Deep Forest Publishing

The DeepChem Book

The DeepChem Book is a step-by-step tutorial series for deep life sciences. The author, Bharath Ramsundar and the DeepChem team, cover the essential tools and techniques for mastering deep learning in life sciences. Tailored for beginners in both machine learning and life sciences, the book builds a repertoire of tools required to perform meaningful work in the dynamic field of life sciences. Going beyond machine learning, the tutorial covers critical aspects of data handling necessary for constructing systems within the deep life sciences. Executed on Google Colab, these tutorials prioritize accessibility and convenience, providing an open avenue for exploration.

"The DeepChem project aims to make high quality open source software for scientific machine learning more accessible to scientists and developers worldwide. We have a particular focus on molecular machine learning and drug discovery, but also support a broad range of applications in bioinformatics, materials science, and computational physics. I started DeepChem while doing my Ph.D. at Stanford, but today DeepChem operates as a global distributed community of researchers spread across many academic and industrial institutions. We hope that you will join our community and help us build!"

- Bharath Ramsundar

www.deepchem.io
www.deepforestsci.com

DeepChem

Bharath Ramsundar

Deep Forest Publishing

The DeepChem Book

Democratizing Deep Learning for Sciences

Bharath Ramsundar and the DeepChem Team

The DeepChem Book

Democratizing Deep-Learning for Drug Discovery Quantum Chemistry, Materials Science and Biology

Bharath Ramsundar and the DeepChem Community

Acknowledgement

We acknowledge the DeepChem community for their contributions and support.

Contents

- ## 1. Introduction To Deepchem

1. The Basic Tools of the Deep Life Sciences
2. Working With Datasets
3. An Introduction To MoleculeNet
4. Molecular Fingerprints
5. Creating Models with TensorFlow and PyTorch
6. Introduction to Graph Convolutions
7. Going Deeper on Molecular Featurizations
8. Working With Splitters
9. Advanced Model Training
10. Creating a high fidelity model from experimental data
11. Putting Multitask Learning to Work
12. Modeling Protein Ligand Interactions
13. Modeling Protein Ligand Interactions With Atomic Convolutions
14. Conditional Generative Adversarial Networks
15. Training a Generative Adversarial Network on MNIST
16. Advanced model training using hyperopt
17. Introduction to Gaussian Processes
18. PytorchLightning Integration

- ## 2. Molecular Machine Learning

1. Molecular Fingerprints
2. Going Deeper on Molecular Featurizations
3. Learning Unsupervised Embeddings for Molecules
4. Atomic Contributions for Molecules
5. Interactive Model Evaluation with Trident Chemwidgets
6. Transfer Learning With ChemBERTa Transformers
7. Training a Normalizing Flow on QM9
8. Large Scale Chemical Screens
9. Introduction to Molecular Attention Transformer
10. Generating molecules with MolGAN
11. Introduction to GROVER

- ## 3. Modeling Proteins

1. Protein Deep Learning

- ## 4. Protein Ligand Modeling

1. Modeling Protein Ligand Interactions
2. Modeling Protein Ligand Interactions With Atomic Convolutions
3. DeepChemXAlphafold

- ## 5. Quantum Chemistry

1. Exploring Quantum Chemistry with GDB1k
2. DeepQMC tutorial
3. Training an Exchange Correlation Functional using Deepchem

- # 6. Bioinformatics

1. Introduction to Bioinformatics
2. Multisequence Alignments
3. Deep probabilistic analysis of single-cell omics data

- # 7. Material Sciences

1. Introduction To Material Science

- # 8. Machine Learning Methods

1. Using Reinforcement Learning to Play Pong
2. Introduction to Model Interpretability
3. Uncertainty In Deep Learning

- # 9. Deep Differential Equations

1. Physics Informed Neural Networks
2. Introducing JaxModel and PINNModel
3. About Neural ODE : Using Torchdiffeq with Deepchem

- # 10. Equivariance

1. Introduction to Equivariance
2. Modeling Protein Ligand Interactions With Atomic Convolutions
3. DeepChemXAlphafold

- # 11. Olfaction

1. Predict Multi Label Odor Descriptors using OpenPOM

The Basic Tools of the Deep Life Sciences

Welcome to DeepChem's introductory tutorial for the deep life sciences. This series of notebooks is a step-by-step guide for you to get to know the new tools and techniques needed to do deep learning for the life sciences. We'll start from the basics, assuming that you're new to machine learning and the life sciences, and build up a repertoire of tools and techniques that you can use to do meaningful work in the life sciences.

Scope: This tutorial will encompass both the machine learning and data handling needed to build systems for the deep life sciences.

Colab

This tutorial and the rest in the sequences are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Why do the DeepChem Tutorial?

1) Career Advancement: Applying AI in the life sciences is a booming industry at present. There are a host of newly funded startups and initiatives at large pharmaceutical and biotech companies centered around AI. Learning and mastering DeepChem will bring you to the forefront of this field and will prepare you to enter a career in this field.

2) Humanitarian Considerations: Disease is the oldest cause of human suffering. From the dawn of human civilization, humans have suffered from pathogens, cancers, and neurological conditions. One of the greatest achievements of the last few centuries has been the development of effective treatments for many diseases. By mastering the skills in this tutorial, you will be able to stand on the shoulders of the giants of the past to help develop new medicine.

3) Lowering the Cost of Medicine: The art of developing new medicine is currently an elite skill that can only be practiced by a small core of expert practitioners. By enabling the growth of open source tools for drug discovery, you can help democratize these skills and open up drug discovery to more competition. Increased competition can help drive down the cost of medicine.

Getting Extra Credit

If you're excited about DeepChem and want to get more involved, there are some things that you can do right now:

- Star DeepChem on GitHub! - https://github.com/deepchem/deepchem
- Join the DeepChem forums and introduce yourself! - https://forum.deepchem.io
- Say hi on the DeepChem gitter - https://gitter.im/deepchem/Lobby
- Make a YouTube video teaching the contents of this notebook.

Prerequisites

This tutorial sequence will assume some basic familiarity with the Python data science ecosystem. We will assume that you have familiarity with libraries such as Numpy, Pandas, and TensorFlow. We'll provide some brief refreshers on basics through the tutorial so don't worry if you're not an expert.

Setup

The first step is to get DeepChem up and running. We recommend using Google Colab to work through this tutorial series. You'll also need to run the following commands to get DeepChem installed on your colab notebook. We are going to use a model based on tensorflow, because of that we've added [tensorflow] to the pip install command to ensure the necessary dependencies are also installed

In [ ]:
```
!pip install --pre deepchem[tensorflow]
```

You can of course run this tutorial locally if you prefer. In this case, don't run the above cell since it will download and install Anaconda on your local machine. In either case, we can now import the `deepchem` package to play with.

In [ ]:
```
import deepchem as dc
dc.__version__
```

Out[ ]: '2.5.0.dev'

Training a Model with DeepChem: A First Example

Deep learning can be used to solve many sorts of problems, but the basic workflow is usually the same. Here are the typical steps you follow.

1. Select the data set you will train your model on (or create a new data set if there isn't an existing suitable one).
2. Create the model.
3. Train the model on the data.
4. Evaluate the model on an independent test set to see how well it works.
5. Use the model to make predictions about new data.

With DeepChem, each of these steps can be as little as one or two lines of Python code. In this tutorial we will walk through a basic example showing the complete workflow to solve a real world scientific problem.

The problem we will solve is predicting the solubility of small molecules given their chemical formulas. This is a very important property in drug development: if a proposed drug isn't soluble enough, you probably won't be able to get enough into the patient's bloodstream to have a therapeutic effect. The first thing we need is a data set of measured solubilities for real molecules. One of the core components of DeepChem is MoleculeNet, a diverse collection of chemical and molecular data sets. For this tutorial, we can use the Delaney solubility data set. The property of solubility in this data set is reported in log(solubility) where solubility is measured in moles/liter.

In [ ]:
```
tasks, datasets, transformers = dc.molnet.load_delaney(featurizer='GraphConv')
train_dataset, valid_dataset, test_dataset = datasets
```

I won't say too much about this code right now. We will see many similar examples in later tutorials. There are two details I do want to draw your attention to. First, notice the `featurizer` argument passed to the `load_delaney()` function. Molecules can be represented in many ways. We therefore tell it which representation we want to use, or in more technical language, how to "featurize" the data. Second, notice that we actually get three different data sets: a training set, a validation set, and a test set. Each of these serves a different function in the standard deep learning workflow.

Now that we have our data, the next step is to create a model. We will use a particular kind of model called a "graph convolutional network", or "graphconv" for short.

In [ ]:
```
model = dc.models.GraphConvModel(n_tasks=1, mode='regression', dropout=0.2,batch_normalize=False)
```

Here again I will not say much about the code. Later tutorials will give lots more information about `GraphConvModel`, as well as other types of models provided by DeepChem.

We now need to train the model on the data set. We simply give it the data set and tell it how many epochs of training to perform (that is, how many complete passes through the data to make).

In [ ]:
```
model.fit(train_dataset, nb_epoch=100)
```

If everything has gone well, we should now have a fully trained model! But do we? To find out, we must evaluate the model on the test set. We do that by selecting an evaluation metric and calling `evaluate()` on the model. For this example, let's use the Pearson correlation, also known as r^2, as our metric. We can evaluate it on both the training set and test set.

In [ ]:
```
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)
print("Training set score:", model.evaluate(train_dataset, [metric], transformers))
print("Test set score:", model.evaluate(test_dataset, [metric], transformers))
```

```
Training set score: {'pearson_r2_score': 0.9323622956442351}
Test set score: {'pearson_r2_score': 0.6898768897014962}
```

Notice that it has a higher score on the training set than the test set. Models usually perform better on the particular data they were trained on than they do on similar but independent data. This is called "overfitting", and it is the reason it is essential to evaluate your model on an independent test set.

Our model still has quite respectable performance on the test set. For comparison, a model that produced totally random outputs would have a correlation of 0, while one that made perfect predictions would have a correlation of 1. Our model does quite well, so now we can use it to make predictions about other molecules we care about.

Since this is just a tutorial and we don't have any other molecules we specifically want to predict, let's just use the first ten molecules from the test set. For each one we print out the chemical structure (represented as a SMILES string) and the predicted log(solubility). To put these predictions in context, we print out the log(solubility) values from the test set

as well.

In [ ]:
```
solubilities = model.predict_on_batch(test_dataset.X[:10])
for molecule, solubility, test_solubility in zip(test_dataset.ids, solubilities, test_dataset.y):
    print(solubility, test_solubility, molecule)
```

```
[-1.8629359] [-1.60114461] c1cc2ccc3cccc4ccc(c1)c2c34
[0.6617248] [0.20848251] Cc1cc(=O)[nH]c(=S)[nH]1
[-0.5705674] [-0.01602738] Oc1ccc(cc1)C2(OC(=O)c3ccccc23)c4ccc(O)cc4
[-2.0929456] [-2.82191713] c1ccc2c(c1)cc3ccc4cccc5ccc2c3c45
[-1.4962314] [-0.52891635] C1=Cc2cccc3cccc1c23
[1.8620405] [1.10168349] CC1CO1
[-0.5858227] [-0.88987406] CCN2c1ccccc1N(C)C(=S)c3cccnc23
[-0.9799993] [-0.52649706] CC12CCC3C(CCc4cc(O)ccc34)C2CCC1=O
[-1.0176951] [-0.76358725] Cn2cc(c1ccccc1)c(=O)c(c2)c3cccc(c3)C(F)(F)F
[0.05622783] [-0.64020358] ClC(Cl)(Cl)C(NC=O)N1C=CN(C=C1)C(NC=O)C(Cl)(Cl)Cl
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro1,
 title={The Basic Tools of the Deep Life Sciences},
 organization={DeepChem},
 author={Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/The_Basic_Tools_of_th
 year={2021},
}
```

Working With Datasets

Data is central to machine learning. This tutorial introduces the `Dataset` class that DeepChem uses to store and manage data. It provides simple but powerful tools for efficiently working with large amounts of data. It also is designed to easily interact with other popular Python frameworks such as NumPy, Pandas, TensorFlow, and PyTorch.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
```

We can now import the `deepchem` package to play with.

In [1]:
```
import deepchem as dc
dc.__version__
```

Out[1]:
```
'2.4.0-rc1.dev'
```

Anatomy of a Dataset

In the last tutorial we loaded the Delaney dataset of molecular solubilities. Let's load it again.

In [2]:
```
tasks, datasets, transformers = dc.molnet.load_delaney(featurizer='GraphConv')
train_dataset, valid_dataset, test_dataset = datasets
```

We now have three Dataset objects: the training, validation, and test sets. What information does each of them contain? We can start to get an idea by printing out the string representation of one of them.

In [3]:
```
print(test_dataset)
```

```
<DiskDataset X.shape: (113,), y.shape: (113, 1), w.shape: (113, 1), ids: ['C1c2ccccc2c3ccc4ccccc4c13' 'COc1ccccc
1Cl'
 'COP(=S)(OC)Oc1cc(Cl)c(Br)cc1Cl' ... 'CCSCCSP(=S)(OC)OC' 'CCC(C)C'
 'COP(=O)(OC)OC(=CCl)c1cc(Cl)c(Cl)cc1Cl'], task_names: ['measured log solubility in mols per litre']>
```

There's a lot of information there, so let's start at the beginning. It begins with the label "DiskDataset". Dataset is an abstract class. It has a few subclasses that correspond to different ways of storing data.

- `DiskDataset` is a dataset that has been saved to disk. The data is stored in a way that can be efficiently accessed, even if the total amount of data is far larger than your computer's memory.
- `NumpyDataset` is an in-memory dataset that holds all the data in NumPy arrays. It is a useful tool when manipulating small to medium sized datasets that can fit entirely in memory.
- `ImageDataset` is a more specialized class that stores some or all of the data in image files on disk. It is useful when working with models that have images as their inputs or outputs.

Now let's consider the contents of the Dataset. Every Dataset stores a list of *samples*. Very roughly speaking, a sample is a single data point. In this case, each sample is a molecule. In other datasets a sample might correspond to an experimental assay, a cell line, an image, or many other things. For every sample the dataset stores the following information.

- The *features*, referred to as `X`. This is the input that should be fed into a model to represent the sample.
- The *labels*, referred to as `y`. This is the desired output from the model. During training, it tries to make the model's output for each sample as close as possible to `y`.
- The *weights*, referred to as `w`. This can be used to indicate that some data values are more important than others. In later tutorials we will see examples of how this is useful.
- An *ID*, which is a unique identifier for the sample. This can be anything as long as it is unique. Sometimes it is just an integer index, but in this dataset the ID is a SMILES string describing the molecule.

Notice that `X`, `y`, and `w` all have 113 as the size of their first dimension. That means this dataset contains 113 samples.

The final piece of information listed in the output is `task_names`. Some datasets contain multiple pieces of information for each sample. For example, if a sample represents a molecule, the dataset might record the results of several

different experiments on that molecule. This dataset has only a single task: "measured log solubility in mols per litre". Also notice that `y` and `w` each have shape (113, 1). The second dimension of these arrays usually matches the number of tasks.

Accessing Data from a Dataset

There are many ways to access the data contained in a dataset. The simplest is just to directly access the `X`, `y`, `w`, and `ids` properties. Each of these returns the corresponding information as a NumPy array.

In [4]:
```
test_dataset.y
```

Out[4]:
```
array([[-1.7065408738415053],
       [0.2911162036252904],
       [-1.4272475857596547],
       [-0.9254664241210759],
       [-1.9526976701170347],
       [1.3514839414275706],
       [-0.8591934405084332],
       [-0.6509069205829855],
       [-0.32900957160729316],
       [0.6082797680572224],
       [1.8295961803473488],
       [1.6213096604219008],
       [1.3751528641463715],
       [0.45632528420252055],
       [1.0532555151706793],
       [-1.1053502367839627],
       [-0.20119738892576 83],
       [0.3479216181504126],
       [-0.9870056231899582],
       [-0.8161160011602158],
       [0.8402352107014712],
       [0.22815686919328],
       [0.06247441016167367],
       [1.040947675356903],
       [-0.5197810887208284],
       [0.8023649343513898],
       [-0.41895147793873655],
       [-2.5964923680684198],
       [1.7443880585596654],
       [0.45206487811313645],
       [0.233837410645792],
       [-1.7917489956291888],
       [0.7739622270888287],
       [1.0011838851893173],
       [-0.05445006806920272],
       [1.1043803882432892],
       [0.7597608734575482],
       [-0.7001382798380905],
       [0.8213000725264304],
       [-1.3136367567094103],
       [0.4567986626568967],
       [-0.5732728540653187],
       [0.4094608172192949],
       [-0.3242757870635329],
       [-0.049716283525442634],
       [-0.39054877067617544],
       [-0.080959261514259 96],
       [-0.2627365879946506],
       [-0.5467636606202616],
       [1.997172153196459],
       [-0.03551492989416198],
       [1.4508934168465344],
       [-0.8639272250521937],
       [0.23904457364392848],
       [0.5278054308132993],
       [-0.48475108309700315],
       [0.2248432200126478],
       [0.3431878336066523],
       [1.5029650468278963],
       [-0.4946920306388995],
       [0.3479216181504126],
       [0.7928973652638694],
       [0.5609419226196206],
       [-0.13965818985688602],
       [-0.13965818985688602],
       [0.15857023640000523],
       [1.6071083067906202],
       [1.9006029485037514],
```

```
       [-0.7171799041956278],
       [-0.8165893796145915],
       [-0.130190620769365566],
       [-0.243801449819609866],
       [-0.149125758944406381],
       [0.9538460397517154],
       [-0.078118990788003741],
       [-0.182262250750727581],
       [0.2532459272752089],
       [0.6887541053011454],
       [0.044012650441008896],
       [-0.5514974451640217],
       [-0.2580028034508905],
       [-0.021313576262881533],
       [-2.4128215277705247],
       [0.07336211461232214],
       [0.9017744097703536],
       [1.9384732248538328],
       [0.8402352107014712],
       [-0.10652169805056463],
       [1.07692443788948],
       [-0.403803367398704],
       [1.2662758196398873],
       [-0.2532690189071302],
       [0.29064282517091444],
       [0.9443784706641951],
       [-0.41563782875810434],
       [-0.7370617992794205],
       [-1.0012069768212388],
       [0.466266231744417061],
       [0.3758509469585975],
       [-0.466289323376338161],
       [1.2662758196398873],
       [-1.4968342185529295],
       [-0.178001844661343441],
       [0.8828392715953128],
       [-0.6083028596891439],
       [-2.170451759130003],
       [0.328986479975371841],
       [0.30058377271281071],
       [0.64615004440730381],
       [1.5058053175541524],
       [-0.007585601085977053],
       [-0.049716283525442634],
       [-0.6849901692980588]], dtype=object)
```

This is a very easy way to access data, but you should be very careful about using it. This requires the data for all samples to be loaded into memory at once. That's fine for small datasets like this one, but for large datasets it could easily take more memory than you have.

A better approach is to iterate over the dataset. That lets it load just a little data at a time, process it, then free the memory before loading the next bit. You can use the `itersamples()` method to iterate over samples one at a time.

In [5]:
```
for X, y, w, id in test_dataset.itersamples():
    print(y, id)
```

```
[-1.70654087] C1c2ccccc2c3ccc4ccccc4c13
[0.2911162] COc1ccccc1Cl
[-1.42724759] COP(=S)(OC)Oc1cc(Cl)c(Br)cc1Cl
[-0.92546642] ClC(Cl)CC(=O)NC2=C(Cl)C(=O)c1ccccc1C2=O
[-1.95269767] ClC(Cl)C(c1ccc(Cl)cc1)c2ccc(Cl)cc2
[1.35148394] COC(=O)C=C
[-0.85919344] CN(C)C(=O)Nc2ccc(Oc1ccc(Cl)cc1)cc2
[-0.65090692] N(=Nc1ccccc1)c2ccccc2
[-0.32900957] CC(C)c1ccc(C)cc1
[0.60827977] Oc1c(Cl)cccc1Cl
[1.82959618] OCC2OC(OC1(CO)OC(CO)C(O)C1O)C(O)C(O)C2O
[1.62130966] OC1C(O)C(O)C(O)C(O)C1O
[1.37515286] Cn2c(=O)n(C)c1ncn(CC(O)CO)c1c2=O
[0.45632528] OCC(NC(=O)C(Cl)Cl)C(O)c1ccc(cc1)N(=O)=O
[1.05325552] CCC(O)(CC)CC
[-1.10535024] CC45CCC2C(CCC3CC1SC1CC23C)C4CCC5O
[-0.20119739] Brc1ccccc1Br
[0.34792162] Oc1c(Cl)cc(Cl)cc1Cl
[-0.98700562] CCCN(CCC)c1c(cc(cc1N(=O)=O)S(N)(=O)=O)N(=O)=O
[-0.816116] C2c1ccccc1N(CCF)C(=O)c3ccccc23
[0.84023521] CC(C)C(=O)C(C)C
[0.22815687] O=C1NC(=O)NC(=O)C1(C(C)C)CC=C(C)C
[0.06247441] c1c(O)C2C(=O)C3cc(O)ccC3OC2cc1(OC)
[1.04094768] Cn1cnc2n(C)c(=O)n(C)c(=O)c12
[-0.51978109] CC(=O)SC4CC1=CC(=O)CCC1(C)C5CCC2(C)C(CCC23CCC(=O)O3)C45
[0.80236493] Cc1ccc(O)cc1C
[-0.41895148] O(c1ccccc1)c2ccccc2
```

[-2.59649237] Clc1cc(Cl)c(cc1Cl)c2cc(Cl)c(Cl)cc2Cl
[1.74438806] NC(=O)c1cccnc1
[0.45206488] Sc1ccccc1
[0.23383741] CNC(=O)Oc1cc(C)cc(C)c1
[-1.791749] ClC1CC2C(C1Cl)C3(Cl)C(=C(Cl)C2(Cl)C3(Cl)Cl)Cl
[0.77396223] CSSC
[1.00118389] NC(=O)c1ccccc1
[-0.05445007] Clc1ccccc1Br
[1.10438039] COC(=O)c1ccccc1OC2OC(COC3OCC(O)C(O)C3O)C(O)C(O)C2O
[0.75976087] CCCCC(O)CC
[-0.70013828] CCN2c1nc(C)cc(C)c1NC(=O)c3cccnc23
[0.82130007] Oc1cc(Cl)cc(Cl)c1
[-1.31363676] Cc1cccc2c1ccc3ccccc32
[0.45679866] CCCCC(CC)CO
[-0.57327285] CC(C)N(C(C)C)C(=O)SCC(=CCl)Cl
[0.40946082] Cc1ccccc1
[-0.32427579] Clc1cccc(n1)C(Cl)(Cl)Cl
[-0.04971628] C1CCC=CCC1
[-0.39054877] CN(C)C(=S)SSC(=S)N(C)C
[-0.08095926] COC1=CC(=O)CC(C)C13Oc2c(Cl)c(OC)cc(OC)c2C3=O
[-0.26273659] CCCCCCCCCCO
[-0.54676366] CCC(C)(C)CC
[1.99717215] CNC(=O)C(C)SCCSP(=O)(OC)(OC)
[-0.03551493] Oc1cc(Cl)c(Cl)c(Cl)c1Cl
[1.45089342] CCCC=O
[-0.86392723] CC4CC3C2CCC1=CC(=O)C=CC1(C)C2(F)C(O)CC3(C)C4(O)C(=O)COC(C)=O
[0.23904457] CCCC
[0.52780543] COc1ccccc1O
[-0.48475108] CC1CC2C3CCC(O)(C(=O)C)C3(C)CC(O)C2(F)C4(C)C=CC(=O)C=C14
[0.22484322] ClC(Cl)C(Cl)(Cl)Cl
[0.34318783] CCOC(=O)c1ccccc1C(=O)OCC
[1.50296505] CC(C)CO
[-0.49469203] CC(C)Cc1ccccc1
[0.34792162] ICI
[0.79289737] CCCC(O)CCC
[0.56094192] CCCCCOC(=O)C
[-0.13965819] Oc1c(Cl)c(Cl)cc(Cl)c1Cl
[-0.13965819] CCCc1ccccc1
[0.15857024] FC(F)(Cl)C(F)(F)Cl
[1.60710831] CC=CC=O
[1.90060295] CN(C)C(=O)N(C)C
[-0.7171799] Cc1cc(C)c(C)cc1C
[-0.81658938] CC(=O)OC3(CCC4C2CCC1=CC(=O)CCC1C2CCC34C)C#C
[-0.13019062] CCOP(=S)(OCC)N2C(=O)c1ccccc1C2=O
[-0.24380145] c1ccccc1NC(=O)c2c(O)cccc2
[-0.14912576] CCN(CC)C(=S)SCC(Cl)=C
[0.95384604] ClCC
[-0.07811899] CC(=O)Nc1cc(NS(=O)(=O)C(F)(F)F)c(C)cc1C
[-0.18226225] O=C(C=CC=Cc2ccc1OCOc1c2)N3CCCCC3
[0.25324593] CC/C=C\C
[0.68875411] CNC(=O)ON=C(CSC)C(C)(C)C
[0.04401265] O=C2NC(=O)C1(CCCCCCC1)C(=O)N2
[-0.55149745] c1(C(C)(C)C)cc(C(C)(C)C)cc(OC(=O)NC)c1
[-0.2580028] Oc2cc(O)c1C(=O)CC(Oc1c2)c3ccc(O)c(O)c3
[-0.02131358] O=C(c1ccccc1)c2ccccc2
[-2.41282153] CCCCCCCCCCCCCCCCCCCC
[0.07336211] N(Nc1ccccc1)c2ccccc2
[0.90177441] CCC(CC)CO
[1.93847322] Oc1ccncc1
[0.84023521] Cl\C=C/Cl
[-0.1065217] CC1CCCC1
[1.07692444] CC(C)CC(C)O
[-0.40380337] O2c1ccc(N)cc1N(C)C(=O)c3cc(C)ccc23
[1.26627582] CC(C)(C)CO
[-0.25326902] CC(C)(C)C(=O)C(Oc1ccc(Cl)cc1)n2cncn2
[0.29064283] Cc1cc(no1)C(=O)NNCc2ccccc2
[0.94437847] CC=C
[-0.41563783] Oc1ccc(Cl)cc1Cc2cc(Cl)ccc2O
[-0.7370618] CCOC(=O)Nc2cccc(OC(=O)Nc1ccccc1)c2
[-1.00120698] O=C1c2ccccc2C(=O)c3ccccc13
[0.46626623] CCCCCCC(C)O
[0.37585095] CC1=C(C(=O)Nc2ccccc2)S(=O)(=O)CCO1
[-0.46628932] CCCCc1ccccc1
[1.26627582] O=C1NC(=O)C(=O)N1
[-1.49683422] COP(=S)(OC)Oc1ccc(Sc2ccc(OP(=S)(OC)OC)cc2)cc1
[-0.17800184] NS(=O)(=O)c1cc(ccc1Cl)C2(O)NC(=O)c3ccccc23
[0.88283927] CC(C)COC(=O)C
[-0.60830286] CC(C)C(C)(C)C
[-2.17045176] Clc1ccc(c(Cl)c1Cl)c2c(Cl)cc(Cl)c(Cl)c2Cl
[0.32898648] N#Cc1ccccc1C#N
[0.30058377] Cc1cccc(c1)N(=O)=O
[0.64615004] FC(F)(F)C(Cl)Br
[1.50580532] CNC(=O)ON=C(SC)C(=O)N(C)C

```
[-0.0075856] CCSCCSP(=S)(OC)OC
[-0.04971628] CCC(C)C
[-0.68499017] COP(=O)(OC)OC(=CCl)c1cc(Cl)c(Cl)cc1Cl
```

Most deep learning models can process a batch of multiple samples all at once. You can use `iterbatches()` to iterate over batches of samples.

In [6]:
```
for X, y, w, ids in test_dataset.iterbatches(batch_size=50):
    print(y.shape)
```

```
(50, 1)
(50, 1)
(13, 1)
```

`iterbatches()` has other features that are useful when training models. For example, `iterbatches(batch_size=100, epochs=10, deterministic=False)` will iterate over the complete dataset ten times, each time with the samples in a different random order.

Datasets can also expose data using the standard interfaces for TensorFlow and PyTorch. To get a `tensorflow.data.Dataset`, call `make_tf_dataset()`. To get a `torch.utils.data.IterableDataset`, call `make_pytorch_dataset()`. See the API documentation for more details.

The final way of accessing data is `to_dataframe()`. This copies the data into a Pandas `DataFrame`. This requires storing all the data in memory at once, so you should only use it with small datasets.

In [7]:
```
test_dataset.to_dataframe()
```

Out[7]:

	X	y	w	ids
0	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.706541	1.0	C1c2ccccc2c3ccc4ccccc4c13
1	<deepchem.feat.mol_graphs.ConvMol object at 0x...	0.291116	1.0	COc1ccccc1Cl
2	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.427248	1.0	COP(=S)(OC)Oc1cc(Cl)c(Br)cc1Cl
3	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.925466	1.0	ClC(Cl)CC(=O)NC2=C(Cl)C(=O)c1ccccc1C2=O
4	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-1.952698	1.0	ClC(Cl)C(c1ccc(Cl)cc1)c2ccc(Cl)cc2
...	...	...	...	...
108	<deepchem.feat.mol_graphs.ConvMol object at 0x...	0.646150	1.0	FC(F)(F)C(Cl)Br
109	<deepchem.feat.mol_graphs.ConvMol object at 0x...	1.505805	1.0	CNC(=O)ON=C(SC)C(=O)N(C)C
110	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.007586	1.0	CCSCCSP(=S)(OC)OC
111	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.049716	1.0	CCC(C)C
112	<deepchem.feat.mol_graphs.ConvMol object at 0x...	-0.684990	1.0	COP(=O)(OC)OC(=CCl)c1cc(Cl)c(Cl)cc1Cl

113 rows × 4 columns

Creating Datasets

Now let's talk about how you can create your own datasets. Creating a `NumpyDataset` is very simple: just pass the arrays containing the data to the constructor. Let's create some random arrays, then wrap them in a NumpyDataset.

In [8]:
```
import numpy as np

X = np.random.random((10, 5))
y = np.random.random((10, 2))
dataset = dc.data.NumpyDataset(X=X, y=y)
print(dataset)
```

```
<NumpyDataset X.shape: (10, 5), y.shape: (10, 2), w.shape: (10, 1), ids: [0 1 2 3 4 5 6 7 8 9], task_names: [0 1
]>
```

Notice that we did not specify weights or IDs. These are optional, as is `y` for that matter. Only `X` is required. Since we left them out, it automatically built `w` and `ids` arrays for us, setting all weights to 1 and setting the IDs to integer indices.

In [9]:
```
dataset.to_dataframe()
```

Out[9]:

	X1	X2	X3	X4	X5	y1	y2	w	ids
0	0.547330	0.919941	0.289138	0.431806	0.776672	0.532579	0.443258	1.0	0
1	0.980867	0.642487	0.460640	0.500153	0.014848	0.678259	0.274029	1.0	1
2	0.953254	0.704446	0.857458	0.378372	0.705789	0.704786	0.901080	1.0	2
3	0.904970	0.729710	0.304247	0.861546	0.917029	0.121747	0.758845	1.0	3
4	0.464144	0.059168	0.600405	0.880529	0.688043	0.595495	0.719861	1.0	4
5	0.820482	0.139002	0.627421	0.129399	0.920024	0.634030	0.464525	1.0	5
6	0.113727	0.551801	0.536189	0.066091	0.311320	0.699331	0.171532	1.0	6
7	0.516131	0.918903	0.429036	0.844973	0.639367	0.464089	0.337989	1.0	7
8	0.809393	0.201450	0.821420	0.841390	0.100026	0.230462	0.376151	1.0	8
9	0.076750	0.389277	0.350371	0.291806	0.127522	0.544606	0.306578	1.0	9

What about creating a DiskDataset? If you have the data in NumPy arrays, you can call `DiskDataset.from_numpy()` to save it to disk. Since this is just a tutorial, we will save it to a temporary directory.

In [10]:
```
import tempfile

with tempfile.TemporaryDirectory() as data_dir:
    disk_dataset = dc.data.DiskDataset.from_numpy(X=X, y=y, data_dir=data_dir)
    print(disk_dataset)
```

```
<DiskDataset X.shape: (10, 5), y.shape: (10, 2), w.shape: (10, 1), ids: [0 1 2 3 4 5 6 7 8 9], task_names: [0 1]
>
```

What about larger datasets that can't fit in memory? What if you have some huge files on disk containing data on hundreds of millions of molecules? The process for creating a DiskDataset from them is slightly more involved. Fortunately, DeepChem's `DataLoader` framework can automate most of the work for you. That is a larger subject, so we will return to it in a later tutorial.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

An Introduction To MoleculeNet

By Bharath Ramsundar | Twitter

One of the most powerful features of DeepChem is that it comes "batteries included" with datasets to use. The DeepChem developer community maintains the MoleculeNet [1] suite of datasets which maintains a large collection of different scientific datasets for use in machine learning applications. The original MoleculeNet suite had 17 datasets mostly focused on molecular properties. Over the last several years, MoleculeNet has evolved into a broader collection of scientific datasets to facilitate the broad use and development of scientific machine learning tools.

These datasets are integrated with the rest of the DeepChem suite so you can conveniently access these through functions in the `dc.molnet` submodule. You've already seen a few examples of these loaders already as you've worked through the tutorial series. The full documentation for the MoleculeNet suite is available in our docs [2].

[1] Wu, Zhenqin, et al. "MoleculeNet: a benchmark for molecular machine learning." Chemical science 9.2 (2018): 513-530.

[2] https://deepchem.readthedocs.io/en/latest/moleculenet.html

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following installation commands. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install DeepChem again on your local machine.

In [ ]:
```
!pip install --pre deepchem
```

We can now import the `deepchem` package to play with.

In [1]:
```
import deepchem as dc
dc.__version__
```

Out[1]:
```
'2.4.0-rc1.dev'
```

MoleculeNet Overview

In the last two tutorials we loaded the Delaney dataset of molecular solubilities. Let's load it one more time.

In [2]:
```
tasks, datasets, transformers = dc.molnet.load_delaney(featurizer='GraphConv', splitter='random')
```

Notice that the loader function we invoke `dc.molnet.load_delaney` lives in the `dc.molnet` submodule of MoleculeNet loaders. Let's take a look at the full collection of loaders available for us

In [3]:
```
[method for method in dir(dc.molnet) if "load_" in method ]
```

Out[3]:
```
['load_bace_classification',
 'load_bace_regression',
 'load_bandgap',
 'load_bbbc001',
 'load_bbbc002',
 'load_bbbp',
 'load_cell_counting',
 'load_chembl',
 'load_chembl25',
 'load_clearance',
 'load_clintox',
 'load_delaney',
 'load_factors',
 'load_function',
 'load_hiv',
 'load_hopv',
 'load_hppb',
 'load_kaggle',
 'load_kinase',
 'load_lipo',
 'load_mp_formation_energy',
 'load_mp_metallicity',
 'load_muv',
 'load_nci',
 'load_pcba',
 'load_pcba_146',
 'load_pcba_2475',
 'load_pdbbind',
 'load_pdbbind_from_dir',
 'load_pdbbind_grid',
 'load_perovskite',
 'load_ppb',
 'load_qm7',
 'load_qm7_from_mat',
 'load_qm7b_from_mat',
 'load_qm8',
 'load_qm9',
 'load_sampl',
 'load_sider',
 'load_sweet',
 'load_thermosol',
 'load_tox21',
 'load_toxcast',
 'load_uspto',
 'load_uv',
 'load_zinc15']
```

The set of MoleculeNet loaders is actively maintained by the DeepChem community and we work on adding new datasets to the collection. Let's see how many datasets there are in MoleculeNet today

In [4]:
```
len([method for method in dir(dc.molnet) if "load_" in method ])
```

Out[4]:
```
46
```

MoleculeNet Dataset Categories

There's a lot of different datasets in MoleculeNet. Let's do a quick overview of the different types of datasets available. We'll break datasets into different categories and list loaders which belong to those categories. More details on each of these datasets can be found at https://deepchem.readthedocs.io/en/latest/moleculenet.html. The original MoleculeNet paper [1] provides details about a subset of these papers. We've marked these datasets as "V1" below. All remaining dataset are "V2" and not documented in the older paper.

Quantum Mechanical Datasets

MoleculeNet's quantum mechanical datasets contain various quantum mechanical property prediction tasks. The current set of quantum mechanical datasets includes QM7, QM7b, QM8, QM9. The associated loaders are

- `dc.molnet.load_qm7` : V1
- `dc.molnet.load_qm7b_from_mat` : V1
- `dc.molnet.load_qm8` : V1
- `dc.molnet.load_qm9` : V1

Physical Chemistry Datasets

The physical chemistry dataset collection contain a variety of tasks for predicting various physical properties of

molecules.

- `dc.molnet.load_delaney` : V1. This dataset is also referred to as ESOL in the original paper.
- `dc.molnet.load_sampl` : V1. This dataset is also referred to as FreeSolv in the original paper.
- `dc.molnet.load_lipo` : V1. This dataset is also referred to as Lipophilicity in the original paper.
- `dc.molnet.load_thermosol` : V2.
- `dc.molnet.load_hppb` : V2.
- `dc.molnet.load_hopv` : V2. This dataset is drawn from a recent publication [3]

Chemical Reaction Datasets

These datasets hold chemical reaction datasets for use in computational retrosynthesis / forward synthesis.

- `dc.molnet.load_uspto`

Biochemical/Biophysical Datasets

These datasets are drawn from various biochemical/biophysical datasets that measure things like the binding affinity of compounds to proteins.

- `dc.molnet.load_pcba` : V1
- `dc.molnet.load_nci` : V2.
- `dc.molnet.load_muv` : V1
- `dc.molnet.load_hiv` : V1
- `dc.molnet.load_ppb` : V2.
- `dc.molnet.load_bace_classification` : V1. This loader loads the classification task for the BACE dataset from the original MoleculeNet paper.
- `dc.molnet.load_bace_regression` : V1. This loader loads the regression task for the BACE dataset from the original MoleculeNet paper.
- `dc.molnet.load_kaggle` : V2. This dataset is from Merck's drug discovery kaggle contest and is described in [4].
- `dc.molnet.load_factors` : V2. This dataset is from [4].
- `dc.molnet.load_uv` : V2. This dataset is from [4].
- `dc.molnet.load_kinase` : V2. This datset is from [4].

Molecular Catalog Datasets

These datasets provide molecular datasets which have no associated properties beyond the raw SMILES formula or structure. These types of datasets are useful for generative modeling tasks.

- `dc.molnet.load_zinc15` : V2
- `dc.molnet.load_chembl` : V2
- `dc.molnet.load_chembl25` : V2

Physiology Datasets

These datasets measure physiological properties of how molecules interact with human patients.

- `dc.molnet.load_bbbp` : V1
- `dc.molnet.load_tox21` : V1
- `dc.molnet.load_toxcast` : V1
- `dc.molnet.load_sider` : V1
- `dc.molnet.load_clintox` : V1
- `dc.molnet.load_clearance` : V2.

Structural Biology Datasets

These datasets contain 3D structures of macromolecules along with associated properties.

- `dc.molnet.load_pdbbind` : V1

Microscopy Datasets

These datasets contain microscopy image datasets, typically of cell lines. These datasets were not in the original MoleculeNet paper.

- `dc.molnet.load_bbbc001` : V2
- `dc.molnet.load_bbbc002` : V2
- `dc.molnet.load_cell_counting` : V2

Materials Properties Datasets

These datasets compute properties of various materials.

- `dc.molnet.load_bandgap` : V2
- `dc.molnet.load_perovskite` : V2
- `dc.molnet.load_mp_formation_energy` : V2
- `dc.molnet.load_mp_metallicity` : V2

[3] Lopez, Steven A., et al. "The Harvard organic photovoltaic dataset." Scientific data 3.1 (2016): 1-7.

[4] Ramsundar, Bharath, et al. "Is multitask deep learning practical for pharma?." Journal of chemical information and modeling 57.8 (2017): 2068-2076.

MoleculeNet Loaders Explained

All MoleculeNet loader functions take the form `dc.molnet.load_X`. Loader functions return a tuple of arguments `(tasks, datasets, transformers)`. Let's walk through each of these return values and explain what we get:

1. `tasks` : This is a list of task-names. Many datasets in MoleculeNet are "multitask". That is, a given datapoint has multiple labels associated with it. These correspond to different measurements or values associated with this datapoint.
2. `datasets` : This field is a tuple of three `dc.data.Dataset` objects `(train, valid, test)`. These correspond to the training, validation, and test set for this MoleculeNet dataset.
3. `transformers` : This field is a list of `dc.trans.Transformer` objects which were applied to this dataset during processing.

This is abstract so let's take a look at each of these fields for the `dc.molnet.load_delaney` function we invoked above. Let's start with `tasks`.

In [5]:
```
tasks
```

Out[5]:
```
['measured log solubility in mols per litre']
```

We have one task in this dataset which corresponds to the measured log solubility in mol/L. Let's now take a look at `datasets`:

In [6]:
```
datasets
```

Out[6]:
```
(<DiskDataset X.shape: (902,), y.shape: (902, 1), w.shape: (902, 1), ids: ['CCC(C)Cl' 'O=C1NC(=O)NC(=O)C1(C(C)C
)CC=C' 'Oc1ccccn1' ...
  'CCCCCCCC(=O)OCC' 'O=Cc1ccccc1' 'CCCC=C(CC)C=O'], task_names: ['measured log solubility in mols per litre']>,
 <DiskDataset X.shape: (113,), y.shape: (113, 1), w.shape: (113, 1), ids: ['CSc1nc(nc(n1)N(C)C)N(C)C' 'CC#N' 'C
CCCCCCC#C' ... 'ClCCBr'
  'CCN(CC)C(=O)CSc1ccc(Cl)nn1' 'CC(=O)OC3CCC4C2CCC1=CC(=O)CCC1(C)C2CCC34C '], task_names: ['measured log solubi
lity in mols per litre']>,
 <DiskDataset X.shape: (113,), y.shape: (113, 1), w.shape: (113, 1), ids: ['CCCCc1c(C)nc(nc1O)N(C)C '
  'Cc3cc2nc1c(=O)[nH]c(=O)nc1n(CC(O)C(O)C(O)CO)c2cc3C'
  'CSc1nc(NC(C)C)nc(NC(C)C)n1' ... 'O=c1[nH]cnc2[nH]ncc12 '
  'CC(=C)C1CC=C(C)C(=O)C1' 'OC(C(=O)c1ccccc1)c2ccccc2'], task_names: ['measured log solubility in mols per litr
e']>)
```

As we mentioned previously, we see that `datasets` is a tuple of 3 datasets. Let's split them out.

In [7]:
```
train, valid, test = datasets
```

In [8]:
```
train
```

Out[8]:
```
<DiskDataset X.shape: (902,), y.shape: (902, 1), w.shape: (902, 1), ids: ['CCC(C)Cl' 'O=C1NC(=O)NC(=O)C1(C(C)C)
CC=C' 'Oc1ccccn1' ...
 'CCCCCCCC(=O)OCC' 'O=Cc1ccccc1' 'CCCC=C(CC)C=O'], task_names: ['measured log solubility in mols per litre']>
```

In [9]:
```
valid
```

Out[9]:
```
<DiskDataset X.shape: (113,), y.shape: (113, 1), w.shape: (113, 1), ids: ['CSc1nc(nc(n1)N(C)C)N(C)C' 'CC#N' 'CC
CCCCCC#C' ... 'ClCCBr'
 'CCN(CC)C(=O)CSc1ccc(Cl)nn1' 'CC(=O)OC3CCC4C2CCC1=CC(=O)CCC1(C)C2CCC34C '], task_names: ['measured log solubil
ity in mols per litre']>
```

In [10]:
```
test
```

Out[10]:
```
<DiskDataset X.shape: (113,), y.shape: (113, 1), w.shape: (113, 1), ids: ['CCCCc1c(C)nc(nc1O)N(C)C '
 'Cc3cc2nc1c(=O)[nH]c(=O)nc1n(CC(O)C(O)C(O)CO)c2cc3C'
 'CSc1nc(NC(C)C)nc(NC(C)C)n1' ... 'O=c1[nH]cnc2[nH]ncc12 '
 'CC(=C)C1CC=C(C)C(=O)C1' 'OC(C(=O)c1ccccc1)c2ccccc2'], task_names: ['measured log solubility in mols per litre
']>
```

Let's peek into one of the datapoints in the `train` dataset.

In [11]:
```
train.X[0]
```

Out[11]:
```
<deepchem.feat.mol_graphs.ConvMol at 0x7fe1ef601438>
```

Note that this is a `dc.feat.mol_graphs.ConvMol` object produced by `dc.feat.ConvMolFeaturizer`. We'll say more about how to control choice of featurization shortly. Finally let's take a look at the `transformers` field:

In [12]:
```
transformers
```

Out[12]:
```
[<deepchem.trans.transformers.NormalizationTransformer at 0x7fe2029bdfd0>]
```

So we see that one transformer was applied, the `dc.trans.NormalizationTransformer`.

After reading through this description so far, you may be wondering what choices are made under the hood. As we've briefly mentioned previously, datasets can be processed with different choices of "featurizers". Can we control the choice of featurization here? In addition, how was the source dataset split into train/valid/test as three different datasets?

You can use the 'featurizer' and 'splitter' keyword arguments and pass in different strings. Common possible choices for 'featurizer' are 'ECFP', 'GraphConv', 'Weave' and 'smiles2img' corresponding to the `dc.feat.CircularFingerprint`, `dc.feat.ConvMolFeaturizer`, `dc.feat.WeaveFeaturizer` and `dc.feat.SmilesToImage` featurizers. Common possible choices for 'splitter' are `None`, 'index', 'random', 'scaffold' and 'stratified' corresponding to no split, `dc.splits.IndexSplitter`, `dc.splits.RandomSplitter`, `dc.splits.SingletaskStratifiedSplitter`. We haven't talked much about splitters yet, but intuitively they're a way to partition a dataset based on different criteria. We'll say more in a future tutorial.

Instead of a string, you also can pass in any `Featurizer` or `Splitter` object. This is very useful when, for example, a Featurizer has constructor arguments you can use to customize its behavior.

In [13]:
```
tasks, datasets, transformers = dc.molnet.load_delaney(featurizer="ECFP", splitter="scaffold")
```

In [14]:
```
(train, valid, test) = datasets
```

In [15]:
```
train
```

Out[15]:
```
<DiskDataset X.shape: (902, 1024), y.shape: (902, 1), w.shape: (902, 1), ids: ['CC(C)=CCCC(C)=CC(=O)' 'CCCC=C'
'CCCCCCCCCCCCCC' ...
 'Nc2cccc3nc1ccccc1cc23 ' 'C1CCCCCC1' 'OC1CCCCCC1'], task_names: ['measured log solubility in mols per litre']>
```

In [16]:
```
train.X[0]
```

Out[16]:
```
array([0., 0., 0., ..., 0., 0., 0.])
```

Note that unlike the earlier invocation we have numpy arrays produced by `dc.feat.CircularFingerprint` instead of `ConvMol` objects produced by `dc.feat.ConvMolFeaturizer`.

Give it a try for yourself. Try invoking MoleculeNet to load some other datasets and experiment with different featurizer/split options and see what happens!

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Molecular Fingerprints

Molecules can be represented in many ways. This tutorial introduces a type of representation called a "molecular fingerprint". It is a very simple representation that often works well for small drug-like molecules.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
```

We can now import the `deepchem` package to play with.

In [1]:
```
import deepchem as dc
dc.__version__
```

Out[1]:
```
'2.4.0-rc1.dev'
```

What is a Fingerprint?

Deep learning models almost always take arrays of numbers as their inputs. If we want to process molecules with them, we somehow need to represent each molecule as one or more arrays of numbers.

Many (but not all) types of models require their inputs to have a fixed size. This can be a challenge for molecules, since different molecules have different numbers of atoms. If we want to use these types of models, we somehow need to represent variable sized molecules with fixed sized arrays.

Fingerprints are designed to address these problems. A fingerprint is a fixed length array, where different elements indicate the presence of different features in the molecule. If two molecules have similar fingerprints, that indicates they contain many of the same features, and therefore will likely have similar chemistry.

DeepChem supports a particular type of fingerprint called an "Extended Connectivity Fingerprint", or "ECFP" for short. They also are sometimes called "circular fingerprints". The ECFP algorithm begins by classifying atoms based only on their direct properties and bonds. Each unique pattern is a feature. For example, "carbon atom bonded to two hydrogens and two heavy atoms" would be a feature, and a particular element of the fingerprint is set to 1 for any molecule that contains that feature. It then iteratively identifies new features by looking at larger circular neighborhoods. One specific feature bonded to two other specific features becomes a higher level feature, and the corresponding element is set for any molecule that contains it. This continues for a fixed number of iterations, most often two.

Let's take a look at a dataset that has been featurized with ECFP.

In [2]:
```
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='ECFP')
train_dataset, valid_dataset, test_dataset = datasets
print(train_dataset)
```

```
<DiskDataset X.shape: (6264, 1024), y.shape: (6264, 12), w.shape: (6264, 12), task_names: ['NR-AR' 'NR-AR-LBD' '
NR-AhR' ... 'SR-HSE' 'SR-MMP' 'SR-p53']>
```

The feature array `X` has shape (6264, 1024). That means there are 6264 samples in the training set. Each one is represented by a fingerprint of length 1024. Also notice that the label array `y` has shape (6264, 12): this is a multitask dataset. Tox21 contains information about the toxicity of molecules. 12 different assays were used to look for signs of toxicity. The dataset records the results of all 12 assays, each as a different task.

Let's also take a look at the weights array.

In [3]:
```
train_dataset.w
```

Out[3]:
```
array([[1.0433141624730409, 1.0369942196531792, 8.53921568627451, ...,
        1.060388945752303, 1.1895710249165168, 1.0700990099009902],
       [1.0433141624730409, 1.0369942196531792, 1.1326397919375812, ...,
        0.0, 1.1895710249165168, 1.0700990099009902],
       [0.0, 0.0, 0.0, ..., 1.060388945752303, 0.0, 0.0],
       ...,
       [0.0, 0.0, 0.0, ..., 0.0, 0.0, 0.0],
       [1.0433141624730409, 1.0369942196531792, 8.53921568627451, ...,
        1.060388945752303, 0.0, 0.0],
       [1.0433141624730409, 1.0369942196531792, 1.1326397919375812, ...,
        1.060388945752303, 1.1895710249165168, 1.0700990099009902]],
      dtype=object)
```

Notice that some elements are 0. The weights are being used to indicate missing data. Not all assays were actually performed on every molecule. Setting the weight for a sample or sample/task pair to 0 causes it to be ignored during fitting and evaluation. It will have no effect on the loss function or other metrics.

Most of the other weights are close to 1, but not exactly 1. This is done to balance the overall weight of positive and negative samples on each task. When training the model, we want each of the 12 tasks to contribute equally, and on each task we want to put equal weight on positive and negative samples. Otherwise, the model might just learn that most of the training samples are non-toxic, and therefore become biased toward identifying other molecules as non-toxic.

Training a Model on Fingerprints

Let's train a model. In earlier tutorials we use `GraphConvModel`, which is a fairly complicated architecture that takes a complex set of inputs. Because fingerprints are so simple, just a single fixed length array, we can use a much simpler type of model.

In [4]:
```
model = dc.models.MultitaskClassifier(n_tasks=12, n_features=1024, layer_sizes=[1000])
```

`MultitaskClassifier` is a simple stack of fully connected layers. In this example we tell it to use a single hidden layer of width 1000. We also tell it that each input will have 1024 features, and that it should produce predictions for 12 different tasks.

Why not train a separate model for each task? We could do that, but it turns out that training a single model for multiple tasks often works better. We will see an example of that in a later tutorial.

Let's train and evaluate the model.

In [5]:
```
import numpy as np

model.fit(train_dataset, nb_epoch=10)
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
print('training set score:', model.evaluate(train_dataset, [metric], transformers))
print('test set score:', model.evaluate(test_dataset, [metric], transformers))
```

```
training set score: {'roc_auc_score': 0.9550063590563469}
test set score: {'roc_auc_score': 0.7781819573695475}
```

Not bad performance for such a simple model and featurization. More sophisticated models do slightly better on this dataset, but not enormously better.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro4,
 title={Molecular Fingerprints},
 organization={DeepChem},
 author={Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Molecular_Fingerprints
 year={2021},
}
```

Creating Models with TensorFlow and PyTorch

In the tutorials so far, we have used standard models provided by DeepChem. This is fine for many applications, but sooner or later you will want to create an entirely new model with an architecture you define yourself. DeepChem provides integration with both TensorFlow (Keras) and PyTorch, so you can use it with models from either of these frameworks.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
```

There are actually two different approaches you can take to using TensorFlow or PyTorch models with DeepChem. It depends on whether you want to use TensorFlow/PyTorch APIs or DeepChem APIs for training and evaluating your model. For the former case, DeepChem's `Dataset` class has methods for easily adapting it to use with other frameworks. `make_tf_dataset()` returns a `tensorflow.data.Dataset` object that iterates over the data. `make_pytorch_dataset()` returns a `torch.utils.data.IterableDataset` that iterates over the data. This lets you use DeepChem's datasets, loaders, featurizers, transformers, splitters, etc. and easily integrate them into your existing TensorFlow or PyTorch code.

But DeepChem also provides many other useful features. The other approach, which lets you use those features, is to wrap your model in a DeepChem `Model` object. Let's look at how to do that.

KerasModel

`KerasModel` is a subclass of DeepChem's `Model` class. It acts as a wrapper around a `tensorflow.keras.Model`. Let's see an example of using it. For this example, we create a simple sequential model consisting of two dense layers.

In [ ]:
```
import deepchem as dc
import tensorflow as tf

keras_model = tf.keras.Sequential([
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dropout(rate=0.5),
    tf.keras.layers.Dense(1)
])
model = dc.models.KerasModel(keras_model, dc.models.losses.L2Loss())
```

For this example, we used the Keras `Sequential` class. Our model consists of a dense layer with ReLU activation, 50% dropout to provide regularization, and a final layer that produces a scalar output. We also need to specify the loss function to use when training the model, in this case L_2 loss. We can now train and evaluate the model exactly as we would with any other DeepChem model. For example, let's load the Delaney solubility dataset. How does our model do at predicting the solubilities of molecules based on their extended-connectivity fingerprints (ECFPs)?

In [ ]:
```
tasks, datasets, transformers = dc.molnet.load_delaney(featurizer='ECFP', splitter='random')
train_dataset, valid_dataset, test_dataset = datasets
model.fit(train_dataset, nb_epoch=50)
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)
print('training set score:', model.evaluate(train_dataset, [metric]))
print('test set score:', model.evaluate(test_dataset, [metric]))
```

```
training set score: {'pearson_r2_score': 0.9766804253639305}
test set score: {'pearson_r2_score': 0.7048814451615332}
```

TorchModel

`TorchModel` works just like `KerasModel`, except it wraps a `torch.nn.Module`. Let's use PyTorch to create another model just like the previous one and train it on the same data.

In [ ]:
```
import torch

pytorch_model = torch.nn.Sequential(
    torch.nn.Linear(1024, 1000),
    torch.nn.ReLU(),
```

```
    torch.nn.Dropout(0.5),
    torch.nn.Linear(1000, 1)
)
model = dc.models.TorchModel(pytorch_model, dc.models.losses.L2Loss())

model.fit(train_dataset, nb_epoch=50)
print('training set score:', model.evaluate(train_dataset, [metric]))
print('test set score:', model.evaluate(test_dataset, [metric]))
```

```
training set score: {'pearson_r2_score': 0.9760781898204121}
test set score: {'pearson_r2_score': 0.6981331812360332}
```

Computing Losses

Now let's see a more advanced example. In the above models, the loss was computed directly from the model's output. Often that is fine, but not always. Consider a classification model that outputs a probability distribution. While it is possible to compute the loss from the probabilities, it is more numerically stable to compute it from the logits.

To do this, we create a model that returns multiple outputs, both probabilities and logits. `KerasModel` and `TorchModel` let you specify a list of "output types". If a particular output has type `'prediction'`, that means it is a normal output that should be returned when you call `predict()`. If it has type `'loss'`, that means it should be passed to the loss function in place of the normal outputs.

Sequential models do not allow multiple outputs, so instead we use a subclassing style model.

In [ ]:
```
class ClassificationModel(tf.keras.Model):

    def __init__(self):
        super(ClassificationModel, self).__init__()
        self.dense1 = tf.keras.layers.Dense(1000, activation='relu')
        self.dense2 = tf.keras.layers.Dense(1)

    def call(self, inputs, training=False):
        y = self.dense1(inputs)
        if training:
            y = tf.nn.dropout(y, 0.5)
        logits = self.dense2(y)
        output = tf.nn.sigmoid(logits)
        return output, logits

keras_model = ClassificationModel()
output_types = ['prediction', 'loss']
model = dc.models.KerasModel(keras_model, dc.models.losses.SigmoidCrossEntropy(), output_types=output_types)
```

We can train our model on the BACE dataset. This is a binary classification task that tries to predict whether a molecule will inhibit the enzyme BACE-1.

In [ ]:
```
tasks, datasets, transformers = dc.molnet.load_bace_classification(feturizer='ECFP', splitter='scaffold')
train_dataset, valid_dataset, test_dataset = datasets
model.fit(train_dataset, nb_epoch=100)
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
print('training set score:', model.evaluate(train_dataset, [metric]))
print('test set score:', model.evaluate(test_dataset, [metric]))
```

```
training set score: {'roc_auc_score': 0.9995809177900399}
test set score: {'roc_auc_score': 0.7629528985507246}
```

Similarly, we will create a custom Classifier Model class to be used with `TorchModel`. Using similar reasoning to the above `KerasModel`, a custom model allows for easy capturing of the unscaled output (logits in Tensorflow) of the second dense layer. The custom class allows definition of how forward pass is done; enabling capture of the logits right before the final sigmoid is applied to produce the prediction.

Finally, an instance of `ClassificationModel` is coupled with a loss function that requires both the prediction and logits to produce an instance of `TorchModel` to train.

In [ ]:
```
class ClassificationModel(torch.nn.Module):

    def __init__(self):
        super(ClassificationModel, self).__init__()
        self.dense1 = torch.nn.Linear(1024, 1000)
        self.dense2 = torch.nn.Linear(1000, 1)

    def forward(self, inputs):
        y = torch.nn.functional.relu( self.dense1(inputs) )
        y = torch.nn.functional.dropout(y, p=0.5, training=self.training)
        logits = self.dense2(y)
        output = torch.sigmoid(logits)
        return output, logits
```

```
torch_model = ClassificationModel()
output_types = ['prediction', 'loss']
model = dc.models.TorchModel(torch_model, dc.models.losses.SigmoidCrossEntropy(), output_types=output_types)
```

We will use the same BACE dataset. As before, the model will try to do a binary classification task that tries to predict whether a molecule will inhibit the enzyme BACE-1.

In [ ]:
```
tasks, datasets, transformers = dc.molnet.load_bace_classification(feturizer='ECFP', splitter='scaffold')
train_dataset, valid_dataset, test_dataset = datasets
model.fit(train_dataset, nb_epoch=100)
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
print('training set score:', model.evaluate(train_dataset, [metric]))
print('test set score:', model.evaluate(test_dataset, [metric]))
```

```
training set score: {'roc_auc_score': 0.9996340015366347}
test set score: {'roc_auc_score': 0.7615036231884058}
```

Other Features

`KerasModel` and `TorchModel` have lots of other features. Here are some of the more important ones.

- Automatically saving checkpoints during training.
- Logging progress to the console, to TensorBoard, or to Weights & Biases.
- Custom loss functions that you define with a function of the form `f(outputs, labels, weights)`.
- Early stopping using the `ValidationCallback` class.
- Loading parameters from pre-trained models.
- Estimating uncertainty in model outputs.
- Identifying important features through saliency mapping.

By wrapping your own models in a `KerasModel` or `TorchModel`, you get immediate access to all these features. See the API documentation for full details on them.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

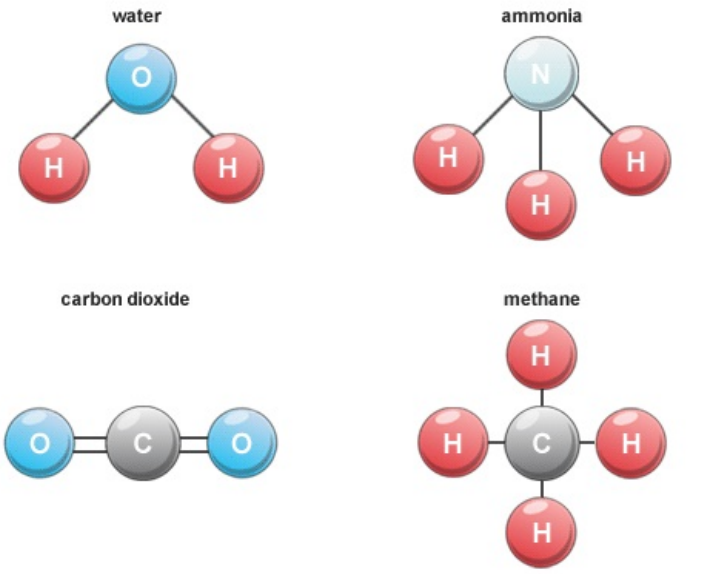
Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro1,
 title={5},
 organization={DeepChem},
 author={Ramsundar, Bharath and Rebel, Alles},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Creating_Models_with_
 year={2021},
}
```

Introduction to Graph Convolutions

In this tutorial we will learn more about "graph convolutions." These are one of the most powerful deep learning tools for working with molecular data. The reason for this is that molecules can be naturally viewed as graphs.



Note how standard chemical diagrams of the sort we're used to from high school lend themselves naturally to visualizing molecules as graphs. In the remainder of this tutorial, we'll dig into this relationship in significantly more detail. This will let us get a deeper understanding of how these systems work.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

Open in Colab

In [3]:
```
!pip install --pre deepchem
```

What are Graph Convolutions?

Consider a standard convolutional neural network (CNN) of the sort commonly used to process images. The input is a grid of pixels. There is a vector of data values for each pixel, for example the red, green, and blue color channels. The data passes through a series of convolutional layers. Each layer combines the data from a pixel and its neighbors to produce a new data vector for the pixel. Early layers detect small scale local patterns, while later layers detect larger, more abstract patterns. Often the convolutional layers alternate with pooling layers that perform some operation such as max or min over local regions.

Graph convolutions are similar, but they operate on a graph. They begin with a data vector for each node of the graph (for example, the chemical properties of the atom that node represents). Convolutional and pooling layers combine information from connected nodes (for example, atoms that are bonded to each other) to produce a new data vector for each node.

Training a GraphConvModel

Let's use the MoleculeNet suite to load the Tox21 dataset. To featurize the data in a way that graph convolutional networks can use, we set the featurizer option to `'GraphConv'`. The MoleculeNet call returns a training set, a validation set, and a test set for us to use. It also returns `tasks`, a list of the task names, and `transformers`, a list of data transformations that were applied to preprocess the dataset. (Most deep networks are quite finicky and require a set of data transformations to ensure that training proceeds stably.)

Note: While importing deepchem, if you see any warnings, ignore them for now. Deepchem is a vast library and there are many things that can cause minor warnings to occur. Almost always, it doesn't require any action from your side.

In [4]:
```
import deepchem as dc

tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='GraphConv')
train_dataset, valid_dataset, test_dataset = datasets
```

Let's now train a graph convolutional network on this dataset. DeepChem has the class `GraphConvModel` that wraps a standard graph convolutional architecture underneath the hood for user convenience. Let's instantiate an object of this class and train it on our dataset.

In [5]:
```
n_tasks = len(tasks)
num_features = train_dataset.X[0].get_atom_features().shape[1]
model = dc.models.torch_models.GraphConvModel(n_tasks, mode='classification',number_input_features=[num_features
model.fit(train_dataset, nb_epoch=50)
```

Out[5]:
```
0.29102970123291017
```

Let's try to evaluate the performance of the model we've trained. For this, we need to define a metric, a measure of model performance. `dc.metrics` holds a collection of metrics already. For this dataset, it is standard to use the ROC-AUC score, the area under the receiver operating characteristic curve (which measures the tradeoff between precision and recall). Luckily, the ROC-AUC score is already available in DeepChem.

To measure the performance of the model under this metric, we can use the convenience function `model.evaluate()`.

In [6]:
```
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
print('Training set score:', model.evaluate(train_dataset, [metric], transformers))
print('Test set score:', model.evaluate(test_dataset, [metric], transformers))
```

```
Training set score: {'roc_auc_score': 0.970785822904073}
Test set score: {'roc_auc_score': 0.7112009940440461}
```

The results are pretty good, and `GraphConvModel` is very easy to use. But what's going on under the hood? Could we build GraphConvModel ourselves? Of course! DeepChem provides Keras layers for all the calculations involved in a graph convolution. We are going to apply the following layers from DeepChem.

- `GraphConv` layer: This layer implements the graph convolution. The graph convolution combines per-node feature vectures in a nonlinear fashion with the feature vectors for neighboring nodes. This "blends" information in local neighborhoods of a graph.
- `GraphPool` layer: This layer does a max-pooling over the feature vectors of atoms in a neighborhood. You can think of this layer as analogous to a max-pooling layer for 2D convolutions but which operates on graphs instead.
- `GraphGather`: Many graph convolutional networks manipulate feature vectors per graph-node. For a molecule for example, each node might represent an atom, and the network would manipulate atomic feature vectors that summarize the local chemistry of the atom. However, at the end of the application, we will likely want to work with a molecule level feature representation. This layer creates a graph level feature vector by combining all the node-level feature vectors.

Apart from this we are going to apply standard neural network layers such as Dense, BatchNormalization and Softmax layer.

Training a custom Graph Convolution network

As you may have seen in the previous tutorials, DeepChem offers both PyTorch and Tensorflow functionalities. However, most of our work moving forward will leverage the PyTorch ecosystem.

Let's look at the Tensorflow implementation first.

Tensorflow

In [7]:
```
from deepchem.models.layers import GraphConv, GraphPool, GraphGather
import tensorflow as tf
import tensorflow.keras.layers as layers

batch_size = 100

class GraphConvModelTensorflow(tf.keras.Model):

  def __init__(self):
    super(GraphConvModelTensorflow, self).__init__()
    self.gc1 = GraphConv(128, activation_fn=tf.nn.tanh)
    self.batch_norm1 = layers.BatchNormalization()
    self.gp1 = GraphPool()
```

```
    self.gc2 = GraphConv(128, activation_fn=tf.nn.tanh)
    self.batch_norm2 = layers.BatchNormalization()
    self.gp2 = GraphPool()

    self.dense1 = layers.Dense(256, activation=tf.nn.tanh)
    self.batch_norm3 = layers.BatchNormalization()
    self.readout = GraphGather(batch_size=batch_size, activation_fn=tf.nn.tanh)

    self.dense2 = layers.Dense(n_tasks*2)
    self.logits = layers.Reshape((n_tasks, 2))
    self.softmax = layers.Softmax()

  def call(self, inputs):
    gc1_output = self.gc1(inputs)
    batch_norm1_output = self.batch_norm1(gc1_output)
    gp1_output = self.gp1([batch_norm1_output] + inputs[1:])

    gc2_output = self.gc2([gp1_output] + inputs[1:])
    batch_norm2_output = self.batch_norm1(gc2_output)
    gp2_output = self.gp2([batch_norm2_output] + inputs[1:])

    dense1_output = self.dense1(gp2_output)
    batch_norm3_output = self.batch_norm3(dense1_output)
    readout_output = self.readout([batch_norm3_output] + inputs[1:])

    logits_output = self.logits(self.dense2(readout_output))
    return self.softmax(logits_output)
```

We can now see more clearly what is happening. There are two convolutional blocks, each consisting of a `GraphConv`, followed by batch normalization, followed by a `GraphPool` to do max pooling. We finish up with a dense layer, another batch normalization, a `GraphGather` to combine the data from all the different nodes, and a final dense layer to produce the global output.

Let's now create the DeepChem model which will be a wrapper around the Keras model that we just created. We will also specify the loss function so the model knows the objective to minimize.

In [8]:
```
model = dc.models.KerasModel(GraphConvModelTensorflow(), loss=dc.models.losses.CategoricalCrossEntropy())
```

What are the inputs to this model? A graph convolution requires a complete description of each molecule, including the list of nodes (atoms) and a description of which ones are bonded to each other. In fact, if we inspect the dataset we see that the feature array contains Python objects of type `ConvMol`.

In [9]:
```
test_dataset.X[0]
```

Out[9]:
```
<deepchem.feat.mol_graphs.ConvMol at 0x7bf66bfa1160>
```

Models expect arrays of numbers as their inputs, not Python objects. We must convert the `ConvMol` objects into the particular set of arrays expected by the `GraphConv`, `GraphPool`, and `GraphGather` layers. Fortunately, the `ConvMol` class includes the code to do this, as well as to combine all the molecules in a batch to create a single set of arrays.

The following code creates a Python generator that given a batch of data generates the lists of inputs, labels, and weights whose values are Numpy arrays. `atom_features` holds a feature vector of length 75 for each atom. The other inputs are required to support minibatching in TensorFlow. `degree_slice` is an indexing convenience that makes it easy to locate atoms from all molecules with a given degree. `membership` determines the membership of atoms in molecules (atom `i` belongs to molecule `membership[i]`). `deg_adjs` is a list that contains adjacency lists grouped by atom degree. For more details, check out the code.

In [10]:
```
from deepchem.metrics import to_one_hot
from deepchem.feat.mol_graphs import ConvMol
import numpy as np

def data_generator(dataset, epochs=1):
  for ind, (X_b, y_b, w_b, ids_b) in enumerate(dataset.iterbatches(batch_size, epochs,
                                                                   deterministic=False, pad_batches=True)):
    multiConvMol = ConvMol.agglomerate_mols(X_b)
    inputs = [multiConvMol.get_atom_features(), multiConvMol.deg_slice, np.array(multiConvMol.membership)]
    for i in range(1, len(multiConvMol.get_deg_adjacency_lists())):
      inputs.append(multiConvMol.get_deg_adjacency_lists()[i])
    labels = [to_one_hot(y_b.flatten(), 2).reshape(-1, n_tasks, 2)]
    weights = [w_b]
    yield (inputs, labels, weights)
```

Now, we can train the model using `fit_generator(generator)` which will use the generator we've defined to train the model.

In [11]:
```
model.fit_generator(data_generator(train_dataset, epochs=50))
```

Out[11]:
```
0.23354644775390626
```

Now that we have trained our graph convolutional method, let's evaluate its performance. We again have to use our defined generator to evaluate model performance.

In [12]:
```
print('Training set score:', model.evaluate_generator(data_generator(train_dataset), [metric], transformers))
print('Test set score:', model.evaluate_generator(data_generator(test_dataset), [metric], transformers))
```

```
Training set score: {'roc_auc_score': 0.8370577643901682}
Test set score: {'roc_auc_score': 0.6610993488016647}
```

PyTorch

Before working on the PyTorch implementation, we must import a few crucial layers from the `torch_models` collection. These are PyTorch implementations of `GraphConv`, `GraphPool` and `GraphGather` which we used in the tensorflow's implementation as well.

In [13]:
```
import torch
import torch.nn as nn
from deepchem.models.torch_models.layers import GraphConv, GraphGather, GraphPool
```

PyTorch's `GraphConv` requires the number of input features to be specified, hence we can extract that piece of information by following steps:

1. First we get a sample from the dataset.
2. Next we slice and separate the node_features (which is the first element of the list, hence the index 0).
3. Finally, we obtain the number of features by finding the shape of the array.

In [14]:
```
sample_batch = next(data_generator(train_dataset))
node_features = sample_batch[0][0]
num_input_features = node_features.shape[1]
print(f"Number of input features: {num_input_features}")
```

```
Number of input features: 75
```

In [17]:
```
class GraphConvModelTorch(nn.Module):
    def __init__(self):
        super(GraphConvModelTorch, self).__init__()

        self.gc1 = GraphConv(out_channel=128, number_input_features=num_input_features, activation_fn=nn.Tanh()
        self.batch_norm1 = nn.BatchNorm1d(128)
        self.gp1 = GraphPool()

        self.gc2 = GraphConv(out_channel=128, number_input_features=128, activation_fn=nn.Tanh())
        self.batch_norm2 = nn.BatchNorm1d(128)
        self.gp2 = GraphPool()

        self.dense1 = nn.Linear(128, 256)
        self.act3 = nn.Tanh()
        self.batch_norm3 = nn.BatchNorm1d(256)
        self.readout = GraphGather(batch_size=batch_size, activation_fn=nn.Tanh())

        self.dense2 = nn.Linear(512, n_tasks * 2)

        self.logits = lambda data: data.view(-1, n_tasks, 2)
        self.softmax = nn.Softmax(dim=-1)

    def forward(self, inputs):
        gc1_output = self.gc1(inputs)
        batch_norm1_output = self.batch_norm1(gc1_output)
        gp1_output = self.gp1([batch_norm1_output] + inputs[1:])

        gc2_output = self.gc2([gp1_output] + inputs[1:])
        batch_norm2_output = self.batch_norm2(gc2_output)
        gp2_output = self.gp2([batch_norm2_output] + inputs[1:])

        dense1_output = self.act3(self.dense1(gp2_output))
        batch_norm3_output = self.batch_norm3(dense1_output)
        readout_output = self.readout([batch_norm3_output] + inputs[1:])

        dense2_output = self.dense2(readout_output)
        logits_output = self.logits(dense2_output)
        softmax_output = self.softmax(logits_output)
        return softmax_output
```

In [18]:
```
model = dc.models.TorchModel(GraphConvModelTorch(), loss=dc.models.losses.CategoricalCrossEntropy())
model.fit_generator(data_generator(train_dataset, epochs=50))
```

Out[18]: 0.2121513557434082

In [19]:
```
print('Training set score:', model.evaluate_generator(data_generator(train_dataset), [metric], transformers))
print('Test set score:', model.evaluate_generator(data_generator(test_dataset), [metric], transformers))
```

```
Training set score: {'roc_auc_score': 0.9838238607233897}
Test set score: {'roc_auc_score': 0.6923516284964811}
```

Success! Both the models we've constructed behave nearly identically to `GraphConvModel`. If you're looking to build your own custom models, you can follow the examples we've provided here to do so. We hope to see exciting constructions from your end soon!

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Going Deeper On Molecular Featurizations

One of the most important steps of doing machine learning on molecular data is transforming the data into a form amenable to the application of learning algorithms. This process is broadly called "featurization" and involves turning a molecule into a vector or tensor of some sort. There are a number of different ways of doing that, and the choice of featurization is often dependent on the problem at hand. We have already seen two such methods: molecular fingerprints, and `ConvMol` objects for use with graph convolutions. In this tutorial we will look at some of the others.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [1]:
```
!pip install -qq --pre deepchem
import deepchem
import warnings
warnings.filterwarnings('ignore')
deepchem.__version__
```

Out[1]:
```
'2.6.0.dev'
```

Featurizers

In DeepChem, a method of featurizing a molecule (or any other sort of input) is defined by a `Featurizer` object. There are three different ways of using featurizers.

1. When using the MoleculeNet loader functions, you simply pass the name of the featurization method to use. We have seen examples of this in earlier tutorials, such as `featurizer='ECFP'` or `featurizer='GraphConv'`.

2. You also can create a Featurizer and directly apply it to molecules. For example:

In [2]:
```
import deepchem as dc

featurizer = dc.feat.CircularFingerprint()
print(featurizer(['CC', 'CCC', 'CCO']))
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
[09:05:59] DEPRECATION WARNING: please use MorganGenerator
[09:05:59] DEPRECATION WARNING: please use MorganGenerator
[09:05:59] DEPRECATION WARNING: please use MorganGenerator
```

3. When creating a new dataset with the DataLoader framework, you can specify a Featurizer to use for processing the data. We will see this in a future tutorial.

We use propane ($CH_3CH_2CH_3$, represented by the SMILES string `'CCC'`) as a running example throughout this tutorial. Many of the featurization methods use conformers of the molecules. A conformer can be generated using the `ConformerGenerator` class in `deepchem.utils.conformers`.

RDKitDescriptors

`RDKitDescriptors` featurizes a molecule by using RDKit to compute values for a list of descriptors. These are basic physical and chemical properties: molecular weight, polar surface area, numbers of hydrogen bond donors and acceptors, etc. This is most useful for predicting things that depend on these high level properties rather than on detailed molecular structure.

Intrinsic to the featurizer is a set of allowed descriptors, which can be accessed using `RDKitDescriptors.allowedDescriptors`. The featurizer uses the descriptors in `rdkit.Chem.Descriptors.descList`, checks if they are in the list of allowed descriptors, and computes the descriptor value for the molecule.

Let's print the values of the first ten descriptors for propane.

In [3]:
```
from rdkit.Chem.Descriptors import descList
```

```
rdkit_featurizer = dc.feat.RDKitDescriptors()
features = rdkit_featurizer(['CCC'])[0]
descriptors = [i[0] for i in descList]
for feature, descriptor in zip(features[:10], descriptors):
    print(descriptor, feature)
```

```
MaxAbsEStateIndex 2.125
MaxEStateIndex 2.125
MinAbsEStateIndex 1.25
MinEStateIndex 1.25
qed 0.3854706587740357
SPS 6.0
MolWt 44.097
HeavyAtomMolWt 36.033
ExactMolWt 44.062600255999996
NumValenceElectrons 20.0
[09:07:17] DEPRECATION WARNING: please use MorganGenerator
[09:07:17] DEPRECATION WARNING: please use MorganGenerator
[09:07:17] DEPRECATION WARNING: please use MorganGenerator
```

Of course, there are many more descriptors than this.

In [4]:
```
print('The number of descriptors present is: ', len(features))
```

```
The number of descriptors present is:  210
```

WeaveFeaturizer and MolGraphConvFeaturizer

We previously looked at graph convolutions, which use `ConvMolFeaturizer` to convert molecules into `ConvMol` objects. Graph convolutions are a special case of a large class of architectures that represent molecules as graphs. They work in similar ways but vary in the details. For example, they may associate data vectors with the atoms, the bonds connecting them, or both. They may use a variety of techniques to calculate new data vectors from those in the previous layer, and a variety of techniques to compute molecule level properties at the end.

DeepChem supports lots of different graph based models. Some of them require molecules to be featurized in slightly different ways. Because of this, there are two other featurizers called `WeaveFeaturizer` and `MolGraphConvFeaturizer`. They each convert molecules into a different type of Python object that is used by particular models. When using any graph based model, just check the documentation to see what featurizer you need to use with it.

CoulombMatrix

All the models we have looked at so far consider only the intrinsic properties of a molecule: the list of atoms that compose it and the bonds connecting them. When working with flexible molecules, you may also want to consider the different conformations the molecule can take on. For example, when a drug molecule binds to a protein, the strength of the binding depends on specific interactions between pairs of atoms. To predict binding strength, you probably want to consider a variety of possible conformations and use a model that takes them into account when making predictions.

The Coulomb matrix is one popular featurization for molecular conformations. Recall that the electrostatic Coulomb interaction between two charges is proportional to $q_1 q_2 / r$ where q_1 and q_2 are the charges and r is the distance between them. For a molecule with N atoms, the Coulomb matrix is a $N \times N$ matrix where each element gives the strength of the electrostatic interaction between two atoms. It contains information both about the charges on the atoms and the distances between them. More information on the functional forms used can be found here.

To apply this featurizer, we first need a set of conformations for the molecule. We can use the `ConformerGenerator` class to do this. It takes a RDKit molecule, generates a set of energy minimized conformers, and prunes the set to only include ones that are significantly different from each other. Let's try running it for propane.

In [5]:
```
from rdkit import Chem

generator = dc.utils.ConformerGenerator(max_conformers=5)
propane_mol = generator.generate_conformers(Chem.MolFromSmiles('CCC'))
print("Number of available conformers for propane: ", len(propane_mol.GetConformers()))
```

```
Number of available conformers for propane:  1
```

It only found a single conformer. This shouldn't be surprising, since propane is a very small molecule with hardly any flexibility. Let's try adding another carbon.

In [6]:
```
butane_mol = generator.generate_conformers(Chem.MolFromSmiles('CCCC'))
print("Number of available conformers for butane: ", len(butane_mol.GetConformers()))
```

```
Number of available conformers for butane:  3
```

Now we can create a Coulomb matrix for our molecule.

In [7]:
```
coulomb_mat = dc.feat.CoulombMatrix(max_atoms=20)
features = coulomb_mat(propane_mol)
print(features)
```

```
[[[36.8581052  12.48684429  7.5619687   2.85945193  2.85804514
    2.85804556  1.4674015   1.46740144  0.91279491  1.14239698
    1.14239675  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [12.48684429 36.8581052  12.48684388  1.46551218  1.45850736
    1.45850732  2.85689525  2.85689538  1.4655122   1.4585072
    1.4585072   0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 7.5619687  12.48684388 36.8581052   0.9127949   1.14239695
    1.14239692  1.46740146  1.46740145  2.85945178  2.85804504
    2.85804493  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 2.85945193  1.46551218  0.9127949   0.5         0.29325367
    0.29325369  0.21256978  0.21256978  0.12268391  0.13960187
    0.13960185  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 2.85804514  1.45850736  1.14239695  0.29325367  0.5
    0.29200271  0.17113413  0.21092513  0.13960186  0.1680002
    0.20540029  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 2.85804556  1.45850732  1.14239692  0.29325369  0.29200271
    0.5         0.21092513  0.17113413  0.13960187  0.20540032
    0.16800016  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.4674015   2.85689525  1.46740146  0.21256978  0.17113413
    0.21092513  0.5         0.29351308  0.21256981  0.2109251
    0.17113412  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.46740144  2.85689538  1.46740145  0.21256978  0.21092513
    0.17113413  0.29351308  0.5         0.21256977  0.17113412
    0.21092513  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.91279491  1.4655122   2.85945178  0.12268391  0.13960186
    0.13960187  0.21256981  0.21256977  0.5         0.29325366
    0.29325365  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.14239698  1.4585072   2.85804504  0.13960187  0.1680002
    0.20540032  0.2109251   0.17113412  0.29325366  0.5
    0.29200266  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.14239675  1.4585072   2.85804493  0.13960185  0.20540029
    0.16800016  0.17113412  0.21092513  0.29325365  0.29200266
    0.5         0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]]]
```

Notice that many elements are 0. To combine multiple molecules in a batch we need all the Coulomb matrices to be the same size, even if the molecules have different numbers of atoms. We specified `max_atoms=20`, so the returned matrix

has size (20, 20). The molecule only has 11 atoms, so only an 11 by 11 submatrix is nonzero.

CoulombMatrixEig

An important feature of Coulomb matrices is that they are invariant to molecular rotation and translation, since the interatomic distances and atomic numbers do not change. Respecting symmetries like this makes learning easier. Rotating a molecule does not change its physical properties. If the featurization does change, then the model is forced to learn that rotations are not important, but if the featurization is invariant then the model gets this property automatically.

Coulomb matrices are not invariant under another important symmetry: permutations of the atoms' indices. A molecule's physical properties do not depend on which atom we call "atom 1", but the Coulomb matrix does. To deal with this, the `CoulombMatrixEig` featurizer was introduced, which uses the eigenvalue spectrum of the Coulumb matrix and is invariant to random permutations of the atom's indices. The disadvantage of this featurization is that it contains much less information (N eigenvalues instead of an $N \times N$ matrix), so models will be more limited in what they can learn.

`CoulombMatrixEig` inherits from `CoulombMatrix` and featurizes a molecule by first computing the Coulomb matrices for different conformers of the molecule and then computing the eigenvalues for each Coulomb matrix. These eigenvalues are then padded to account for variation in number of atoms across molecules.

In [8]:
```
coulomb_mat_eig = dc.feat.CoulombMatrixEig(max_atoms=20)
features = coulomb_mat_eig(propane_mol)
print(features)
```

```
[[60.07620303 29.62963149 22.75497781  0.5713786   0.28781332  0.28548338
  0.27558187  0.18163794  0.17460999  0.17059719  0.16640098  0.
  0.          0.          0.          0.          0.          0.
  0.          0.        ]]
```

SMILES Tokenization and Numericalization

So far, we have looked at featurization techniques that translate the implicit structural and physical information in SMILES data into more explicit features that help our models learn and make predictions. In this section, we will preprocess SMILES strings into a format that can be fed to sequence models, such as 1D convolutional neural networks and transformers, and enables these models to learn their own representations of molecular properties.

To prepare SMILES strings for a sequence model, we break them down into lists of substrings (called tokens) and turn them into lists of integer values (numericalization). Sequence models use those integer values as indices of an embedding matrix, which contains a vector of floating-point numbers for each token in the vocabulary. These embedding vectors are updated during model training. This process allows the sequence model to learn its own representations of the molecular properties implicit in the training data.

We will use DeepChem's `BasicSmilesTokenizer` and the Tox21 dataset from MoleculeNet to demonstrate the process of tokenizing SMILES.

In [9]:
```
import numpy as np
```

In [10]:
```
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer="Raw")
train_dataset, valid_dataset, test_dataset = datasets
print(train_dataset)
```

```
<DiskDataset X.shape: (6264,), y.shape: (6264, 12), w.shape: (6264, 12), task_names: ['NR-AR' 'NR-AR-LBD' 'NR-AhR' ... 'SR-HSE' 'SR-MMP' 'SR-p53']>
```

We loaded the datasets with `featurizer="Raw"`. Now we obtain the SMILES from their `ids` attributes.

In [11]:
```
train_smiles = train_dataset.ids
valid_smiles = valid_dataset.ids
test_smiles = test_dataset.ids
print(train_smiles[:5])
```

```
['CC(O)(P(=O)(O)O)P(=O)(O)O' 'CC(C)(C)OOC(C)(C)CCC(C)(C)OOC(C)(C)C'
 'OC[C@H](O)[C@@H](O)[C@H](O)CO'
 'CCCCCCCC(=O)[O-].CCCCCCCC(=O)[O-].[Zn+2]' 'CC(C)COC(=O)C(C)C']
```

Next we define our tokenizer and `map` it onto all our data to convert the SMILES strings into lists of tokens. The `BasicSmilesTokenizer` breaks down SMILES roughly at atom level.

In [12]:
```
tokenizer = dc.feat.smiles_tokenizer.BasicSmilesTokenizer()
```

```
train_tok = list(map(tokenizer.tokenize, train_smiles))
valid_tok = list(map(tokenizer.tokenize, valid_smiles))
test_tok = list(map(tokenizer.tokenize, test_smiles))
print(train_tok[0])
len(train_tok)
```

```
['C', 'C', '(', 'O', ')', '(', 'P', '(', '=', 'O', ')', '(', 'O', ')', 'O', ')', 'P', '(', '=', 'O', ')', '(', '
O', ')', 'O']
```

Out[12]: 6264

Now we have tokenized versions of all SMILES strings in our dataset. To convert those into lists of integer values we first need to create a list of all possible tokens in our dataset. That list is called the vocabulary. We also add the empty string `""` to our vocabulary in order to correctly handle trailing zeros when decoding zero-padded numericalized SMILES.

In [13]:
```
flatten = lambda l: [item for items in l for item in items]

all_toks = flatten(train_tok) + flatten(valid_tok) + flatten(test_tok)
vocab = sorted(set(all_toks + [""]))
print(vocab[:12], "...", vocab[-12:])
len(vocab)
```

```
['', '#', '(', ')', '-', '.', '/', '1', '2', '3', '4', '5'] ... ['[n+]', '[n-]', '[nH+]', '[nH]', '[o+]', '[s+]'
, '[se]', '\\', 'c', 'n', 'o', 's']
```

Out[13]: 128

To numericalize tokenized SMILES strings we create a `str2int` dictionary which assigns a number to each token in the dictionary. We also create the reverse `int2str` dictionary and define the corresponding `encode` and `decode` functions. Finally we `map` the `encode` function on the tokenized data to obtain numericalized SMILES data.

In [14]:
```
str2int = {s:i for i, s in enumerate(vocab)}
int2str = {i:s for i, s in enumerate(vocab)}
print(f"str2int: {dict(list(str2int.items())[:5])} ...")
print(f"int2str: {dict(list(int2str.items())[:5])} ...")
```

```
str2int: {'': 0, '#': 1, '(': 2, ')': 3, '-': 4} ...
int2str: {0: '', 1: '#', 2: '(', 3: ')', 4: '-'} ...
```

In [15]:
```
encode = lambda s: [str2int[tok] for tok in s]
decode = lambda i: [int2str[num] for num in i]
print(train_smiles[0])
print(encode(train_tok[0]))
print("".join(decode(encode(train_tok[0]))))
```

```
CC(O)(P(=O)(O)O)P(=O)(O)O
[19, 19, 2, 24, 3, 2, 25, 2, 16, 24, 3, 2, 24, 3, 24, 3, 25, 2, 16, 24, 3, 2, 24, 3, 24]
CC(O)(P(=O)(O)O)P(=O)(O)O
```

In [16]:
```
train_num = list(map(encode, train_tok))
valid_num = list(map(encode, valid_tok))
test_num = list(map(encode, test_tok))
print(train_num[0])
```

```
[19, 19, 2, 24, 3, 2, 25, 2, 16, 24, 3, 2, 24, 3, 24, 3, 25, 2, 16, 24, 3, 2, 24, 3, 24]
```

Lastly, we would like to combine all molecules in a dataset in an `np.array` so they can be served to a model in batches. To achieve that, all sequences have to be of the same length. As in the CoulombMatrix section, we achieve that by appending zeros up to a fixed value.

In [17]:
```
max_len = max(map(len, train_num + valid_num + test_num))
max_len
```

Out[17]: 240

The longest sequence across all Tox21 datasets has length `240`, so we use that as our fixed length. We create a `zero_pad` function, `map` it to all numericalized SMILES, and turn them into `np.array`s.

In [18]:
```
zero_pad = lambda x: x + [0] * (max_len - len(x))

train_numpad = np.array(list(map(zero_pad, train_num)))
valid_numpad = np.array(list(map(zero_pad, valid_num)))
test_numpad = np.array(list(map(zero_pad, test_num)))
train_numpad
```

Out[18]:
```
array([[19, 19,  2, ...,  0,  0,  0],
       [19, 19,  2, ...,  0,  0,  0],
       [24, 19, 42, ...,  0,  0,  0],
       ...,
       [24, 16, 19, ...,  0,  0,  0],
       [19, 19,  2, ...,  0,  0,  0],
       [19, 19,  2, ...,  0,  0,  0]])
```

We can check that the zero-padded data still converts back to the correct SMILES string using the `decode` function on a random datapoint.

In [19]:
```
idx = np.random.randint(0, train_numpad.shape[0], 1).item()
print(train_smiles[idx])
print("".join(decode(train_numpad[idx])))
```

```
Cc1cc(C(C)(C)c2ccc(O)c(C)c2)ccc1O
Cc1cc(C(C)(C)c2ccc(O)c(C)c2)ccc1O
```

The padded data passes the test. It is now in the correct format to be used for training of a sequence model, but it doesn't yet interface nicely with DeepChem's training framework. To change that, we define a `tokenize_smiles` function that combines all the steps spelled out above to process a single datapoint. Additionally, we define a `SmilesFeaturizer` that uses our custom `tokenize_smiles` function in its `_featurize` method and instanciate it as `smiles_featurizer` passing it our `vocab` and `max_len`.

In [20]:
```
def tokenize_smiles(x, vocab, max_len):
    tokenizer = dc.feat.smiles_tokenizer.BasicSmilesTokenizer()
    str2int = {s:i for i, s in enumerate(vocab)}
    encode = lambda s: [str2int[tok] for tok in s]
    zero_pad = lambda x: x + [0] * (max_len - len(x))
    x = tokenizer.tokenize(x)
    x = encode(x)
    x = zero_pad(x)
    return np.array(x)

class SmilesFeaturizer(dc.feat.Featurizer):
    def __init__(self, feat_func, vocab, max_len):
        self.feat_func = feat_func
        self.vocab = vocab
        self.max_len = max_len

    def _featurize(self, x):
        return self.feat_func(x, self.vocab, self.max_len)

smiles_featurizer = SmilesFeaturizer(tokenize_smiles, vocab, max_len)
```

Finally, we use the `smiles_featurizer` to create new Tox21 datasets that contain tokenized and numericalized SMILES in their `X` attribute.

In [21]:
```
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer=smiles_featurizer)
print(datasets[0].X)
```

```
[09:24:48] WARNING: not removing hydrogen atom without neighbors
[[19 19  2 ...  0  0  0]
 [19 19  2 ...  0  0  0]
 [24 19 42 ...  0  0  0]
 ...
 [24 16 19 ...  0  0  0]
 [19 19  2 ...  0  0  0]
 [19 19  2 ...  0  0  0]]
```

The datasets are now ready to be used with your custom DeepChem sequence model. Don't forget to wrap your model into the appropriate DeepChem model class.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro7,
 title={Going Deeper on Molecular Featurizations},
 organization={DeepChem},
 author={Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Going_Deeper_on_Molecu
 year={2021},
}
```

Working With Splitters

When using machine learning, you typically divide your data into training, validation, and test sets. The MoleculeNet loaders do this automatically. But how should you divide up the data? This question seems simple at first, but it turns out to be quite complicated. There are many ways of splitting up data, and which one you choose can have a big impact on the reliability of your results. This tutorial introduces some of the splitting methods provided by DeepChem.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [2]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

Out[2]:
```
'2.7.1'
```

Splitters

In DeepChem, a method of splitting samples into multiple datasets is defined by a `Splitter` object. Choosing an appropriate method for your data is very important. Otherwise, your trained model may seem to work much better than it really does.

Consider a typical drug development pipeline. You might begin by screening many thousands of molecules to see if they bind to your target of interest. Once you find one that seems to work, you try to optimize it by testing thousands of minor variations on it, looking for one that binds more strongly. Then perhaps you test it in animals and find it has unacceptable toxicity, so you try more variations to fix the problems.

This has an important consequence for chemical datasets: they often include lots of molecules that are very similar to each other. If you split the data into training and test sets in a naive way, the training set will include many molecules that are very similar to the ones in the test set, even if they are not exactly identical. As a result, the model may do very well on the test set, but then fail badly when you try to use it on other data that is less similar to the training data.

Let's take a look at a few of the splitters found in DeepChem.

- General Splitters
  - `RandomSplitter`
  - `RandomGroupSplitter`
  - `RandomStratifiedSplitter`
  - `SingletaskStratifiedSplitter`
  - `IndexSplitter`
  - `SpecifiedSplitter`
  - `TaskSplitter`
- Molecular Splitters
  - `ScaffoldSplitter`
  - `MolecularWeightSplitter`
  - `MaxMinSplitter`
  - `ButinaSplitter`
  - `FingerprintSplitter`

Let's take a look how different splitters work.

RandomSplitter

This is one of the simplest splitters. It just selects samples for the training, validation, and test sets in a completely random way.

Didn't we just say that's a bad idea? Well, it depends on your data. If every sample is truly independent of every other, then this is just as good a way as any to split the data. There is no universally best choice of splitter. It all depends on your particular dataset, and for some datasets this is a fine choice.

RandomStratifiedSplitter

Some datasets are very unbalanced: only a tiny fraction of all samples are positive. In that case, random splitting may sometimes lead to the validation or test set having few or even no positive samples for some tasks. That makes it unable to evaluate performance.

`RandomStratifiedSplitter` addresses this by dividing up the positive and negative samples evenly. If you ask for a 80/10/10 split, the validation and test sets will contain not just 10% of samples, but also 10% of the positive samples for each task.

ScaffoldSplitter

This splitter tries to address the problem discussed above where many molecules are very similar to each other. It identifies the scaffold that forms the core of each molecule, and ensures that all molecules with the same scaffold are put into the same dataset. This is still not a perfect solution, since two molecules may have different scaffolds but be very similar in other ways, but it usually is a large improvement over random splitting.

ButinaSplitter

This is another splitter that tries to address the problem of similar molecules. It clusters them based on their molecular fingerprints, so that ones with similar fingerprints will tend to be in the same dataset. The time required by this splitting algorithm scales as the square of the number of molecules, so it is mainly useful for small to medium sized datasets.

SpecifiedSplitter

This splitter leaves everything up to the user. You tell it exactly which samples to put in each dataset. This is useful when you know in advance that a particular splitting is appropriate for your data.

An example is temporal splitting. Consider a research project where you are continually generating and testing new molecules. As you gain more data, you periodically retrain your model on the steadily growing dataset, then use it to predict results for other not yet tested molecules. A good way of validating whether this works is to pick a particular cutoff date, train the model on all data you had at that time, and see how well it predicts other data that was generated later.

TaskSplitter

Provides a simple interface for splitting datasets task-wise.

For some learning problems, the training and test datasets should have different tasks entirely. This is a different paradigm from the usual Splitter, which ensures that split datasets have different data points, not different tasks. This method improves multi-task learning and problem decomposition situations by enhancing their efficiency and performance.

SingletaskStratifiedSplitter

Another way of splitting data, particularly for classification tasks with imbalanced class distributions is the single-task stratified splitter. The single-task stratified splitter maintains the class distribution in the original dataset across training, validation and test sets. This is crucial when working with imbalanced datasets where some classes may be under-represented.

FingerprintSplitter

Class for doing data splits based on the Tanimoto similarity(Tanimoto similarity measures overlap between two sets succinctly) between ECFP4 fingerprints(ECFP4 fingerprints encode unique parts of molecules for efficient comparison).

This class tries to split the data such that the molecules in each dataset are as different as possible from the ones in the other datasets. This makes it a very stringent test of models. Predicting the test and validation sets may require extrapolating far outside the training data.It splits molecular datasets using Tanimoto similarity scores calculated from ECFP4 fingerprints. ECFP4, based on Morgan fingerprints, encodes molecular substructures.

MolecularWeightSplitter

Another splitter performs data splits based on molecular weight

Effect of Using Different Splitters

Let's look at an example. We will load the Tox21 toxicity dataset using random, fingerprint, scaffold, and Butina splitting. For each one we train a model and evaluate it on the training and test sets.

In [3]:
```
import deepchem as dc

splitters = ['random', 'scaffold', 'butina', 'fingerprint']
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
for splitter in splitters:
    tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='ECFP', splitter=splitter)
    train_dataset, valid_dataset, test_dataset = datasets
    model = dc.models.MultitaskClassifier(n_tasks=len(tasks), n_features=1024, layer_sizes=[1000])
    model.fit(train_dataset, nb_epoch=10)
    print('splitter:', splitter)
    print('training set score:', model.evaluate(train_dataset, [metric], transformers))
    print('test set score:', model.evaluate(test_dataset, [metric], transformers))
    print()
```

```
splitter: random
training set score: {'roc_auc_score': 0.9554904185889012}
test set score: {'roc_auc_score': 0.7854105497196335}

splitter: scaffold
training set score: {'roc_auc_score': 0.958752269558084}
test set score: {'roc_auc_score': 0.6849149319233084}

splitter: butina
training set score: {'roc_auc_score': 0.9584914471889929}
test set score: {'roc_auc_score': 0.6061155305251504}

splitter: fingerprint
training set score: {'roc_auc_score': 0.954193849465875}
test set score: {'roc_auc_score': 0.6235667313881933}
```

All of them produce very similar performance on the training set, but the random splitter has much higher performance on the test set. Scaffold splitting has a lower test set score, and Butina splitting is even lower. Does that mean random splitting is better? No! It means random splitting doesn't give you an accurate measure of how well your model works. Because the test set contains lots of molecules that are very similar to ones in the training set, it isn't truly independent. It makes the model appear to work better than it really does. Scaffold splitting and Butina splitting give a better indication of what you can expect on independent data in the future.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [5]:
```
@manual{Intro8,
 title={Working With Splitters},
 organization={DeepChem},
 author={Eastman, Peter, Mohapatra, Bibhusundar and Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Working_With_Splitters
 year={2021},
}
```

Advanced Model Training

In the tutorials so far we have followed a simple procedure for training models: load a dataset, create a model, call `fit()`, evaluate it, and call ourselves done. That's fine for an example, but in real machine learning projects the process is usually more complicated. In this tutorial we will look at a more realistic workflow for training a model.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following installation commands. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install DeepChem in your local machine again.

In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

Hyperparameter Optimization

Let's start by loading the HIV dataset. It classifies over 40,000 molecules based on whether they inhibit HIV replication.

In [1]:
```
import deepchem as dc

tasks, datasets, transformers = dc.molnet.load_hiv(featurizer='ECFP', split='scaffold')
train_dataset, valid_dataset, test_dataset = datasets
```

Now let's train a model on it. We will use a `MultitaskClassifier`, which is just a stack of dense layers. But that still leaves a lot of options. How many layers should there be, and how wide should each one be? What dropout rate should we use? What learning rate?

These are called hyperparameters. The standard way to select them is to try lots of values, train each model on the training set, and evaluate it on the validation set. This lets us see which ones work best.

You could do that by hand, but usually it's easier to let the computer do it for you. DeepChem provides a selection of hyperparameter optimization algorithms, which are found in the `dc.hyper` package. For this example we'll use `GridHyperparamOpt`, which is the most basic method. We just give it a list of options for each hyperparameter and it exhaustively tries all combinations of them.

The lists of options are defined by a `dict` that we provide. For each of the model's arguments, we provide a list of values to try. In this example we consider three possible sets of hidden layers: a single layer of width 500, a single layer of width 1000, or two layers each of width 1000. We also consider two dropout rates (20% and 50%) and two learning rates (0.001 and 0.0001).

In [2]:
```
params_dict = {
    'n_tasks': [len(tasks)],
    'n_features': [1024],
    'layer_sizes': [[500], [1000], [1000, 1000]],
    'dropouts': [0.2, 0.5],
    'learning_rate': [0.001, 0.0001]
}
optimizer = dc.hyper.GridHyperparamOpt(dc.models.MultitaskClassifier)
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
best_model, best_hyperparams, all_results = optimizer.hyperparam_search(
        params_dict, train_dataset, valid_dataset, metric, transformers)
```

`hyperparam_search()` returns three arguments: the best model it found, the hyperparameters for that model, and a full listing of the validation score for every model. Let's take a look at the last one.

In [3]:
```
all_results
```

Out[3]:

```
{'_dropouts_0.200000_layer_sizes[500]_learning_rate_0.001000_n_features_1024_n_tasks_1': 0.759624393738977,
 '_dropouts_0.200000_layer_sizes[500]_learning_rate_0.000100_n_features_1024_n_tasks_1': 0.7680791323731138,
 '_dropouts_0.500000_layer_sizes[500]_learning_rate_0.001000_n_features_1024_n_tasks_1': 0.7623870149911817,
 '_dropouts_0.500000_layer_sizes[500]_learning_rate_0.000100_n_features_1024_n_tasks_1': 0.7552282358416618,
 '_dropouts_0.200000_layer_sizes[1000]_learning_rate_0.001000_n_features_1024_n_tasks_1': 0.7689915858318636,
 '_dropouts_0.200000_layer_sizes[1000]_learning_rate_0.000100_n_features_1024_n_tasks_1': 0.7619292572996277,
 '_dropouts_0.500000_layer_sizes[1000]_learning_rate_0.001000_n_features_1024_n_tasks_1': 0.7641491524593376,
 '_dropouts_0.500000_layer_sizes[1000]_learning_rate_0.000100_n_features_1024_n_tasks_1': 0.7609877155594749,
 '_dropouts_0.200000_layer_sizes[1000, 1000]_learning_rate_0.001000_n_features_1024_n_tasks_1': 0.770716980207721,
 '_dropouts_0.200000_layer_sizes[1000, 1000]_learning_rate_0.000100_n_features_1024_n_tasks_1': 0.7750327625906329,
 '_dropouts_0.500000_layer_sizes[1000, 1000]_learning_rate_0.001000_n_features_1024_n_tasks_1': 0.725972314079953,
 '_dropouts_0.500000_layer_sizes[1000, 1000]_learning_rate_0.000100_n_features_1024_n_tasks_1': 0.7546280986674505}
```


We can see a few general patterns. Using two layers with the larger learning rate doesn't work very well. It seems the deeper model requires a smaller learning rate. We also see that 20% dropout usually works better than 50%. Once we narrow down the list of models based on these observations, all the validation scores are very close to each other, probably close enough that the remaining variation is mainly noise. It doesn't seem to make much difference which of the remaining hyperparameter sets we use, so let's arbitrarily pick a single layer of width 1000 and learning rate of 0.0001.

Early Stopping

There is one other important hyperparameter we haven't considered yet: how long we train the model for. `GridHyperparamOpt` trains each for a fixed, fairly small number of epochs. That isn't necessarily the best number.

You might expect that the longer you train, the better your model will get, but that isn't usually true. If you train too long, the model will usually start overfitting to irrelevant details of the training set. You can tell when this happens because the validation set score stops increasing and may even decrease, while the score on the training set continues to improve.

Fortunately, we don't need to train lots of different models for different numbers of steps to identify the optimal number. We just train it once, monitor the validation score, and keep whichever parameters maximize it. This is called "early stopping". DeepChem's `ValidationCallback` class can do this for us automatically. In the example below, we have it compute the validation set's ROC AUC every 1000 training steps. If you add the `save_dir` argument, it will also save a copy of the best model parameters to disk.

In [4]:
```
model = dc.models.MultitaskClassifier(n_tasks=len(tasks),
                                      n_features=1024,
                                      layer_sizes=[1000],
                                      dropouts=0.2,
                                      learning_rate=0.0001)
callback = dc.models.ValidationCallback(valid_dataset, 1000, metric)
model.fit(train_dataset, nb_epoch=50, callbacks=callback)
```

```
Step 1000 validation: roc_auc_score=0.759757
Step 2000 validation: roc_auc_score=0.770685
Step 3000 validation: roc_auc_score=0.771588
Step 4000 validation: roc_auc_score=0.777862
Step 5000 validation: roc_auc_score=0.773894
Step 6000 validation: roc_auc_score=0.763762
Step 7000 validation: roc_auc_score=0.766361
Step 8000 validation: roc_auc_score=0.767026
Step 9000 validation: roc_auc_score=0.761239
Step 10000 validation: roc_auc_score=0.761279
Step 11000 validation: roc_auc_score=0.765363
Step 12000 validation: roc_auc_score=0.769481
Step 13000 validation: roc_auc_score=0.768523
Step 14000 validation: roc_auc_score=0.761306
Step 15000 validation: roc_auc_score=0.77397
Step 16000 validation: roc_auc_score=0.764848
```

Out[4]:
```
0.8040038299560547
```

Learning Rate Schedules

In the examples above we use a fixed learning rate throughout training. In some cases it works better to vary the learning rate during training. To do this in DeepChem, we simply specify a `LearningRateSchedule` object instead of a number for the `learning_rate` argument. In the following example we use a learning rate that decreases exponentially. It starts at 0.0002, then gets multiplied by 0.9 after every 1000 steps.

In [5]:
```
learning_rate = dc.models.optimizers.ExponentialDecay(0.0002, 0.9, 1000)
```

```
model = dc.models.MultitaskClassifier(n_tasks=len(tasks),
                                      n_features=1024,
                                      layer_sizes=[1000],
                                      dropouts=0.2,
                                      learning_rate=learning_rate)
model.fit(train_dataset, nb_epoch=50, callbacks=callback)
```

```
Step 1000 validation: roc_auc_score=0.736547
Step 2000 validation: roc_auc_score=0.758979
Step 3000 validation: roc_auc_score=0.768361
Step 4000 validation: roc_auc_score=0.764898
Step 5000 validation: roc_auc_score=0.775253
Step 6000 validation: roc_auc_score=0.779898
Step 7000 validation: roc_auc_score=0.76991
Step 8000 validation: roc_auc_score=0.771515
Step 9000 validation: roc_auc_score=0.773796
Step 10000 validation: roc_auc_score=0.776977
Step 11000 validation: roc_auc_score=0.778866
Step 12000 validation: roc_auc_score=0.777066
Step 13000 validation: roc_auc_score=0.77616
Step 14000 validation: roc_auc_score=0.775646
Step 15000 validation: roc_auc_score=0.772785
Step 16000 validation: roc_auc_score=0.769975
```

Out[5]: 0.22854619979858398

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro9,
 title={Advanced Model Training},
 organization={DeepChem},
 author={Eastman, Peter and Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Advanced_Model_Traini
 year={2021},
}
```

Creating a High Fidelity Dataset from Experimental Data

In this tutorial, we will look at what is involved in creating a new Dataset from experimental data. As we will see, the mechanics of creating the Dataset object is only a small part of the process. Most real datasets need significant cleanup and QA before they are suitable for training models.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

Working With Data Files

Suppose you were given data collected by an experimental collaborator. You would like to use this data to construct a machine learning model.

How do you transform this data into a dataset capable of creating a useful model?

Building models from novel data can present several challenges. Perhaps the data was not recorded in a convenient manner. Additionally, perhaps the data contains noise. This is a common occurrence with, for example, biological assays due to the large number of external variables and the difficulty and cost associated with collecting multiple samples. This is a problem because you do not want your model to fit to this noise.

Hence, there are two primary challenges:

- Parsing data
- De-noising data

In this tutorial, we will walk through an example of curating a dataset from an excel spreadsheet of experimental drug measurements. Before we dive into this example though, let's do a brief review of DeepChem's input file handling and featurization capabilities.

Input Formats

DeepChem supports a whole range of input files. For example, accepted input formats include .csv, .sdf, .fasta, .png, .tif and other file formats. The loading for a particular file format is governed by the `Loader` class associated with that format. For example, to load a .csv file we use the `CSVLoader` class. Here's an example of a .csv file that fits the requirements of `CSVLoader`.

1. A column containing SMILES strings.
2. A column containing an experimental measurement.
3. (Optional) A column containing a unique compound identifier.

Here's an example of a potential input file.

Compound ID	measured log solubility in mols per litre	smiles
benzothiazole	-1.5	c2ccc1scnc1c2

Here the "smiles" column contains the SMILES string, the "measured log solubility in mols per litre" contains the experimental measurement, and "Compound ID" contains the unique compound identifier.

Data Featurization

Most machine learning algorithms require that input data form vectors. However, input data for drug-discovery datasets routinely come in the form of lists of molecules and associated experimental readouts. To load the data, we use a subclass of `dc.data.DataLoader` such as `dc.data.CSVLoader` or `dc.data.SDFLoader`. Users can subclass `dc.data.DataLoader` to load arbitrary file formats. All loaders must be passed a `dc.feat.Featurizer` object, which specifies how to transform molecules into vectors. DeepChem provides a number of different subclasses of `dc.feat.Featurizer`.

Parsing data

In order to read in the data, we will use the pandas data analysis library.

In order to convert the drug names into smiles strings, we will use pubchempy. This isn't a standard DeepChem dependency, but you can install this library with `conda install pubchempy`.

In [ ]:
```
!conda install pubchempy
```

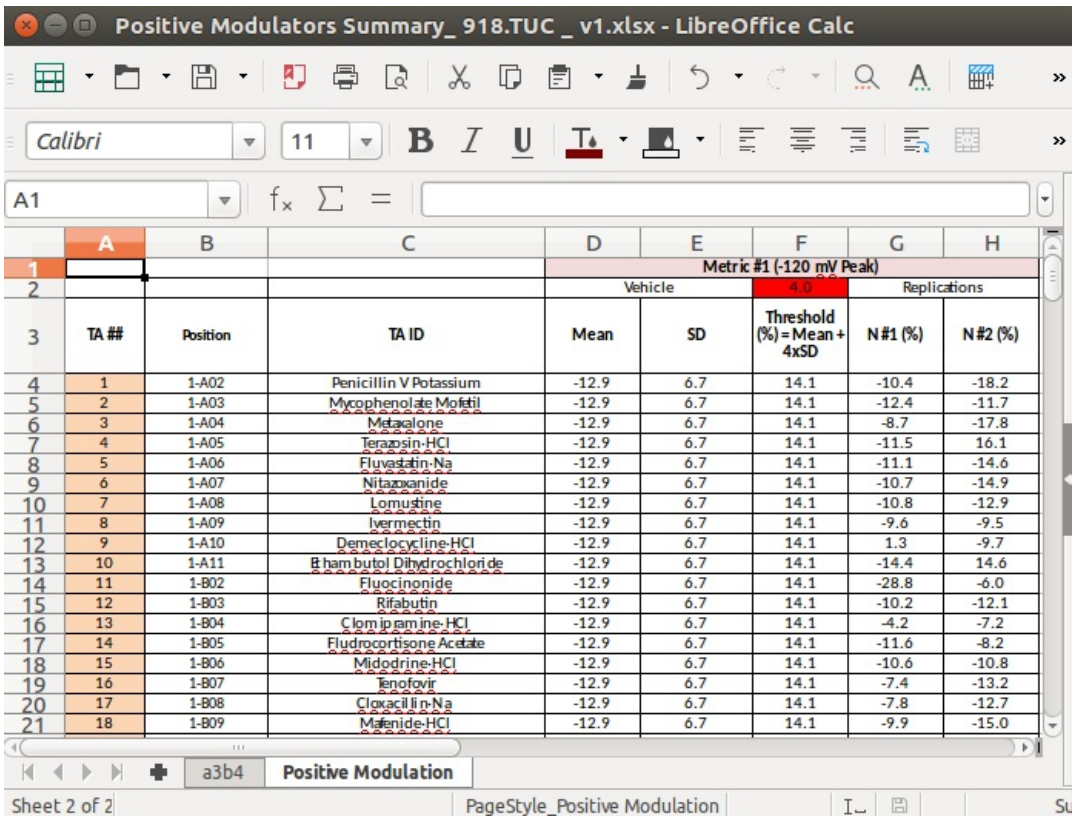
In [1]:
```
import os
import pandas as pd
from pubchempy import get_cids, get_compounds
```

Pandas is magic but it doesn't automatically know where to find your data of interest. You likely will have to look at it first using a GUI.

We will now look at a screenshot of this dataset as rendered by LibreOffice.

To do this, we will import Image and os.

In [2]:
```
import os
from IPython.display import Image, display
current_dir = os.path.dirname(os.path.realpath('__file__'))
data_screenshot = os.path.join(current_dir, 'assets/dataset_preparation_gui.png')
display(Image(filename=data_screenshot))
```



We see the data of interest is on the second sheet, and contained in columns "TA ID", "N #1 (%)", and "N #2 (%)".

Additionally, it appears much of this spreadsheet was formatted for human readability (multicolumn headers, column labels with spaces and symbols, etc.). This makes the creation of a neat dataframe object harder. For this reason we will cut everything that is unnecesary or inconvenient.

In [3]:
```
import deepchem as dc
dc.utils.download_url(
    'https://github.com/deepchem/deepchem/raw/master/datasets/Positive%20Modulators%20Summary_%20918.TUC%20_%20
    current_dir,
    'Positive Modulators Summary_ 918.TUC _ v1.xlsx'
)
```

In [4]:
```
raw_data_file = os.path.join(current_dir, 'Positive Modulators Summary_ 918.TUC _ v1.xlsx')
raw_data_excel = pd.ExcelFile(raw_data_file)

# second sheet only
raw_data = raw_data_excel.parse(raw_data_excel.sheet_names[1])
```

In [5]:
```
# preview 5 rows of raw dataframe
raw_data.loc[raw_data.index[:5]]
```

Out[5]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Metric #1 (-120 mV Peak)	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7
0	NaN	NaN	NaN	Vehicle	NaN	4	Replications	NaN
1	TA ##	Position	TA ID	Mean	SD	Threshold (%) = Mean + 4xSD	N #1 (%)	N #2 (%)
2	1	1-A02	Penicillin V Potassium	-12.8689	6.74705	14.1193	-10.404	-18.1929
3	2	1-A03	Mycophenolate Mofetil	-12.8689	6.74705	14.1193	-12.4453	-11.7175
4	3	1-A04	Metaxalone	-12.8689	6.74705	14.1193	-8.65572	-17.7753

Note that the actual row headers are stored in row 1 and not 0 above.

In [6]:
```
# remove column labels (rows 0 and 1), as we will replace them
# only take data given in columns "TA ID" "N #1 (%)" (3) and "N #2 (%)" (4)
raw_data = raw_data.iloc[2:, [2, 6, 7]]

# reset the index so we keep the label but number from 0 again
raw_data.reset_index(inplace=True)

## rename columns
raw_data.columns = ['label', 'drug', 'n1', 'n2']
```

In [7]:
```
# preview cleaner dataframe
raw_data.loc[raw_data.index[:5]]
```

Out[7]:

	label	drug	n1	n2
0	2	Penicillin V Potassium	-10.404	-18.1929
1	3	Mycophenolate Mofetil	-12.4453	-11.7175
2	4	Metaxalone	-8.65572	-17.7753
3	5	Terazosin·HCl	-11.5048	16.0825
4	6	Fluvastatin·Na	-11.1354	-14.553

This formatting is closer to what we need.

Now, let's take the drug names and get smiles strings for them (format needed for DeepChem).

In [8]:
```
drugs = raw_data['drug'].values
```

For many of these, we can retreive the smiles string via the canonical_smiles attribute of the `get_compounds` object (using `pubchempy`)

In [9]:
```
get_compounds(drugs[1], 'name')
```

Out[9]:
```
[Compound(5281078)]
```

In [10]:
```
get_compounds(drugs[1], 'name')[0].canonical_smiles
```

Out[10]:
```
'CC1=C2COC(=O)C2=C(C(=C1OC)CC=C(C)CCC(=O)OCCN3CCOCC3)O'
```

However, some of these drug names have variables spaces and symbols (·, (±), etc.), and names that may not be readable by pubchempy.

For this task, we will do a bit of hacking via regular expressions. Also, we notice that all ions are written in a shortened form that will need to be expanded. For this reason we use a dictionary, mapping the shortened ion names to versions recognizable to pubchempy.

Unfortunately you may have several corner cases that will require more hacking.

In [11]:
```
import re

ion_replacements = {
    'HBr': ' hydrobromide',
    '2Br': ' dibromide',
    'Br': ' bromide',
    'HCl': ' hydrochloride',
    '2H2O': ' dihydrate',
    'H20': ' hydrate',
    'Na': ' sodium'
```

```
}

ion_keys = ['H20', 'HBr', 'HCl', '2Br', '2H2O', 'Br', 'Na']

def compound_to_smiles(cmpd):
    # remove spaces and irregular characters
    compound = re.sub(r'([^\s\w]|_)+', '', cmpd)

    # replace ion names if needed
    for ion in ion_keys:
        if ion in compound:
            compound = compound.replace(ion, ion_replacements[ion])

    # query for cid first in order to avoid timeouterror
    cid = get_cids(compound, 'name')[0]
    smiles = get_compounds(cid)[0].canonical_smiles

    return smiles
```

Now let's actually convert all these compounds to smiles. This conversion will take a few minutes so might not be a bad spot to go grab a coffee or tea and take a break while this is running! Note that this conversion will sometimes fail so we've added some error handling to catch these cases below.

In [12]:
```
smiles_map = {}
for i, compound in enumerate(drugs):
    try:
        smiles_map[compound] = compound_to_smiles(compound)
    except:
        print("Errored on %s" % i)
        continue
```

```
Errored on 162
Errored on 303
```

In [13]:
```
smiles_data = raw_data
# map drug name to smiles string
smiles_data['drug'] = smiles_data['drug'].apply(lambda x: smiles_map[x] if x in smiles_map else None)
```

In [14]:
```
# preview smiles data
smiles_data.loc[smiles_data.index[:5]]
```

Out[14]:

	label	drug	n1	n2
0	2	CC1(C(N2C(S1)C(C2=O)NC(=O)COC3=CC=CC=C3)C(=O)[...	-10.404	-18.1929
1	3	CC1=C2COC(=O)C2=C(C(=C1OC)CC=C(C)CCC(=O)OCCN3C...	-12.4453	-11.7175
2	4	CC1=CC(=CC(=C1)OCC2CNC(=O)O2)C	-8.65572	-17.7753
3	5	COC1=C(C=C2C(=C1)C(=NC(=N2)N3CCN(CC3)C(=O)C4CC...	-11.5048	16.0825
4	6	CC(C)N1C2=CC=CC=C2C(=C1C=CC(CC(CC(=O)[O-])O)O)...	-11.1354	-14.553

Hooray, we have mapped each drug name to its corresponding smiles code.

Now, we need to look at the data and remove as much noise as possible.

De-noising data

In machine learning, we know that there is no free lunch. You will need to spend time analyzing and understanding your data in order to frame your problem and determine the appropriate model framework. Treatment of your data will depend on the conclusions you gather from this process.

Questions to ask yourself:

- What are you trying to accomplish?
- What is your assay?
- What is the structure of the data?
- Does the data make sense?
- What has been tried previously?

For this project (respectively):

- I would like to build a model capable of predicting the affinity of an arbitrary small molecule drug to a particular ion channel protein
- For an input drug, data describing channel inhibition
- A few hundred drugs, with n=2

- Will need to look more closely at the dataset*
- Nothing on this particular protein

*This will involve plotting, so we will import matplotlib and seaborn. We will also need to look at molecular structures, so we will import rdkit. We will also use the seaborn library which you can install with `conda install seaborn`.

In [15]:
```
import matplotlib.pyplot as plt
%matplotlib inline

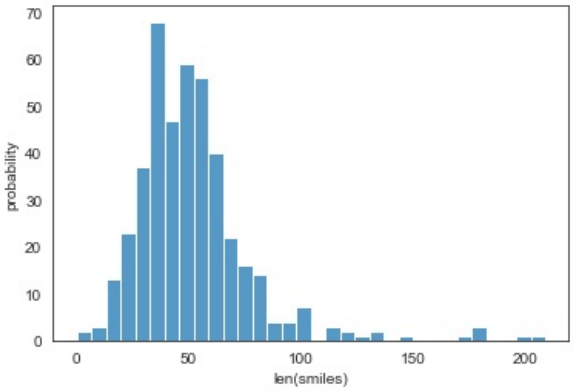
import seaborn as sns
sns.set_style('white')

from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Draw, PyMol, rdFMCS
from rdkit.Chem.Draw import IPythonConsole
from rdkit import rdBase
import numpy as np
```

Our goal is to build a small molecule model, so let's make sure our molecules are all small. This can be approximated by the length of each smiles string.

In [16]:
```
smiles_data['len'] = [len(i) if i is not None else 0 for i in smiles_data['drug']]
smiles_lens = [len(i) if i is not None else 0 for i in smiles_data['drug']]
sns.histplot(smiles_lens)
plt.xlabel('len(smiles)')
plt.ylabel('probability')
```

Out[16]:
```
Text(0, 0.5, 'probability')
```



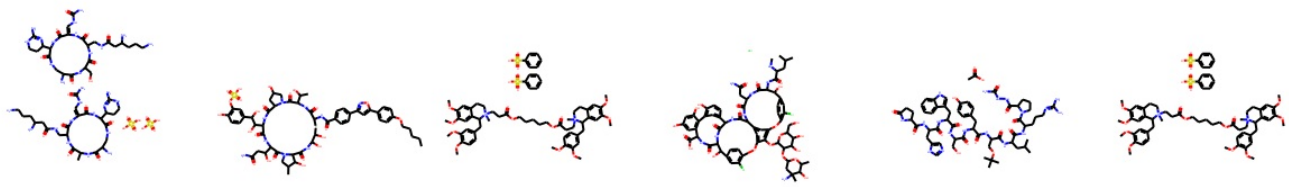
Some of these look rather large, len(smiles) > 150. Let's see what they look like.

In [17]:
```
# indices of large looking molecules
suspiciously_large = np.where(np.array(smiles_lens) > 150)[0]

# corresponding smiles string
long_smiles = smiles_data.loc[smiles_data.index[suspiciously_large]]['drug'].values

# look
Draw._MolsToGridImage([Chem.MolFromSmiles(i) for i in long_smiles], molsPerRow=6)
```

Out[17]:



As suspected, these are not small molecules, so we will remove them from the dataset. The argument here is that these molecules could register as inhibitors simply because they are large. They are more likely to sterically block the channel, rather than diffuse inside and bind (which is what we are interested in).

The lesson here is to remove data that does not fit your use case.

In [18]:
```
# drop large molecules
smiles_data = smiles_data[~smiles_data['drug'].isin(long_smiles)]
```

Now, let's look at the numerical structure of the dataset.

First, check for NaNs.

In [19]:
```
nan_rows = smiles_data[smiles_data.isnull().T.any().T]
nan_rows[['n1', 'n2']]
```

Out[19]:

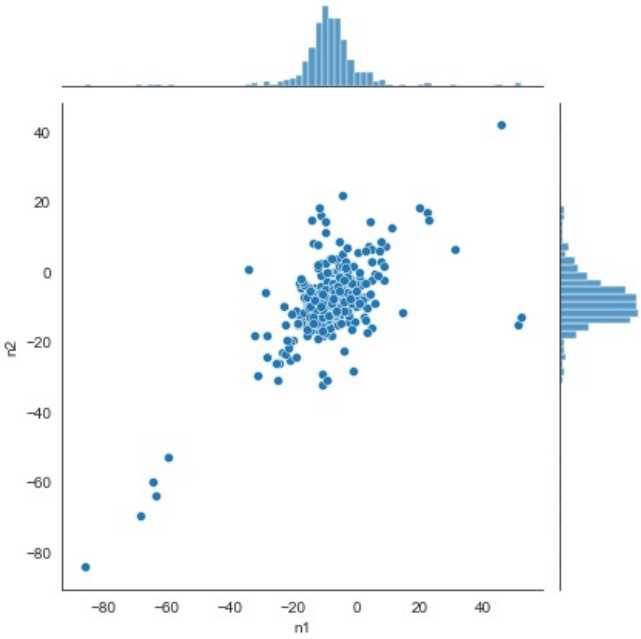
	n1	n2
62	NaN	-7.8266
162	-12.8456	-11.4627
175	NaN	-6.61225
187	NaN	-8.23326
233	-8.21781	NaN
262	NaN	-12.8788
288	NaN	-2.34264
300	NaN	-8.19936
301	NaN	-10.4633
303	-5.61374	8.42267
311	NaN	-8.78722

I don't trust n=1, so I will throw these out.

Then, let's examine the distribution of n1 and n2.

In [20]:
```
df = smiles_data.dropna(axis=0, how='any')
# seaborn jointplot will allow us to compare n1 and n2, and plot each marginal
sns.jointplot(x='n1', y='n2', data=smiles_data)
```

Out[20]:
```
<seaborn.axisgrid.JointGrid at 0x14c4e37d0>
```



We see that most of the data is contained in the gaussian-ish blob centered a bit below zero. We see that there are a few clearly active datapoints located in the bottom left, and one on the top right. These are all distinguished from the majority of the data. How do we handle the data in the blob?

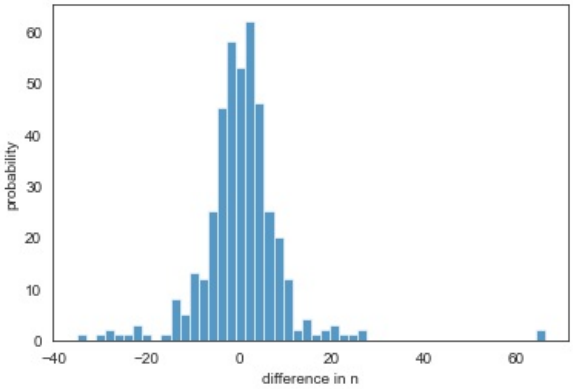
Because n1 and n2 represent the same measurement, ideally they would be of the same value. This plot should be tightly aligned to the diagonal, and the pearson correlation coefficient should be 1. We see this is not the case. This helps gives us an idea of the error of our assay.

Let's look at the error more closely, plotting in the distribution of (n1-n2).

In [21]:
```
diff_df = df['n1'] - df['n2']

sns.histplot(diff_df)
plt.xlabel('difference in n')
plt.ylabel('probability')
```

Out[21]:
```
Text(0, 0.5, 'probability')
```



This looks pretty gaussian, let's get the 95% confidence interval by fitting a gaussian via scipy, and taking 2*the standard deviation

In [22]:
```
from scipy import stats
mean, std = stats.norm.fit(np.asarray(diff_df, dtype=np.float32))
ci_95 = std*2
ci_95
```

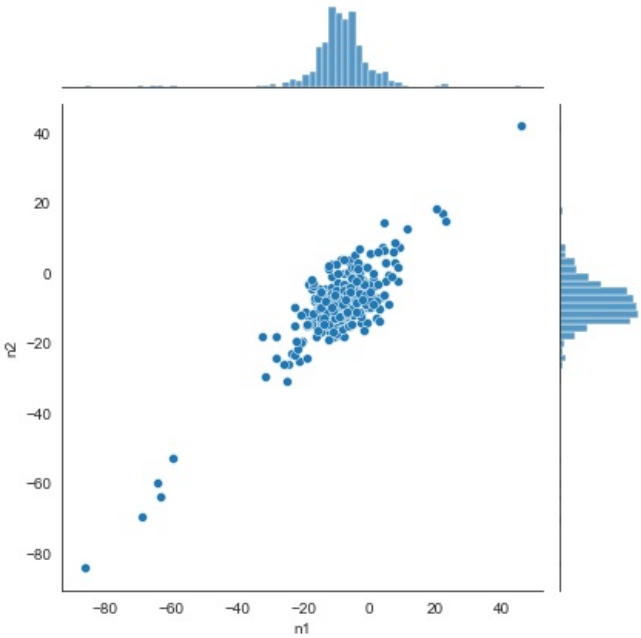
Out[22]:
```
17.75387954711914
```

Now, I don't trust the data outside of the confidence interval, and will therefore drop these datapoints from df.

For example, in the plot above, at least one datapoint has n1-n2 > 60. This is disconcerting.

In [23]:
```
noisy = diff_df[abs(diff_df) > ci_95]
df = df.drop(noisy.index)
sns.jointplot(x='n1', y='n2', data=df)
```

Out[23]:
```
<seaborn.axisgrid.JointGrid at 0x15a363c10>
```



Now that data looks much better!

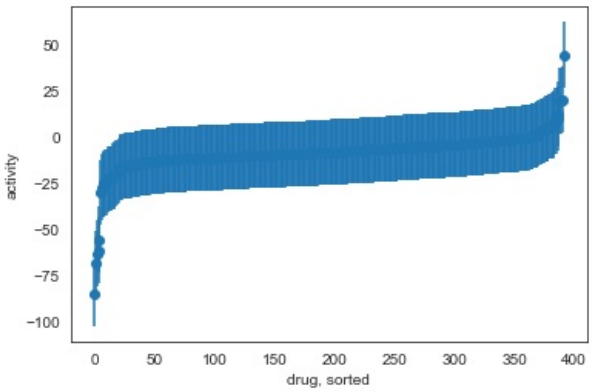
So, let's average n1 and n2, and take the error bar to be ci_95.

In [24]:
```
avg_df = df[['label', 'drug']].copy()
n_avg = df[['n1', 'n2']].mean(axis=1)
avg_df['n'] = n_avg
avg_df.sort_values('n', inplace=True)
```

Now, let's look at the sorted data with error bars.

In [25]:
```
plt.errorbar(np.arange(avg_df.shape[0]), avg_df['n'], yerr=ci_95, fmt='o')
plt.xlabel('drug, sorted')
plt.ylabel('activity')
```

Out[25]:
```
Text(0, 0.5, 'activity')
```



Now, let's identify our active compounds.

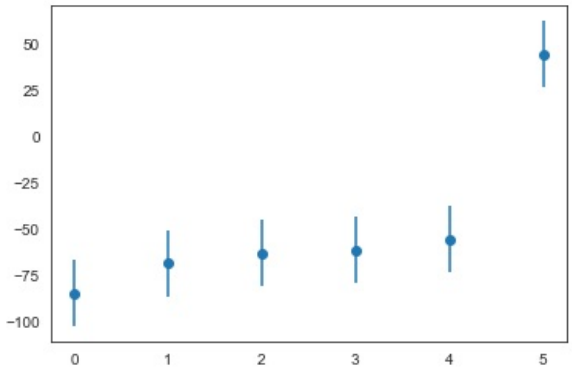
In my case, this required domain knowledge. Having worked in this area, and having consulted with professors specializing on this channel, I am interested in compounds where the absolute value of the activity is greater than 25. This relates to the desired drug potency we would like to model.

If you are not certain how to draw the line between active and inactive, this cutoff could potentially be treated as a hyperparameter.

In [26]:
```
actives = avg_df[abs(avg_df['n'])-ci_95 > 25]['n']

plt.errorbar(np.arange(actives.shape[0]), actives, yerr=ci_95, fmt='o')
```

Out[26]:
```
<ErrorbarContainer object of 3 artists>
```



In [27]:
```
# summary
print (raw_data.shape, avg_df.shape, len(actives.index))
```

```
(430, 5) (392, 3) 6
```

In summary, we have:

- Removed data that did not address the question we hope to answer (small molecules only)
- Dropped NaNs
- Determined the noise of our measurements
- Removed exceptionally noisy datapoints
- Identified actives (using domain knowledge to determine a threshold)

Determine model type, final form of dataset, and sanity load

Now, what model framework should we use?

Given that we have 392 datapoints and 6 actives, this data will be used to build a low data one-shot classifier (10.1021/acscentsci.6b00367). If there were datasets of similar character, transfer learning could potentially be used, but this is not the case at the moment.

Let's apply logic to our dataframe in order to cast it into a binary format, suitable for classification.

In [28]:
```
# 1 if condition for active is met, 0 otherwise
avg_df.loc[:, 'active'] = (abs(avg_df['n'])-ci_95 > 25).astype(int)
```

Now, save this to file.

In [29]:
```
avg_df.to_csv('modulators.csv', index=False)
```

Now, we will convert this dataframe to a DeepChem dataset.

In [30]:
```
dataset_file = 'modulators.csv'
task = ['active']
featurizer_func = dc.feat.ConvMolFeaturizer()

loader = dc.data.CSVLoader(tasks=task, feature_field='drug', featurizer=featurizer_func)
dataset = loader.create_dataset(dataset_file)
```

Lastly, it is often advantageous to numerically transform the data in some way. For example, sometimes it is useful to normalize the data, or to zero the mean. This depends in the task at hand.

Built into DeepChem are many useful transformers, located in the deepchem.transformers.transformers base class.

Because this is a classification model, and the number of actives is low, I will apply a balancing transformer. I treated this transformer as a hyperparameter when I began training models. It proved to unambiguously improve model performance.

In [31]:
```
transformer = dc.trans.BalancingTransformer(dataset=dataset)
dataset = transformer.transform(dataset)
```

Now let's save the balanced dataset object to disk, and then reload it as a sanity check.

In [32]:
```
dc.utils.save_to_disk(dataset, 'balanced_dataset.joblib')
balanced_dataset = dc.utils.load_from_disk('balanced_dataset.joblib')
```

Tutorial written by Keri McKiernan (github.com/kmckiern) on September 8, 2016

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Bibliography

[2] Anderson, Eric, Gilman D. Veith, and David Weininger. "SMILES, a line notation and computerized interpreter for chemical structures." US Environmental Protection Agency, Environmental Research Laboratory, 1987.

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro10,
 title={Creating a high fidelity model from experimental data},
 organization={DeepChem},
 author={Eastman, Peter and Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/tree/master/examples/tutorials}},
 year={2021},
}
```

Putting Multitask Learning to Work

This notebook walks through the creation of multitask models on MUV [1]. The goal is to demonstrate how multitask methods can provide improved performance in situations with little or very unbalanced data.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

The MUV dataset is a challenging benchmark in molecular design that consists of 17 different "targets" where there are only a few "active" compounds per target. There are 93,087 compounds in total, yet no task has more than 30 active compounds, and many have even less. Training a model with such a small number of positive examples is very challenging. Multitask models address this by training a single model that predicts all the different targets at once. If a feature is useful for predicting one task, it often is useful for predicting several other tasks as well. Each added task makes it easier to learn important features, which improves performance on other tasks [2].

To get started, let's load the MUV dataset. The MoleculeNet loader function automatically splits it into training, validation, and test sets. Because there are so few positive examples, we use stratified splitting to ensure the test set has enough of them to evaluate.

In [1]:
```
import deepchem as dc
import numpy as np

tasks, datasets, transformers = dc.molnet.load_muv(split='stratified')
train_dataset, valid_dataset, test_dataset = datasets
```

Now let's train a model on it. We'll use a MultitaskClassifier, which is a simple stack of fully connected layers.

In [2]:
```
n_tasks = len(tasks)
n_features = train_dataset.get_data_shape()[0]
model = dc.models.MultitaskClassifier(n_tasks, n_features)
model.fit(train_dataset)
```

Out[2]:
```
0.0004961589723825455
```

Let's see how well it does on the test set. We loop over the 17 tasks and compute the ROC AUC for each one.

In [3]:
```
y_true = test_dataset.y
y_pred = model.predict(test_dataset)
metric = dc.metrics.roc_auc_score
for i in range(n_tasks):
    score = metric(dc.metrics.to_one_hot(y_true[:,i]), y_pred[:,i])
    print(tasks[i], score)
```

```
MUV-466 0.9207684040838259
MUV-548 0.7480655561526062
MUV-600 0.9927995701235895
MUV-644 0.9974207415368082
MUV-652 0.7823481998925309
MUV-689 0.6636843990686011
MUV-692 0.6319093677234462
MUV-712 0.7787838079885365
MUV-713 0.7910711087229088
MUV-733 0.4401307540748701
MUV-737 0.34679383843811573
MUV-810 0.9564571019165323
MUV-832 0.9991044241447251
MUV-846 0.7519881783987103
MUV-852 0.8516747268493642
MUV-858 0.5906591438294824
MUV-859 0.5962954008166774
```

Not bad! Recall that random guessing would produce a ROC AUC score of 0.5, and a perfect predictor would score 1.0. Most of the tasks did much better than random guessing, and many of them are above 0.9.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Bibliography

[1] https://pubs.acs.org/doi/10.1021/ci8002649

[2] https://pubs.acs.org/doi/abs/10.1021/acs.jcim.7b00146

In [ ]:

Tutorial Part 13: Modeling Protein-Ligand Interactions

By Nathan C. Frey | Twitter and Bharath Ramsundar | Twitter

In this tutorial, we'll walk you through the use of machine learning and molecular docking methods to predict the binding energy of a protein-ligand complex. Recall that a ligand is some small molecule which interacts (usually non-covalently) with a protein. Molecular docking performs geometric calculations to find a “binding pose” with a small molecule interacting with a protein in a suitable binding pocket (that is, a region on the protein which has a groove in which the small molecule can rest).

The structure of proteins can be determined experimentally with techniques like Cryo-EM or X-ray crystallography. This can be a powerful tool for structure-based drug discovery. For more info on docking, read the AutoDock Vina paper and the `deepchem.dock` documentation. There are many graphical user and command line interfaces (like AutoDock) for performing molecular docking. Here, we show how docking can be performed programmatically with DeepChem, which enables automation and easy integration with machine learning pipelines.

As you work through the tutorial, you'll trace an arc including

1. Loading a protein-ligand complex dataset (PDBbind)
2. Performing programmatic molecular docking
3. Featurizing protein-ligand complexes with interaction fingerprints
4. Fitting a random forest model and predicting binding affinities

To start the tutorial, we'll use a simple pre-processed dataset file that comes in the form of a gzipped file. Each row is a molecular system, and each column represents a different piece of information about that system. For instance, in this example, every row reflects a protein-ligand complex, and the following columns are present: a unique complex identifier; the SMILES string of the ligand; the binding affinity (Ki) of the ligand to the protein in the complex; a Python `list` of all lines in a PDB file for the protein alone; and a Python `list` of all lines in a ligand file for the ligand alone.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5 minutes to run to completion and install your environment.

In [1]:
```
!pip install -q condacolab
import condacolab
condacolab.install()
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the syst
em package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
     Everything looks OK!
```

In [2]:
```
!conda install -c conda-forge openmm
```

```
Collecting package metadata (current_repodata.json): - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / done
Solving environment: \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / done

# All requested packages already installed.
```

In [3]:
```
!pip install deepchem
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: deepchem in /usr/local/lib/python3.10/site-packages (2.7.1)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/site-packages (from deepchem) (1.24.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/site-packages (from deepchem) (1.2.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/site-packages (from deepchem) (1.2.0)
Requirement already satisfied: scipy<1.9 in /usr/local/lib/python3.10/site-packages (from deepchem) (1.8.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/site-packages (from deepchem) (2.0.1)
Requirement already satisfied: rdkit in /usr/local/lib/python3.10/site-packages (from deepchem) (2023.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/site-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/site-packages (from pandas->deepchem) (
2023.3)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/site-packages (from pandas->deepchem)
(2023.3)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/site-packages (from rdkit->deepchem) (9.5.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/site-packages (from scikit-lear
n->deepchem) (3.1.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/site-packages (from python-dateutil>=2.8.2-
>pandas->deepchem) (1.16.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the syst
em package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [4]:
```
!conda install -c conda-forge pdbfixer
```

```
Collecting package metadata (current_repodata.json): - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - done
Solving environment: | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ done

# All requested packages already installed.
```

In [5]:
```
!conda install -c conda-forge vina
```

```
Collecting package metadata (current_repodata.json): - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ done
Solving environment: / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / done

# All requested packages already installed.
```

Protein-ligand complex data

It is really helpful to visualize proteins and ligands when doing docking. Unfortunately, Google Colab doesn't currently support the Jupyter widgets we need to do that visualization. Install `MDTraj` and `nglview` on your local machine to view the protein-ligand complexes we're working with.

In [6]:
```
!pip install -q mdtraj nglview
# !jupyter-nbextension enable nglview --py --sys-prefix  # for jupyter notebook
# !jupyter labextension install  nglview-js-widgets  # for jupyter lab
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the syst
em package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [8]:
```
import os
import numpy as np
import pandas as pd

import tempfile

from rdkit import Chem
from rdkit.Chem import AllChem
import deepchem as dc

from deepchem.utils import download_url, load_from_disk
```

```
Skipped loading modules with pytorch-geometric dependency, missing a dependency. No module named 'torch_geometri
c'
Skipped loading modules with pytorch-geometric dependency, missing a dependency. cannot import name 'DMPNN' from
'deepchem.models.torch_models' (/usr/local/lib/python3.10/site-packages/deepchem/models/torch_models/__init__.py
)
Skipped loading modules with pytorch-lightning dependency, missing a dependency. No module named 'pytorch_lightn
ing'
Skipped loading some Jax models, missing a dependency. No module named 'haiku'
```

To illustrate the docking procedure, here we'll use a csv that contains SMILES strings of ligands as well as PDB files for the ligand and protein targets from PDBbind. Later, we'll use the labels to train a model to predict binding affinities. We'll also show how to download and featurize PDBbind to train a model from scratch.

In [9]:
```
data_dir = dc.utils.get_data_dir()
dataset_file = os.path.join(data_dir, "pdbbind_core_df.csv.gz")

if not os.path.exists(dataset_file):
    print('File does not exist. Downloading file...')
    download_url("https://s3-us-west-1.amazonaws.com/deepchem.io/datasets/pdbbind_core_df.csv.gz")
    print('File downloaded...')

raw_dataset = load_from_disk(dataset_file)
raw_dataset = raw_dataset[['pdb_id', 'smiles', 'label']]
```

Let's see what `raw_dataset` looks like:

In [10]:
```
raw_dataset.head(2)
```

Out[10]:

	pdb_id	smiles	label
0	2d3u	CC1CCCCC1S(O)(O)NC1CC(C2CCC(CN)CC2)SC1C(O)O	6.92
1	3cyx	CC(C)(C)NC(O)C1CC2CCCCC2C[NH+]1CC(O)C(CC1CCCC...	8.00

Fixing PDB files

Next, let's get some PDB protein files for visualization and docking. We'll use the PDB IDs from our `raw_dataset` and download the pdb files directly from the Protein Data Bank using `pdbfixer`. We'll also sanitize the structures with RDKit. This ensures that any problems with the protein and ligand files (non-standard residues, chemical validity, etc.) are corrected. Feel free to modify these cells and pdbids to consider new protein-ligand complexes. We note here that PDB files are complex and human judgement is required to prepare protein structures for docking. DeepChem includes a number of docking utilites to assist you with preparing protein files, but results should be inspected before docking is attempted.

In [11]:
```
from openmm.app import PDBFile
from pdbfixer import PDBFixer

from deepchem.utils.vina_utils import prepare_inputs
```

In [12]:
```
# consider one protein-ligand complex for visualization
pdbid = raw_dataset['pdb_id'].iloc[1]
ligand = raw_dataset['smiles'].iloc[1]
```

In [13]:
```
%%time
fixer = PDBFixer(pdbid=pdbid)
PDBFile.writeFile(fixer.topology, fixer.positions, open('%s.pdb' % (pdbid), 'w'))

p, m = None, None
# fix protein, optimize ligand geometry, and sanitize molecules
try:
    p, m = prepare_inputs('%s.pdb' % (pdbid), ligand)
except:
    print('%s failed PDB fixing' % (pdbid))

if p and m:  # protein and molecule are readable by RDKit
    print(pdbid, p.GetNumAtoms())
    Chem.rdmolfiles.MolToPDBFile(p, '%s.pdb' % (pdbid))
    Chem.rdmolfiles.MolToPDBFile(m, 'ligand_%s.pdb' % (pdbid))
```

```
<timed exec>:7: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
Warning: importing 'simtk.openmm' is deprecated.  Import 'openmm' instead.
3cyx 1510
CPU times: user 2.04 s, sys: 157 ms, total: 2.2 s
Wall time: 4.32 s
```

Visualization

If you're outside of Colab, you can expand these cells and use `MDTraj` and `nglview` to visualize proteins and ligands.

In [14]:
```
import mdtraj as md
import nglview

from IPython.display import display, Image
```

Let's take a look at the first protein ligand pair in our dataset:

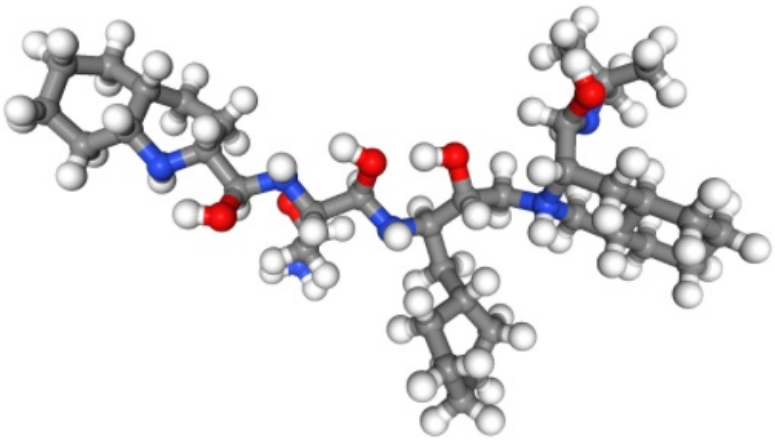
In [15]:
```
protein_mdtraj = md.load_pdb('3cyx.pdb')
ligand_mdtraj = md.load_pdb('ligand_3cyx.pdb')
```

We'll use the convenience function `nglview.show_mdtraj` in order to view our proteins and ligands. Note that this will only work if you uncommented the above cell, installed nglview, and enabled the necessary notebook extensions.

In [16]:
```
v = nglview.show_mdtraj(ligand_mdtraj)
```

In [17]:
```
display(v)  # interactive view outside Colab
```

```
NGLWidget()
```



Now that we have an idea of what the ligand looks like, let's take a look at our protein:

In [18]:
```
view = nglview.show_mdtraj(protein_mdtraj)
display(view)  # interactive view outside Colab
```

```
NGLWidget()
```



Molecular Docking

Ok, now that we've got our data and basic visualization tools up and running, let's see if we can use molecular docking to estimate the binding affinities between our protein ligand systems.

There are three steps to setting up a docking job, and you should experiment with different settings. The three things we need to specify are 1) how to identify binding pockets in the target protein; 2) how to generate poses (geometric configurations) of a ligand in a binding pocket; and 3) how to "score" a pose. Remember, our goal is to identify candidate ligands that strongly interact with a target protein, which is reflected by the score.

DeepChem has a simple built-in method for identifying binding pockets in proteins. It is based on the convex hull method. The method works by creating a 3D polyhedron (convex hull) around a protein structure and identifying the surface atoms of the protein as the ones closest to the convex hull. Some biochemical properties are considered, so the method is not purely geometrical. It has the advantage of having a low computational cost and is good enough for our purposes.

In [19]:
```
finder = dc.dock.binding_pocket.ConvexHullPocketFinder()
pockets = finder.find_pockets('3cyx.pdb')
len(pockets)  # number of identified pockets
```

Out[19]:
```
36
```

Pose generation is quite complex. Luckily, using DeepChem's pose generator will install the AutoDock Vina engine under

the hood, allowing us to get up and running generating poses quickly.

In [20]:
```
vpg = dc.dock.pose_generation.VinaPoseGenerator()
```

We could specify a pose scoring function from `deepchem.dock.pose_scoring`, which includes things like repulsive and hydrophobic interactions and hydrogen bonding. Vina will take care of this, so instead we'll allow Vina to compute scores for poses.

In [7]:
```
!mkdir -p vina_test
```

In [21]:
```
%%time
complexes, scores = vpg.generate_poses(molecular_complex=('3cyx.pdb', 'ligand_3cyx.pdb'),  # protein-ligand fil
                                       out_dir='vina_test',
                                       generate_scores=True
                                      )
```

```
CPU times: user 41min 4s, sys: 21.9 s, total: 41min 26s
Wall time: 28min 32s
```

```
/usr/local/lib/python3.10/site-packages/vina/vina.py:260: DeprecationWarning: `np.int` is a deprecated alias for
the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is
safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If yo
u wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#dep
recations
  self._voxels = np.ceil(np.array(box_size) / self._spacing).astype(np.int)
```

We used the default value for `num_modes` when generating poses, so Vina will return the 9 lowest energy poses it found in units of `kcal/mol`.

In [22]:
```
scores
```

Out[22]:
```
[-9.484, -9.405, -9.195, -9.151, -8.9, -8.696, -8.687, -8.633, -8.557]
```

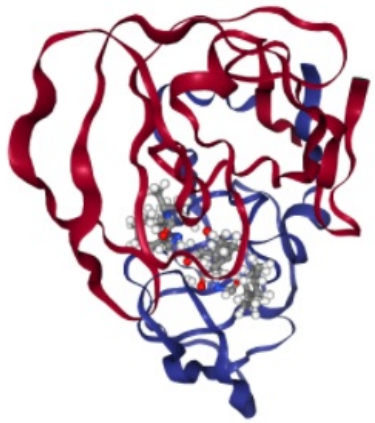
Can we view the complex with both protein and ligand? Yes, but we'll need to combine the molecules into a single RDkit molecule.

In [23]:
```
complex_mol = Chem.CombineMols(complexes[0][0], complexes[0][1])
```

Let's now visualize our complex. We can see that the ligand slots into a pocket of the protein.

In [24]:
```
v = nglview.show_rdkit(complex_mol)
display(v)
```

```
NGLWidget()
```



Now that we understand each piece of the process, we can put it all together using DeepChem's `Docker` class. Docker creates a generator that yields tuples of posed complexes and docking scores.

In [25]:
```
docker = dc.dock.docking.Docker(pose_generator=vpg)
posed_complex, score = next(docker.dock(molecular_complex=('3cyx.pdb', 'ligand_3cyx.pdb'),
                                        use_pose_generator_scores=True))
```

Modeling Binding Affinity

Docking is a useful, albeit coarse-grained tool for predicting protein-ligand binding affinities. However, it takes some time, especially for large-scale virtual screenings where we might be considering different protein targets and thousands of potential ligands. We might naturally ask then, can we train a machine learning model to predict docking scores? Let's try and find out!

We'll show how to download the PDBbind dataset. We can use the loader in MoleculeNet to get the 4852 protein-ligand complexes from the "refined" set or the entire "general" set in PDBbind. For simplicity, we'll stick with the ~100 complexes we've already processed to train our models.

Next, we'll need a way to transform our protein-ligand complexes into representations which can be used by learning algorithms. Ideally, we'd have neural protein-ligand complex fingerprints, but DeepChem doesn't yet have a good learned fingerprint of this sort. We do however have well-tuned manual featurizers that can help us with our challenge here.

We'll make use of two types of fingerprints in the rest of the tutorial, the `CircularFingerprint` and `ContactCircularFingerprint`. DeepChem also has voxelizers and grid descriptors that convert a 3D volume containing an arragment of atoms into a fingerprint. These featurizers are really useful for understanding protein-ligand complexes since they allow us to translate complexes into vectors that can be passed into a simple machine learning algorithm. First, we'll create circular fingerprints. These convert small molecules into a vector of fragments.

In [26]:
```
pdbids = raw_dataset['pdb_id'].values
ligand_smiles = raw_dataset['smiles'].values
```

In [29]:
```
%%time
for (pdbid, ligand) in zip(pdbids, ligand_smiles):
  fixer = PDBFixer(url='https://files.rcsb.org/download/%s.pdb' % (pdbid))
  PDBFile.writeFile(fixer.topology, fixer.positions, open('%s.pdb' % (pdbid), 'w'))

  p, m = None, None
  # skip pdb fixing for speed
  try:
    p, m = prepare_inputs('%s.pdb' % (pdbid), ligand, replace_nonstandard_residues=False,
                          remove_heterogens=False, remove_water=False,
                          add_hydrogens=False)
  except:
    print('%s failed sanitization' % (pdbid))

  if p and m:  # protein and molecule are readable by RDKit
    Chem.rdmolfiles.MolToPDBFile(p, '%s.pdb' % (pdbid))
    Chem.rdmolfiles.MolToPDBFile(m, 'ligand_%s.pdb' % (pdbid))
```

```
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:11:45] UFFTYPER: Unrecognized atom type: S_5+4 (7)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
3cyx failed sanitization
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:02] UFFTYPER: Warning: hybridization set to SP3 for atom 17
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:04] UFFTYPER: Warning: hybridization set to SP3 for atom 6
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:06] UFFTYPER: Warning: hybridization set to SP3 for atom 1
[15:12:06] UFFTYPER: Unrecognized atom type: S_5+4 (21)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:23] UFFTYPER: Warning: hybridization set to SP3 for atom 20
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
```

```
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:31] UFFTYPER: Warning: hybridization set to SP3 for atom 19
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:35] UFFTYPER: Warning: hybridization set to SP3 for atom 29
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:13:03] UFFTYPER: Unrecognized atom type: S_5+4 (39)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:13:37] UFFTYPER: Warning: hybridization set to SP3 for atom 33
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:01] UFFTYPER: Unrecognized atom type: S_5+4 (11)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:02] UFFTYPER: Unrecognized atom type: S_5+4 (47)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:14] UFFTYPER: Unrecognized atom type: S_5+4 (1)
```

```
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:27] UFFTYPER: Warning: hybridization set to SP3 for atom 6
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:33] UFFTYPER: Unrecognized atom type: S_5+4 (47)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:43] UFFTYPER: Unrecognized atom type: S_5+4 (28)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:55] UFFTYPER: Warning: hybridization set to SP3 for atom 17
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:57] UFFTYPER: Warning: hybridization set to SP3 for atom 6
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:08] Explicit valence for atom # 388 O, 3, is greater than permitted
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:15] UFFTYPER: Warning: hybridization set to SP3 for atom 9
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:19] UFFTYPER: Unrecognized atom type: S_5+4 (6)
3utu failed sanitization
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:29] UFFTYPER: Unrecognized atom type: S_5+4 (1)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
```

```
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:39] UFFTYPER: Unrecognized atom type: S_5+4 (19)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:43] UFFTYPER: Unrecognized atom type: S_5+4 (21)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:57] UFFTYPER: Unrecognized atom type: S_5+4 (9)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:16:01] UFFTYPER: Warning: hybridization set to SP3 for atom 18
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:16:21] UFFTYPER: Warning: hybridization set to SP3 for atom 17
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:16:42] UFFTYPER: Warning: hybridization set to SP3 for atom 10
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:19] UFFTYPER: Unrecognized atom type: S_5+4 (13)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
```

```
ction in deepchem.utils.docking_utils.
[15:17:25] UFFTYPER: Unrecognized atom type: S_5+4 (10)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:27] UFFTYPER: Unrecognized atom type: S_5+4 (6)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:28] UFFTYPER: Warning: hybridization set to SP3 for atom 11
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:46] UFFTYPER: Unrecognized atom type: S_5+4 (8)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:58] UFFTYPER: Unrecognized atom type: S_5+4 (4)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:02] UFFTYPER: Unrecognized atom type: S_5+4 (9)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:15] UFFTYPER: Unrecognized atom type: S_5+4 (1)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:32] UFFTYPER: Unrecognized atom type: S_5+4 (23)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:35] UFFTYPER: Unrecognized atom type: S_5+4 (22)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:42] UFFTYPER: Warning: hybridization set to SP3 for atom 8
[15:18:42] UFFTYPER: Warning: hybridization set to SP3 for atom 24
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
```

```
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:01] UFFTYPER: Warning: hybridization set to SP3 for atom 16
[15:19:01] UFFTYPER: Unrecognized atom type: S_5+4 (20)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:02] UFFTYPER: Unrecognized atom type: S_5+4 (6)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:05] UFFTYPER: Unrecognized atom type: S_5+4 (6)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
1hfs failed sanitization
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:22] UFFTYPER: Warning: hybridization set to SP3 for atom 20
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:41] Explicit valence for atom # 1800 C, 5, is greater than permitted
[15:19:41] UFFTYPER: Unrecognized atom type: S_5+4 (11)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:42] UFFTYPER: Warning: hybridization set to SP3 for atom 11
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:57] UFFTYPER: Warning: hybridization set to SP3 for atom 9
[15:19:57] UFFTYPER: Warning: hybridization set to SP3 for atom 23
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 8
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 12
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 34
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 41
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
CPU times: user 4min 9s, sys: 3.31 s, total: 4min 12s
Wall time: 8min 19s
```

In [28]:
```
proteins = [f for f in os.listdir('.') if len(f) == 8 and f.endswith('.pdb')]
ligands = [f for f in os.listdir('.') if f.startswith('ligand') and f.endswith('.pdb')]
```

We'll do some clean up to make sure we have a valid ligand file for every valid protein. The lines here will compare the PDB IDs between the ligand and protein files and remove any proteins that don't have corresponding ligands.

In [30]:
```
# Handle failed sanitizations
failures = set([f[:-4] for f in proteins]) - set([f[7:-4] for f in ligands])
for pdbid in failures:
  proteins.remove(pdbid + '.pdb')
```

In [31]:
```
len(proteins), len(ligands)
```

Out[31]:
```
(190, 190)
```

In [32]:
```
pdbids = [f[:-4] for f in proteins]
small_dataset = raw_dataset[raw_dataset['pdb_id'].isin(pdbids)]
labels = small_dataset.label
```

In [33]:
```
fp_featurizer = dc.feat.CircularFingerprint(size=2048)
```

In [34]:
```
features = fp_featurizer.featurize([Chem.MolFromPDBFile(l) for l in ligands])
```

In [35]:
```
dataset = dc.data.NumpyDataset(X=features, y=labels, ids=pdbids)
train_dataset, test_dataset = dc.splits.RandomSplitter().train_test_split(dataset, seed=42)
```

The convenience loader `dc.molnet.load_pdbbind` will take care of downloading and featurizing the pdbbind dataset under the hood for us. This will take quite a bit of time and compute, so the code to do it is commented out. Uncomment it and grab a cup of coffee if you'd like to featurize all of PDBbind's refined set. Otherwise, you can continue with the small dataset we constructed above.

In [36]:
```
# # Uncomment to featurize all of PDBBind's "refined" set
# pdbbind_tasks, (train_dataset, valid_dataset, test_dataset), transformers = dc.molnet.load_pdbbind(
#     featurizer=fp_featurizer, set_name="refined", reload=True,
#     data_dir='pdbbind_data', save_dir='pdbbind_data')
```

Now, we're ready to do some learning!

To fit a deepchem model, first we instantiate one of the provided (or user-written) model classes. In this case, we have a created a convenience class to wrap around any ML model available in Sci-Kit Learn that can in turn be used to interoperate with deepchem. To instantiate an `SklearnModel`, you will need (a) task_types, (b) model_params, another `dict` as illustrated below, and (c) a `model_instance` defining the type of model you would like to fit, in this case a `RandomForestRegressor`.

In [37]:
```
from sklearn.ensemble import RandomForestRegressor

from deepchem.utils.evaluate import Evaluator
import pandas as pd
```

In [38]:
```
seed = 42 # Set a random seed to get stable results
sklearn_model = RandomForestRegressor(n_estimators=100, max_features='sqrt')
sklearn_model.random_state = seed
model = dc.models.SklearnModel(sklearn_model)
model.fit(train_dataset)
```

Note that the
R^2
value for the test set indicates that the model isn't producing meaningful outputs. It turns out that predicting binding affinities is *hard*. This tutorial isn't meant to show how to create a state-of-the-art model for predicting binding affinities, but it gives you the tools to generate your own datasets with molecular docking, featurize complexes, and train models.

In [39]:
```
# use Pearson correlation so metrics are > 0
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)

evaluator = Evaluator(model, train_dataset, [])
train_r2score = evaluator.compute_model_performance([metric])
print("RF Train set R^2 %f" % (train_r2score["pearson_r2_score"]))

evaluator = Evaluator(model, test_dataset, [])
test_r2score = evaluator.compute_model_performance([metric])
print("RF Test set R^2 %f" % (test_r2score["pearson_r2_score"]))
```

```
RF Train set R^2 0.888697
RF Test set R^2 0.007797
```

We're using a very small dataset and an overly simplistic representation, so it's no surprise that the test set performance is quite bad.

In [40]:
```
# Compare predicted and true values
list(zip(model.predict(train_dataset), train_dataset.y))[:5]
```

Out[40]:
```
[(6.862549999999994, 7.4),
 (6.616400000000008, 6.85),
 (4.852004999999995, 3.4),
 (6.43060000000001, 6.72),
 (8.66322999999999, 11.06)]
```

In [41]:
```
list(zip(model.predict(test_dataset), test_dataset.y))[:5]
```

Out[41]:
```
[(5.960549999999999, 4.21),
 (6.051305714285715, 8.7),
 (5.799900000000003, 6.39),
 (6.433881666666665, 4.94),
 (6.7465399999999995, 9.21)]
```

The protein-ligand complex view.

In the previous section, we featurized only the ligand. This time, let's see if we can do something sensible with our protein-ligand fingerprints that make use of our structural information. To start with, we need to re-featurize the dataset but using the contact fingerprint this time.

In [42]:
```
fp_featurizer = dc.feat.ContactCircularFingerprint(size=2048)
```

In [43]:
```
features = fp_featurizer.featurize(zip(ligands, proteins))
dataset = dc.data.NumpyDataset(X=features, y=labels, ids=pdbids)
train_dataset, test_dataset = dc.splits.RandomSplitter().train_test_split(dataset, seed=42)
```

```
[15:21:40] Explicit valence for atom # 3 C, 5, is greater than permitted
Mol [H]OC([H])([H])[C@]1([H])O[C@@]2([H])SC([H])([H])(N([H])C([H])([H])C([H])([H])[H])N([H])[C@@]2([H])[C@@]([H]
)(O[H])[C@]1([H])O[H] failed sanitization
[15:21:47] Explicit valence for atom # 214 O, 3, is greater than permitted
Mol CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CN
C3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H
]2CCCN2C(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2
CCCN2C(=O)CNC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=
O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)
NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@@H]2CSSC[C@H](NC(=O
)[C@@H](NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]3CCCN
3C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[
C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(N)=O)NC(=O)CN)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)C(
=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@
H]([C@@H](C)CC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@@H](CC(=O)O)C(=
O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N2)CSSC[C@@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@
H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)NCC(=O)N[C@@H](CC
2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C
2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CNC3=C2C=CC
=C3)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@
H](CCC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N2C3C(=O)O32)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)CNC
(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@H](C(C)C)NC1=O.CC[C@H](C)[C@H](
NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@]12C(CC(=O)O)C13C(C)[C@H](
N)C(=O)N32)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H]
(CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(
=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C
)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=C(O)C=C2
)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[
C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=
O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](
CCCCN)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=
O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C
@H](C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC3=CNC=N3)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCCNC(=N)N)C(=
O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@
@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](
CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N3CCC[C@H]3C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)
N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC3=CNC=N3)C(=O)N3CCC[C@H]
3C(=O)N[C@H](C(=O)N[C@@H](CSSC[C@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)N)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](
CCCNC(=N)N)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCC
N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H]
(CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)
N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@]34C(=O)O3C4C3=CC=C(O)C=C3)[C@@H](C)O)C(=O)N[C@@H
](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C
@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H]
(C)C(=O)NCC(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H]
(C(=O)NCC(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)
N[C@H](C=O)CCC(=O)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@
H](C)CC)[C@@H](C)CC)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC2=O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H](CC2=CN
C=N2)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC2=CNC
3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)CNC1=O)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O
)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](N)CC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@
H](CO)CC1CCC(OS(O)(O)O)CC1.CO[C@H]1CC[C@H](S(O)(O)N[C@@H](C[C@@H](O)NC[C@H]2CC[C@H](CN)CC2)C(O)N2CCC[C@H]2[C@H](
O)NCC2CCC(C(N)N)CC2)CC1Cl.O.O.O
```

[15:22:27] Explicit valence for atom # 5358 O, 3, is greater than permitted
Mol CC(O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H]1CC(O)=OO=C([C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CNC=N2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCNC(=N)N)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC

(CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]1CCCN1
C(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)N
C(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@
H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C
@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](C)NC
(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](
CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](N)C(C)C)C(C)
C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H]
(C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C
(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[
C@@H](C)CC)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@
H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=
O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)
O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O
)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(
=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N1CCC[C@H]1C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H
](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)
C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)
C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C
=C1)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(
=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O
)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C
=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@
@H](C)C=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C.CC[C@H](C)[C@H](NC(=O)CN
C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=
C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=
O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO
)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)N)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CC1=CC=C(O
)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O
)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H]
(CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C
@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(
=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H
](CCCNC(=N)N)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N
[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C
@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N
[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)
C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)
N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H]
(C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C
@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC
(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC
NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)N)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(=O)N[C@@
H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](
CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C
(=O)N1CCC[C@H]1C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@
H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)
O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1
=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=
O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC
C(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@
H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CC1

0.0.0.

NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC

O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]
](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCC
N)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(
=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](
C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(
=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@
H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)N
C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCC
N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=
C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)
CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@
H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](
CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(
=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(
=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@
H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C
@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H
](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](CCCCN)NC(=O)[C@@H](NC(=
O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C
CC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)C
NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](NC(=O)
)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1
=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O
)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[
C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H
](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)CNC(=O)[C
@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)
NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[
C@@H](NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(
=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C
@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C
C(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](
CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)
[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H
]1CCCN1C(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN
1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](N)CCC(N)=O)C(C)C)C(C)C)[C@@H](C
)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@
H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)
[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C
@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C
@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@
H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](
C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@
H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)O.Cl.Cl.Cl.Cl.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0

[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C

(=O)O)NC(=O)[C@H](CCCCN)NC1=O)C(C)C.N=C(N)NCCCCC(=O)O.O.O.

1C=CC=C2)C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1CC1N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1OC1=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H]1CCCNC2=[N+](N2)O=C(N)CC[C@@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)NC1=O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C1CCC1N)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC(C)

)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(
=O)CNC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C
)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CC
(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](
CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=
CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H
](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O
)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCC
NC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=
O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H
](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H]
(CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@
@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[
C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](C
H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC
(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CN)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](
C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@
@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CCC
NC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=C
C=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N
)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C
(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C
)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=
N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@
H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C
CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C
@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC
(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCC
N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C
(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C
@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C
=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N1CCC[C
@H]1C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C
(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)
C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N
[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)
C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)
C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CCSC)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H]
(CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]
(CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N
[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)
O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CO)C(=O)N[C@@
H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)
N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)
C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=
CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C
(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=CC=
C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O
)O)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)
N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(N)=
O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H]
(CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@
H](C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)

=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H]1CCC(=O)NC(N)CCC[C@@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC2N3C(N)=[N+]23)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC2=OO2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CS)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCSC)NC1=O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC

NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](

@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC

(NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C
C1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C
C1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N
1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC
(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1C
CCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O
)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](
CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](
NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(
=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)N
C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C
(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](
C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C
=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@
@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C
@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C
@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO
)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=
O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC
(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C
@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)N
C(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C
@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O

C(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O

O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)

[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C

CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)
CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C
(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C
C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O
)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O
)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC
C(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)
NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC
(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=
O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=
O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=
O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)N
C(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(
=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC
NC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)
[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C
@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](N)CO)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=
O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O
)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N
[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]
(NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C
C1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C
C1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N
1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC
(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1C
CCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O
)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](
CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](
NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(
=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O
)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](N)CO)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H
](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)
C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H
](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N
[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.0.0.0

0.0.0.

C2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC2=OO2)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H]2CC(O)=ON2C(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](N)CCCCN)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)CCC(=O)ONCCCC[C@@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N2CCC[C@H]2C(=O)N[C@H]2CSSC[C@@H](C=O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H]3CCCN3C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(=O)O)NC2=O)[C@@H](C)CC)[C@@H](C)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C(O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)C1=O.CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CO)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O

C(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC1=O.CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC2=NN2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H]2CC(O)=O3N2C3(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]3CCCN3C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C4)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)CNC(=O)[C@H](CS)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CS)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]3CCCN3C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CC(N)=O)NC(=O)C(CCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@]3(N)CC3CCN)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)CC2C(N)=O)C2CC(=O)O2)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)CSSC[C@@H](C=O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC1=O.C[C@@H]1ONC(O)[C@@H]1C[C@H](N)C(O)O.C[C@@H]1ONC(O)[C@@H]1C[C@H](N)C(O)O.C[C@H]1ONC(O)[C@@H]1C[C@H](N)C(O)O.0.0.0

CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@H]3O[C@H](CO)[C@@H](O)[C@H](O)[C@H]3NC(C)O)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC3=ON3)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H]3CCCN3C(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H]3CCCN3C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)C(C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC2=OO2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N1)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)C(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CSSC[C@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCN1NC1=N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](N)CO)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NC1=O)C(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(=O)N[C@@H]1O[C@H](CO)[C@@H](O)[C@H](O)[C@H]1NC(C)O)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(=O)N[C@@H]1O[C@H](CO)[C@@H](O)[C@H](O)[C@H]1NC(C)O)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC

)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCSC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C0)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)NCC(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CS)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CS)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CS)C(=O)N[C@@H](C)C(=O)N[C@H](

@H](CCCNC(=N)N)C(=O)O)[C@@H](C)O)C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC3=CNC=N3)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H]4CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H]4CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H]4CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)

@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N3CCC[C@H]3C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](CC(N)=O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C3CN3C)C(=O)N[C@@H](C3C4C(=O)O43)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N2)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)C(=O)N[C@H]2CC(N)=ONC(=O)C[C@H](NC(=O)[C@H](CC(C)C)NC2=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2OC2=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N1)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1CC=CC=C1)C(=O)N[C@@H](CCS)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@]1(C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=C(O)C=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H]2CSSC[C@@H](C(=O)N[C@H](C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C3=OC4567=O89%10N(C(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)N3)[C@@]8([C@@H](C)O)C9(=O)[N+]%104[C@H]5C607)C(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(=O)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H]([C@@H](C)O)NC(=O)CNC(=O)[C@H](CC(=O)N[C@@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)NCC(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O



(C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N
[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](
CC2=CNC=N2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2
C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](C
C2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(
=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[
C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=
O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC
(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)NCC(=O)N[C@@H](C)
C(=O)N[C@@H](CS)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N2CCC[C@H]2C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(
=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CNC=N2)C(=O)NCC(=O)
N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O
)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C
@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C
O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O
)N[C@H](C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC2
=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@
@H](CCCNC(=N)N)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N
[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(
=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](C
)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@
H](CC(C)C)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)
C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)NCC(=O)N
[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=
C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=
CNC=N2)C(=O)N[C@@]23C(=O)O24OC34)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)
C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@
@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[
C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)O)NC(=O)[C@H
](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCCN
C(=N)N)NC1=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O.CC[C@H](C)[C@H]
(NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C
@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
](CCCCN)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O
)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H]1CCCN1C(=O)[C@@H]1CC
CN1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H
](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(
=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H
](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=
O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC
(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H]
(CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(
N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)
[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O
)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O
)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C
C(C)C)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)
NC(=O)[C@H](CO)NC(=O)[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O
)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)C
(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@H](C=O)CCC(N)=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C
@@H](C)O)C(C)C)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C
@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=
N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CN
C(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C
@H](CC1=CNC=N1)NC(=O)[C@@H]1CCCN1C(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
S)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@
H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)
NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)
[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](
(C)C)C(=O)N[C@@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC2=C1C=
CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)
N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H]
(C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C(

CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@H](C=O)CCC(N)=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C.O.O.O



)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)
[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@
@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)
O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)
[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)O.CC[C@H](C)[C@H](NC(=O)[C@H](CC(C)C)
NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)N
C(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=
O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](N
C(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](
C)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(N)=O
)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(
=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
(CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC=ON)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=
O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
C(=O)O)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CO)NC(=O)[C@H](CS)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](N
C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=
O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)
O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](N
C(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]
(CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=C
NC2=C1C=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(
=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[
C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)N
C(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
(CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](
C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=
O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)N
C(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CNC2=
C1C=CC=C2)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[
C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC
(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](
C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)N
C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCC(=O)O)
NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[
C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](
CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC
1=CNC=N1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H
](CCCCN)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC
=C2)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C
CCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@
H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)
)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)
NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)CNC(
=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)CNC(
=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)

NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(
=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C
@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C
@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C
@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)
=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC
(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)
[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)
NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@@
H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)
[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[
C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](
NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](N)CCC(N)=O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)
O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C
)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C
(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@
H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@
H](C)CC)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@
H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)
[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@
@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)
C)[C@@H](C)CC)[C@@H](C)O)C(=O)O.Cl.Cl.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.
O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O

1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC
(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H
](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[
C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC
C(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@
H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H
](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O
)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H
](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C
@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(
=O)[C@@H](N)CC(N)=O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](
C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C
@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C
@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@H](
C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C
(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O
)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)
C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C
(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(N
)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O
)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(
=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C
@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C
@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H]
(C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[
C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](CC1=CNC2=C1C
=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@
@H](C)C(=O)N[C@@H](CS)C(=O)NCC(=O)N1O=C(O)C[C@H]1C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)
C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O
)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC
N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@
H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H
](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N
[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)
N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(
=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC
CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CC

)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(
=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@
H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](
C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(
=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N
)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)
NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(
N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=
C2)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)
N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=
O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C
@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H
](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC
(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(
=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C
@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)
N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(
N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCC
N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H
](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC
C(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H]
(C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H]
(CS)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@]1(C(=O)N[C@@H](CC(
C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N2CCC[C@H]2
C(=O)N[C@@H](CC2O=C2O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H]2CCCNC(=N)NNC(=O)C
C[C@@H](C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N3CCC[C@H]3C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H]
(CO)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC3C4CN34)C(=O)N[C@@H](CCC
(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@
@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H
](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC3=C
C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)
C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)
C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)
C)C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCN3C4=NN43)C(=O)N[C@@H](CCCCN)
C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N3CCC[C@H]3C(=O)N[C@@H](
CCCNC3=NN3)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[
C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CN
C=N3)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C
@@H](CCCCN)C(=O)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C3CCCN3)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)C
C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]
(CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC
=C3)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)
[C@H](C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC
(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC
CCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CO)NC2=O)[C@@H](C)O)CC1CCN)C(C)C)[C@@H](C)O)C(C)C)C(
C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@
@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)C)
C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H]
(NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[
C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)C(C)C)[C@@H](C)CC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](
C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=C
NC=N1)C(=O)O)[C@@H](C)O.CC[C@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCN1N=C1N)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H
]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=CC=C
1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=
O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=
O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](C
C(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC
(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CS)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N
C(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C
@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H]
(CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC
CNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CO)NC
(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)CNC(=O)[C@H](CO)
NC(=O)[C@@H]1CCCC2NNC2(=O)CC[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H
](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](
CC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]
(NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)[C@
H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=C
NC=N2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(N)=O)NC(=
O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O
)[C@H](CO)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](C
C2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(N)=O
)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](C
CSC)NC(=O)[C@H](CS)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O
)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C
C(C)C)NC(=O)CNC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)CN

C(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN[C@@](CO)(NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]3CCCN3C(=O)[C@@H](NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC3CN3)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)CNC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CS)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CCSC)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)CNC(=O)CNC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)CNC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](

CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)N
C(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(
=O)[C@@H](NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=C(O)C=C
3)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)
=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H
](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC3=CC=C(
O)C=C3)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC(C)C)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@
H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H]3CCCN3C(=O)[
C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCSC)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC
CCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC3=
CC=CC=C3)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
(CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)
[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(N)=O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C
@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C

=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)

```
](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC
(=O)CNC(=O)CNC(=O)[C@@H]4CCCN4C(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](CC4=CC=C(O)C=C4)
NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)
C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CC4=CNC5=C4C=CC=C5)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC4=CC=CC=C4)
NC(=O)[C@@H](NC(=O)[C@@H]4CSSC[C@H](NC(=O)[C@@H](NC(=O)[C@H](CC5=CNC=N5)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC5=C
C=C(O)C=C5)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=
O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC5=CC=C(O)C=C5)NC(=O)[C@H](CC5=CC=CC=C5)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)
=O)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC5=CC=CC=C5)NC(=O)[C@H](CC5=CNC=N5)NC
(=O)[C@@H]5CCCN5C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H]5CCCN5C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC5=CC=CC
=C5)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC5=CNC6=C5C=CC=C6)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](N
C(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC5=CNC=N5)NC(=O)[C@H](CCC(N)=O)NC(
=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]5CSSC[C@H](NC(=O)[C@@H](NC(=O)[C@H
](CC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@
H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CNC7=C
6C=CC=C7)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](
CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H
](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CS)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC6=CNC=N6)NC(=O
)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@@H]
6CCCN6C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O
)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C
@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=CC=C6)N
C(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC(N)=O)NC(=O)[
C@@H](NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C
@@H]6CCCN6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)
[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC
(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)
C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC
(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H]6CCCN
6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC6
=CC=C(O)C=C6)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(
=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=
O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7
)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=
C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[
C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](
(CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CNC=N6)NC(=
O)CNC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O
)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=
O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@@H]6O[C@H](CO)[C@@H](O[C@@H]7O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)N6O=C6[C@H](CC(N)=O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CNC=N6)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)N[C@@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)[C@@H]6CCCN6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)C(=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC6=CC=C(O)C=C6)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC6=CNC7=C6C=CC=C7)C(=O)N[C@@H](CC(N)=O)C(=O)N5)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N4)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N2)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)CSSC[C@@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)[N+]23CCC[C@H]2C(=O)O3)
NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CNC=N2)N
```

C(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC1=O.CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CN

(CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CO)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CNC=N6)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)N[C@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)[C@@H]6CCCN6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)CNC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)CNC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C

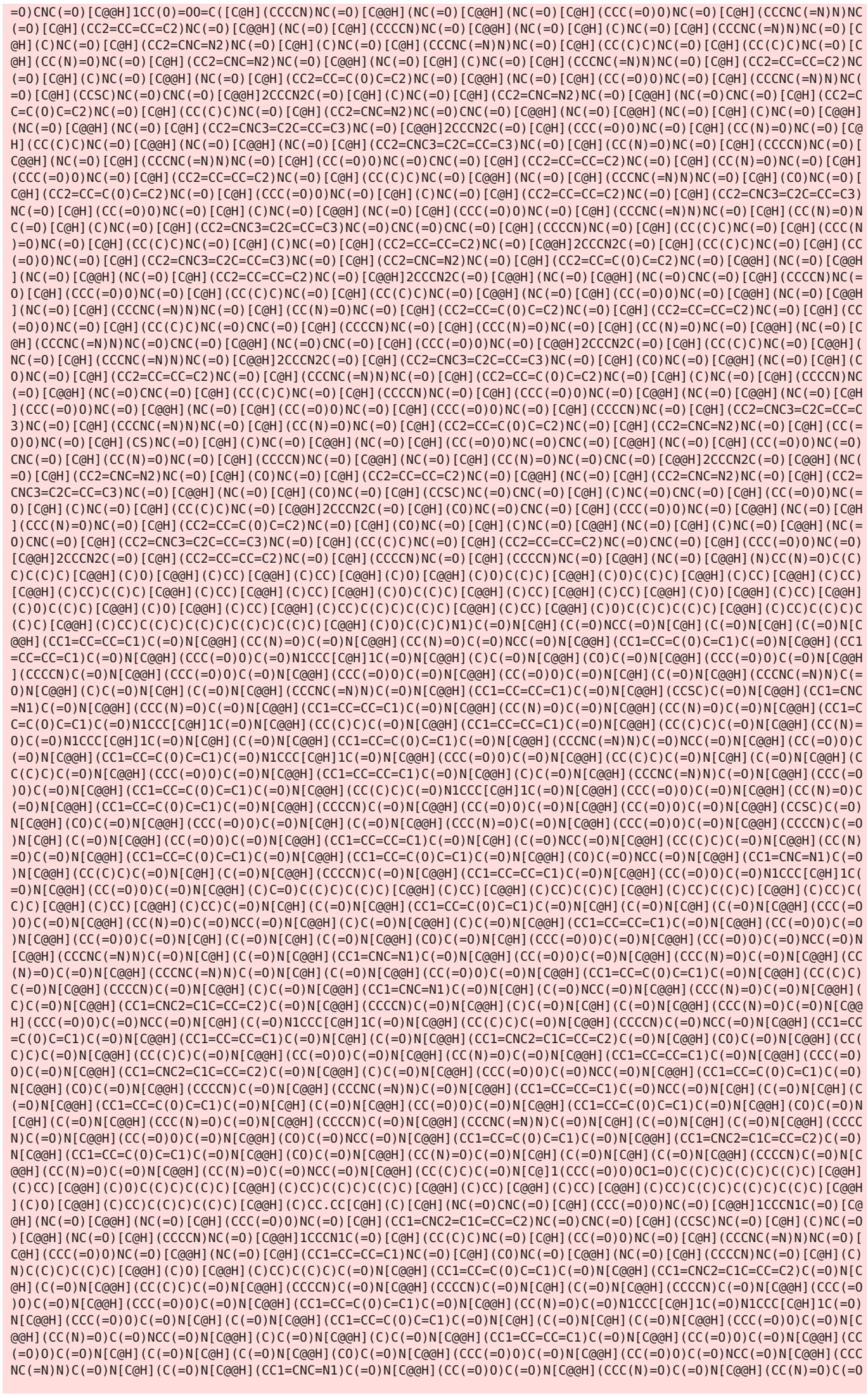
C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@H](

CNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](
CS)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C
1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=
O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=C
C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](
CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(
=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O
)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C
(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=
O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=
O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](
CCC(=O)O)C(=O)N[C@H]1CCC(=O)N2NC2CCC[C@@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)
C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H
](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC
CN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC
[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC2=ON2
)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H]
(CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC
2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC
CN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O
)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H]
(CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H
](CC2=CC=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](
CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)
N[C@H](C(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H]2C3(C4C5CN54)O=C23N[C@@H]
(CCC2CN2)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O
)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O
)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H]
(CC2=CNC=N2)C(=O)N[C@@H](CC2=CNC=N2)C2=OO2)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H
](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(
C)C)C(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C)NC
(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC
CNC(=N)N)NC(=O)[C@H](CS)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O
)[C@H](C)NC(=O)[C@H](CCCN2N=C2N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)C)C)NC(=O)[C@H](CS)NC(=O)[C@H](CCCCN)N
C(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C(C)C)NC(=O)
[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(N)=O)N
C(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=
O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)[C
@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=
O)[C@H](CCSC)NC1=O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O
)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@
H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@
H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O
)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O
)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)
CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C).CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC1=CNC2=C1
C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(
=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)
NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN
)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=C
C=CC=C1)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(N)=O
)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O
)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@
@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O
)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC
CCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@@H
](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC
CCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)
O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1
)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)
[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC
(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](
CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=
O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC
(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[
C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)
[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)
=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)
[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC
(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=
O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC
(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[
C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)
[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=C
C=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](
CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](
CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)
NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](N)CC(N)=O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)
[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C
(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](
CCSC)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC
NC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=
CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)

N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@H]1CCC(=O)ONCCCC[C@@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCSC)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)NCC(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)

O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC2=OO2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C=O)CCCCN)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C23C45CN426NC536=N)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C(C)C)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H]1CCC(=O)OO=C(O)CC[C@H](N)C(=O)N1)C(C)C)[C@@H](C)CC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCCN)C(=O)O)[C@@H](C)O.Cl.Cl.Cl.Cl.Cl.0.0.0

)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)



)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@]12C(=O)O1C2CC(=O)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)C(=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](CC(=O)O)C(=O)N1CCC[C@H]1C=O)C(C)C)C(C)C)[C@@H](C)CC.O.O.O

[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O.CC[C@H](C)[C@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)CNC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CS)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CO)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CO)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](

@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](N)CCCCN)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C1=O[C@@](CO)(C2=OO2)N1)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CS)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CS)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@]12O=C13O1C2013)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C.CO[C@H](C1CCCCC1)C(O)N1CC2NNC(NC(O)C3CCC(N4CCN(C)CC4)CC3)C2C1.CO[C@H](C1CCCCC1)C(O)N1CC2NNC(NC(O)C3CCC(N4CCN(C)CC4)CC3)C2C1.0.0.0

=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CSSC[C@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@@H](N)CO)C(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](C(C)C)C(=O)NCC(=O)N2CCC[C@H]2C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H]([C@@H](C)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N2CCC[C@H]2C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)

2CNC3C(N)NCNC32)[C@H](O)[C@@H]1O.C[S@@H](CC[C@H](N)C(O)O)C[C@H]1O[C@@H](N2CNC3C(N)NCNC32)[C@H](O)[C@@H]1O.O.O.O.
O.O.O.O.

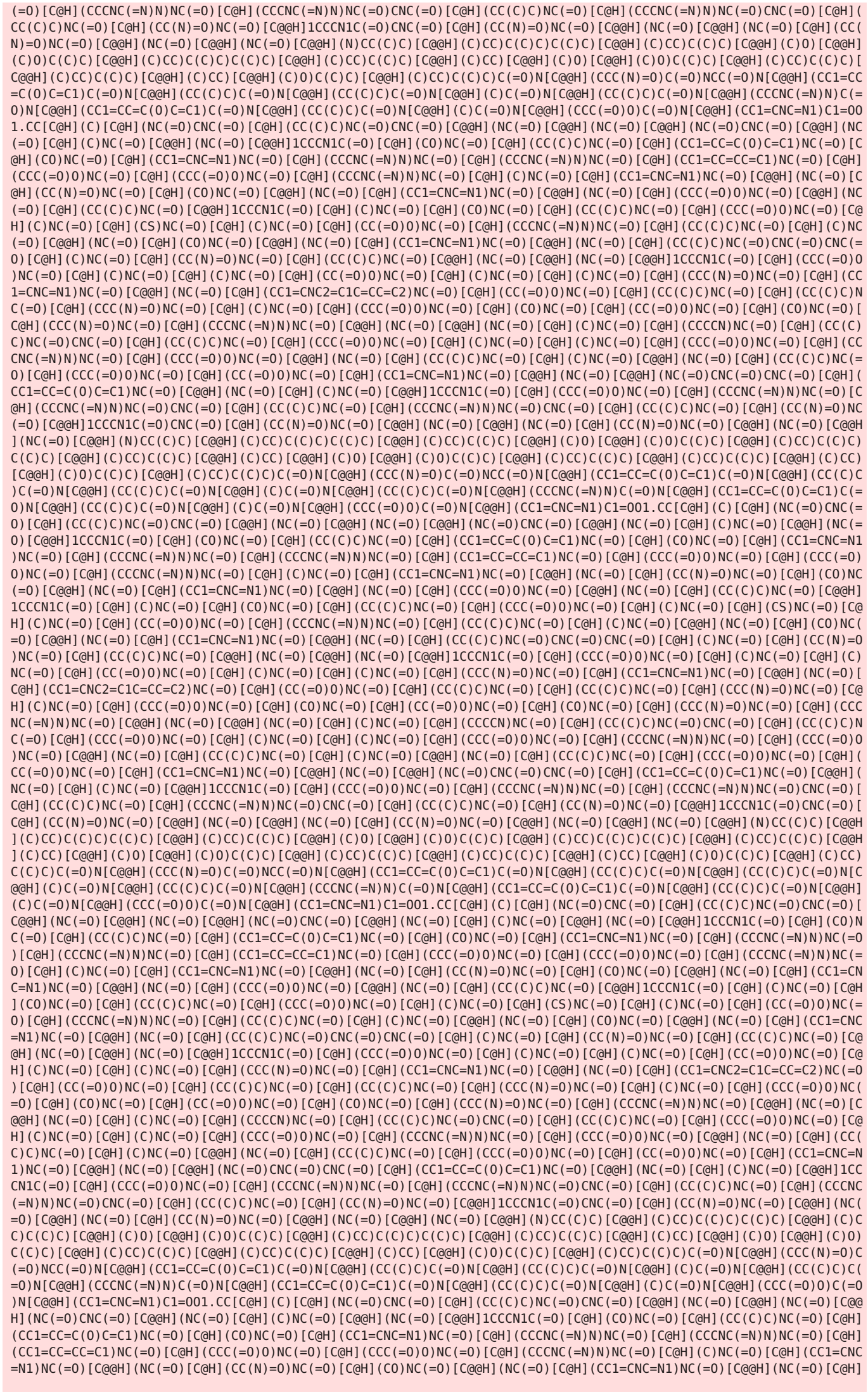
0.0.0.

N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CC
CN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H
](N)CC(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](
C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(
C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]
(CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(
C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)
C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN
1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](
CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]
(CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC
(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C
@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C
@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(
=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](
CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C
C(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2
=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C
@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O
)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC
C(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]
(NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O
)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C
)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(N)=O)
NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)
C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H]
(C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[
C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N
[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@
H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[
C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=
CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC
(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO
)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)
NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)
N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)
O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC
(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=
O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO
)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)
NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)



NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CC=CC=C1)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](C)NC(=O)[C@@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](CC(N)=O)NC(=O)[C@@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)C(=O)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](C)NC(=O)[C@@H](CCCCN)NC(=O)[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H](CC(N)=O)NC(=O)[C@@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H](CC(N)=O)NC(=O)[C@@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](CC1=CC=C(O)C=C1

```
](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O
)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](
C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)C
NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(
=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H
](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(
=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
C(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H
](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)
NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C
@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(
C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC
(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)O.CC[C@H](C)[C@H](NC(=O)C
)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)C
NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](
NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CN
C=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C
O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C
@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O
)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO
)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(
N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H
](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(
=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O
)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H]
(CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(
=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@
@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(
=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(
=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[
C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1
=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)N
C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C
@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C
@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)
[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H
](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](C
C(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)
[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)
[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[
C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@
@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC
```



(CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC

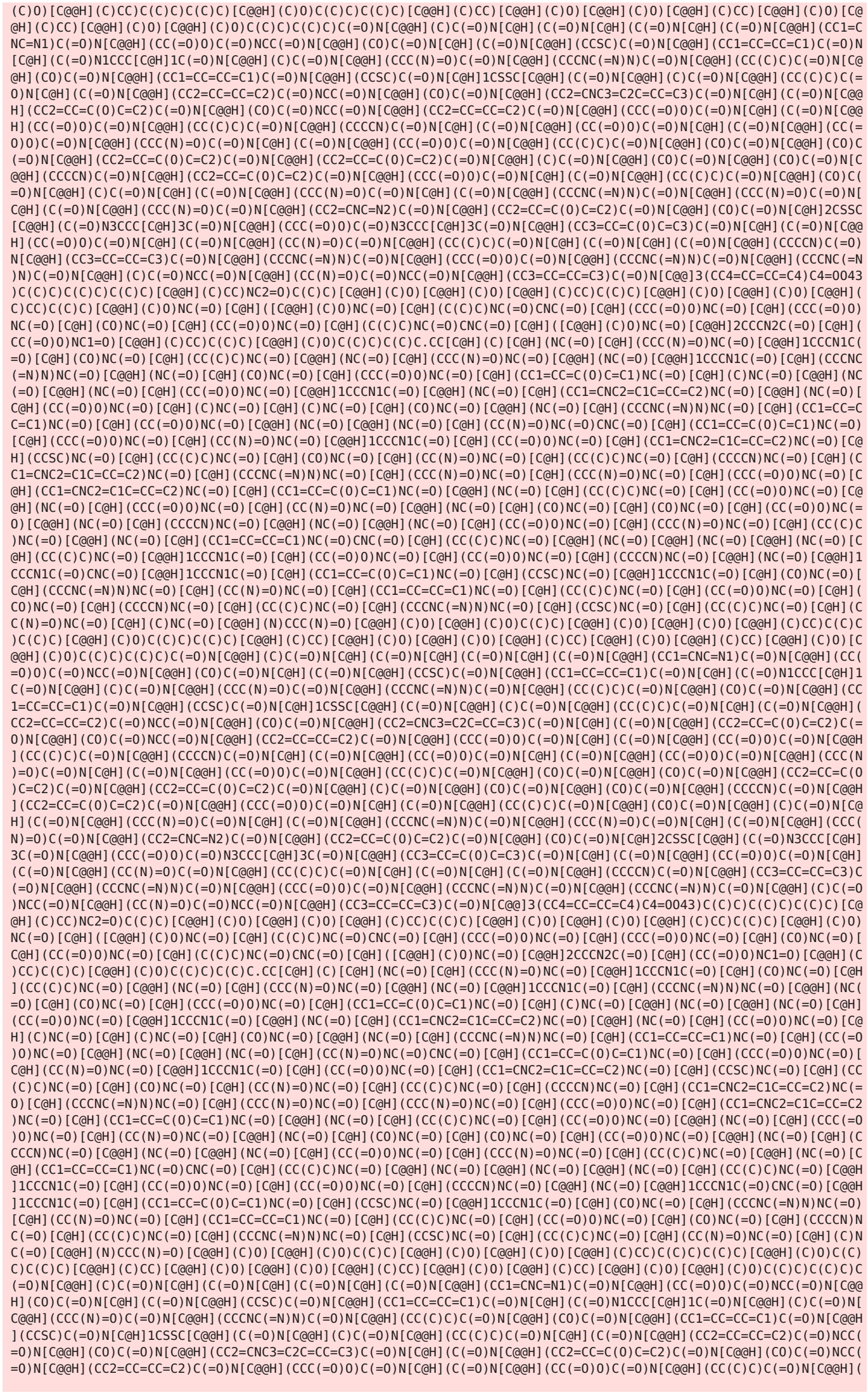
.0.0.0

CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@]34CO3C4=O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](C)O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2

](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO

C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN

C(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(=O)N[C@@H](C)C(=O)N[C@H]([C@@H](C)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)C1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H]




CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N
[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2
=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O
)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)
=O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=
CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O
)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)
C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N
)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC(N)=
O)C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[
C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O
)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@
H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C
(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](
NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H
](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1
CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(
=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](NC(=O)
[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@
@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC
(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(
=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(
O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O
)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=
O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O
)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=
O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=
CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C
@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O
)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C
@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C
@H](C(=O)N[C@H](C(=O)N[C@@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@H](C(=O
)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[
C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSS
C[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C
@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C
2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O
)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[
C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@
@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O
)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[
C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC
2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)
N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(
=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O
)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](
CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)NC2=O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(
=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)
[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[
C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)N
C(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]
(CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O
)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@
H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(
N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC
(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=
O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC
(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C
@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@
@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)
NC(=O)[C@@H](NC(=O)[C@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H]1CS
SC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C
@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C
2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O
)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[
C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@
@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O
)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[
C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C
@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C
3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)NC(=O)N[C@H](C(=O)N[C@H](

C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.0.0.0

.0.0.0

```
CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@]1([C@@H](C)O)OC1=O)C(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(=O)N[C@@H]1O[C@H](CO)[C@@H](O[C@H]2O[C@H](CO)[C@@H](O)[C@H](O)[C@H]2NC(C)O)[C@H](O)[C@H]1NC(C)O)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1C2=OO21)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N1CCC[C@H]1C=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC.0.0.0
```

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Modeling Protein-Ligand Interactions with Atomic Convolutions

By Nathan C. Frey | Twitter and Bharath Ramsundar | Twitter

This DeepChem tutorial introduces the Atomic Convolutional Neural Network. We'll see the structure of the `AtomicConvModel` and write a simple program to run Atomic Convolutions.

ACNN Architecture

ACNN's directly exploit the local three-dimensional structure of molecules to hierarchically learn more complex chemical features by optimizing both the model and featurization simultaneously in an end-to-end fashion.

The atom type convolution makes use of a neighbor-listed distance matrix to extract features encoding local chemical environments from an input representation (Cartesian atomic coordinates) that does not necessarily contain spatial locality. The following methods are used to build the ACNN architecture:

- **Distance Matrix**
  The distance matrix R is constructed from the Cartesian atomic coordinates X. It calculates distances from the distance tensor D. The distance matrix construction accepts as input a $(N, 3)$ coordinate matrix C. This matrix is "neighbor listed" into a (N, M) matrix R.

```
R = tf.reduce_sum(tf.multiply(D, D), 3)     # D: Distance Tensor
    R = tf.sqrt(R)                              # R: Distance Matrix
    return R
```

- **Atom type convolution**
  The output of the atom type convolution is constructed from the distance matrix R and atomic number matrix Z. The matrix R is fed into a (1x1) filter with stride 1 and depth of N_{at}, where N_{at} is the number of unique atomic numbers (atom types) present in the molecular system. The atom type convolution kernel is a step function that operates on the neighbor distance matrix R.

- **Radial Pooling layer**
  Radial Pooling is basically a dimensionality reduction process that down-samples the output of the atom type convolutions. The reduction process prevents overfitting by providing an abstracted form of representation through feature binning, as well as reducing the number of parameters learned. Mathematically, radial pooling layers pool over tensor slices (receptive fields) of size (1x M x1) with stride 1 and a depth of N_r, where N_r is the number of desired radial filters and M is the maximum number of neighbors.

- **Atomistic fully connected network**
  Atomic Convolution layers are stacked by feeding the flattened (

N
,
N_{at}
,
N_r
) output of the radial pooling layer into the atom type convolution operation. Finally, we feed the tensor row-wise (per-atom) into a fully-connected network. The same fully connected weights and biases are used for each atom in a given molecule.

Now that we have seen the structural overview of ACNNs, we'll try to get deeper into the model and see how we can train it and what we expect as the output.

For the training, we will use the publicly available PDBbind dataset. In this example, every row reflects a protein-ligand complex and the target is the binding affinity (K_i) of the ligand to the protein in the complex.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5 minutes to run to completion and install your environment.

In [ ]:
```
!pip install -q condacolab
import condacolab
condacolab.install()
!/usr/local/bin/conda info -e
```

In [ ]:
```
!/usr/local/bin/conda install -c conda-forge pycosat mdtraj pdbfixer openmm -y -q  # needed for AtomicConvs
```

In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

In [ ]:
```
import deepchem as dc
import os

import numpy as np
import tensorflow as tf

import matplotlib.pyplot as plt

from rdkit import Chem

from deepchem.molnet import load_pdbbind
from deepchem.models import AtomicConvModel
from deepchem.feat import AtomicConvFeaturizer
```

Getting protein-ligand data

If you worked through Tutorial 13 on modeling protein-ligand interactions, you'll already be familiar with how to obtain a set of data from PDBbind for training our model. Since we explored molecular complexes in detail in the previous tutorial), this time we'll simply initialize an `AtomicConvFeaturizer` and load the PDBbind dataset directly using MolNet.

In [ ]:
```
f1_num_atoms = 100  # maximum number of atoms to consider in the ligand
f2_num_atoms = 1000  # maximum number of atoms to consider in the protein
max_num_neighbors = 12  # maximum number of spatial neighbors for an atom

acf = AtomicConvFeaturizer(frag1_num_atoms=f1_num_atoms,
                      frag2_num_atoms=f2_num_atoms,
                      complex_num_atoms=f1_num_atoms+f2_num_atoms,
                      max_num_neighbors=max_num_neighbors,
                      neighbor_cutoff=4)
```

`load_pdbbind` allows us to specify if we want to use the entire protein or only the binding pocket (`pocket=True`) for featurization. Using only the pocket saves memory and speeds up the featurization. We can also use the "core" dataset

of ~200 high-quality complexes for rapidly testing our model, or the larger "refined" set of nearly 5000 complexes for more datapoints and more robust training/validation. On Colab, it takes only a minute to featurize the core PDBbind set! This is pretty incredible, and it means you can quickly experiment with different featurizations and model architectures.

In [ ]:
```
%%time
tasks, datasets, transformers = load_pdbbind(featurizer=acf,
                                             save_dir='.',
                                             data_dir='.',
                                             pocket=True,
                                             reload=False,
                                             set_name='core')
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
CPU times: user 43.2 s, sys: 18.6 s, total: 1min 1s
Wall time: 1min 10s
```

Unfortunately, if you try to use the "refined" dataset, there are some complexes that cannot be featurized. To resolve this issue, rather than increasing `complex_num_atoms`, simply omit the lines of the dataset that have an `x` value of `None`

In [ ]:
```
class MyTransformer(dc.trans.Transformer):
  def transform_array(x, y, w, ids):
    kept_rows = x != None
    return x[kept_rows], y[kept_rows], w[kept_rows], ids[kept_rows],

datasets = [d.transform(MyTransformer) for d in datasets]
```

In [ ]:
```
datasets
```

Out[ ]:
```
(<DiskDataset X.shape: (154, 9), y.shape: (154,), w.shape: (154,), ids: ['1mq6' '3pe2' '2wtv' ... '3f3c' '4gqq'
'2x00'], task_names: [0]>,
 <DiskDataset X.shape: (19, 9), y.shape: (19,), w.shape: (19,), ids: ['3ivg' '4de1' '4tmn' ... '2vw5' '1w3l' '2
zjw'], task_names: [0]>,
 <DiskDataset X.shape: (20, 9), y.shape: (20,), w.shape: (20,), ids: ['1kel' '2w66' '2xnb' ... '2qbp' '3lka' '1
qi0'], task_names: [0]>)
```

In [ ]:
```
train, val, test = datasets
```

Training the model

Now that we've got our dataset, let's go ahead and initialize an `AtomicConvModel` to train. Keep the input parameters the same as those used in `AtomicConvFeaturizer`, or else we'll get errors. `layer_sizes` controls the number of layers and the size of each dense layer in the network. We choose these hyperparameters to be the same as those used in the original paper.

In [ ]:
```
acm = AtomicConvModel(n_tasks=1,
                      frag1_num_atoms=f1_num_atoms,
                      frag2_num_atoms=f2_num_atoms,
                      complex_num_atoms=f1_num_atoms+f2_num_atoms,
                      max_num_neighbors=max_num_neighbors,
                      batch_size=12,
                      layer_sizes=[32, 32, 16],
                      learning_rate=0.003,
                      )
```

In [ ]:
```
losses, val_losses = [], []
```

In [ ]:
```
%%time
max_epochs = 50

metric = dc.metrics.Metric(dc.metrics.score_function.rms_score)
step_cutoff = len(train)//12
def val_cb(model, step):
  if step%step_cutoff!=0:
      return
  val_losses.append(model.evaluate(val, metrics=[metric])['rms_score']**2)  # L2 Loss
  losses.append(model.evaluate(train, metrics=[metric])['rms_score']**2)  # L2 Loss

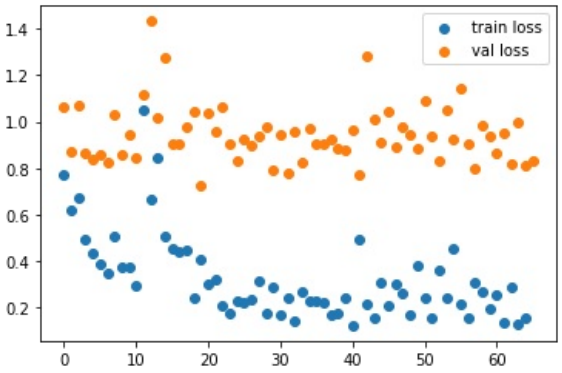
acm.fit(train, nb_epoch=max_epochs, max_checkpoints_to_keep=1,
                callbacks=[val_cb])
```

```
CPU times: user 2min 41s, sys: 11.4 s, total: 2min 53s
Wall time: 2min 47s
```

The loss curves are not exactly smooth, which is unsurprising because we are using 154 training and 19 validation

datapoints. Increasing the dataset size may help with this, but will also require greater computational resources.

In [ ]:
```
f, ax = plt.subplots()
ax.scatter(range(len(losses)), losses, label='train loss')
ax.scatter(range(len(val_losses)), val_losses, label='val loss')
plt.legend(loc='upper right');
```



The ACNN paper showed a Pearson
R^2
score of 0.912 and 0.448 for a random 80/20 split of the PDBbind core train/test sets. Here, we've used an 80/10/10 training/validation/test split and achieved similar performance for the training set (0.943). We can see from the performance on the training, validation, and test sets (and from the results in the paper) that the ACNN can learn chemical interactions from small training datasets, but struggles to generalize. Still, it is pretty amazing that we can train an `AtomicConvModel` with only a few lines of code and start predicting binding affinities!
From here, you can experiment with different hyperparameters, more challenging splits, and the "refined" set of PDBbind to see if you can reduce overfitting and come up with a more robust model.

In [ ]:
```
score = dc.metrics.Metric(dc.metrics.score_function.pearson_r2_score)
for tvt, ds in zip(['train', 'val', 'test'], datasets):
  print(tvt, acm.evaluate(ds, metrics=[score]))
```

```
train {'pearson_r2_score': 0.9311347622675604
val {'pearson_r2_score': 0.5162870575992874}
test {'pearson_r2_score': 0.4756633065901693}
```

Further reading

We have explored the ACNN architecture and used the PDBbind dataset to train an ACNN to predict protein-ligand binding energies. For more information, read the original paper that introduced ACNNs: Gomes, Joseph, et al. "Atomic convolutional networks for predicting protein-ligand binding affinity." arXiv preprint arXiv:1703.10603 (2017). There are many other methods and papers on predicting binding affinities. Here are a few interesting ones to check out: predictions using only ligands or proteins, molecular docking with deep learning, and AtomNet.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Conditional Generative Adversarial Network

A Generative Adversarial Network (GAN) is a type of generative model. It consists of two parts called the "generator" and the "discriminator". The generator takes random values as input and transforms them into an output that (hopefully) resembles the training data. The discriminator takes a set of samples as input and tries to distinguish the real training samples from the ones created by the generator. Both of them are trained together. The discriminator tries to get better and better at telling real from false data, while the generator tries to get better and better at fooling the discriminator.

A Conditional GAN (CGAN) allows additional inputs to the generator and discriminator that their output is conditioned on. For example, this might be a class label, and the GAN tries to learn how the data distribution varies between classes.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands.

In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

For this example, we will create a data distribution consisting of a set of ellipses in 2D, each with a random position, shape, and orientation. Each class corresponds to a different ellipse. Let's randomly generate the ellipses. For each one we select a random center position, X and Y size, and rotation angle. We then create a transformation matrix that maps the unit circle to the ellipse.

In [1]:
```
import deepchem as dc
import numpy as np
import tensorflow as tf

n_classes = 4
class_centers = np.random.uniform(-4, 4, (n_classes, 2))
class_transforms = []
for i in range(n_classes):
    xscale = np.random.uniform(0.5, 2)
    yscale = np.random.uniform(0.5, 2)
    angle = np.random.uniform(0, np.pi)
    m = [[xscale*np.cos(angle), -yscale*np.sin(angle)],
         [xscale*np.sin(angle), yscale*np.cos(angle)]]
    class_transforms.append(m)
class_transforms = np.array(class_transforms)
```

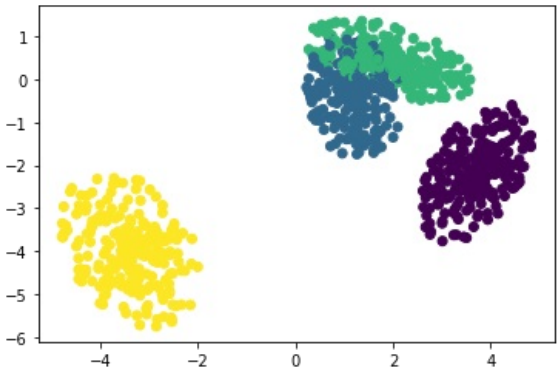
This function generates random data from the distribution. For each point it chooses a random class, then a random position in that class' ellipse.

In [2]:
```
def generate_data(n_points):
    classes = np.random.randint(n_classes, size=n_points)
    r = np.random.random(n_points)
    angle = 2*np.pi*np.random.random(n_points)
    points = (r*np.array([np.cos(angle), np.sin(angle)])).T
    points = np.einsum('ijk,ik->ij', class_transforms[classes], points)
    points += class_centers[classes]
    return classes, points
```

Let's plot a bunch of random points drawn from this distribution to see what it looks like. Points are colored based on their class label.

In [3]:
```
%matplotlib inline
import matplotlib.pyplot as plot
classes, points = generate_data(1000)
plot.scatter(x=points[:,0], y=points[:,1], c=classes)
```

Out[3]:
```
<matplotlib.collections.PathCollection at 0x1584692d0>
```



Now let's create the model for our CGAN. DeepChem's GAN class makes this very easy. We just subclass it and implement a few methods. The two most important are:

- `create_generator()` constructs a model implementing the generator. The model takes as input a batch of random noise plus any condition variables (in our case, the one-hot encoded class of each sample). Its output is a synthetic sample that is supposed to resemble the training data.
- `create_discriminator()` constructs a model implementing the discriminator. The model takes as input the samples to evaluate (which might be either real training data or synthetic samples created by the generator) and the condition variables. Its output is a single number for each sample, which will be interpreted as the probability that the sample is real training data.

In this case, we use very simple models. They just concatenate the inputs together and pass them through a few dense layers. Notice that the final layer of the discriminator uses a sigmoid activation. This ensures it produces an output between 0 and 1 that can be interpreted as a probability.

We also need to implement a few methods that define the shapes of the various inputs. We specify that the random noise provided to the generator should consist of ten numbers for each sample; that each data sample consists of two numbers (the X and Y coordinates of a point in 2D); and that the conditional input consists of `n_classes` numbers for each sample (the one-hot encoded class index).

In [4]:
```
from tensorflow.keras.layers import Concatenate, Dense, Input

class ExampleGAN(dc.models.GAN):

  def get_noise_input_shape(self):
    return (10,)

  def get_data_input_shapes(self):
    return [(2,)]

  def get_conditional_input_shapes(self):
    return [(n_classes,)]

  def create_generator(self):
    noise_in = Input(shape=(10,))
    conditional_in = Input(shape=(n_classes,))
    gen_in = Concatenate()([noise_in, conditional_in])
    gen_dense1 = Dense(30, activation=tf.nn.relu)(gen_in)
    gen_dense2 = Dense(30, activation=tf.nn.relu)(gen_dense1)
    generator_points = Dense(2)(gen_dense2)
    return tf.keras.Model(inputs=[noise_in, conditional_in], outputs=[generator_points])

  def create_discriminator(self):
    data_in = Input(shape=(2,))
    conditional_in = Input(shape=(n_classes,))
    discrim_in = Concatenate()([data_in, conditional_in])
    discrim_dense1 = Dense(30, activation=tf.nn.relu)(discrim_in)
    discrim_dense2 = Dense(30, activation=tf.nn.relu)(discrim_dense1)
    discrim_prob = Dense(1, activation=tf.sigmoid)(discrim_dense2)
    return tf.keras.Model(inputs=[data_in, conditional_in], outputs=[discrim_prob])

gan = ExampleGAN(learning_rate=1e-4)
```

Now to fit the model. We do this by calling `fit_gan()`. The argument is an iterator that produces batches of training data. More specifically, it needs to produce dicts that map all data inputs and conditional inputs to the values to use for them. In our case we can easily create as much random data as we need, so we define a generator that calls the `generate_data()` function defined above for each new batch.

In [5]:
```
def iterbatches(batches):
  for i in range(batches):
```

```
    classes, points = generate_data(gan.batch_size)
    classes = dc.metrics.to_one_hot(classes, n_classes)
    yield {gan.data_inputs[0]: points, gan.conditional_inputs[0]: classes}

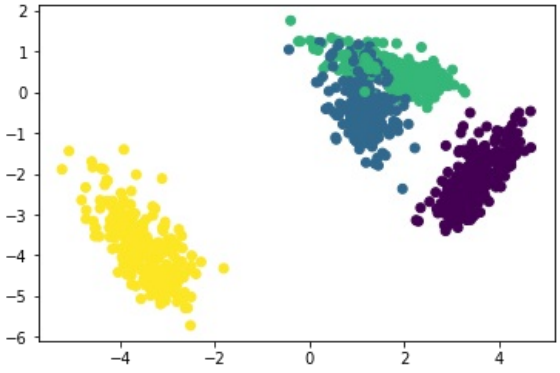
gan.fit_gan(iterbatches(5000))
```

```
Ending global_step 999: generator average loss 0.87121, discriminator average loss 1.08472
Ending global_step 1999: generator average loss 0.968357, discriminator average loss 1.17393
Ending global_step 2999: generator average loss 0.710444, discriminator average loss 1.37858
Ending global_step 3999: generator average loss 0.699195, discriminator average loss 1.38131
Ending global_step 4999: generator average loss 0.694203, discriminator average loss 1.3871
TIMING: model fitting took 31.352 s
```

Have the trained model generate some data, and see how well it matches the training distribution we plotted before.

In [6]:
```
classes, points = generate_data(1000)
one_hot_classes = dc.metrics.to_one_hot(classes, n_classes)
gen_points = gan.predict_gan_generator(conditional_inputs=[one_hot_classes])
plot.scatter(x=gen_points[:,0], y=gen_points[:,1], c=classes)
```

Out[6]:
```
<matplotlib.collections.PathCollection at 0x160dedf50>
```



Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Training a Generative Adversarial Network on MNIST

In this tutorial, we will train a Generative Adversarial Network (GAN) on the MNIST dataset. This is a large collection of 28x28 pixel images of handwritten digits. We will try to train a network to produce new images of handwritten digits.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

To begin, let's import all the libraries we'll need and load the dataset (which comes bundled with Tensorflow).

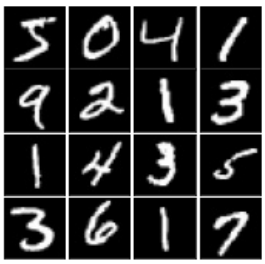
In [1]:
```
import deepchem as dc
import tensorflow as tf
from deepchem.models.optimizers import ExponentialDecay
from tensorflow.keras.layers import Conv2D, Conv2DTranspose, Dense, Reshape
import matplotlib.pyplot as plot
import matplotlib.gridspec as gridspec
%matplotlib inline

mnist = tf.keras.datasets.mnist.load_data(path='mnist.npz')
images = mnist[0][0].reshape((-1, 28, 28, 1))/255
dataset = dc.data.NumpyDataset(images)
```

Let's view some of the images to get an idea of what they look like.

In [2]:
```
def plot_digits(im):
  plot.figure(figsize=(3, 3))
  grid = gridspec.GridSpec(4, 4, wspace=0.05, hspace=0.05)
  for i, g in enumerate(grid):
    ax = plot.subplot(g)
    ax.set_xticks([])
    ax.set_yticks([])
    ax.imshow(im[i,:,:,0], cmap='gray')

plot_digits(images)
```



Now we can create our GAN. Like in the last tutorial, it consists of two parts:

1. The generator takes random noise as its input and produces output that will hopefully resemble the training data.
2. The discriminator takes a set of samples as input (possibly training data, possibly created by the generator), and tries to determine which are which.

This time we will use a different style of GAN called a Wasserstein GAN (or WGAN for short). In many cases, they are found to produce better results than conventional GANs. The main difference between the two is in the discriminator (often called a "critic" in this context). Instead of outputting the probability of a sample being real training data, it tries to learn how to measure the distance between the training distribution and generated distribution. That measure can then be directly used as a loss function for training the generator.

We use a very simple model. The generator uses a dense layer to transform the input noise into a 7x7 image with eight channels. That is followed by two convolutional layers that upsample it first to 14x14, and finally to 28x28.

The discriminator does roughly the same thing in reverse. Two convolutional layers downsample the image first to 14x14, then to 7x7. A final dense layer produces a single number as output. In the last tutorial we used a sigmoid activation to produce a number between 0 and 1 that could be interpreted as a probability. Since this is a WGAN, we

instead use a softplus activation. It produces an unbounded positive number that can be interpreted as a distance.

In [3]:
```
class DigitGAN(dc.models.WGAN):

  def get_noise_input_shape(self):
    return (10,)

  def get_data_input_shapes(self):
    return [(28, 28, 1)]

  def create_generator(self):
    return tf.keras.Sequential([
        Dense(7*7*8, activation=tf.nn.relu),
        Reshape((7, 7, 8)),
        Conv2DTranspose(filters=16, kernel_size=5, strides=2, activation=tf.nn.relu, padding='same'),
        Conv2DTranspose(filters=1, kernel_size=5, strides=2, activation=tf.sigmoid, padding='same')
    ])

  def create_discriminator(self):
    return tf.keras.Sequential([
        Conv2D(filters=32, kernel_size=5, strides=2, activation=tf.nn.leaky_relu, padding='same'),
        Conv2D(filters=64, kernel_size=5, strides=2, activation=tf.nn.leaky_relu, padding='same'),
        Dense(1, activation=tf.math.softplus)
    ])

gan = DigitGAN(learning_rate=ExponentialDecay(0.001, 0.9, 5000))
```

Now to train it. As in the last tutorial, we write a generator to produce data. This time the data is coming from a dataset, which we loop over 100 times.

One other difference is worth noting. When training a conventional GAN, it is important to keep the generator and discriminator in balance thoughout training. If either one gets too far ahead, it becomes very difficult for the other one to learn.

WGANs do not have this problem. In fact, the better the discriminator gets, the cleaner a signal it provides and the easier it becomes for the generator to learn. We therefore specify `generator_steps=0.2` so that it will only take one step of training the generator for every five steps of training the discriminator. This tends to produce faster training and better results.

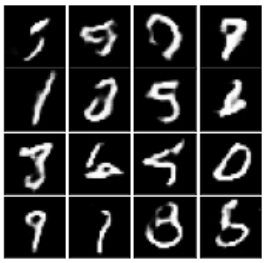
In [4]:
```
def iterbatches(epochs):
  for i in range(epochs):
    for batch in dataset.iterbatches(batch_size=gan.batch_size):
      yield {gan.data_inputs[0]: batch[0]}

gan.fit_gan(iterbatches(100), generator_steps=0.2, checkpoint_interval=5000)
```

```
Ending global_step 4999: generator average loss 0.340072, discriminator average loss -0.0234236
Ending global_step 9999: generator average loss 0.52308, discriminator average loss -0.00702729
Ending global_step 14999: generator average loss 0.572661, discriminator average loss -0.00635684
Ending global_step 19999: generator average loss 0.560454, discriminator average loss -0.00534357
Ending global_step 24999: generator average loss 0.556055, discriminator average loss -0.00620613
Ending global_step 29999: generator average loss 0.541958, discriminator average loss -0.00734233
Ending global_step 34999: generator average loss 0.540904, discriminator average loss -0.00736641
Ending global_step 39999: generator average loss 0.524298, discriminator average loss -0.00650514
Ending global_step 44999: generator average loss 0.503931, discriminator average loss -0.00563732
Ending global_step 49999: generator average loss 0.528964, discriminator average loss -0.00590612
Ending global_step 54999: generator average loss 0.510892, discriminator average loss -0.00562366
Ending global_step 59999: generator average loss 0.494756, discriminator average loss -0.00533636
TIMING: model fitting took 4197.860 s
```

Let's generate some data and see how the results look.

In [5]:
```
plot_digits(gan.predict_gan_generator(batch_size=16))
```



Not too bad. Many of the generated images look plausibly like handwritten digits. A larger model trained for a longer time can do much better, of course.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Advanced model training using hyperopt

In the Advanced Model Training tutorial we have already taken a look into hyperparameter optimasation using GridHyperparamOpt in the deepchem pacakge. In this tutorial, we will take a look into another hyperparameter tuning library called hyperopt.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

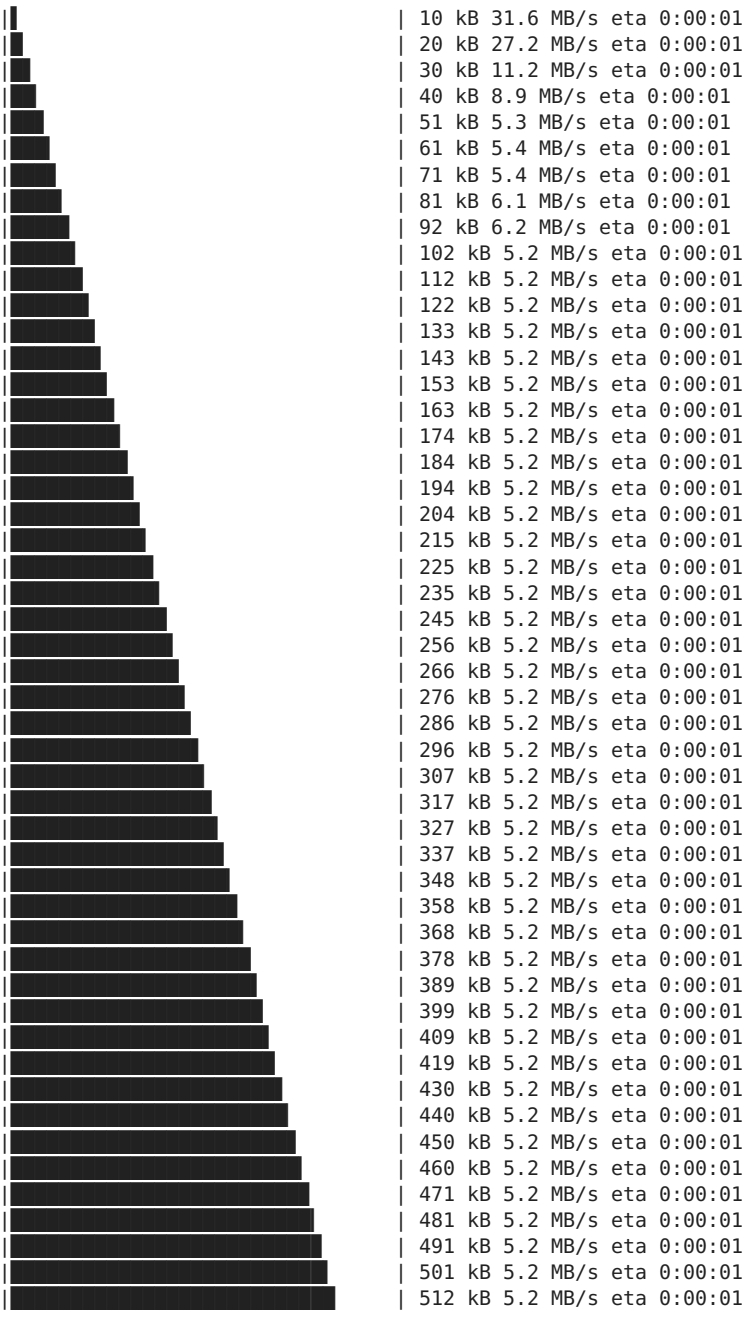


Setup

To run DeepChem and Hyperopt within Colab, you'll need to run the following installation commands. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install DeepChem and Hyperopt in your local machine again.

In [1]:
```
!pip install deepchem
!pip install hyperopt
```

```
Collecting deepchem
  Downloading deepchem-2.6.1-py3-none-any.whl (608 kB)
```



```
     |████████████████████████████████| 522 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 532 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 542 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 552 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 563 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 573 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 583 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 593 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 604 kB 5.2 MB/s eta 0:00:01
     |████████████████████████████████| 608 kB 5.2 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.4.1)
Collecting numpy>=1.21
  Downloading numpy-1.21.5-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (15.7 MB)
     |████████████████████████████████| 15.7 MB 25.3 MB/s
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.3.5)
Collecting rdkit-pypi
  Downloading rdkit_pypi-2021.9.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20.6 MB)
     |████████████████████████████████| 20.6 MB 1.4 MB/s
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem) (2
018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
pandas->deepchem) (1.15.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from rdkit-pypi->deepchem) (7.1
.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn
->deepchem) (3.1.0)
Installing collected packages: numpy, rdkit-pypi, deepchem
  Attempting uninstall: numpy
    Found existing installation: numpy 1.19.5
    Uninstalling numpy-1.19.5:
      Successfully uninstalled numpy-1.19.5
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This
behaviour is the source of the following dependency conflicts.
yellowbrick 1.3.post1 requires numpy<1.20,>=1.16.0, but you have numpy 1.21.5 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.2.9 which is incompatible.
Successfully installed deepchem-2.6.1 numpy-1.21.5 rdkit-pypi-2021.9.4
Requirement already satisfied: hyperopt in /usr/local/lib/python3.7/dist-packages (0.1.2)
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages (from hyperopt) (2.6.3)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from hyperopt) (0.16.0)
Requirement already satisfied: pymongo in /usr/local/lib/python3.7/dist-packages (from hyperopt) (4.0.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from hyperopt) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from hyperopt) (1.21.5)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from hyperopt) (4.62.3)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from hyperopt) (1.15.0)
```

Hyperparameter Optimization via hyperopt

Let's start by loading the HIV dataset. It classifies over 40,000 molecules based on whether they inhibit HIV replication.

In [2]:
```
import deepchem as dc
tasks, datasets, transformers = dc.molnet.load_hiv(featurizer='ECFP', split='scaffold')
train_dataset, valid_dataset, test_dataset = datasets
```

```
'split' is deprecated.  Use 'splitter' instead.
```

Now, lets import the hyperopt library, which we will be using to fund the best parameters

In [3]:
```
from hyperopt import hp, fmin, tpe, Trials
```

Then we have to declare a dictionary with all the hyperparameters and their range that you will be tuning them in. This dictionary will serve as the search space for the hyperopt. Some basic ways of declaring the ranges in the dictionary are:

- hp.choice('label',[*choices*]) : this is used to specify a list of choices
- hp.uniform('label' ,low=*low_value* ,high=*high_value*) : this is used to specify a uniform distibution between the low and high values. The values between them can be any real number, not necessaarily an integer.

Here, we are going to use a multitaskclassifier to classify the HIV dataset and hence the appropriate search space is as follows.

In [ ]:
```
search_space = {
    'layer_sizes': hp.choice('layer_sizes',[[500], [1000], [2000],[1000,1000]]),
    'dropouts': hp.uniform('dropout',low=0.2, high=0.5),
    'learning_rate': hp.uniform('learning_rate',high=0.001, low=0.0001)
```

```
}
```

We should then declare a function to be minimized by the hyperopt. So, here we should use the function to minimize our multitaskclassifier model. Additionally, we are using a validation callback to validate the classifier for every 1000 steps, then we are passing the best score as the return. The metric used here is 'roc_auc_score', which needs to be maximized. To maximize a non-negative value is equivalent to minimize its opposite number, hence we are returning the negative of the validation score.

In [ ]:
```
import tempfile
#tempfile is used to save the best checkpoint later in the program.

metric = dc.metrics.Metric(dc.metrics.roc_auc_score)

def fm(args):
  save_dir = tempfile.mkdtemp()
  model = dc.models.MultitaskClassifier(n_tasks=len(tasks),n_features=1024,layer_sizes=args['layer_sizes'],drop
  #validation callback that saves the best checkpoint, i.e the one with the maximum score.
  validation=dc.models.ValidationCallback(valid_dataset, 1000, [metric],save_dir=save_dir,transformers=transfor

  model.fit(train_dataset, nb_epoch=25,callbacks=validation)

  #restoring the best checkpoint and passing the negative of its validation score to be minimized.
  model.restore(model_dir=save_dir)
  valid_score = model.evaluate(valid_dataset, [metric], transformers)

  return -1*valid_score['roc_auc_score']
```

Here, we are calling the fmin function of the hyperopt, where we pass on the function to be minimized, the algorithm to be followed, max number of evals and a trials object. The Trials object is used to keep All hyperparameters, loss, and other information, this means you can access them after running optimization. Also, trials can help you to save important information and later load and then resume the optimization process.

Moreover, for the algorithm there are three choice which can be used without any additional configuration. they are :-

- Random Search - rand.suggest
- TPE (Tree Parzen Estimators) - tpe.suggest
- Adaptive TPE - atpe.suggest

In [ ]:
```
trials=Trials()
best = fmin(fm,
                space= search_space,
                algo=tpe.suggest,
                max_evals=15,
                trials = trials)
```

```
  0%|          | 0/15 [00:00<?, ?it/s, best loss: ?]Step 1000 validation: roc_auc_score=0.777648
Step 2000 validation: roc_auc_score=0.755485
Step 3000 validation: roc_auc_score=0.739519
Step 4000 validation: roc_auc_score=0.764756
Step 5000 validation: roc_auc_score=0.757006
Step 6000 validation: roc_auc_score=0.752609
Step 7000 validation: roc_auc_score=0.763002
Step 8000 validation: roc_auc_score=0.749202
  7%|▋         | 1/15 [05:37<1:18:46, 337.58s/it, best loss: -0.7776476459925534]Step 1000 validation: roc_auc_s
core=0.750455
Step 2000 validation: roc_auc_score=0.783594
Step 3000 validation: roc_auc_score=0.775872
Step 4000 validation: roc_auc_score=0.768825
Step 5000 validation: roc_auc_score=0.769555
Step 6000 validation: roc_auc_score=0.765324
Step 7000 validation: roc_auc_score=0.771146
Step 8000 validation: roc_auc_score=0.760138
 13%|█▎        | 2/15 [07:05<41:16, 190.51s/it, best loss: -0.7835939030962179]  Step 1000 validation: roc_auc_s
core=0.744178
Step 2000 validation: roc_auc_score=0.765406
Step 3000 validation: roc_auc_score=0.76532
Step 4000 validation: roc_auc_score=0.769255
Step 5000 validation: roc_auc_score=0.77029
Step 6000 validation: roc_auc_score=0.768024
Step 7000 validation: roc_auc_score=0.764157
Step 8000 validation: roc_auc_score=0.756805
 20%|██        | 3/15 [09:40<34:53, 174.42s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.714572
Step 2000 validation: roc_auc_score=0.770712
Step 3000 validation: roc_auc_score=0.777914
Step 4000 validation: roc_auc_score=0.76923
Step 5000 validation: roc_auc_score=0.774823
Step 6000 validation: roc_auc_score=0.775927
Step 7000 validation: roc_auc_score=0.777054
Step 8000 validation: roc_auc_score=0.778508
```

```
 27%|██▋       | 4/15 [12:12<30:22, 165.66s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.743939
Step 2000 validation: roc_auc_score=0.759478
Step 3000 validation: roc_auc_score=0.738839
Step 4000 validation: roc_auc_score=0.751084
Step 5000 validation: roc_auc_score=0.740504
Step 6000 validation: roc_auc_score=0.753612
Step 7000 validation: roc_auc_score=0.71802
Step 8000 validation: roc_auc_score=0.761025
 33%|███▎      | 5/15 [17:40<37:21, 224.16s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.74099
Step 2000 validation: roc_auc_score=0.767516
Step 3000 validation: roc_auc_score=0.767338
Step 4000 validation: roc_auc_score=0.775691
Step 5000 validation: roc_auc_score=0.768731
Step 6000 validation: roc_auc_score=0.755029
Step 7000 validation: roc_auc_score=0.767115
Step 8000 validation: roc_auc_score=0.764744
 40%|███▉      | 6/15 [22:48<37:54, 252.71s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.713761
Step 2000 validation: roc_auc_score=0.759518
Step 3000 validation: roc_auc_score=0.765853
Step 4000 validation: roc_auc_score=0.771976
Step 5000 validation: roc_auc_score=0.772762
Step 6000 validation: roc_auc_score=0.773206
Step 7000 validation: roc_auc_score=0.775565
Step 8000 validation: roc_auc_score=0.768521
 47%|████▋     | 7/15 [27:53<35:58, 269.84s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.717178
Step 2000 validation: roc_auc_score=0.754258
Step 3000 validation: roc_auc_score=0.767905
Step 4000 validation: roc_auc_score=0.762917
Step 5000 validation: roc_auc_score=0.766162
Step 6000 validation: roc_auc_score=0.767581
Step 7000 validation: roc_auc_score=0.770746
Step 8000 validation: roc_auc_score=0.77597
 53%|█████▎    | 8/15 [30:36<27:29, 235.64s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.74314
Step 2000 validation: roc_auc_score=0.757408
Step 3000 validation: roc_auc_score=0.76668
Step 4000 validation: roc_auc_score=0.768104
Step 5000 validation: roc_auc_score=0.746377
Step 6000 validation: roc_auc_score=0.745282
Step 7000 validation: roc_auc_score=0.74113
Step 8000 validation: roc_auc_score=0.734482
 60%|██████    | 9/15 [36:53<28:00, 280.04s/it, best loss: -0.7835939030962179]Step 1000 validation: roc_auc_sco
re=0.743204
Step 2000 validation: roc_auc_score=0.76912
Step 3000 validation: roc_auc_score=0.769981
Step 4000 validation: roc_auc_score=0.784163
Step 5000 validation: roc_auc_score=0.77536
Step 6000 validation: roc_auc_score=0.779237
Step 7000 validation: roc_auc_score=0.782344
Step 8000 validation: roc_auc_score=0.779085
 67%|██████▋   | 10/15 [38:23<18:26, 221.33s/it, best loss: -0.7841634210268469]Step 1000 validation: roc_auc_sc
ore=0.743565
Step 2000 validation: roc_auc_score=0.765063
Step 3000 validation: roc_auc_score=0.75284
Step 4000 validation: roc_auc_score=0.759978
Step 5000 validation: roc_auc_score=0.74255
Step 6000 validation: roc_auc_score=0.721809
Step 7000 validation: roc_auc_score=0.729863
Step 8000 validation: roc_auc_score=0.73075
 73%|███████▎  | 11/15 [44:07<17:15, 258.91s/it, best loss: -0.7841634210268469]Step 1000 validation: roc_auc_sc
ore=0.695949
Step 2000 validation: roc_auc_score=0.765082
Step 3000 validation: roc_auc_score=0.756256
Step 4000 validation: roc_auc_score=0.771923
Step 5000 validation: roc_auc_score=0.758841
Step 6000 validation: roc_auc_score=0.759393
Step 7000 validation: roc_auc_score=0.765971
Step 8000 validation: roc_auc_score=0.747064
 80%|████████  | 12/15 [48:54<13:21, 267.23s/it, best loss: -0.7841634210268469]Step 1000 validation: roc_auc_sc
ore=0.757871
Step 2000 validation: roc_auc_score=0.765296
Step 3000 validation: roc_auc_score=0.769748
Step 4000 validation: roc_auc_score=0.776487
Step 5000 validation: roc_auc_score=0.775009
Step 6000 validation: roc_auc_score=0.779539
Step 7000 validation: roc_auc_score=0.763165
Step 8000 validation: roc_auc_score=0.772093
 87%|████████▋ | 13/15 [50:22<07:06, 213.15s/it, best loss: -0.7841634210268469]Step 1000 validation: roc_auc_sc
ore=0.720166
```

```
Step 2000 validation: roc_auc_score=0.768489
Step 3000 validation: roc_auc_score=0.782853
Step 4000 validation: roc_auc_score=0.785556
Step 5000 validation: roc_auc_score=0.78583
Step 6000 validation: roc_auc_score=0.786569
Step 7000 validation: roc_auc_score=0.779249
Step 8000 validation: roc_auc_score=0.783423
 93%|█████████ | 14/15 [51:52<02:55, 175.93s/it, best loss: -0.7865693280913189]Step 1000 validation: roc_auc_sc
ore=0.743232
Step 2000 validation: roc_auc_score=0.762007
Step 3000 validation: roc_auc_score=0.771809
Step 4000 validation: roc_auc_score=0.755023
Step 5000 validation: roc_auc_score=0.769812
Step 6000 validation: roc_auc_score=0.769867
Step 7000 validation: roc_auc_score=0.777354
Step 8000 validation: roc_auc_score=0.775313
100%|██████████| 15/15 [56:47<00:00, 227.13s/it, best loss: -0.7865693280913189]
```

The code below is used to print the best hyperparameters found by the hyperopt.

In [ ]:
```
print("Best: {}".format(best))
```

```
Best: {'dropout': 0.3749846096922802, 'layer_sizes': 0, 'learning_rate': 0.0007544819475363869}
```

The hyperparameter found here may not be necessarily the best one, but gives a general idea on which parameters are effective. To get mroe accurate results, one has to increase the number of validation epochs and the epochs the model fit. But doing so may increase the time in finding the best hyperparameters.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Introduction to Gaussian Processes

In the world of cheminformatics and machine learning, models are often trees (random forest, XGBoost, etc.) or artifical neural networks (deep neural networks, graph convolutional networks, etc.). These models are known as "Frequentist" models. However, there is another category known as Bayesian models. Today we will be experimenting with a Bayesian model implemented in scikit-learn known as gaussian processes (GP). For a deeper dive on GP, there is a great tutorial paper on how GP works for regression. There is also an academic paper that applies GP to a real world problem.

As a short intro, GP allows us to build up our statistical model using an infinite number of Gaussian functions over our n-dimensional space, where n is the number of features. However, we pick these functions based on how well they fit the data we pass it. We end up with a statistical model built from an *ensemble* of Gaussian functions which can actually vary quite a bit. The result is that for points we have trained the model on, the variance in our ensemble should be very low. For test set points close to the training set points, the variance should be higher but still low as the ensemble was picked to predict well in its neighborhood. For points far from the training set points, however, we did not pick our ensemble of Gaussian functions to fit them so we'd expect the variance in our ensemble to be high. In this way, we end up with a statistical model that allows for a natural generation of uncertainty.

Colab

This tutorial and the rest in the sequences are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

The first step is to get DeepChem up and running. We recommend using Google Colab to work through this tutorial series. You'll need to run the following commands to get DeepChem installed on your colab notebook.

In [1]:
```
%pip install --pre deepchem
```

```
Requirement already satisfied: deepchem in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (2.5.0.de
v20210319222130)
Requirement already satisfied: scikit-learn in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from
deepchem) (1.0.2)
Requirement already satisfied: numpy in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from deepch
em) (1.19.1)
Requirement already satisfied: pandas in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from deepc
hem) (1.3.1)
Requirement already satisfied: joblib in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from deepc
hem) (1.1.0)
Requirement already satisfied: scipy in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from deepch
em) (1.6.2)
Requirement already satisfied: python-dateutil>=2.7.3 in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-pack
ages (from pandas->deepchem) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from
pandas->deepchem) (2021.3)
Requirement already satisfied: six>=1.5 in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packages (from pyt
hon-dateutil>=2.7.3->pandas->deepchem) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /home/ozone/miniconda3/envs/mol/lib/python3.7/site-packag
es (from scikit-learn->deepchem) (2.2.0)
```

Gaussian Processes

As stated earlier, GP is already implemented in scikit-learn so we will be using DeepChem's scikit-learn wrapper. SklearnModel is a subclass of DeepChem's Model class. It acts as a wrapper around a sklearn.base.BaseEstimator.

Here we import deepchem and the GP regressor model from sklearn.

In [1]:
```
import deepchem as dc
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, WhiteKernel

import numpy as np
import matplotlib.pyplot as plt
```

Loading data

Next we need a dataset that presents a regression problem. For this tutorial we will be using the BACE dataset from MoleculeNet.

In [2]:
```
tasks, datasets, transformers = dc.molnet.load_bace_regression(featurizer='ecfp', splitter='random')
train_dataset, valid_dataset, test_dataset = datasets
```

I always like to get a close look at what the objects in my code are storing. We see that tasks is a list of tasks that we are trying to predict. The transformer is a NormalizationTransformer that normalizes the outputs (y values) of the dataset.

In [3]:
```
print(f'The tasks are: {tasks}')
print(f'The transformers are: {transformers}')
print(f'The transformer normalizes the outputs (y values): {transformers[0].transform_y}')
```

```
The tasks are: ['pIC50']
The transformers are: [<deepchem.trans.transformers.NormalizationTransformer object at 0x7fc04401b190>]
The transformer normalizes the outputs (y values): True
```

Here we see that the data has already been split into a training set, a validation set, and a test set. We will train the model on the training set and test the accuracy of the model on the test set. If we were to do any hyperparameter tuning, we would use the validation set. The split was ~80/10/10 train/valid/test.

In [4]:
```
print(train_dataset)
print(valid_dataset)
print(test_dataset)
```

```
<DiskDataset X.shape: (1210, 1024), y.shape: (1210, 1), w.shape: (1210, 1), task_names: ['pIC50']>
<DiskDataset X.shape: (151, 1024), y.shape: (151, 1), w.shape: (151, 1), ids: ['Fc1ncccc1-c1cc(ccc1)C1(N=C(N)N(C
)C1=O)c1cn(nc1)CC(CC)CC'
 'S1(=O)(=O)N(c2cc(cc3n(cc(CC1)c23)CC)C(=O)NC(Cc1ccccc1)C(=O)C[NH2+]C1CCOCC1)C'
 's1ccnc1-c1cc(ccc1)CC(NC(=O)[C@@H](OC)C)C(O)C[NH2+]C1CC2(Oc3ncc(cc13)CC(C)(C)C)CCC2'
 ...
 'S(=O)(=O)(Nc1cc(cc(c1)C(C)(C)C)C1([NH2+]CC(O)C(NC(=O)C)Cc2cc(F)cc(F)c2)CCCCC1)C'
 'O=C1N(C)C(=N[C@]1(c1cc(nc(c1)CC)CC)c1cc(ccc1)-c1cncnc1)N'
 'Clc1cc2CC(N=C(NC(Cc3ccccc3)C=3NC(=O)c4c(N=3)cccc4)c2cc1)(C)C'], task_names: ['pIC50']>
<DiskDataset X.shape: (152, 1024), y.shape: (152, 1), w.shape: (152, 1), ids: ['Clc1ccc(cc1)CC(NC(=O)C)C(O)C[NH2
+]C1CC2(Oc3ncc(cc13)CC(C)(C)C)CCC2'
 'Fc1cc(cc(F)c1)CC(NC(=O)c1cc(cc(Oc2ccc(F)cc2)c1)C(=O)N(CCC)CCC)C(O)C[NH2+]Cc1cc(OC)ccc1'
 'O1c2c(cc(cc2)C2CCCCC2)C2(N=C(N)N(C)C2=O)CC1(C)C' ...
 'S(=O)(=O)(N(C)c1cc(cc(c1)COCC([NH3+])(Cc1ccccc1)C(F)F)C(=O)NC(C)c1ccc(F)cc1)C'
 'O1CCCC1CN1C(=O)C(N=C1N)(C1CCCCC1)c1ccccc1'
 'Fc1cc(cc(c1)C#C)CC(NC(=O)COC)C(O)C[NH2+]C1CC2(Oc3ncc(cc13)CC(C)(C)C)CCC2'], task_names: ['pIC50']>
```

Using the SklearnModel

Here we first create the model using the GaussianProcessRegressor we imported from sklearn. Then we wrap it in DeepChem's SklearnModel. To learn more about the model, you can either read the sklearn API or run help(GaussianProcessRegressor) in a code block.

As you see, the values I picked for the parameters seem awfully specific. This is because I needed to do some hyperparameter tuning beforehand to get model that wasn't wildly overfitting the training set. You can learn more about how I tuned the model in the Appendix at the end of this tutorial.

In [5]:
```
output_variance = 7.908735015054668
length_scale = 6.452349252677817
noise_level = 0.10475507755839343
kernel = output_variance**2 * RBF(length_scale=length_scale, length_scale_bounds='fixed') + WhiteKernel(noise_l
alpha = 4.989499481123432e-09

sklearn_gpr = GaussianProcessRegressor(kernel=kernel, alpha=alpha)
model = dc.models.SklearnModel(sklearn_gpr)
```

Then we fit our model to the data and see how it performs both on the training set and on the test set.

In [6]:
```
model.fit(train_dataset)
metric1 = dc.metrics.Metric(dc.metrics.mean_squared_error)
metric2 = dc.metrics.Metric(dc.metrics.r2_score)
print(f'Training set score: {model.evaluate(train_dataset, [metric1, metric2])}')
print(f'Test set score: {model.evaluate(test_dataset, [metric1, metric2])}')
```

```
Training set score: {'mean_squared_error': 0.0457129375800123, 'r2_score': 0.9542870624199877}
Test set score: {'mean_squared_error': 0.20503945381118496, 'r2_score': 0.7850242035806018}
```

Analyzing the Results

We can also visualize how well the predicted values match up to the measured values. First we need a function that

allows us to obtain both the mean predicted value and the standard deviation of the value. This is done by sampling 100 predictions from each set of inputs X and calculating the mean and standard deviation.

In [7]:
```
def predict_with_error(dc_model, X, y_transformer):
    samples = model.model.sample_y(X, 100)
    means = y_transformer.untransform(np.mean(samples, axis=1))
    stds = y_transformer.y_stds[0] * np.std(samples, axis=1)

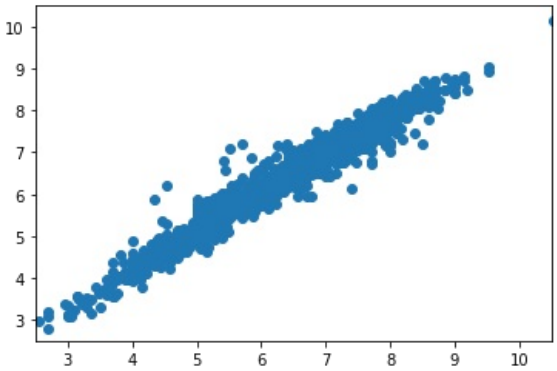
    return means, stds
```

For our training set, we see a pretty good correlation between the measured values (x-axis) and the predicted values (y-axis). Note that we use the transformer from earlier to untransform our predicted values.

In [8]:
```
y_meas_train = transformers[0].untransform(train_dataset.y)
y_pred_train, y_pred_train_stds = predict_with_error(model, train_dataset.X, transformers[0])

plt.xlim([2.5, 10.5])
plt.ylim([2.5, 10.5])
plt.scatter(y_meas_train, y_pred_train)
```

Out[8]:
```
<matplotlib.collections.PathCollection at 0x7fc0431b45d0>
```

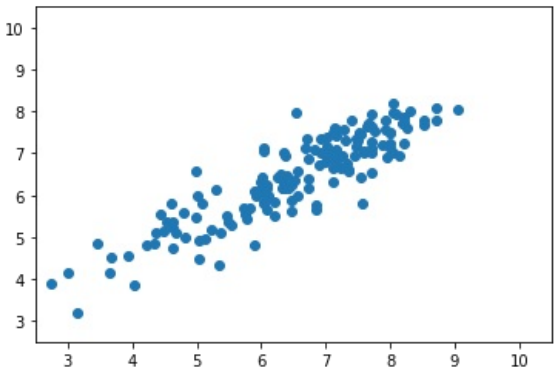


We now do the same for our test set. We see a fairly good correlation! However, it is certainly not as tight. This is reflected in the difference between the R2 scores calculated above.

In [9]:
```
y_meas_test = transformers[0].untransform(test_dataset.y)
y_pred_test, y_pred_test_stds = predict_with_error(model, test_dataset.X, transformers[0])

plt.xlim([2.5, 10.5])
plt.ylim([2.5, 10.5])
plt.scatter(y_meas_test, y_pred_test)
```

Out[9]:
```
<matplotlib.collections.PathCollection at 0x7fc04023b590>
```



We can also write a function to calculate how many of the predicted values fall within the predicted error range. This is done by counting up how many samples have a true error smaller than its standard deviation calculated earlier. One standard deviation is a 68% confidence interval.

In [10]:
```
def percent_within_std(y_meas, y_pred, y_std):
    assert len(y_meas) == len(y_pred) and len(y_meas) == len(y_std), 'length of y_meas and y_pred must be the sa

    count_within_error = 0
    for i in range(len(y_meas)):
        if abs(y_meas[i][0]-y_pred[i]) < y_std[i]:
            count_within_error += 1

    return count_within_error/len(y_meas)
```

For the train set, >90% of the samples are within a standard deviation. In comparison, only ~70% of the samples are

within a standard deviation for the test set. A standard deviation is a 68% confidence interval so we see that for the training set, the uncertainty is close. However, this model overpredicts uncertainty on the training set.

In [11]:
```
percent_within_std(y_meas_train, y_pred_train, y_pred_train_stds)
```

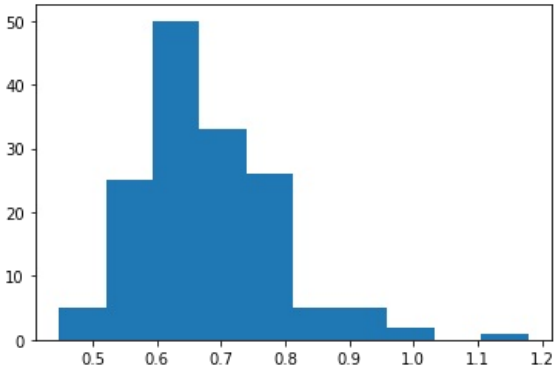
Out[11]: 0.9355371900826446

In [12]:
```
percent_within_std(y_meas_test, y_pred_test, y_pred_test_stds)
```

Out[12]: 0.7368421052631579

We can also take a look at the distributions of the standard deviations for the test set predictions. We see a very roughly Gaussian distribution in the predicted errors.

In [13]:
```
plt.hist(y_pred_test_stds)
plt.show()
```



For now, this is the end of our tutorial. We plan to follow up soon with a deeper dive into uncertainty estimation and in particular, calibrated uncertainty estimation. We will see you then!

Appendix: Hyperparameter Optimization

As hyperparameter optimization is outside the scope of this tutorial, I will not explain how to use Optuna to tune hyperparameters. But the code is still included for the sake of completeness.

In [ ]:
```
%pip install optuna
import optuna
```

In [ ]:
```
def get_model(trial):
    output_variance = trial.suggest_float('output_variance', 0.1, 10, log=True)
    length_scale = trial.suggest_float('length_scale', 1e-5, 1e5, log=True)
    noise_level = trial.suggest_float('noise_level', 1e-5, 1e5, log=True)

    params = {
        'kernel': output_variance**2 * RBF(length_scale=length_scale, length_scale_bounds='fixed') + WhiteKerne
        'alpha': trial.suggest_float('alpha', 1e-12, 1e-5, log=True),
    }

    sklearn_gpr = GaussianProcessRegressor(**params)
    return dc.models.SklearnModel(sklearn_gpr)

def objective(trial):
    model = get_model(trial)
    model.fit(train_dataset)

    metric = dc.metrics.Metric(dc.metrics.mean_squared_error)
    return model.evaluate(valid_dataset, [metric])['mean_squared_error']
```

In [ ]:
```
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)
```

In [ ]:
```
print(study.best_params)
```

```
{'output_variance': 0.38974570882583015, 'length_scale': 5.375387643239208, 'noise_level': 0.0016265333497286342
, 'alpha': 1.1273318360324618e-11}
```

Pytorch-Lightning Integration for DeepChem Models

In this tutorial we will go through how to setup a deepchem model inside the pytorch-lightning framework. Lightning is a pytorch framework which simplifies the process of experimenting with pytorch models easier. A few key functionalities offered by pytorch lightning which deepchem users can find useful are:

1. Multi-gpu training functionalities: pytorch-lightning provides easy multi-gpu, multi-node training. It also simplifies the process of launching multi-gpu, multi-node jobs across different cluster infrastructure, e.g. AWS, slurm based clusters.
2. Reducing boilerplate pytorch code: lightning takes care of details like, `optimizer.zero_grad(), model.train(), model.eval()`. Lightning also provides experiment logging functionality, for e.g. irrespective of training on CPU, GPU, multi-nodes the user can use the method `self.log` inside the trainer and it will appropriately log the metrics.
3. Features that can speed up training: half-precision training, gradient checkpointing, code profiling.



Setup

- This notebook assumes that you have already installed deepchem, if you have not follow the instructions at the deepchem installation page: https://deepchem.readthedocs.io/en/latest/get_started/installation.html.
- Install pytorch lightning following the instructions on lightning's home page: https://www.pytorchlightning.ai/

In [1]:
```
!pip install --pre deepchem
!pip install pytorch_lightning
```

```
Requirement already satisfied: deepchem in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-pa
ckages (2.6.1.dev20220119163852)
Requirement already satisfied: numpy>=1.21 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site
-packages (from deepchem) (1.22.0)
Requirement already satisfied: scikit-learn in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/sit
e-packages (from deepchem) (1.0.2)
Requirement already satisfied: pandas in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-pack
ages (from deepchem) (1.4.0)
Collecting rdkit-pypi
  Downloading rdkit_pypi-2021.9.5.1-cp38-cp38-macosx_11_0_arm64.whl (15.9 MB)
     |████████████████████████████████| 15.9 MB 6.8 MB/s eta 0:00:01
Requirement already satisfied: joblib in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-pack
ages (from deepchem) (1.1.0)
Requirement already satisfied: scipy in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-packa
ges (from deepchem) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pytho
n3.8/site-packages (from scikit-learn->deepchem) (3.0.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pyt
hon3.8/site-packages (from pandas->deepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/sit
e-packages (from pandas->deepchem) (2021.3)
Requirement already satisfied: Pillow in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-pack
ages (from rdkit-pypi->deepchem) (8.4.0)
Requirement already satisfied: six>=1.5 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-pa
ckages (from python-dateutil>=2.8.1->pandas->deepchem) (1.16.0)
Installing collected packages: rdkit-pypi
Successfully installed rdkit-pypi-2021.9.5.1
Requirement already satisfied: pytorch_lightning in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.
8/site-packages (1.5.8)
Requirement already satisfied: typing-extensions in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.
8/site-packages (from pytorch_lightning) (4.0.1)
Requirement already satisfied: numpy>=1.17.2 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/si
te-packages (from pytorch_lightning) (1.22.0)
Requirement already satisfied: torch>=1.7.* in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/sit
e-packages (from pytorch_lightning) (1.10.2)
Requirement already satisfied: tensorboard>=2.2.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3
.8/site-packages (from pytorch_lightning) (2.7.0)
Requirement already satisfied: tqdm>=4.41.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/sit
e-packages (from pytorch_lightning) (4.62.3)
Requirement already satisfied: fsspec[http]!=2021.06.0,>=2021.05.0 in /Users/princychahal/mambaforge/envs/keras_
try_5/lib/python3.8/site-packages (from pytorch_lightning) (2022.1.0)
Requirement already satisfied: packaging>=17.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/
site-packages (from pytorch_lightning) (21.3)
Requirement already satisfied: PyYAML>=5.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site
-packages (from pytorch_lightning) (6.0)
Requirement already satisfied: pyDeprecate==0.3.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3
.8/site-packages (from pytorch_lightning) (0.3.1)
Processing /Users/princychahal/Library/Caches/pip/wheels/8e/70/28/3d6ccd6e315f65f245da085482a2e1c7d14b90b30f239e
```

```
2cf4/future-0.18.2-py3-none-any.whl
Requirement already satisfied: torchmetrics>=0.4.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python
3.8/site-packages (from pytorch_lightning) (0.7.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /Users/princychahal/mambaforge/envs/kera
s_try_5/lib/python3.8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (0.6.0)
Requirement already satisfied: absl-py>=0.4 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/sit
e-packages (from tensorboard>=2.2.0->pytorch_lightning) (1.0.0)
Requirement already satisfied: grpcio>=1.24.3 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/s
ite-packages (from tensorboard>=2.2.0->pytorch_lightning) (1.43.0)
Requirement already satisfied: requests<3,>=2.21.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python
3.8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (2.27.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pyth
on3.8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (2.3.3)
Requirement already satisfied: wheel>=0.26 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site
-packages (from tensorboard>=2.2.0->pytorch_lightning) (0.37.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /Users/princychahal/mambaforge/envs/keras_try
_5/lib/python3.8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (0.4.6)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /Users/princychahal/mambaforge/envs/keras_try_5/
lib/python3.8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (1.8.1)
Requirement already satisfied: setuptools>=41.0.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3
.8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (60.5.0)
Requirement already satisfied: werkzeug>=0.11.15 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.
8/site-packages (from tensorboard>=2.2.0->pytorch_lightning) (2.0.2)
Requirement already satisfied: markdown>=2.6.8 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/
site-packages (from tensorboard>=2.2.0->pytorch_lightning) (3.3.6)
Requirement already satisfied: protobuf>=3.6.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/
site-packages (from tensorboard>=2.2.0->pytorch_lightning) (3.18.1)
Requirement already satisfied: aiohttp; extra == "http" in /Users/princychahal/mambaforge/envs/keras_try_5/lib/p
ython3.8/site-packages (from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning) (3.8.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/p
ython3.8/site-packages (from packaging>=17.0->pytorch_lightning) (3.0.7)
Requirement already satisfied: six in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-package
s (from absl-py>=0.4->tensorboard>=2.2.0->pytorch_lightning) (1.16.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pyth
on3.8/site-packages (from requests<3,>=2.21.0->tensorboard>=2.2.0->pytorch_lightning) (1.26.8)
Requirement already satisfied: certifi>=2017.4.17 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3
.8/site-packages (from requests<3,>=2.21.0->tensorboard>=2.2.0->pytorch_lightning) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0; python_version >= "3" in /Users/princychahal/mambaforg
e/envs/keras_try_5/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard>=2.2.0->pytorch_lightning)
(2.0.10)
Requirement already satisfied: idna<4,>=2.5; python_version >= "3" in /Users/princychahal/mambaforge/envs/keras_
try_5/lib/python3.8/site-packages (from requests<3,>=2.21.0->tensorboard>=2.2.0->pytorch_lightning) (3.3)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pyt
hon3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard>=2.2.0->pytorch_lightning) (4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pyth
on3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard>=2.2.0->pytorch_lightning) (0.2.7)
Requirement already satisfied: rsa<5,>=3.1.4; python_version >= "3.6" in /Users/princychahal/mambaforge/envs/ker
as_try_5/lib/python3.8/site-packages (from google-auth<3,>=1.6.3->tensorboard>=2.2.0->pytorch_lightning) (4.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/p
ython3.8/site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard>=2.2.0->pytorch_lightning) (1.3.0)
Requirement already satisfied: importlib-metadata>=4.4; python_version < "3.10" in /Users/princychahal/mambaforg
e/envs/keras_try_5/lib/python3.8/site-packages (from markdown>=2.6.8->tensorboard>=2.2.0->pytorch_lightning) (4.
10.1)
Requirement already satisfied: aiosignal>=1.1.2 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8
/site-packages (from aiohttp; extra == "http"->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning) (1.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.
8/site-packages (from aiohttp; extra == "http"->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning) (1.2.0)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /Users/princychahal/mambaforge/envs/keras_try_5/li
b/python3.8/site-packages (from aiohttp; extra == "http"->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning
) (4.0.2)
Requirement already satisfied: multidict<7.0,>=4.5 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python
3.8/site-packages (from aiohttp; extra == "http"->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning) (6.0.2
)
Requirement already satisfied: yarl<2.0,>=1.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/s
ite-packages (from aiohttp; extra == "http"->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning) (1.7.2)
Requirement already satisfied: attrs>=17.3.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/si
te-packages (from aiohttp; extra == "http"->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch_lightning) (21.4.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/pytho
n3.8/site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard>=2.2.0->pytorch_lightning) (0
.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/
site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard>=2.2.0->pytorch_ligh
tning) (3.1.1)
Requirement already satisfied: zipp>=0.5 in /Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-p
ackages (from importlib-metadata>=4.4; python_version < "3.10"->markdown>=2.6.8->tensorboard>=2.2.0->pytorch_lig
htning) (3.7.0)
Installing collected packages: future
Successfully installed future-0.18.2
```

Import the relevant packages.

In [2]:
```
import deepchem as dc
from deepchem.models import GCNModel
```

```
import pytorch_lightning as pl
import torch
from torch.nn import functional as F
from torch import nn
import pytorch_lightning as pl
from pytorch_lightning.core.lightning import LightningModule
from torch.optim import Adam
import numpy as np
import torch
```

Deepchem Example

Below we show an example of a Graph Convolution Network (GCN). Note that this is a simple example which uses a GCNModel to predict the label from an input sequence. We do not showcase the complete functionality of deepchem in this example as we want to restructure the deepchem code and adapt it so that it can be easily plugged into pytorch-lightning. This example was inspired from the `GCNModel` documentation present here.

Prepare the dataset: for training our deepchem models we need a dataset that we can use to train the model. Below we prepare a sample dataset for the purposes of this tutorial. Below we also directly use the featurized to encode examples for the dataset.

In [3]:
```
smiles = ["C1CCC1", "CCC"]
labels = [0., 1.]
featurizer = dc.feat.MolGraphConvFeaturizer()
X = featurizer.featurize(smiles)
dataset = dc.data.NumpyDataset(X=X, y=labels)
```

Setup the model: now we initialize the Graph Convolutional Network model that we will use in our training.

In [4]:
```
model = GCNModel(
    mode='classification',
    n_tasks=1,
    batch_size=2,
    learning_rate=0.001
)
```

```
[16:00:37] /Users/princychahal/Documents/github/dgl/src/runtime/tensordispatch.cc:43: TensorDispatcher: dlopen f
ailed: Using backend: pytorch
dlopen(/Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-packages/dgl-0.8-py3.8-macosx-11.0-arm
64.egg/dgl/tensoradapter/pytorch/libtensoradapter_pytorch_1.10.2.dylib, 1): image not found
```

Train the model: fit the model on our training dataset, also specify the number of epochs to run.

In [5]:
```
loss = model.fit(dataset, nb_epoch=5)
print(loss)
```

```
0.18830760717391967
/Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-packages/torch/autocast_mode.py:141: UserWarn
ing: User provided device_type of 'cuda', but CUDA is not available. Disabling
  warnings.warn('User provided device_type of \'cuda\', but CUDA is not available. Disabling')
```

Pytorch-Lightning + Deepchem example

Now we will look at an example of the GCN model adapt for Pytorch-Lightning. For using Pytorch-Lightning there are two important components:

1. `LightningDataModule`: This module defines who the data is prepared and fed into the model so that the model can use it for training. The module defines the train dataloader function which are directly used by the trainer to generate data for the `LightningModule`. To learn more about the `LightningDataModule` refer to the datamodules documentation.
2. `LightningModule`: This module defines the training, validation steps for our model. We can use this module to initialize our model based on the hyperparameters. There are a number of boilerplate functions which we use directly to track our experiments, for example we can save all the hyperparameters that we used for training using the `self.save_hyperparameters()` method. For more details on how to use this module refer to the lightningmodules documentation.

Setup the torch dataset: Note that here we need to create a custome `SmilesDataset` so that we can easily interface with the deepchem featurizers. For this interface we need to define a collate method so that we can create batches for the dataset.

In [6]:
```
# prepare LightningDataModule
class SmilesDataset(torch.utils.data.Dataset):
    def __init__(self, smiles, labels):
        assert len(smiles) == len(labels)
```

```
        featurizer = dc.feat.MolGraphConvFeaturizer()
        X = featurizer.featurize(smiles)
        self._samples = dc.data.NumpyDataset(X=X, y=labels)

    def __len__(self):
        return len(self._samples)

    def __getitem__(self, index):
        return (
            self._samples.X[index],
            self._samples.y[index],
            self._samples.w[index],
        )


class SmilesDatasetBatch:
    def __init__(self, batch):
        X = [np.array([b[0] for b in batch])]
        y = [np.array([b[1] for b in batch])]
        w = [np.array([b[2] for b in batch])]
        self.batch_list = [X, y, w]


def collate_smiles_dataset_wrapper(batch):
    return SmilesDatasetBatch(batch)
```

Create the GCN specific lightning module: in this part we use an object of the `SmilesDataset` created above to create the `SmilesDatasetModule`

In [7]:
```
class SmilesDatasetModule(pl.LightningDataModule):
    def __init__(self, train_smiles, train_labels, batch_size):
        super().__init__()
        self._train_smiles = train_smiles
        self._train_labels = train_labels
        self._batch_size = batch_size

    def setup(self, stage):
        self.train_dataset = SmilesDataset(
            self._train_smiles,
            self._train_labels,
        )

    def train_dataloader(self):
        return torch.utils.data.DataLoader(
            self.train_dataset,
            batch_size=self._batch_size,
            collate_fn=collate_smiles_dataset_wrapper,
            shuffle=True,
        )
```

Create the lightning module: in this part we create the GCN specific lightning module. This class specifies the logic flow for the training step. We also create the required models, optimizers and losses for the training flow.

In [8]:
```
# prepare the LightningModule
class GCNModule(pl.LightningModule):
    def __init__(self, mode, n_tasks, learning_rate):
        super().__init__()
        self.save_hyperparameters(
            "mode",
            "n_tasks",
            "learning_rate",
        )
        self.gcn_model = GCNModel(
            mode=self.hparams.mode,
            n_tasks=self.hparams.n_tasks,
            learning_rate=self.hparams.learning_rate,
        )
        self.pt_model = self.gcn_model.model
        self.loss = self.gcn_model._loss_fn

    def configure_optimizers(self):
        return self.gcn_model.optimizer._create_pytorch_optimizer(
            self.pt_model.parameters(),
        )

    def training_step(self, batch, batch_idx):
        batch = batch.batch_list
        inputs, labels, weights = self.gcn_model._prepare_batch(batch)
        outputs = self.pt_model(inputs)

        if isinstance(outputs, torch.Tensor):
            outputs = [outputs]
```

```
        if self.gcn_model._loss_outputs is not None:
            outputs = [outputs[i] for i in self.gcn_model._loss_outputs]

        loss_outputs = self.loss(outputs, labels, weights)

        self.log(
            "train_loss",
            loss_outputs,
            on_epoch=True,
            sync_dist=True,
            reduce_fx="mean",
            prog_bar=True,
        )

        return loss_outputs
```

Create the relevant objects

In [9]:
```
# create module objects
smiles_datasetmodule = SmilesDatasetModule(
    train_smiles=["C1CCC1", "CCC", "C1CCC1", "CCC", "C1CCC1", "CCC", "C1CCC1", "CCC", "C1CCC1", "CCC"],
    train_labels=[0., 1., 0., 1., 0., 1., 0., 1., 0., 1.],
    batch_size=2,
)

gcnmodule = GCNModule(
    mode="classification",
    n_tasks=1,
    learning_rate=1e-3,
)
```

Lightning Trainer

Trainer is the wrapper which builds on top of the `LightningDataModule` and `LightningModule`. When constructing the lightning trainer you can also specify the number of epochs, max-steps to run, number of GPUs, number of nodes to be used for trainer. Lightning trainer acts as a wrapper over your distributed training setup and this way you are able to build your models in a way you would build them in a simple way for your local runs.

In [10]:
```
trainer = pl.Trainer(
    max_epochs=5,
)
```

```
GPU available: False, used: False
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
```

Call the fit function to run model training

In [11]:
```
# train
trainer.fit(
    model=gcnmodule,
    datamodule=smiles_datasetmodule,
)
```

```
  | Name     | Type | Params
---------------------------------
0 | pt_model | GCN  | 29.4 K
---------------------------------
29.4 K    Trainable params
0         Non-trainable params
29.4 K    Total params
0.118     Total estimated model params size (MB)
/Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-packages/pytorch_lightning/trainer/data_loadi
ng.py:132: UserWarning: The dataloader, train_dataloader, does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` (try 8 which is the number of cpus on this machine)
in the `DataLoader` init to improve performance.
  rank_zero_warn(
/Users/princychahal/mambaforge/envs/keras_try_5/lib/python3.8/site-packages/pytorch_lightning/trainer/data_loadi
ng.py:428: UserWarning: The number of training samples (5) is smaller than the logging interval Trainer(log_ever
y_n_steps=50). Set a lower value for log_every_n_steps if you want to see logs for the training epoch.
  rank_zero_warn(
Training: 0it [00:00, ?it/s]
```

In [ ]:
```
```

Compiling Deepchem Torch Models

Deep Learning models typically involve millions or even billions of parameters (as in the case of LLMs) that need to be fine-tuned through an iterative training processes. During training, these models process vast amounts of data to learn patterns and features effectively. This data-intensive nature, combined with the computational complexity of operations like matrix multiplications and backpropagation, leads to lengthy training times that can span days, weeks, or even months on standard hardware. Additionally, the need for multiple training runs to experiment with the different model and hyperparameter configurations further extends the overall development time.

Effective optimization techniques can significantly reduce training times, lower computational costs, and improve model performance. This makes optimization particularly crucial in research and industrial settings where faster iterations can accelerate scientific discoveries, product development, and the deployment of AI solutions. Moreover, as models grow larger and more sophisticated, optimization plays a vital role in making advanced AI accessible and practical for a wider range of applications and environments.

To address the need for optimization of Deep Larning models and as an improvement over existing methods, PyTorch introduced the `torch.compile()` function in PyTorch 2.0 to allow faster training and inference of the models. `torch.compile()` works by compiling PyTorch code into optimised kernels using a JIT(Just in Time) compiler. Different models show varying levels of improvement in run times depending on their architecture and batch size when compiled. Compared to existing methods like TorchScript or FX tracing, `compile()` also offers advantages such as the ability to handle arbitrary Python code and conditional graph-breaking flow of the inputs to the models. This allows `compile()` to work with minimal or no code modification to the model.

DeepChem has builtin support for compiling PyTorch models using `torch.compile()` and using this feature, users can efficiently run PyTorch models and achieve significant performance gains. This tutorial contains the steps for compiling DeepChem PyTorch models, benchmarking and evaluating their performance with the uncompiled models.

> NOTE: DeepChem contains many models with varying architecture and complexity. Not all models will show significant improvements in run times when compiled. It is recommended to test the models with and without compilation to determine the performance improvements.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Compilation Process

This section gives an introductory explanation about the compilation process of PyTorch models and assumes prior knowledge about forward pass, backward pass and computational graphs in neural networks. If you're unfamiliar with these concepts, you can refer to these slides for a basic understanding. Alternatively, you can proceed to the next section to learn how to compile and benchmark DeepChem models without delving into the internal details of the compilation process.

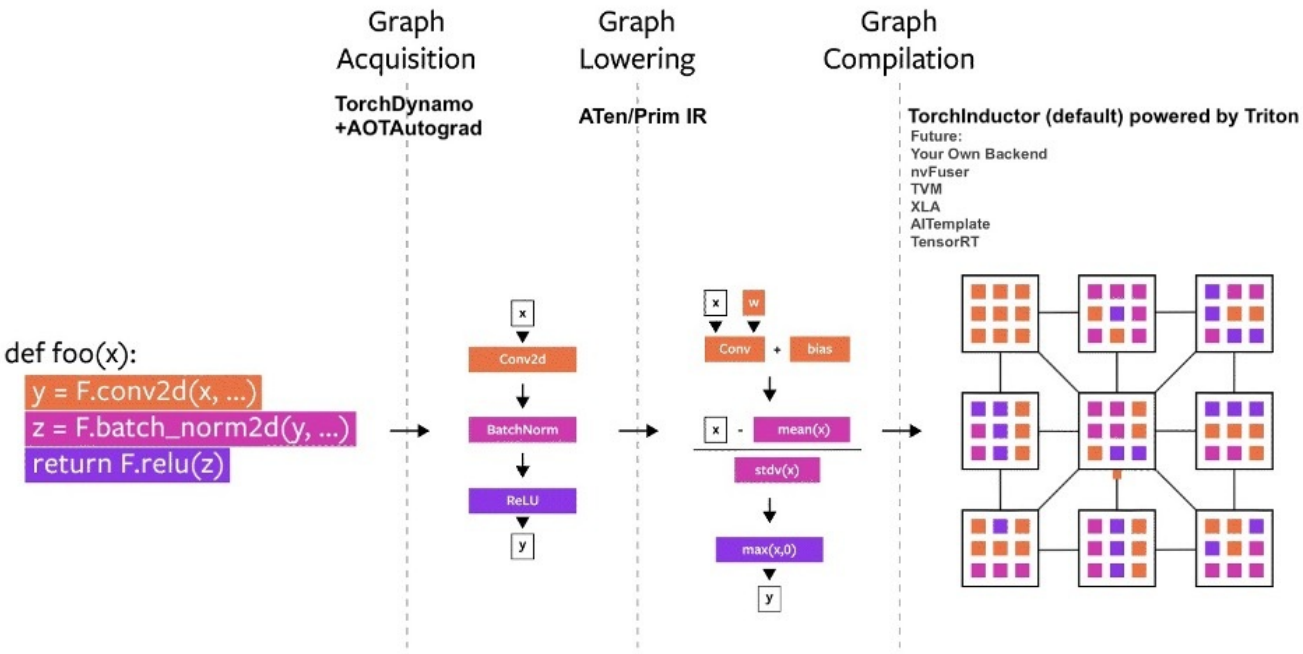


Image taken from PyTorch2.0 Introductory Blog

The compilation process is split into multiple steps which uses many new technologies that were introduced in PyTorch 2.0. The process is as follows:

1. **Graph Acquisition:** During the compilation process, TorchDynamo and AOTAutograd are used for capturing the forward and backward pass graphs respectively. AOTAutograd allows the backward graph to be captured ahead of time without needing a backward pass to be performed.
2. **Graph Lowering:** The captured graph that could be composed of the 2000+ PyTorch operators is lowered into a collection of ~250 Prim and ~750 ATen operators.
3. **Graph Compilation:** In this step optimised low-level kernels are generated for the target accelerator using a suitable backend compiler. TorchInductor is the default backend compiler used for this purpose.

Deepchem uses the `torch.compile()` function that implements all the above steps internally to compile the models. The compiled model can be used for training, evaluation and inference.

For more information on the compilation process, refer to PyTorch2.0 Introductory Blog that does a deep dive into the compilation process, technical decisions and future features for the compile function. You can also refer to the Huggingface blog, Optimize inference using torch.compile() that benchmarks many common PyTorch models and shows the performance improvements when compiled.

Compiling Models

The compile function is only available in DeepChem for models that use PyTorch as the backend (i.e inherits `TorchModel` class). You can see the complete list of models that are available in DeepChem and their backends in the DeepChem Documentation here.

This tutorial contains the steps to load a DeepChem model, compile it and evaluate the performance improvements when compiled for both training and inference. Refer to the documentation of DeepChem's compile function to read more about the different parameters you can pass to the function and their usage.

If you just want to compile the model, you can add the line `model.compile()` after initialising the model. You **DO NOT** have to make any changes to the rest of your code.

Some of the things to keep in mind when compiling models are:

1. **Selecting the right mode:** The modes can be `default`, `reduce-overhead`, `max-autotune` or `max-autotune-no-cudagraphs`. Out of this `reduce-overhead` and `max-autotune` modes requires `triton` to be installed. Refer to the PyTorch docs on `torch.compile` for more information on the modes.
2. **Setting `fullgraph` parameter:** If True (default `False`), `torch.compile` will require that the entire function be capturable into a single graph. If this is not possible (that is, if there are graph breaks), then the function will raise an error.
3. **Experimenting with different parameter configuration:** Different parameter configurations can give different speedups based on the model, batch size and the device used for training/inference. Experiment with a few

parameter combinations to check which one gives better results.

In this tutorial, we will be using DMPNN model and Freesolv Dataset for training and inference of the models.

In [ ]:
```
!pip install --pre deepchem
!pip install torch_geometric #required for DMPNN model
!pip install triton          #required for reduce-overhead mode
```

```
Collecting deepchem
  Downloading deepchem-2.8.1.dev20240624214143-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 6.9 MB/s eta 0:00:00
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.4.2)
Requirement already satisfied: numpy<2 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from deepchem) (2.0.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.2.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.12.1)
Requirement already satisfied: scipy>=1.10.1 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.11.4)
Collecting rdkit (from deepchem)
  Downloading rdkit-2024.3.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (35.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 35.1/35.1 MB 14.6 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem) (
2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem)
(2024.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from rdkit->deepchem) (9.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n->deepchem) (3.5.0)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->deep
chem) (1.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->deepchem) (1.16.0)
Installing collected packages: rdkit, deepchem
Successfully installed deepchem-2.8.1.dev20240624214143 rdkit-2024.3.1
Collecting torch_geometric
  Downloading torch_geometric-2.5.3-py3-none-any.whl (1.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.1/1.1 MB 10.8 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (4.66.4)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (1.11.4)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (2023.6.
0)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (3.1.4)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (3.9.5)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (2.31.
0)
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (3.1.
2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (1
.2.2)
Requirement already satisfied: psutil>=5.8.0 in /usr/local/lib/python3.10/dist-packages (from torch_geometric) (
5.9.5)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->torch_
geometric) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->torch_geo
metric) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->torch
_geometric) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->tor
ch_geometric) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->torch_ge
ometric) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp-
>torch_geometric) (4.0.3)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch_ge
ometric) (2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
s->torch_geometric) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->torch_geo
metric) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->tor
ch_geometric) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->tor
ch_geometric) (2024.6.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->torc
h_geometric) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n->torch_geometric) (3.5.0)
Installing collected packages: torch_geometric
Successfully installed torch_geometric-2.5.3
Requirement already satisfied: triton in /usr/local/lib/python3.10/dist-packages (2.3.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from triton) (3.15.3)
```

In [ ]:
```
import torch

import datetime
import numpy as np
import deepchem as dc

import matplotlib.pyplot as plt
```

```
WARNING:deepchem.feat.molecule_featurizers.rdkit_descriptors:No normalization for SPS. Feature removed!
WARNING:deepchem.feat.molecule_featurizers.rdkit_descriptors:No normalization for AvgIpc. Feature removed!
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/tensorflow/python/util/deprecation.py:588: calli
ng function (from tensorflow.python.eager.polymorphic_function.polymorphic_function) with experimental_relax_sha
pes is deprecated and will be removed in a future version.
Instructions for updating:
experimental_relax_shapes is deprecated, use reduce_retracing instead
WARNING:deepchem.models.torch_models:Skipped loading modules with pytorch-geometric dependency, missing a depend
ency. No module named 'dgl'
WARNING:deepchem.models:Skipped loading modules with pytorch-lightning dependency, missing a dependency. No modu
le named 'lightning'
WARNING:deepchem.models:Skipped loading some Jax models, missing a dependency. No module named 'haiku'
```

In [ ]:
```
torch._dynamo.config.cache_size_limit = 64
```

In [ ]:
```
tasks, datasets, transformers = dc.molnet.load_freesolv(featurizer=dc.feat.DMPNNFeaturizer(), splitter='random'
train_dataset, valid_dataset, test_dataset = datasets
len(train_dataset), len(valid_dataset), len(test_dataset)

model = dc.models.DMPNNModel()
```

The below line is the only addition you have to make to the code for compiling the model. You can pass in the other arguments too to the `compile()` function if they are required.

In [ ]:
```
model.compile()
```

```
/usr/lib/python3.10/multiprocessing/popen_fork.py:66: RuntimeWarning: os.fork() was called. os.fork() is incompa
tible with multithreaded code, and JAX is multithreaded, so this will likely lead to a deadlock.
  self.pid = os.fork()
```

In [ ]:
```
model.fit(train_dataset, nb_epoch=10)

metrics = [dc.metrics.Metric(dc.metrics.mean_squared_error)]
print(f"Training MSE: {model.evaluate(train_dataset, metrics=metrics)}")
print(f"Validation MSE: {model.evaluate(valid_dataset, metrics=metrics)}")
print(f"Test MSE: {model.evaluate(test_dataset, metrics=metrics)}")
```

```
Training MSE: {'mean_squared_error': 0.04699941161198689}
Validation MSE: {'mean_squared_error': 0.18010469643557037}
Test MSE: {'mean_squared_error': 0.043559911545479245}
```

Benchmarking model Speedups

This section contains the steps for benchmarking the performance of models after compilation process for both training and inference. We are using the same model(DMPNN) and dataset(FreSolv) in this section too. The steps for compilation and benchmarking is same for other models as well.

To account for the initial performance overhead of kernel compilation in compiled models, median values are employed as the performance metric throughout the tutorial for calculating speedup.

The below two functions, `time_torch_function` and `get_time_track_callback` can be used for tracking the time taken for inference and training respectively.

The implementation of `time_torch_function` is taken from the PyTorch official `torch.compile` tutorial here.

We use `get_time_track_callback` to make a callback that can track the time taken for each batch during training as DeepChem does not provide a direct way to track the time taken per batch during training. We can use this callback by passing it as an argument to `model.fit()` function.

In [ ]:
```
def time_torch_function(fn):
    start = torch.cuda.Event(enable_timing=True)
    end = torch.cuda.Event(enable_timing=True)
    start.record()
    result = fn()
    end.record()
    torch.cuda.synchronize()
    return result, start.elapsed_time(end) / 1000

track_dict = {}
prev_time_dict = {}
```

```
def get_time_track_callback(track_dict, track_name, track_interval):
    track_dict[track_name] = []
    prev_time_dict[track_name] = datetime.datetime.now()
    def callback(model, step):
        if step % track_interval == 0:
            elapsed_time = datetime.datetime.now() - prev_time_dict[track_name]
            track_dict[track_name].append(elapsed_time.total_seconds())
            prev_time_dict[track_name] = datetime.datetime.now()
    return callback
```

Tracking Training Time

In [ ]:
```
model = dc.models.DMPNNModel()
model_compiled = dc.models.DMPNNModel()
model_compiled.compile(mode='reduce-overhead')

track_interval = 20
eager_dict_name = "eager_train"
compiled_dict_name = "compiled_train"

eager_train_callback = get_time_track_callback(track_dict, eager_dict_name, track_interval)
model.fit(train_dataset, nb_epoch=10, callbacks=[eager_train_callback])

compiled_train_callback = get_time_track_callback(track_dict, compiled_dict_name, track_interval)
model_compiled.fit(train_dataset, nb_epoch=10, callbacks=[compiled_train_callback])
```

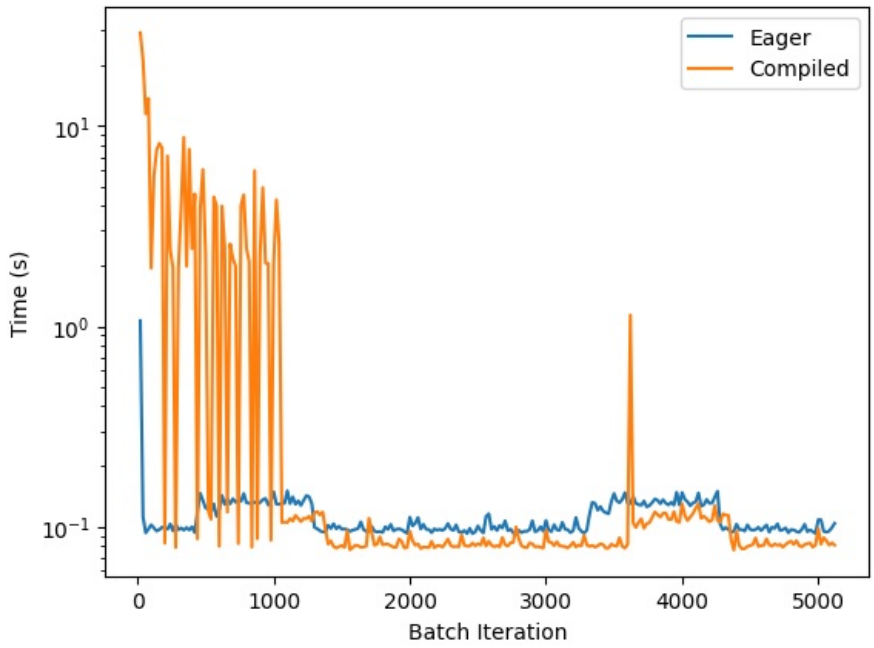
Out[ ]:
```
0.06506308714548746
```

In [ ]:
```
eager_train_times = track_dict[eager_dict_name]
compiled_train_times = track_dict[compiled_dict_name]

print(f"Eager Times (first 15): {[f'{t:.3f}' for t in eager_train_times[:15]]}")
print(f"Compiled Times (first 15): {[f'{t:.3f}' for t in compiled_train_times[:15]]}")
print(f"Total Eager Time: {sum(eager_train_times)}")
print(f"Total Compiled Time: {sum(compiled_train_times)}")
print(f"Eager Median: {np.median(eager_train_times)}")
print(f"Compiled Median: {np.median(compiled_train_times)}")
print(f"Median Speedup: {((np.median(eager_train_times) / np.median(compiled_train_times)) - 1) * 100:.2f}%")
```

```
Eager Times (first 15): ['1.067', '0.112', '0.093', '0.097', '0.102', '0.098', '0.095', '0.097', '0.099', '0.098
', '0.097', '0.103', '0.095', '0.103', '0.096']
Compiled Times (first 15): ['29.184', '21.463', '11.503', '13.742', '1.951', '5.595', '7.568', '8.201', '7.761',
'0.083', '7.087', '2.421', '1.961', '0.079', '1.948']
Total Eager Time: 29.176121000000023
Total Compiled Time: 243.32460400000022
Eager Median: 0.100118
Compiled Median: 0.0843535
Median Speedup: 18.69%
```

In [ ]:
```
x_vals = np.arange(1, len(eager_train_times) + 1) * track_interval
plt.plot(x_vals, eager_train_times, label="Eager")
plt.plot(x_vals, compiled_train_times, label="Compiled")
plt.yscale('log', base= 10)
plt.ylabel('Time (s)')
plt.xlabel('Batch Iteration')
plt.legend()
plt.show()
```



Looking at the graph, there is a significant difference in the time taken for the compiled and uncompiled versions of the model for the starting many steps. After that the time taken by the compiled model stabilises below the uncompiled model. This is because the compilation is done JIT when the model is first run and the optimized kernels are generated after a few passes.

Tracking Inference Time

In [ ]:
```
model = dc.models.DMPNNModel()
model_compiled = dc.models.DMPNNModel()
model_compiled.compile(mode='reduce-overhead')

iters = 100

eager_predict_times = []
compiled_predict_times = []

for i in range(iters):
    for X, y, w, ids in test_dataset.iterbatches(64, pad_batches=True):
        with torch.no_grad():
            _, eager_time = time_torch_function(lambda: model.predict_on_batch(X))
            _, compiled_time = time_torch_function(lambda: model_compiled.predict_on_batch(X))

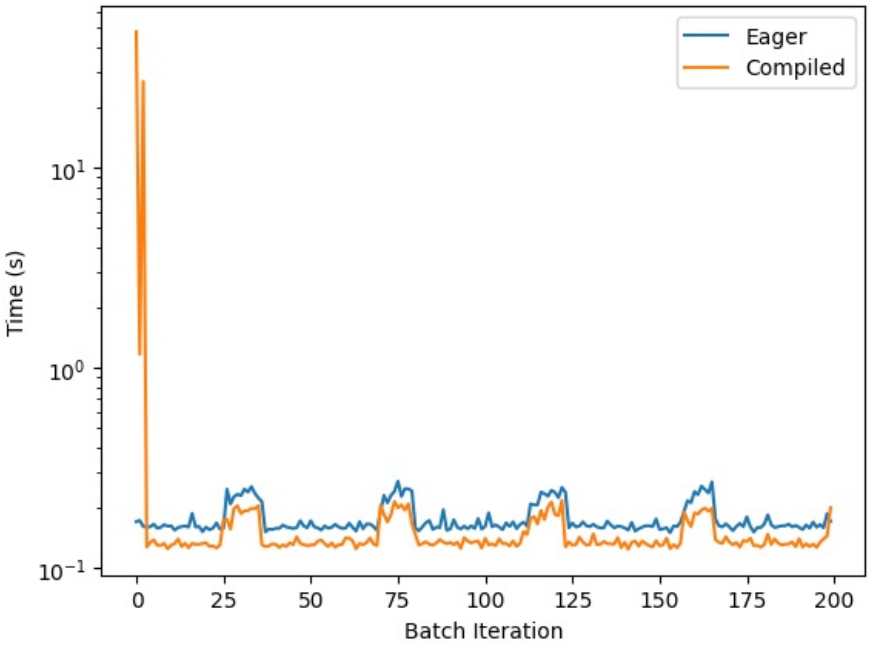
        eager_predict_times.append(eager_time)
        compiled_predict_times.append(compiled_time)
```

In [ ]:
```
print(f"Eager Times (first 15): {[f'{t:.3f}' for t in eager_predict_times[:15]]}")
print(f"Compiled Times (first 15): {[f'{t:.3f}' for t in compiled_predict_times[:15]]}")
print(f"Total Eager Time: {sum(eager_predict_times)}")
print(f"Total Compiled Time: {sum(compiled_predict_times)}")
print(f"Eager Median: {np.median(eager_predict_times)}")
print(f"Compiled Median: {np.median(compiled_predict_times)}")
print(f"Median Speedup: {((np.median(eager_predict_times) / np.median(compiled_predict_times)) - 1) * 100:.2f}%
```

```
Eager Times (first 15): ['0.170', '0.173', '0.161', '0.160', '0.160', '0.165', '0.158', '0.159', '0.164', '0.161
', '0.162', '0.154', '0.159', '0.161', '0.162']
Compiled Times (first 15): ['47.617', '1.168', '26.927', '0.127', '0.134', '0.138', '0.130', '0.130', '0.133', '
0.125', '0.130', '0.132', '0.139', '0.128', '0.133']
Total Eager Time: 35.297711242675796
Total Compiled Time: 104.20891365814221
Eager Median: 0.1617226104736328
Compiled Median: 0.1332385482788086
Median Speedup: 21.38%
```

In [ ]:
```
plt.plot(eager_predict_times, label= "Eager")
plt.plot(compiled_predict_times, label= "Compiled")
plt.ylabel('Time (s)')
plt.xlabel('Batch Iteration')
plt.yscale('log', base= 10)
plt.legend()
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x7c7a040c9c30>
```



As with the results we got training, the first few runs for inference also takes significantly more time due to the same reason as mentioned before.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Molecular Fingerprints

Molecules can be represented in many ways. This tutorial introduces a type of representation called a "molecular fingerprint". It is a very simple representation that often works well for small drug-like molecules.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
```

We can now import the `deepchem` package to play with.

In [1]:
```
import deepchem as dc
dc.__version__
```

Out[1]:
```
'2.4.0-rc1.dev'
```

What is a Fingerprint?

Deep learning models almost always take arrays of numbers as their inputs. If we want to process molecules with them, we somehow need to represent each molecule as one or more arrays of numbers.

Many (but not all) types of models require their inputs to have a fixed size. This can be a challenge for molecules, since different molecules have different numbers of atoms. If we want to use these types of models, we somehow need to represent variable sized molecules with fixed sized arrays.

Fingerprints are designed to address these problems. A fingerprint is a fixed length array, where different elements indicate the presence of different features in the molecule. If two molecules have similar fingerprints, that indicates they contain many of the same features, and therefore will likely have similar chemistry.

DeepChem supports a particular type of fingerprint called an "Extended Connectivity Fingerprint", or "ECFP" for short. They also are sometimes called "circular fingerprints". The ECFP algorithm begins by classifying atoms based only on their direct properties and bonds. Each unique pattern is a feature. For example, "carbon atom bonded to two hydrogens and two heavy atoms" would be a feature, and a particular element of the fingerprint is set to 1 for any molecule that contains that feature. It then iteratively identifies new features by looking at larger circular neighborhoods. One specific feature bonded to two other specific features becomes a higher level feature, and the corresponding element is set for any molecule that contains it. This continues for a fixed number of iterations, most often two.

Let's take a look at a dataset that has been featurized with ECFP.

In [2]:
```
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='ECFP')
train_dataset, valid_dataset, test_dataset = datasets
print(train_dataset)
```

```
<DiskDataset X.shape: (6264, 1024), y.shape: (6264, 12), w.shape: (6264, 12), task_names: ['NR-AR' 'NR-AR-LBD' '
NR-AhR' ... 'SR-HSE' 'SR-MMP' 'SR-p53']>
```

The feature array `X` has shape (6264, 1024). That means there are 6264 samples in the training set. Each one is represented by a fingerprint of length 1024. Also notice that the label array `y` has shape (6264, 12): this is a multitask dataset. Tox21 contains information about the toxicity of molecules. 12 different assays were used to look for signs of toxicity. The dataset records the results of all 12 assays, each as a different task.

Let's also take a look at the weights array.

In [3]:
```
train_dataset.w
```

Out[3]:
```
array([[1.0433141624730409, 1.0369942196531792, 8.53921568627451, ...,
        1.060388945752303, 1.1895710249165168, 1.0700990099009902],
       [1.0433141624730409, 1.0369942196531792, 1.1326397919375812, ...,
        0.0, 1.1895710249165168, 1.0700990099009902],
       [0.0, 0.0, 0.0, ..., 1.060388945752303, 0.0, 0.0],
       ...,
       [0.0, 0.0, 0.0, ..., 0.0, 0.0, 0.0],
       [1.0433141624730409, 1.0369942196531792, 8.53921568627451, ...,
        1.060388945752303, 0.0, 0.0],
       [1.0433141624730409, 1.0369942196531792, 1.1326397919375812, ...,
        1.060388945752303, 1.1895710249165168, 1.0700990099009902]],
      dtype=object)
```

Notice that some elements are 0. The weights are being used to indicate missing data. Not all assays were actually performed on every molecule. Setting the weight for a sample or sample/task pair to 0 causes it to be ignored during fitting and evaluation. It will have no effect on the loss function or other metrics.

Most of the other weights are close to 1, but not exactly 1. This is done to balance the overall weight of positive and negative samples on each task. When training the model, we want each of the 12 tasks to contribute equally, and on each task we want to put equal weight on positive and negative samples. Otherwise, the model might just learn that most of the training samples are non-toxic, and therefore become biased toward identifying other molecules as non-toxic.

Training a Model on Fingerprints

Let's train a model. In earlier tutorials we use `GraphConvModel`, which is a fairly complicated architecture that takes a complex set of inputs. Because fingerprints are so simple, just a single fixed length array, we can use a much simpler type of model.

In [4]:
```
model = dc.models.MultitaskClassifier(n_tasks=12, n_features=1024, layer_sizes=[1000])
```

`MultitaskClassifier` is a simple stack of fully connected layers. In this example we tell it to use a single hidden layer of width 1000. We also tell it that each input will have 1024 features, and that it should produce predictions for 12 different tasks.

Why not train a separate model for each task? We could do that, but it turns out that training a single model for multiple tasks often works better. We will see an example of that in a later tutorial.

Let's train and evaluate the model.

In [5]:
```
import numpy as np

model.fit(train_dataset, nb_epoch=10)
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
print('training set score:', model.evaluate(train_dataset, [metric], transformers))
print('test set score:', model.evaluate(test_dataset, [metric], transformers))
```

```
training set score: {'roc_auc_score': 0.9550063590563469}
test set score: {'roc_auc_score': 0.7781819573695475}
```

Not bad performance for such a simple model and featurization. More sophisticated models do slightly better on this dataset, but not enormously better.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro4,
 title={Molecular Fingerprints},
 organization={DeepChem},
 author={Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Molecular_Fingerprints
 year={2021},
}
```

Going Deeper On Molecular Featurizations

One of the most important steps of doing machine learning on molecular data is transforming the data into a form amenable to the application of learning algorithms. This process is broadly called "featurization" and involves turning a molecule into a vector or tensor of some sort. There are a number of different ways of doing that, and the choice of featurization is often dependent on the problem at hand. We have already seen two such methods: molecular fingerprints, and `ConvMol` objects for use with graph convolutions. In this tutorial we will look at some of the others.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [1]:
```
!pip install -qq --pre deepchem
import deepchem
import warnings
warnings.filterwarnings('ignore')
deepchem.__version__
```

Out[1]:
```
'2.6.0.dev'
```

Featurizers

In DeepChem, a method of featurizing a molecule (or any other sort of input) is defined by a `Featurizer` object. There are three different ways of using featurizers.

1. When using the MoleculeNet loader functions, you simply pass the name of the featurization method to use. We have seen examples of this in earlier tutorials, such as `featurizer='ECFP'` or `featurizer='GraphConv'`.

2. You also can create a Featurizer and directly apply it to molecules. For example:

In [2]:
```
import deepchem as dc

featurizer = dc.feat.CircularFingerprint()
print(featurizer(['CC', 'CCC', 'CCO']))
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
[09:05:59] DEPRECATION WARNING: please use MorganGenerator
[09:05:59] DEPRECATION WARNING: please use MorganGenerator
[09:05:59] DEPRECATION WARNING: please use MorganGenerator
```

3. When creating a new dataset with the DataLoader framework, you can specify a Featurizer to use for processing the data. We will see this in a future tutorial.

We use propane ($CH_3CH_2CH_3$, represented by the SMILES string `'CCC'`) as a running example throughout this tutorial. Many of the featurization methods use conformers of the molecules. A conformer can be generated using the `ConformerGenerator` class in `deepchem.utils.conformers`.

RDKitDescriptors

`RDKitDescriptors` featurizes a molecule by using RDKit to compute values for a list of descriptors. These are basic physical and chemical properties: molecular weight, polar surface area, numbers of hydrogen bond donors and acceptors, etc. This is most useful for predicting things that depend on these high level properties rather than on detailed molecular structure.

Intrinsic to the featurizer is a set of allowed descriptors, which can be accessed using `RDKitDescriptors.allowedDescriptors`. The featurizer uses the descriptors in `rdkit.Chem.Descriptors.descList`, checks if they are in the list of allowed descriptors, and computes the descriptor value for the molecule.

Let's print the values of the first ten descriptors for propane.

In [3]:
```
from rdkit.Chem.Descriptors import descList
```

```
rdkit_featurizer = dc.feat.RDKitDescriptors()
features = rdkit_featurizer(['CCC'])[0]
descriptors = [i[0] for i in descList]
for feature, descriptor in zip(features[:10], descriptors):
    print(descriptor, feature)
```

```
MaxAbsEStateIndex 2.125
MaxEStateIndex 2.125
MinAbsEStateIndex 1.25
MinEStateIndex 1.25
qed 0.3854706587740357
SPS 6.0
MolWt 44.097
HeavyAtomMolWt 36.033
ExactMolWt 44.062600255999996
NumValenceElectrons 20.0
[09:07:17] DEPRECATION WARNING: please use MorganGenerator
[09:07:17] DEPRECATION WARNING: please use MorganGenerator
[09:07:17] DEPRECATION WARNING: please use MorganGenerator
```

Of course, there are many more descriptors than this.

In [4]:
```
print('The number of descriptors present is: ', len(features))
```

```
The number of descriptors present is:  210
```

WeaveFeaturizer and MolGraphConvFeaturizer

We previously looked at graph convolutions, which use `ConvMolFeaturizer` to convert molecules into `ConvMol` objects. Graph convolutions are a special case of a large class of architectures that represent molecules as graphs. They work in similar ways but vary in the details. For example, they may associate data vectors with the atoms, the bonds connecting them, or both. They may use a variety of techniques to calculate new data vectors from those in the previous layer, and a variety of techniques to compute molecule level properties at the end.

DeepChem supports lots of different graph based models. Some of them require molecules to be featurized in slightly different ways. Because of this, there are two other featurizers called `WeaveFeaturizer` and `MolGraphConvFeaturizer`. They each convert molecules into a different type of Python object that is used by particular models. When using any graph based model, just check the documentation to see what featurizer you need to use with it.

CoulombMatrix

All the models we have looked at so far consider only the intrinsic properties of a molecule: the list of atoms that compose it and the bonds connecting them. When working with flexible molecules, you may also want to consider the different conformations the molecule can take on. For example, when a drug molecule binds to a protein, the strength of the binding depends on specific interactions between pairs of atoms. To predict binding strength, you probably want to consider a variety of possible conformations and use a model that takes them into account when making predictions.

The Coulomb matrix is one popular featurization for molecular conformations. Recall that the electrostatic Coulomb interaction between two charges is proportional to $q_1 q_2 / r$ where q_1 and q_2 are the charges and r is the distance between them. For a molecule with N atoms, the Coulomb matrix is a $N \times N$ matrix where each element gives the strength of the electrostatic interaction between two atoms. It contains information both about the charges on the atoms and the distances between them. More information on the functional forms used can be found here.

To apply this featurizer, we first need a set of conformations for the molecule. We can use the `ConformerGenerator` class to do this. It takes a RDKit molecule, generates a set of energy minimized conformers, and prunes the set to only include ones that are significantly different from each other. Let's try running it for propane.

In [5]:
```
from rdkit import Chem

generator = dc.utils.ConformerGenerator(max_conformers=5)
propane_mol = generator.generate_conformers(Chem.MolFromSmiles('CCC'))
print("Number of available conformers for propane: ", len(propane_mol.GetConformers()))
```

```
Number of available conformers for propane:  1
```

It only found a single conformer. This shouldn't be surprising, since propane is a very small molecule with hardly any flexibility. Let's try adding another carbon.

In [6]:
```
butane_mol = generator.generate_conformers(Chem.MolFromSmiles('CCCC'))
print("Number of available conformers for butane: ", len(butane_mol.GetConformers()))
```

```
Number of available conformers for butane:  3
```

Now we can create a Coulomb matrix for our molecule.

In [7]:
```
coulomb_mat = dc.feat.CoulombMatrix(max_atoms=20)
features = coulomb_mat(propane_mol)
print(features)
```

```
[[[36.8581052  12.48684429  7.5619687   2.85945193  2.85804514
    2.85804556  1.4674015   1.46740144  0.91279491  1.14239698
    1.14239675  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [12.48684429 36.8581052  12.48684388  1.46551218  1.45850736
    1.45850732  2.85689525  2.85689538  1.4655122   1.4585072
    1.4585072   0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 7.5619687  12.48684388 36.8581052   0.9127949   1.14239695
    1.14239692  1.46740146  1.46740145  2.85945178  2.85804504
    2.85804493  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 2.85945193  1.46551218  0.9127949   0.5         0.29325367
    0.29325369  0.21256978  0.21256978  0.12268391  0.13960187
    0.13960185  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 2.85804514  1.45850736  1.14239695  0.29325367  0.5
    0.29200271  0.17113413  0.21092513  0.13960186  0.1680002
    0.20540029  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 2.85804556  1.45850732  1.14239692  0.29325369  0.29200271
    0.5         0.21092513  0.17113413  0.13960187  0.20540032
    0.16800016  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.4674015   2.85689525  1.46740146  0.21256978  0.17113413
    0.21092513  0.5         0.29351308  0.21256981  0.2109251
    0.17113412  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.46740144  2.85689538  1.46740145  0.21256978  0.21092513
    0.17113413  0.29351308  0.5         0.21256977  0.17113412
    0.21092513  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.91279491  1.4655122   2.85945178  0.12268391  0.13960186
    0.13960187  0.21256981  0.21256977  0.5         0.29325366
    0.29325365  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.14239698  1.4585072   2.85804504  0.13960187  0.1680002
    0.20540032  0.2109251   0.17113412  0.29325366  0.5
    0.29200266  0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 1.14239675  1.4585072   2.85804493  0.13960185  0.20540029
    0.16800016  0.17113412  0.21092513  0.29325365  0.29200266
    0.5         0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]
  [ 0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.
    0.          0.          0.          0.          0.        ]]]
```

Notice that many elements are 0. To combine multiple molecules in a batch we need all the Coulomb matrices to be the same size, even if the molecules have different numbers of atoms. We specified `max_atoms=20`, so the returned matrix

has size (20, 20). The molecule only has 11 atoms, so only an 11 by 11 submatrix is nonzero.

CoulombMatrixEig

An important feature of Coulomb matrices is that they are invariant to molecular rotation and translation, since the interatomic distances and atomic numbers do not change. Respecting symmetries like this makes learning easier. Rotating a molecule does not change its physical properties. If the featurization does change, then the model is forced to learn that rotations are not important, but if the featurization is invariant then the model gets this property automatically.

Coulomb matrices are not invariant under another important symmetry: permutations of the atoms' indices. A molecule's physical properties do not depend on which atom we call "atom 1", but the Coulomb matrix does. To deal with this, the `CoulombMatrixEig` featurizer was introduced, which uses the eigenvalue spectrum of the Coulumb matrix and is invariant to random permutations of the atom's indices. The disadvantage of this featurization is that it contains much less information (N eigenvalues instead of an $N \times N$ matrix), so models will be more limited in what they can learn.

`CoulombMatrixEig` inherits from `CoulombMatrix` and featurizes a molecule by first computing the Coulomb matrices for different conformers of the molecule and then computing the eigenvalues for each Coulomb matrix. These eigenvalues are then padded to account for variation in number of atoms across molecules.

In [8]:
```
coulomb_mat_eig = dc.feat.CoulombMatrixEig(max_atoms=20)
features = coulomb_mat_eig(propane_mol)
print(features)
```

```
[[60.07620303 29.62963149 22.75497781  0.5713786   0.28781332  0.28548338
  0.27558187  0.18163794  0.17460999  0.17059719  0.16640098  0.
  0.          0.          0.          0.          0.          0.
  0.          0.        ]]
```

SMILES Tokenization and Numericalization

So far, we have looked at featurization techniques that translate the implicit structural and physical information in SMILES data into more explicit features that help our models learn and make predictions. In this section, we will preprocess SMILES strings into a format that can be fed to sequence models, such as 1D convolutional neural networks and transformers, and enables these models to learn their own representations of molecular properties.

To prepare SMILES strings for a sequence model, we break them down into lists of substrings (called tokens) and turn them into lists of integer values (numericalization). Sequence models use those integer values as indices of an embedding matrix, which contains a vector of floating-point numbers for each token in the vocabulary. These embedding vectors are updated during model training. This process allows the sequence model to learn its own representations of the molecular properties implicit in the training data.

We will use DeepChem's `BasicSmilesTokenizer` and the Tox21 dataset from MoleculeNet to demonstrate the process of tokenizing SMILES.

In [9]:
```
import numpy as np
```

In [10]:
```
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer="Raw")
train_dataset, valid_dataset, test_dataset = datasets
print(train_dataset)
```

```
<DiskDataset X.shape: (6264,), y.shape: (6264, 12), w.shape: (6264, 12), task_names: ['NR-AR' 'NR-AR-LBD' 'NR-AhR' ... 'SR-HSE' 'SR-MMP' 'SR-p53']>
```

We loaded the datasets with `featurizer="Raw"`. Now we obtain the SMILES from their `ids` attributes.

In [11]:
```
train_smiles = train_dataset.ids
valid_smiles = valid_dataset.ids
test_smiles = test_dataset.ids
print(train_smiles[:5])
```

```
['CC(O)(P(=O)(O)O)P(=O)(O)O' 'CC(C)(C)OOC(C)(C)CCC(C)(C)OOC(C)(C)C'
 'OC[C@H](O)[C@@H](O)[C@H](O)CO'
 'CCCCCCCC(=O)[O-].CCCCCCCC(=O)[O-].[Zn+2]' 'CC(C)COC(=O)C(C)C']
```

Next we define our tokenizer and `map` it onto all our data to convert the SMILES strings into lists of tokens. The `BasicSmilesTokenizer` breaks down SMILES roughly at atom level.

In [12]:
```
tokenizer = dc.feat.smiles_tokenizer.BasicSmilesTokenizer()
```

```
train_tok = list(map(tokenizer.tokenize, train_smiles))
valid_tok = list(map(tokenizer.tokenize, valid_smiles))
test_tok = list(map(tokenizer.tokenize, test_smiles))
print(train_tok[0])
len(train_tok)
```

```
['C', 'C', '(', 'O', ')', '(', 'P', '(', '=', 'O', ')', '(', 'O', ')', 'O', ')', 'P', '(', '=', 'O', ')', '(', '
O', ')', 'O']
```

Out[12]: 6264

Now we have tokenized versions of all SMILES strings in our dataset. To convert those into lists of integer values we first need to create a list of all possible tokens in our dataset. That list is called the vocabulary. We also add the empty string `""` to our vocabulary in order to correctly handle trailing zeros when decoding zero-padded numericalized SMILES.

In [13]:
```
flatten = lambda l: [item for items in l for item in items]

all_toks = flatten(train_tok) + flatten(valid_tok) + flatten(test_tok)
vocab = sorted(set(all_toks + [""]))
print(vocab[:12], "...", vocab[-12:])
len(vocab)
```

```
['', '#', '(', ')', '-', '.', '/', '1', '2', '3', '4', '5'] ... ['[n+]', '[n-]', '[nH+]', '[nH]', '[o+]', '[s+]'
, '[se]', '\\', 'c', 'n', 'o', 's']
```

Out[13]: 128

To numericalize tokenized SMILES strings we create a `str2int` dictionary which assigns a number to each token in the dictionary. We also create the reverse `int2str` dictionary and define the corresponding `encode` and `decode` functions. Finally we `map` the `encode` function on the tokenized data to obtain numericalized SMILES data.

In [14]:
```
str2int = {s:i for i, s in enumerate(vocab)}
int2str = {i:s for i, s in enumerate(vocab)}
print(f"str2int: {dict(list(str2int.items())[:5])} ...")
print(f"int2str: {dict(list(int2str.items())[:5])} ...")
```

```
str2int: {'': 0, '#': 1, '(': 2, ')': 3, '-': 4} ...
int2str: {0: '', 1: '#', 2: '(', 3: ')', 4: '-'} ...
```

In [15]:
```
encode = lambda s: [str2int[tok] for tok in s]
decode = lambda i: [int2str[num] for num in i]
print(train_smiles[0])
print(encode(train_tok[0]))
print("".join(decode(encode(train_tok[0]))))
```

```
CC(O)(P(=O)(O)O)P(=O)(O)O
[19, 19, 2, 24, 3, 2, 25, 2, 16, 24, 3, 2, 24, 3, 24, 3, 25, 2, 16, 24, 3, 2, 24, 3, 24]
CC(O)(P(=O)(O)O)P(=O)(O)O
```

In [16]:
```
train_num = list(map(encode, train_tok))
valid_num = list(map(encode, valid_tok))
test_num = list(map(encode, test_tok))
print(train_num[0])
```

```
[19, 19, 2, 24, 3, 2, 25, 2, 16, 24, 3, 2, 24, 3, 24, 3, 25, 2, 16, 24, 3, 2, 24, 3, 24]
```

Lastly, we would like to combine all molecules in a dataset in an `np.array` so they can be served to a model in batches. To achieve that, all sequences have to be of the same length. As in the CoulombMatrix section, we achieve that by appending zeros up to a fixed value.

In [17]:
```
max_len = max(map(len, train_num + valid_num + test_num))
max_len
```

Out[17]: 240

The longest sequence across all Tox21 datasets has length `240`, so we use that as our fixed length. We create a `zero_pad` function, `map` it to all numericalized SMILES, and turn them into `np.array`s.

In [18]:
```
zero_pad = lambda x: x + [0] * (max_len - len(x))

train_numpad = np.array(list(map(zero_pad, train_num)))
valid_numpad = np.array(list(map(zero_pad, valid_num)))
test_numpad = np.array(list(map(zero_pad, test_num)))
train_numpad
```

Out[18]:
```
array([[19, 19,  2, ...,  0,  0,  0],
       [19, 19,  2, ...,  0,  0,  0],
       [24, 19, 42, ...,  0,  0,  0],
       ...,
       [24, 16, 19, ...,  0,  0,  0],
       [19, 19,  2, ...,  0,  0,  0],
       [19, 19,  2, ...,  0,  0,  0]])
```

We can check that the zero-padded data still converts back to the correct SMILES string using the `decode` function on a random datapoint.

In [19]:
```
idx = np.random.randint(0, train_numpad.shape[0], 1).item()
print(train_smiles[idx])
print("".join(decode(train_numpad[idx])))
```

```
Cc1cc(C(C)(C)c2ccc(O)c(C)c2)ccc1O
Cc1cc(C(C)(C)c2ccc(O)c(C)c2)ccc1O
```

The padded data passes the test. It is now in the correct format to be used for training of a sequence model, but it doesn't yet interface nicely with DeepChem's training framework. To change that, we define a `tokenize_smiles` function that combines all the steps spelled out above to process a single datapoint. Additionally, we define a `SmilesFeaturizer` that uses our custom `tokenize_smiles` function in its `_featurize` method and instanciate it as `smiles_featurizer` passing it our `vocab` and `max_len`.

In [20]:
```
def tokenize_smiles(x, vocab, max_len):
    tokenizer = dc.feat.smiles_tokenizer.BasicSmilesTokenizer()
    str2int = {s:i for i, s in enumerate(vocab)}
    encode = lambda s: [str2int[tok] for tok in s]
    zero_pad = lambda x: x + [0] * (max_len - len(x))
    x = tokenizer.tokenize(x)
    x = encode(x)
    x = zero_pad(x)
    return np.array(x)

class SmilesFeaturizer(dc.feat.Featurizer):
    def __init__(self, feat_func, vocab, max_len):
        self.feat_func = feat_func
        self.vocab = vocab
        self.max_len = max_len

    def _featurize(self, x):
        return self.feat_func(x, self.vocab, self.max_len)

smiles_featurizer = SmilesFeaturizer(tokenize_smiles, vocab, max_len)
```

Finally, we use the `smiles_featurizer` to create new Tox21 datasets that contain tokenized and numericalized SMILES in their `X` attribute.

In [21]:
```
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer=smiles_featurizer)
print(datasets[0].X)
```

```
[09:24:48] WARNING: not removing hydrogen atom without neighbors
[[19 19  2 ...  0  0  0]
 [19 19  2 ...  0  0  0]
 [24 19 42 ...  0  0  0]
 ...
 [24 16 19 ...  0  0  0]
 [19 19  2 ...  0  0  0]
 [19 19  2 ...  0  0  0]]
```

The datasets are now ready to be used with your custom DeepChem sequence model. Don't forget to wrap your model into the appropriate DeepChem model class.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Intro7,
 title={Going Deeper on Molecular Featurizations},
 organization={DeepChem},
 author={Ramsundar, Bharath},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Going_Deeper_on_Molecu
 year={2021},
}
```

Learning Unsupervised Embeddings for Molecules

In this tutorial, we will use a `SeqToSeq` model to generate fingerprints for classifying molecules. This is based on the following paper, although some of the implementation details are different: Xu et al., "Seq2seq Fingerprint: An Unsupervised Deep Molecular Embedding for Drug Discovery" (https://doi.org/10.1145/3107411.3107424).

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

Learning Embeddings with SeqToSeq

Many types of models require their inputs to have a fixed shape. Since molecules can vary widely in the numbers of atoms and bonds they contain, this makes it hard to apply those models to them. We need a way of generating a fixed length "fingerprint" for each molecule. Various ways of doing this have been designed, such as the Extended-Connectivity Fingerprints (ECFPs) we used in earlier tutorials. But in this example, instead of designing a fingerprint by hand, we will let a `SeqToSeq` model learn its own method of creating fingerprints.

A `SeqToSeq` model performs sequence to sequence translation. For example, they are often used to translate text from one language to another. It consists of two parts called the "encoder" and "decoder". The encoder is a stack of recurrent layers. The input sequence is fed into it, one token at a time, and it generates a fixed length vector called the "embedding vector". The decoder is another stack of recurrent layers that performs the inverse operation: it takes the embedding vector as input, and generates the output sequence. By training it on appropriately chosen input/output pairs, you can create a model that performs many sorts of transformations.

In this case, we will use SMILES strings describing molecules as the input sequences. We will train the model as an autoencoder, so it tries to make the output sequences identical to the input sequences. For that to work, the encoder must create embedding vectors that contain all information from the original sequence. That's exactly what we want in a fingerprint, so perhaps those embedding vectors will then be useful as a way to represent molecules in other models!

Let's start by loading the data. We will use the MUV dataset. It includes 74,501 molecules in the training set, and 9313 molecules in the validation set, so it gives us plenty of SMILES strings to work with.

In [1]:
```
import deepchem as dc
tasks, datasets, transformers = dc.molnet.load_muv(split='stratified')
train_dataset, valid_dataset, test_dataset = datasets
train_smiles = train_dataset.ids
valid_smiles = valid_dataset.ids
```

We need to define the "alphabet" for our `SeqToSeq` model, the list of all tokens that can appear in sequences. (It's also possible for input and output sequences to have different alphabets, but since we're training it as an autoencoder, they're identical in this case.) Make a list of every character that appears in any training sequence.

In [2]:
```
tokens = set()
for s in train_smiles:
  tokens = tokens.union(set(c for c in s))
tokens = sorted(list(tokens))
```

Create the model and define the optimization method to use. In this case, learning works much better if we gradually decrease the learning rate. We use an `ExponentialDecay` to multiply the learning rate by 0.9 after each epoch.

In [3]:
```
from deepchem.models.optimizers import Adam, ExponentialDecay
max_length = max(len(s) for s in train_smiles)
batch_size = 100
batches_per_epoch = len(train_smiles)/batch_size
model = dc.models.SeqToSeq(tokens,
                           tokens,
                           max_length,
                           encoder_layers=2,
                           decoder_layers=2,
                           embedding_dimension=256,
```

```
                          model_dir='fingerprint',
                          batch_size=batch_size,
                          learning_rate=ExponentialDecay(0.001, 0.9, batches_per_epoch))
```

Let's train it! The input to `fit_sequences()` is a generator that produces input/output pairs. On a good GPU, this should take a few hours or less.

In [4]:
```
def generate_sequences(epochs):
  for i in range(epochs):
    for s in train_smiles:
      yield (s, s)

model.fit_sequences(generate_sequences(40))
```

Let's see how well it works as an autoencoder. We'll run the first 500 molecules from the validation set through it, and see how many of them are exactly reproduced.

In [5]:
```
predicted = model.predict_from_sequences(valid_smiles[:500])
count = 0
for s,p in zip(valid_smiles[:500], predicted):
  if ''.join(p) == s:
    count += 1
print('reproduced', count, 'of 500 validation SMILES strings')
```

```
reproduced 161 of 500 validation SMILES strings
```

Now we'll trying using the encoder as a way to generate molecular fingerprints. We compute the embedding vectors for all molecules in the training and validation datasets, and create new datasets that have those as their feature vectors. The amount of data is small enough that we can just store everything in memory.

In [6]:
```
import numpy as np
train_embeddings = model.predict_embeddings(train_smiles)
train_embeddings_dataset = dc.data.NumpyDataset(train_embeddings,
                                                train_dataset.y,
                                                train_dataset.w.astype(np.float32),
                                                train_dataset.ids)

valid_embeddings = model.predict_embeddings(valid_smiles)
valid_embeddings_dataset = dc.data.NumpyDataset(valid_embeddings,
                                                valid_dataset.y,
                                                valid_dataset.w.astype(np.float32),
                                                valid_dataset.ids)
```

For classification, we'll use a simple fully connected network with one hidden layer.

In [7]:
```
classifier = dc.models.MultitaskClassifier(n_tasks=len(tasks),
                                                      n_features=256,
                                                      layer_sizes=[512])
classifier.fit(train_embeddings_dataset, nb_epoch=10)
```

Out[7]:
```
0.0014195525646209716
```

Find out how well it worked. Compute the ROC AUC for the training and validation datasets.

In [8]:
```
metric = dc.metrics.Metric(dc.metrics.roc_auc_score, np.mean, mode="classification")
train_score = classifier.evaluate(train_embeddings_dataset, [metric], transformers)
valid_score = classifier.evaluate(valid_embeddings_dataset, [metric], transformers)
print('Training set ROC AUC:', train_score)
print('Validation set ROC AUC:', valid_score)
```

```
Training set ROC AUC: {'mean-roc_auc_score': 0.9598792603154332}
Validation set ROC AUC: {'mean-roc_auc_score': 0.7251350862464794}
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Synthetic Feasibility

Synthetic feasibility is a problem when running large scale enumerations. Often molecules that are enumerated are very difficult to make and thus not worth inspection, even if their other chemical properties are good in silico. This tutorial goes through how to train the ScScore model [1].

The idea of the model is to train on pairs of molecules where one molecule is "more complex" than the other. The neural network then can make scores which attempt to keep this pairwise ordering of molecules. The final result is a model which can give a relative complexity of a molecule.

The paper trains on every reaction in reaxys, declaring products more complex than reactions. Since this training set is prohibitively expensive we will instead train on arbitrary molecules declaring one more complex if its SMILES string is longer. In the real world you can use whatever measure of complexity makes sense for the project.

In this tutorial, we'll use the Tox21 dataset to train our simple synthetic feasibility model.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

Make The Datasets

Let's begin by loading some molecules to work with. We load Tox21, specifying `splitter=None` so everything will be returned as a single dataset.

In [1]:
```
import deepchem as dc
tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='Raw', splitter=None)
molecules = datasets[0].X
```

Because ScScore is trained on relative complexities, we want the `X` tensor in our dataset to have 3 dimensions `(sample_id, molecule_id, features)`. The `molecule_id` dimension has size 2 because a sample is a pair of molecules. The label is 1 if the first molecule is more complex than the second molecule. The function `create_dataset` we introduce below pulls random pairs of SMILES strings out of a given list and ranks them according to this complexity measure.

In the real world you could use purchase cost, or number of reaction steps required as your complexity score.

In [2]:
```
from rdkit import Chem
import random
from deepchem.feat import CircularFingerprint
import numpy as np


def create_dataset(fingerprints, smiles_lens, ds_size=100000):
    """
    m1: list of np.Array
        fingerprints for molecules
    m2: list of int
        length of a molecules SMILES string

    returns:
        dc.data.Dataset for input into ScScore Model

    Dataset.X
        shape is (sample_id, molecule_id, features)
    Dataset.y
        shape is (sample_id,)
        values is 1 if the 0th index molecule is more complex
                  0 if the 1st index molecule is more complex
    """
    X, y = [], []
    all_data = list(zip(fingerprints, smiles_lens))
    while len(y) < ds_size:
```

```
        i1 = random.randrange(0, len(smiles_lens))
        i2 = random.randrange(0, len(smiles_lens))
        m1 = all_data[i1]
        m2 = all_data[i2]
        if m1[1] == m2[1]:
            continue
        if m1[1] > m2[1]:
            y.append(1.0)
        else:
            y.append(0.0)
        X.append([m1[0], m2[0]])
    return dc.data.NumpyDataset(np.array(X), np.expand_dims(np.array(y), axis=1))
```

With our complexity ranker in place we can now construct our dataset. Let's start by randomly splitting the list of molecules into training and test sets.

In [3]:
```
molecule_ds = dc.data.NumpyDataset(np.array(molecules))
splitter = dc.splits.RandomSplitter()
train_mols, test_mols = splitter.train_test_split(molecule_ds)
```

We'll featurize all our molecules with the ECFP fingerprint with chirality (matching the source paper), and will then construct our pairwise dataset using the function defined above. We are using Circular Fingerprint featurizer, and defining parameters such as the fingerprint size n_features, fingerprint radius radius, and whether to consider chirality chiral. The Circular Fingerprint is a popular type of molecular fingerprint that encodes the structural information of molecules.

In [4]:
```
n_features = 1024
featurizer = dc.feat.CircularFingerprint(size=n_features, radius=2, chiral=True)
train_features = featurizer.featurize(train_mols.X)
train_smiles_len = [len(Chem.MolToSmiles(x)) for x in train_mols.X]
train_dataset = create_dataset(train_features, train_smiles_len)
```

Now that we have our dataset created, let's train a `ScScoreModel` on this dataset.

In [5]:
```
model = dc.models.ScScoreModel(n_features=n_features)
model.fit(train_dataset, nb_epoch=20)
```

Out[5]:
```
0.03494557857513428
```

Model Performance

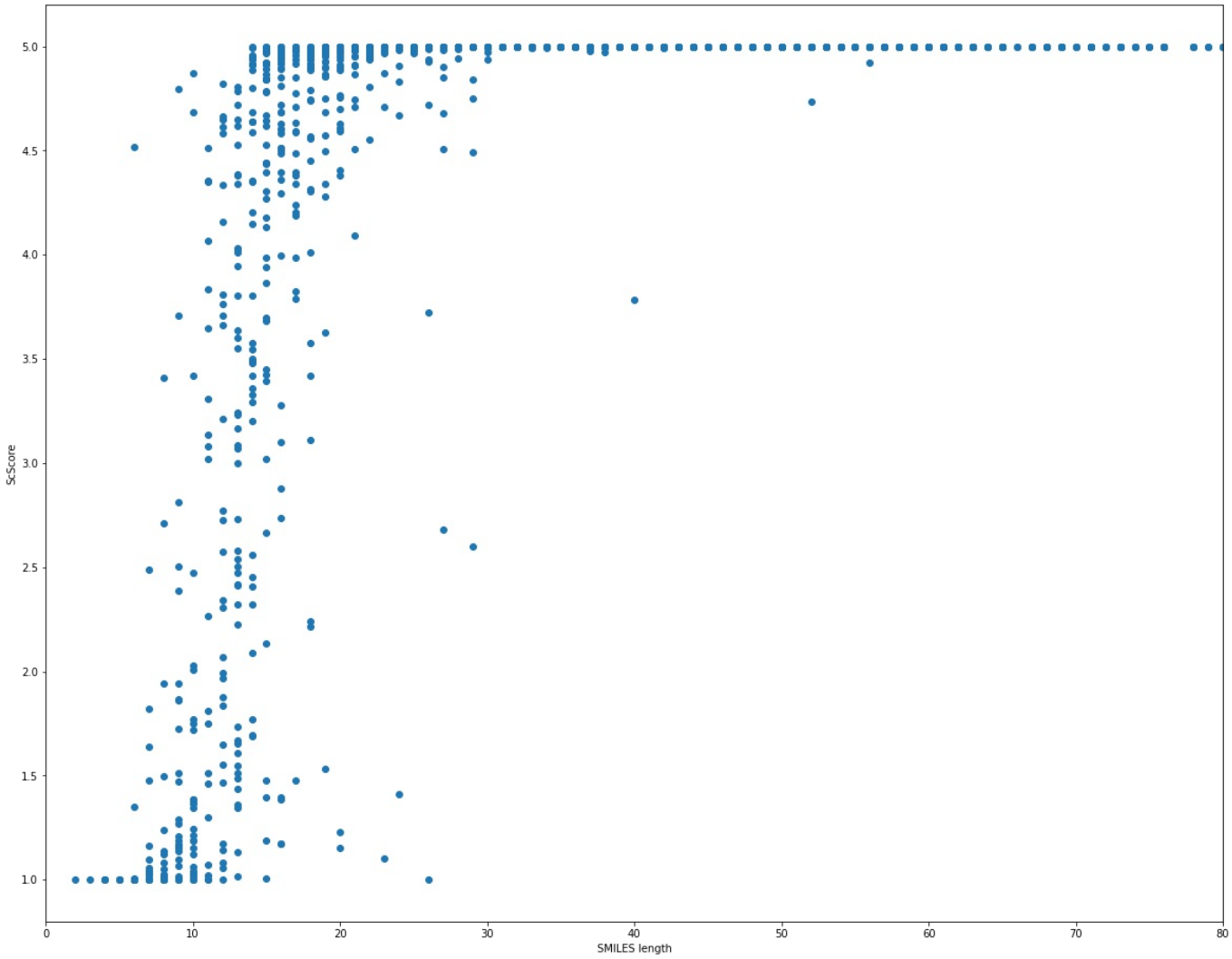
Lets evaluate how well the model does on our holdout molecules. The SaScores should track the length of SMILES strings from never before seen molecules.

In [6]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [7]:
```
mol_scores = model.predict_mols(test_mols.X)
smiles_lengths = [len(Chem.MolToSmiles(x)) for x in test_mols.X]
```

Let's now plot the length of the smiles string of the molecule against the SaScore using matplotlib.

In [8]:
```
plt.figure(figsize=(20,16))
plt.scatter(smiles_lengths, mol_scores)
plt.xlim(0,80)
plt.xlabel("SMILES length")
plt.ylabel("ScScore")
plt.show()
```



As we can see the model generally tracks SMILES length. It has good enrichment between 8 and 30 characters and gets both small and large SMILES strings extremes dead on.

Now you can train your own models on more meaningful metrics than SMILES length!

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Bibliography:

[1] https://pubs.acs.org/doi/abs/10.1021/acs.jcim.7b00622

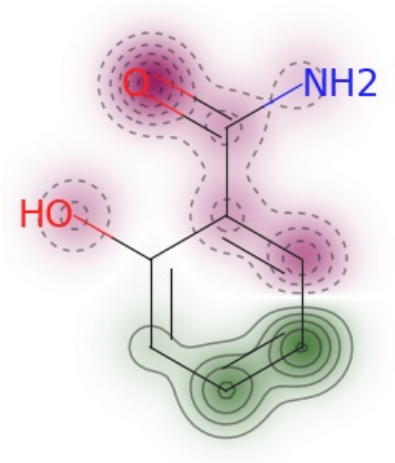
Calculating Atomic Contributions for Molecules Based on a Graph Convolutional QSAR Model

In an earlier tutorial we introduced the concept of model interpretability: understanding why a model produced the result it did. In this tutorial we will learn about atomic contributions, a useful tool for interpreting models that operate on molecules.

The idea is simple: remove a single atom from the molecule and see how the model's prediction changes. The "atomic contribution" for an atom is defined as the difference in activity between the whole molecule, and the fragment remaining after atom removal. It is a measure of how much that atom affects the prediction.

Contributions are also known as "attributions", "coloration", etc. in the literature. This is a model interpretation method [1], analogous to Similarity maps [2] in the QSAR domain, or occlusion methods in other fields (image classification, etc). Present implementation was used in [4].

Mariia Matveieva, Pavel Polishchuk. Institute of Molecular and Translational Medicine, Palacky University, Olomouc, Czech Republic.



Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

Open in Colab

Setup

To run DeepChem within Colab, you'll need to run the following installation commands. This will take about 5 minutes to run to completion and install your environment. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install Anaconda on your local machine.

In [1]:
```
!curl -Lo conda_installer.py https://raw.githubusercontent.com/deepchem/deepchem/master/scripts/colab_install.py
import conda_installer
conda_installer.install()
!/root/miniconda/bin/conda info -e
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3457  100  3457    0     0  24692      0 --:--:-- --:--:-- --:--:-- 24692
add /root/miniconda/lib/python3.7/site-packages to PYTHONPATH
python version: 3.7.12
fetching installer from https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
done
installing miniconda to /root/miniconda
done
installing openmm, pdbfixer
added conda-forge to channels
done
conda packages installation finished!
# conda environments:
#
base                  *  /root/miniconda
```

In [2]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

```
Collecting deepchem
  Downloading deepchem-2.6.0.dev20211215231347-py3-none-any.whl (608 kB)
     |▏                               | 10 kB 25.3 MB/s eta 0:00:01
     |▎                               | 20 kB 27.0 MB/s eta 0:00:01
     |▍                               | 30 kB 30.7 MB/s eta 0:00:01
     |▌                               | 40 kB 34.1 MB/s eta 0:00:01
     |▋                               | 51 kB 37.7 MB/s eta 0:00:01
     |▊                               | 61 kB 40.1 MB/s eta 0:00:01
     |▉                               | 71 kB 36.7 MB/s eta 0:00:01
     |▉                               | 81 kB 38.0 MB/s eta 0:00:01
     |█                               | 92 kB 39.3 MB/s eta 0:00:01
     |█                               | 102 kB 34.8 MB/s eta 0:00:01
     |█▏                              | 112 kB 34.8 MB/s eta 0:00:01
     |█▏                              | 122 kB 34.8 MB/s eta 0:00:01
     |█▎                              | 133 kB 34.8 MB/s eta 0:00:01
     |█▍                              | 143 kB 34.8 MB/s eta 0:00:01
     |█▌                              | 153 kB 34.8 MB/s eta 0:00:01
     |█▌                              | 163 kB 34.8 MB/s eta 0:00:01
     |█▋                              | 174 kB 34.8 MB/s eta 0:00:01
     |█▊                              | 184 kB 34.8 MB/s eta 0:00:01
     |█▊                              | 194 kB 34.8 MB/s eta 0:00:01
     |█▉                              | 204 kB 34.8 MB/s eta 0:00:01
     |██                              | 215 kB 34.8 MB/s eta 0:00:01
     |██                              | 225 kB 34.8 MB/s eta 0:00:01
     |██▏                             | 235 kB 34.8 MB/s eta 0:00:01
     |██▏                             | 245 kB 34.8 MB/s eta 0:00:01
     |██▎                             | 256 kB 34.8 MB/s eta 0:00:01
     |██▍                             | 266 kB 34.8 MB/s eta 0:00:01
     |██▍                             | 276 kB 34.8 MB/s eta 0:00:01
     |██▌                             | 286 kB 34.8 MB/s eta 0:00:01
     |██▋                             | 296 kB 34.8 MB/s eta 0:00:01
     |██▋                             | 307 kB 34.8 MB/s eta 0:00:01
     |██▊                             | 317 kB 34.8 MB/s eta 0:00:01
     |██▉                             | 327 kB 34.8 MB/s eta 0:00:01
     |██▉                             | 337 kB 34.8 MB/s eta 0:00:01
     |███                             | 348 kB 34.8 MB/s eta 0:00:01
     |███                             | 358 kB 34.8 MB/s eta 0:00:01
     |███▏                            | 368 kB 34.8 MB/s eta 0:00:01
     |███▏                            | 378 kB 34.8 MB/s eta 0:00:01
     |███▎                            | 389 kB 34.8 MB/s eta 0:00:01
     |███▍                            | 399 kB 34.8 MB/s eta 0:00:01
     |███▍                            | 409 kB 34.8 MB/s eta 0:00:01
     |███▌                            | 419 kB 34.8 MB/s eta 0:00:01
     |███▋                            | 430 kB 34.8 MB/s eta 0:00:01
     |███▋                            | 440 kB 34.8 MB/s eta 0:00:01
     |███▊                            | 450 kB 34.8 MB/s eta 0:00:01
     |███▉                            | 460 kB 34.8 MB/s eta 0:00:01
     |███▉                            | 471 kB 34.8 MB/s eta 0:00:01
     |████                            | 481 kB 34.8 MB/s eta 0:00:01
     |████                            | 491 kB 34.8 MB/s eta 0:00:01
     |████▏                           | 501 kB 34.8 MB/s eta 0:00:01
     |████▏                           | 512 kB 34.8 MB/s eta 0:00:01
     |████▎                           | 522 kB 34.8 MB/s eta 0:00:01
     |████▍                           | 532 kB 34.8 MB/s eta 0:00:01
     |████▍                           | 542 kB 34.8 MB/s eta 0:00:01
     |████▌                           | 552 kB 34.8 MB/s eta 0:00:01
     |████▋                           | 563 kB 34.8 MB/s eta 0:00:01
     |████▋                           | 573 kB 34.8 MB/s eta 0:00:01
     |████▊                           | 583 kB 34.8 MB/s eta 0:00:01
     |████▉                           | 593 kB 34.8 MB/s eta 0:00:01
     |████▉                           | 604 kB 34.8 MB/s eta 0:00:01
     |█████                           | 608 kB 34.8 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.19.5)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.0)
Collecting rdkit-pypi
  Downloading rdkit_pypi-2021.9.3-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20.6 MB)
     |████████████████████████████████| 20.6 MB 1.3 MB/s
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.0.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.4.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.5)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem) (2
018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
pandas->deepchem) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn
->deepchem) (3.0.0)
Installing collected packages: rdkit-pypi, deepchem
Successfully installed deepchem-2.6.0.dev20211215231347 rdkit-pypi-2021.9.3
```

Out[2]: '2.6.0.dev'

A classification QSAR model for blood-brain barrier permeability

BBB permeability is the ability of compounds to enter the central nervous system. Here we use a dataset of relatively small compounds which are transported by diffusion without any carriers. The property is defined as log10(concentration in brain / concentration in blood). Compounds with a positive value (and 0) are labeled active, and others are labeled inactive. After modelling we will identify atoms favorable and unfavorable for diffusion.

First let's create the dataset. The molecules are stored in an SDF file.

In [8]:
```
import os
import pandas as pd
import deepchem as dc
import numpy as np
from rdkit import Chem
from rdkit.Chem import AllChem
from rdkit.Chem import Draw, PyMol, rdFMCS
from rdkit.Chem.Draw import IPythonConsole
from rdkit import rdBase
from deepchem import metrics
from IPython.display import Image, display
from rdkit.Chem.Draw import SimilarityMaps
import tensorflow as tf

current_dir = os.path.dirname(os.path.realpath('__file__'))
dc.utils.download_url(
    'https://raw.githubusercontent.com/deepchem/deepchem/master/examples/tutorials/assets/atomic_contributions_
    current_dir,
    'logBB.sdf'
)
DATASET_FILE =os.path.join(current_dir, 'logBB.sdf')
# Create RDKit mol objects, since we will need them later.
mols = [m for m in Chem.SDMolSupplier(DATASET_FILE) if m is not None ]
loader = dc.data.SDFLoader(tasks=["logBB_class"],
                           featurizer=dc.feat.ConvMolFeaturizer(),
                           sanitize=True)
dataset = loader.create_dataset(DATASET_FILE, shard_size=2000)
```

Now let's build and train a GraphConvModel.

In [10]:
```
np.random.seed(2020)
tf.random.set_seed(2020)
```

In [11]:
```
m = dc.models.GraphConvModel(1, mode="classification", batch_normalize=False, batch_size=100)
m.fit(dataset, nb_epoch=10)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_14:0", shape=(331,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_po
ol_1/Reshape_13:0", shape=(331, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras
_model/graph_pool_1/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a
large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_17:0", shape=(1646,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_p
ool_1/Reshape_16:0", shape=(1646, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_pool_1/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_20:0", shape=(1359,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_p
ool_1/Reshape_19:0", shape=(1359, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_pool_1/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_23:0", shape=(148,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_po
ol_1/Reshape_22:0", shape=(148, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras
_model/graph_pool_1/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a
large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/
Reshape_11:0", shape=(331,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_co
nv_1/Reshape_10:0", shape=(331, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras
_model/graph_conv_1/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a l
arge amount of memory.
  "shape. This may consume a large amount of memory." % value)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_13:0", shape=(1646,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_12:0", shape=(1646, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_1:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_15:0", shape=(1359,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_14:0", shape=(1359, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_2:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_17:0", shape=(148,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_16:0", shape=(148, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_3:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_19:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_18:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_21:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_20:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_23:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_22:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_25:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_24:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_27:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_26:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_8:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_29:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_28:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_9:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_14:0", shape=(331,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_13:0", shape=(331, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_17:0", shape=(1646,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_16:0", shape=(1646, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_20:0", shape=(1359,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_19:0", shape=(1359, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
```

```
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_23:0", shape=(148,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_22:0", shape=(148, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_14:0", shape=(334,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_13:0", shape=(334, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_17:0", shape=(1838,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_16:0", shape=(1838, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_20:0", shape=(1458,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_19:0", shape=(1458, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_23:0", shape=(120,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Reshape_22:0", shape=(120, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_11:0", shape=(334,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_10:0", shape=(334, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_13:0", shape=(1838,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_12:0", shape=(1838, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_1:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_15:0", shape=(1458,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_14:0", shape=(1458, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_2:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_17:0", shape=(120,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Reshape_16:0", shape=(120, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/Cast_3:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_14:0", shape=(334,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_13:0", shape=(334, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_17:0", shape=(1838,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_16:0", shape=(1838, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_20:0", shape=(1458,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Reshape_19:0", shape=(1458, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a lar
```

```
ge amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Re
shape_23:0", shape=(120,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool
/Reshape_22:0", shape=(120, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_mod
el/graph_pool/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large
amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_14:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_p
ool_1/Reshape_13:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_pool_1/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_17:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_p
ool_1/Reshape_16:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_pool_1/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_20:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_p
ool_1/Reshape_19:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_pool_1/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/
Reshape_11:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_c
onv_1/Reshape_10:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_conv_1/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a
large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/
Reshape_13:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_c
onv_1/Reshape_12:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_conv_1/Cast_1:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/
Reshape_15:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_c
onv_1/Reshape_14:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_conv_1/Cast_2:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Re
shape_14:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_poo
l/Reshape_13:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_m
odel/graph_pool/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a lar
ge amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Re
shape_17:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_poo
l/Reshape_16:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_m
odel/graph_pool/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a lar
ge amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Re
shape_20:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_poo
l/Reshape_19:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_m
odel/graph_pool/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a lar
ge amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool_1/
Reshape_23:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_p
ool_1/Reshape_22:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model/graph_pool_1/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_conv_1/
Reshape_17:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_c
onv_1/Reshape_16:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
```

```
as_model/graph_conv_1/Cast_3:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume
a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model/graph_pool/Re
shape_23:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model/graph_poo
l/Reshape_22:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_m
odel/graph_pool/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a lar
ge amount of memory.
  "shape. This may consume a large amount of memory." % value)
```

Out[11]: 0.5348201115926107

Let's load a test set and see how well it works.

In [12]:
```
current_dir = os.path.dirname(os.path.realpath('__file__'))
dc.utils.download_url(
    'https://raw.githubusercontent.com/deepchem/deepchem/master/examples/tutorials/assets/atomic_contributions_
    current_dir,
    'logBB_test_.sdf'
)
TEST_DATASET_FILE = os.path.join(current_dir, 'logBB_test_.sdf')
loader = dc.data.SDFLoader(tasks=["p_np"], sanitize=True,
                           featurizer=dc.feat.ConvMolFeaturizer())
test_dataset = loader.create_dataset(TEST_DATASET_FILE, shard_size=2000)
pred =  m.predict(test_dataset)
pred = np.argmax(np.squeeze(pred),axis=1)
ba = metrics.balanced_accuracy_score(y_true=test_dataset.y, y_pred=pred)
print(ba)
```

```
0.7444444444444445
```

The balanced accuracy is high enough. Now let's proceed to model interpretation and estimate the contributions of individual atoms to the prediction.

A fragment dataset

Now let's prepare a dataset of fragments based on the training set. (Any other unseen data set of interest can also be used). These fragments will be used to evaluate the contributions of individual atoms.

For each molecule we will generate a list of ConvMol objects. Specifying `per_atom_fragmentation=True` tells it to iterate over all heavy atoms and featurize a single-atom-depleted version of the molecule with each one removed.

In [13]:
```
loader = dc.data.SDFLoader(tasks=[],# dont need task (moreover, passing the task can lead to inconsitencies in (
                           featurizer=dc.feat.ConvMolFeaturizer(per_atom_fragmentation=True),
                           sanitize=True)
frag_dataset = loader.create_dataset(DATASET_FILE, shard_size=5000)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray
from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.7/dist-packages/deepchem/data/data_loader.py:885: VisibleDeprecationWarning: Creating an
ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different len
gths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarra
y
  return np.array(features), valid_inds
```

The dataset still has the same number of samples as the original training set, but each sample is now represented as a list of ConvMol objects (one for each fragment) rather than a single ConvMol.

IMPORTANT: The order of fragments depends on the input format. If SDF, the fragment order is the same as the atom order in corresponding mol blocks. If SMILES (i.e. csv with molecules represented as SMILES), then the order is given by RDKit CanonicalRankAtoms

In [14]:
```
print(frag_dataset.X.shape)
```

```
(298,)
```

We really want to treat each fragment as a separate sample. We can use a FlatteningTransformer to flatten the fragments lists.

In [15]:
```
tr = dc.trans.FlatteningTransformer(frag_dataset)
frag_dataset = tr.transform(frag_dataset)
print(frag_dataset.X.shape)
```

```
(5111,)
```

Predicting atomic contributions to activity

Now we will predict the activity for molecules and for their fragments. Then, for each fragment, we'll find the activity difference: the change in activity when removing one atom.

Note: Here, in classification context, we use the probability output of the model as the activity. So the contribution is the probability difference, i.e. "how much a given atom increases/decreases the probability of the molecule being active."

In [16]:
```
# whole  molecules
pred = np.squeeze(m.predict(dataset))[:, 1] # probabilitiy of class 1
pred = pd.DataFrame(pred, index=dataset.ids, columns=["Molecule"])  # turn to dataframe for convinience

# fragments
pred_frags = np.squeeze(m.predict(frag_dataset))[:, 1]
pred_frags = pd.DataFrame(pred_frags, index=frag_dataset.ids, columns=["Fragment"])
```

We take the difference to find the atomic contributions.

In [17]:
```
# merge 2 dataframes by molecule names
df = pd.merge(pred_frags, pred, right_index=True, left_index=True)
# find contribs
df['Contrib'] = df["Molecule"] - df["Fragment"]
```

In [18]:
```
df
```

Out[18]:

	Fragment	Molecule	Contrib
C#CC1(O)CCC2C3C(C)CC4=C(CCC(=O)C4)C3CCC21C	0.756537	0.811550	0.055013
C#CC1(O)CCC2C3C(C)CC4=C(CCC(=O)C4)C3CCC21C	0.752759	0.811550	0.058791
C#CC1(O)CCC2C3C(C)CC4=C(CCC(=O)C4)C3CCC21C	0.747012	0.811550	0.064538
C#CC1(O)CCC2C3C(C)CC4=C(CCC(=O)C4)C3CCC21C	0.815878	0.811550	-0.004328
C#CC1(O)CCC2C3C(C)CC4=C(CCC(=O)C4)C3CCC21C	0.741805	0.811550	0.069745
...	...	...	...
c1cncc(C2CCCN2)c1	0.780473	0.813031	0.032559
c1cncc(C2CCCN2)c1	0.722649	0.813031	0.090383
c1cncc(C2CCCN2)c1	0.721607	0.813031	0.091425
c1cncc(C2CCCN2)c1	0.683299	0.813031	0.129732
c1cncc(C2CCCN2)c1	0.674451	0.813031	0.138581

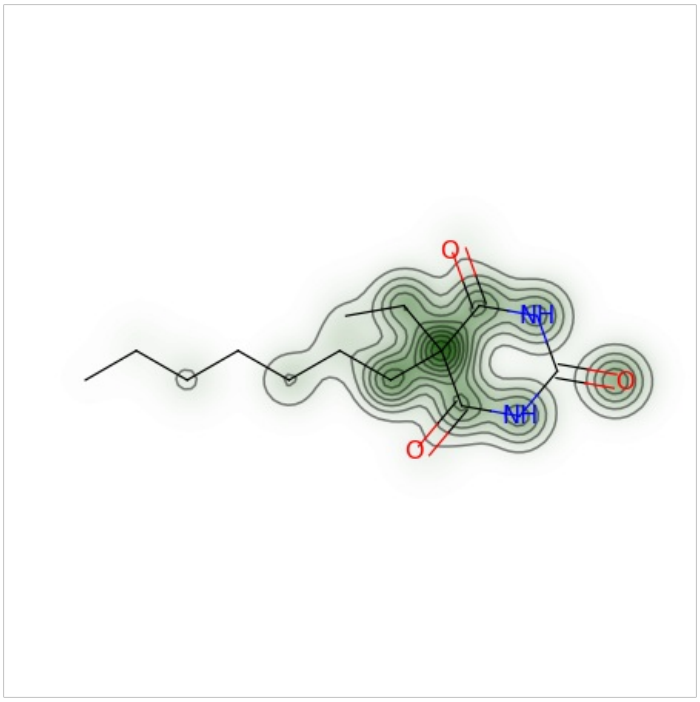
5111 rows × 3 columns

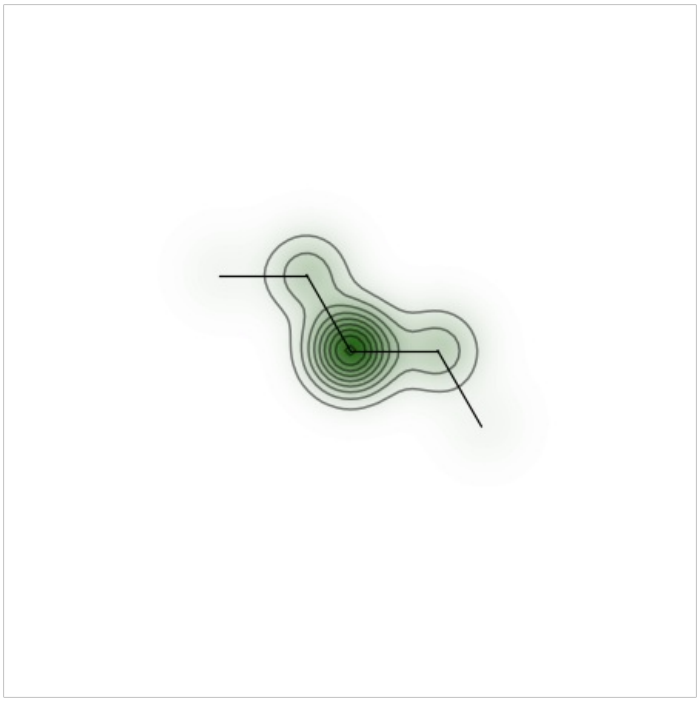
We can use the SimilarityMaps feature of RDKit to visualize the results. Each atom is colored by how it affects activity.

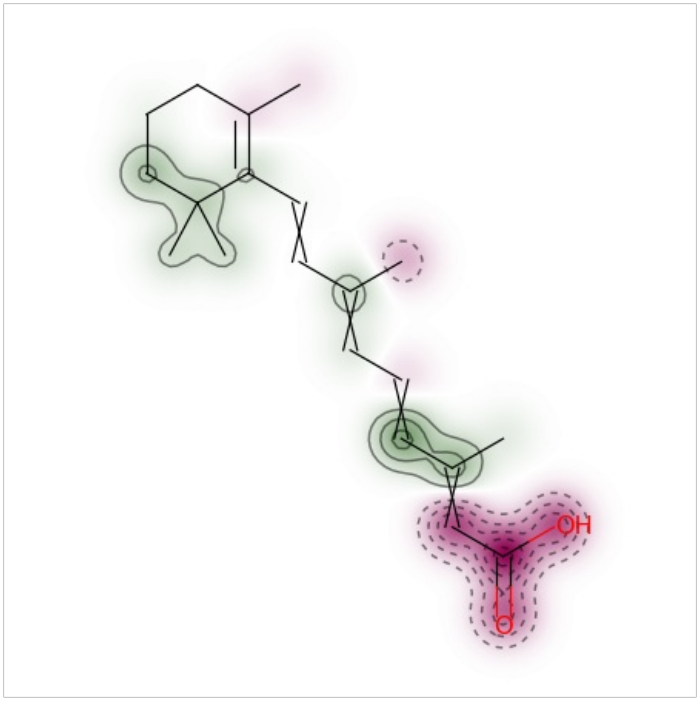
In [19]:
```
def vis_contribs(mols, df, smi_or_sdf = "sdf"):
    # input format of file, which was used to create dataset determines the order of atoms,
    # so we take it into account for correct mapping!
    maps = []
    for mol  in mols:
        wt = {}
        if smi_or_sdf == "smi":
            for n,atom in enumerate(Chem.rdmolfiles.CanonicalRankAtoms(mol)):
                wt[atom] = df.loc[mol.GetProp("_Name"),"Contrib"][n]
        if smi_or_sdf == "sdf":
            for n,atom in enumerate(range(mol.GetNumHeavyAtoms())):
                wt[atom] = df.loc[Chem.MolToSmiles(mol),"Contrib"][n]
        maps.append(SimilarityMaps.GetSimilarityMapFromWeights(mol,wt))
    return maps
```

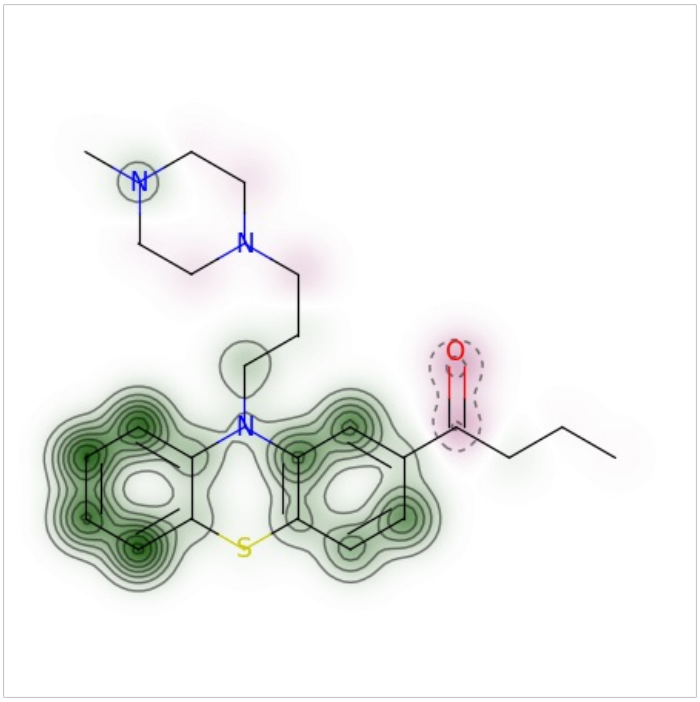
Let's look at some pictures:

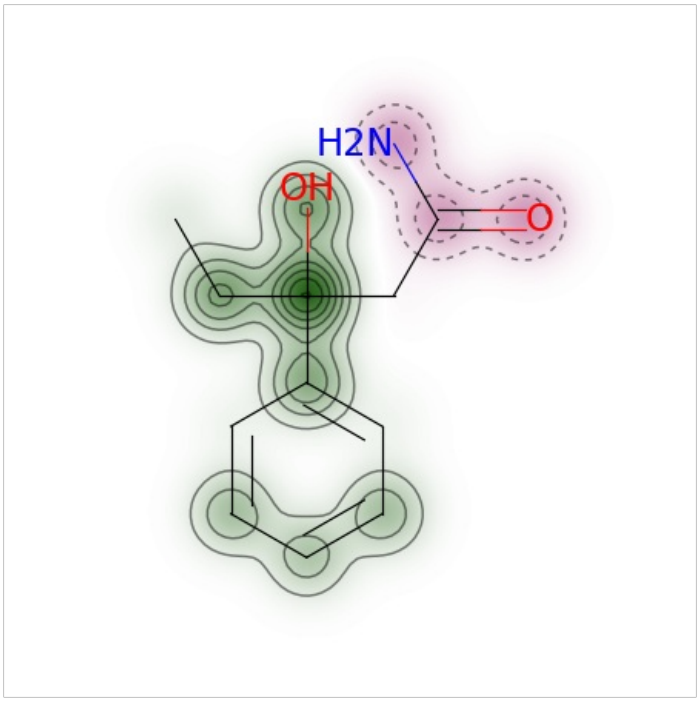
In [20]:
```
np.random.seed(2000)
maps = vis_contribs(np.random.choice(np.array(mols),10), df)
```

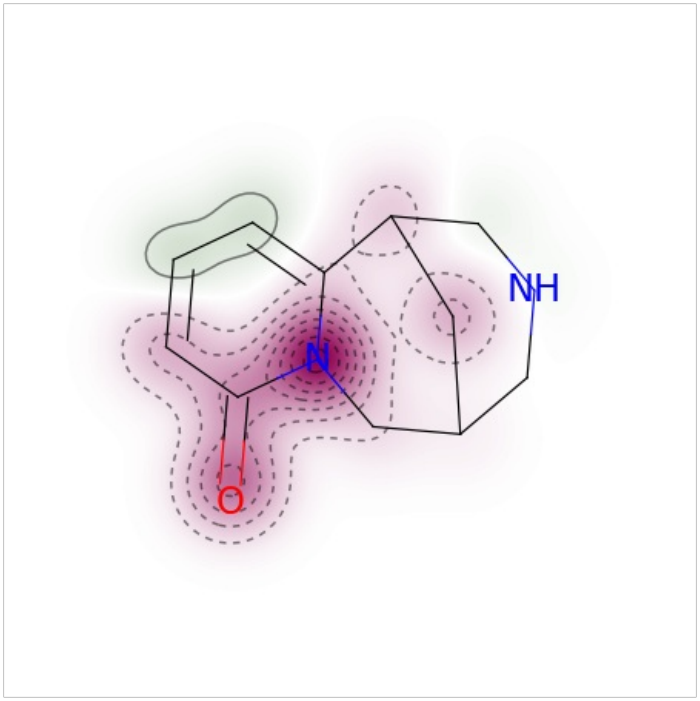

O
NH
CO
NH
O

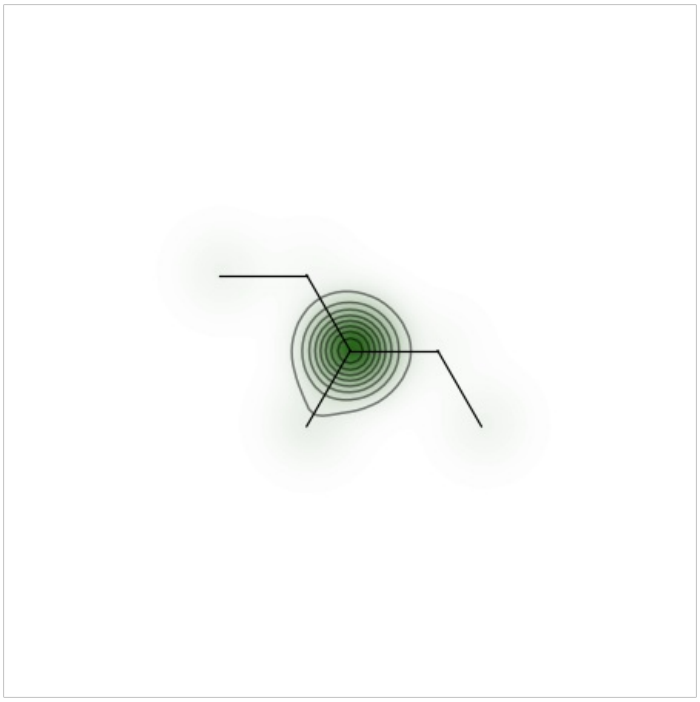


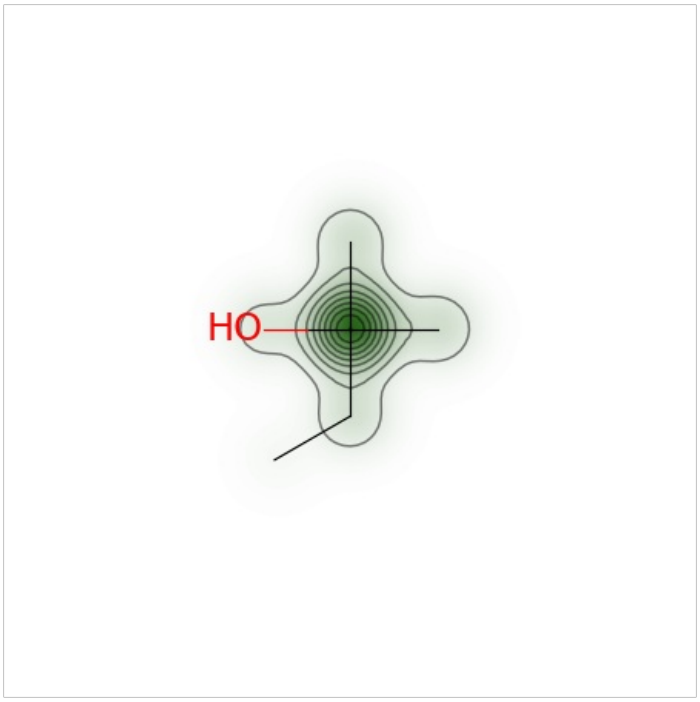

OH
O

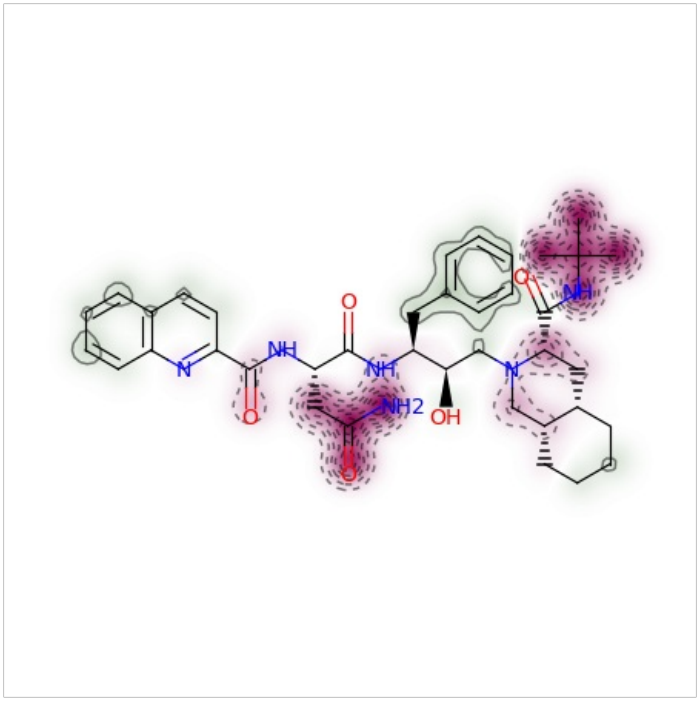

N
N
O
N
S

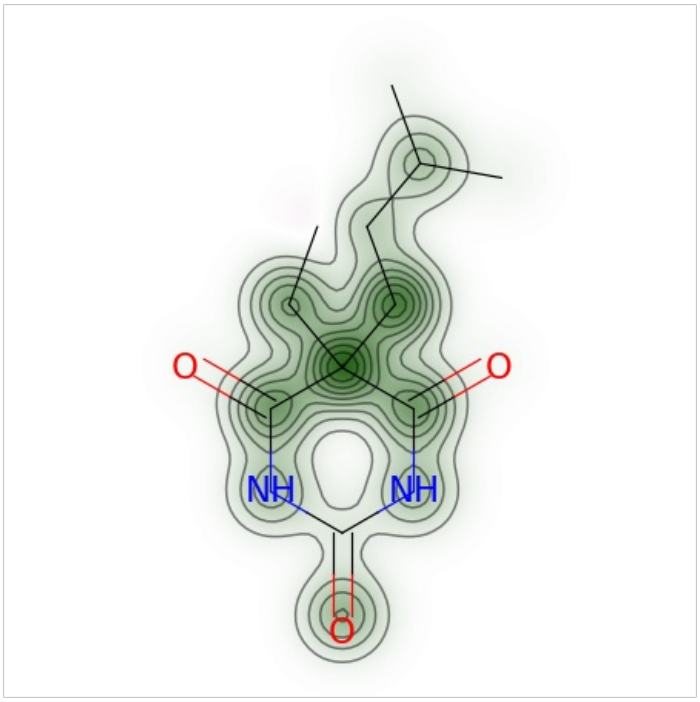

H2N
OH
O


NH
N
O




HO


O
NH
N
NH
O
NH
NH2
O
OH
N
O
NH



We can see that aromatics or aliphatics have a positive impact on blood-brain barrier permeability, while polar or charged heteroatoms have a negative influence. This is generally consistent with literature data.

A regression task

The example above used a classification model. The same techniques can also be used for regression models. Let's look at a regression task, aquatic toxicity (towards the water organism T. pyriformis).

Toxicity is defined as log10(IGC50) (concentration that inhibits colony growth by 50%). Toxicophores for T. pyriformis will be identified by atomic contributions.

All the above steps are the same: load data, featurize, build a model, create dataset of fragments, find contributions, and visualize them.

Note: this time as it is regression, contributions will be in activity units, not probability.

In [21]:
```
current_dir = os.path.dirname(os.path.realpath('__file__'))
dc.utils.download_url(
    'https://raw.githubusercontent.com/deepchem/deepchem/master/examples/tutorials/assets/atomic_contributions_
    current_dir,
    'Tetrahymena_pyriformis_Work_set_OCHEM.sdf'
)
DATASET_FILE =os.path.join(current_dir, 'Tetrahymena_pyriformis_Work_set_OCHEM.sdf')

# create RDKit mol objects, we will need them later
mols = [m for m in Chem.SDMolSupplier(DATASET_FILE) if m is not None ]
loader = dc.data.SDFLoader(tasks=["IGC50"],
                           featurizer=dc.feat.ConvMolFeaturizer(), sanitize=True)
dataset = loader.create_dataset(DATASET_FILE, shard_size=5000)
```

Create and train the model.

In [22]:
```
np.random.seed(2020)
tf.random.set_seed(2020)
m = dc.models.GraphConvModel(1, mode="regression", batch_normalize=False)
m.fit(dataset, nb_epoch=40)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
3/Reshape_14:0", shape=(291,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/grap
h_pool_3/Reshape_13:0", shape=(291, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_k
eras_model_1/graph_pool_3/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may con
sume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
3/Reshape_17:0", shape=(910,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/grap
h_pool_3/Reshape_16:0", shape=(910, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_k
eras_model_1/graph_pool_3/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may con
sume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_20:0", shape=(663,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_19:0", shape=(663, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_23:0", shape=(28,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_22:0", shape=(28, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_11:0", shape=(291,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_10:0", shape=(291, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_13:0", shape=(910,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_12:0", shape=(910, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_1:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_15:0", shape=(663,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_14:0", shape=(663, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_2:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_17:0", shape=(28,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_16:0", shape=(28, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_3:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_19:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_18:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_21:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_20:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_23:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_22:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_25:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_24:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_27:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_26:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_8:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_29:0", shape=(0,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_28:0", shape=(0, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_9:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
```

```
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_14:0", shape=(291,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_13:0", shape=(291, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_17:0", shape=(910,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_16:0", shape=(910, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_20:0", shape=(663,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_19:0", shape=(663, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_23:0", shape=(28,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Reshape_22:0", shape=(28, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_2/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_14:0", shape=(307,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_13:0", shape=(307, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_17:0", shape=(944,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_16:0", shape=(944, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_20:0", shape=(693,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_19:0", shape=(693, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_23:0", shape=(16,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Reshape_22:0", shape=(16, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_3/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_11:0", shape=(307,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_10:0", shape=(307, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_13:0", shape=(944,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_12:0", shape=(944, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_1:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_15:0", shape=(693,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_14:0", shape=(693, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_2:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Converting sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_17:0", shape=(16,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Reshape_16:0", shape=(16, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_3/Cast_3:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consu
```

```
me a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_14:0", shape=(307,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/grap
h_pool_2/Reshape_13:0", shape=(307, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_k
eras_model_1/graph_pool_2/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may con
sume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_17:0", shape=(944,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/grap
h_pool_2/Reshape_16:0", shape=(944, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_k
eras_model_1/graph_pool_2/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may con
sume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_20:0", shape=(693,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/grap
h_pool_2/Reshape_19:0", shape=(693, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_k
eras_model_1/graph_pool_2/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may con
sume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_23:0", shape=(16,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph
_pool_2/Reshape_22:0", shape=(16, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv_ker
as_model_1/graph_pool_2/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may consu
me a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
3/Reshape_14:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_3/Reshape_13:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_3/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
3/Reshape_17:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_3/Reshape_16:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_3/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
3/Reshape_20:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_3/Reshape_19:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_3/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
3/Reshape_23:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_3/Reshape_22:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_3/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_
3/Reshape_11:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_conv_3/Reshape_10:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_conv_3/Cast:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may con
sume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_
3/Reshape_13:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_conv_3/Reshape_12:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_conv_3/Cast_1:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_
3/Reshape_15:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_conv_3/Reshape_14:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_conv_3/Cast_2:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_conv_
3/Reshape_17:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_conv_3/Reshape_16:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
```

```
_keras_model_1/graph_conv_3/Cast_3:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_14:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_2/Reshape_13:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_2/Cast_4:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_17:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_2/Reshape_16:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_2/Cast_5:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_20:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_2/Reshape_19:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_2/Cast_6:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/indexed_slices.py:450: UserWarning: Convertin
g sparse IndexedSlices(IndexedSlices(indices=Tensor("gradient_tape/private__graph_conv_keras_model_1/graph_pool_
2/Reshape_23:0", shape=(None,), dtype=int32), values=Tensor("gradient_tape/private__graph_conv_keras_model_1/gra
ph_pool_2/Reshape_22:0", shape=(None, 64), dtype=float32), dense_shape=Tensor("gradient_tape/private__graph_conv
_keras_model_1/graph_pool_2/Cast_7:0", shape=(2,), dtype=int32))) to a dense Tensor of unknown shape. This may c
onsume a large amount of memory.
  "shape. This may consume a large amount of memory." % value)
```

Out[22]:
```
0.12407124519348145
```

Load the test dataset and check the model's performance.

In [23]:
```
current_dir = os.path.dirname(os.path.realpath('__file__'))
dc.utils.download_url(
    'https://raw.githubusercontent.com/deepchem/deepchem/master/examples/tutorials/assets/atomic_contributions_
    current_dir,
    'Tetrahymena_pyriformis_Test_set_OCHEM.sf'
)



TEST_DATASET_FILE = os.path.join(current_dir, 'Tetrahymena_pyriformis_Test_set_OCHEM.sdf')
loader = dc.data.SDFLoader(tasks=["IGC50"], sanitize= True,
                           featurizer=dc.feat.ConvMolFeaturizer())
test_dataset = loader.create_dataset(TEST_DATASET_FILE, shard_size=2000)
pred = m.predict(test_dataset)
mse = metrics.mean_squared_error(y_true=test_dataset.y, y_pred=pred)
r2 = metrics.r2_score(y_true=test_dataset.y, y_pred=pred)
print(mse)
print(r2)
```

```
0.2381780323921622
0.784334539071699
```

Load the training set again, but this time set `per_atom_fragmentation=True`.

In [24]:
```
loader = dc.data.SDFLoader(tasks=[], # dont need any task
                           sanitize=True,
                           featurizer=dc.feat.ConvMolFeaturizer(per_atom_fragmentation=True))
frag_dataset = loader.create_dataset(DATASET_FILE, shard_size=5000)
tr = dc.trans.FlatteningTransformer(frag_dataset) # flatten dataset and add ids to each fragment
frag_dataset = tr.transform(frag_dataset)
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray
from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
/usr/local/lib/python3.7/dist-packages/deepchem/data/data_loader.py:885: VisibleDeprecationWarning: Creating an
ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different len
gths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarra
y
  return np.array(features), valid_inds
```

Compute the activity differences.

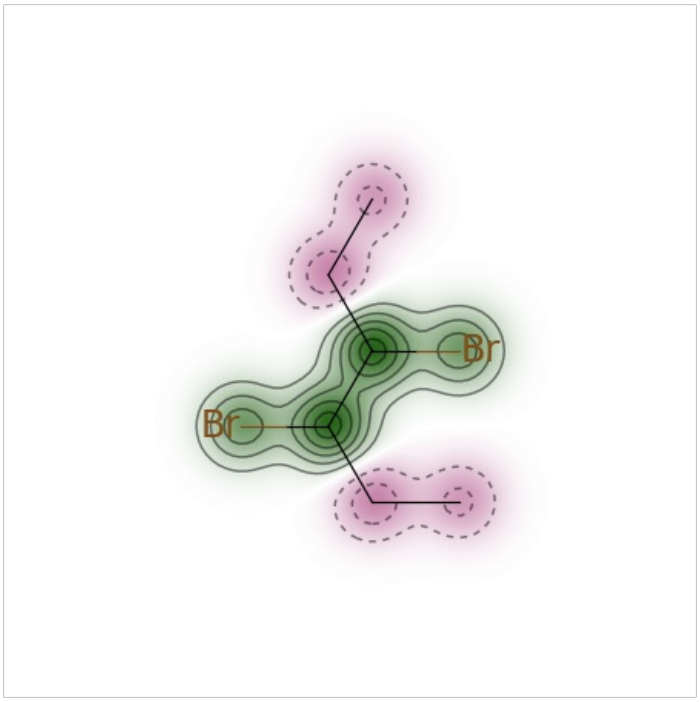
In [25]:
```
# whole molecules
pred = m.predict(dataset)
pred = pd.DataFrame(pred, index=dataset.ids, columns=["Molecule"])  # turn to dataframe for convenience
```

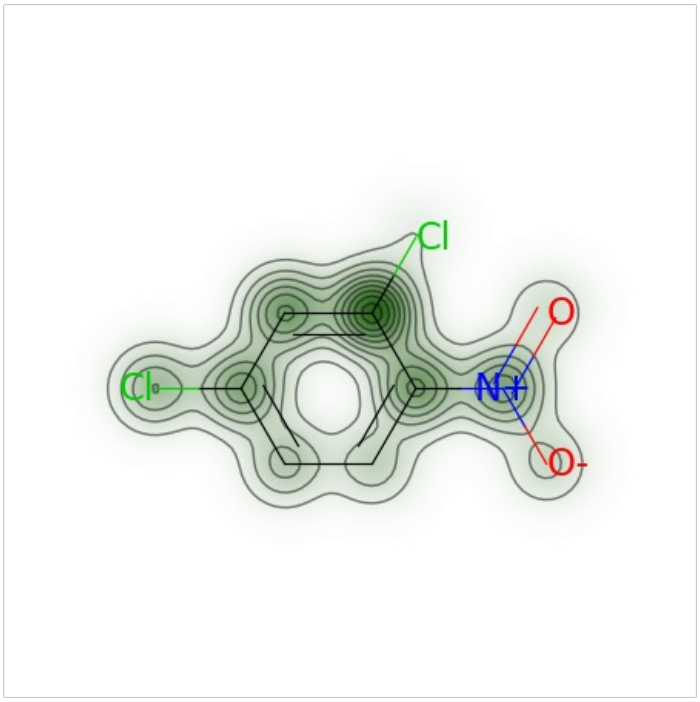
```
# fragments
pred_frags = m.predict(frag_dataset)
pred_frags = pd.DataFrame(pred_frags, index=frag_dataset.ids, columns=["Fragment"])  # turn to dataframe for co

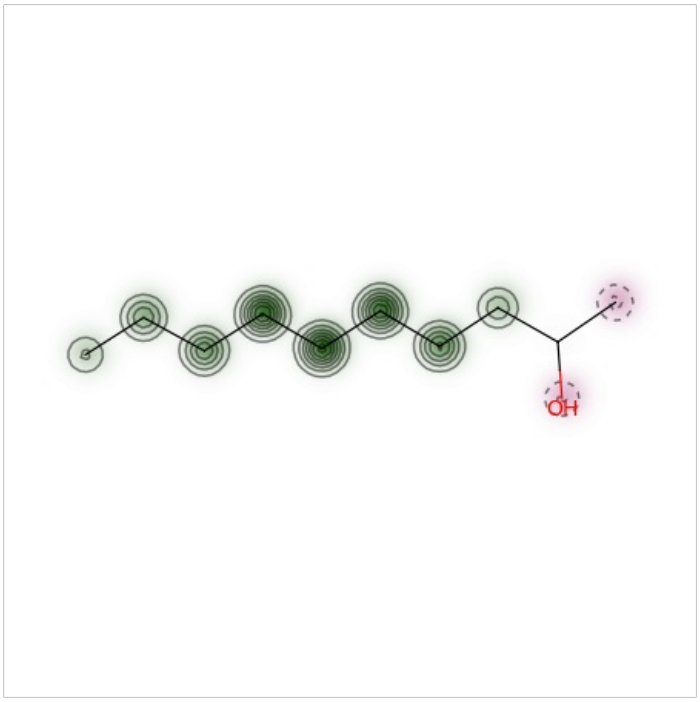
# merge 2 dataframes by molecule names
df = pd.merge(pred_frags, pred, right_index=True, left_index=True)
# find contribs
df['Contrib'] = df["Molecule"] - df["Fragment"]
```

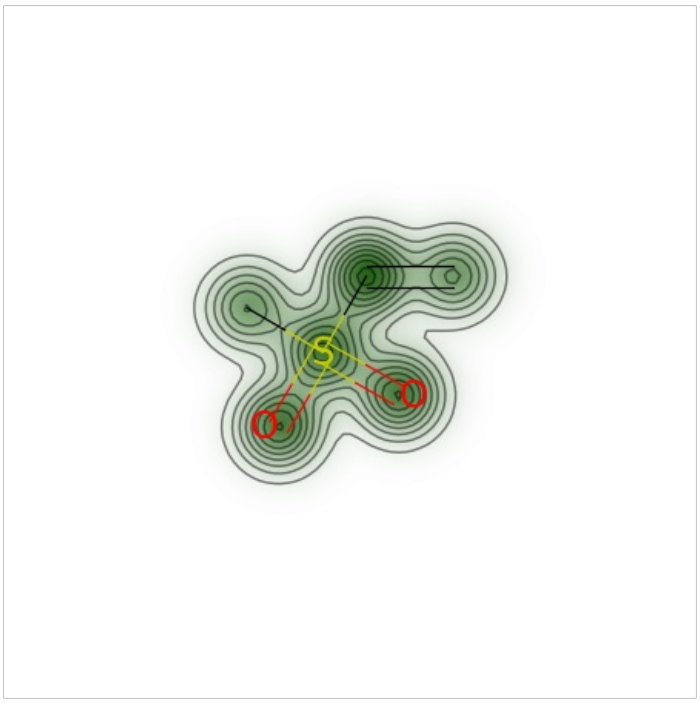
Lets take some molecules with moderate activity (not extremely active/inactive) and visualize the atomic contributions.

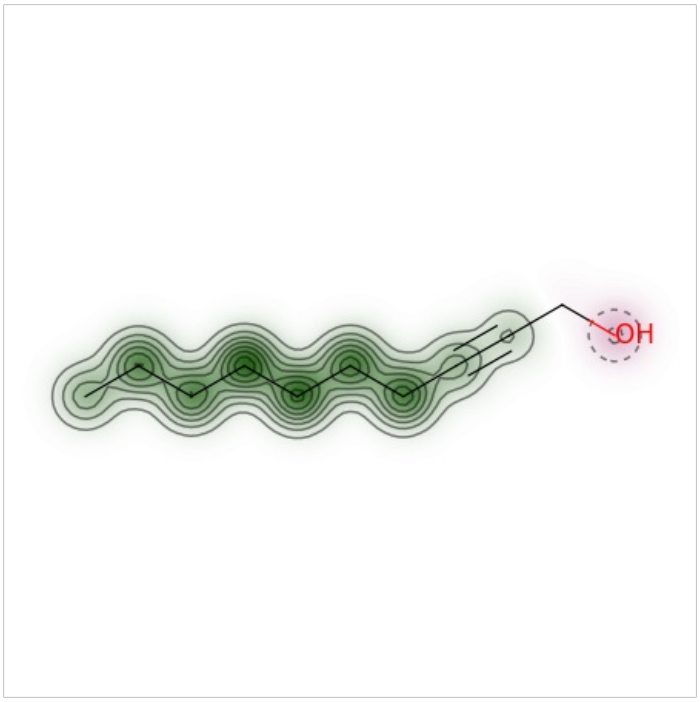
In [26]:
```
maps = vis_contribs([mol for mol in mols if float(mol.GetProp("IGC50"))>3 and float(mol.GetProp("IGC50"))<4][:1
```

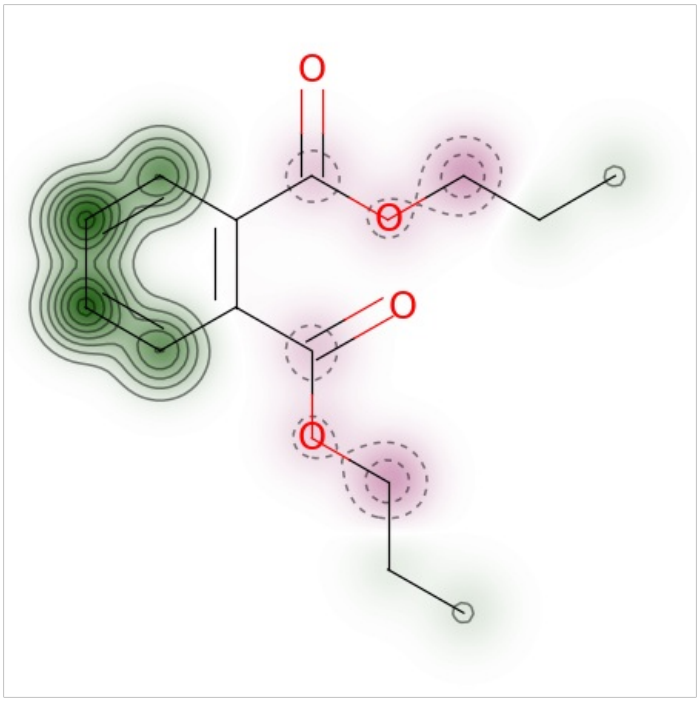


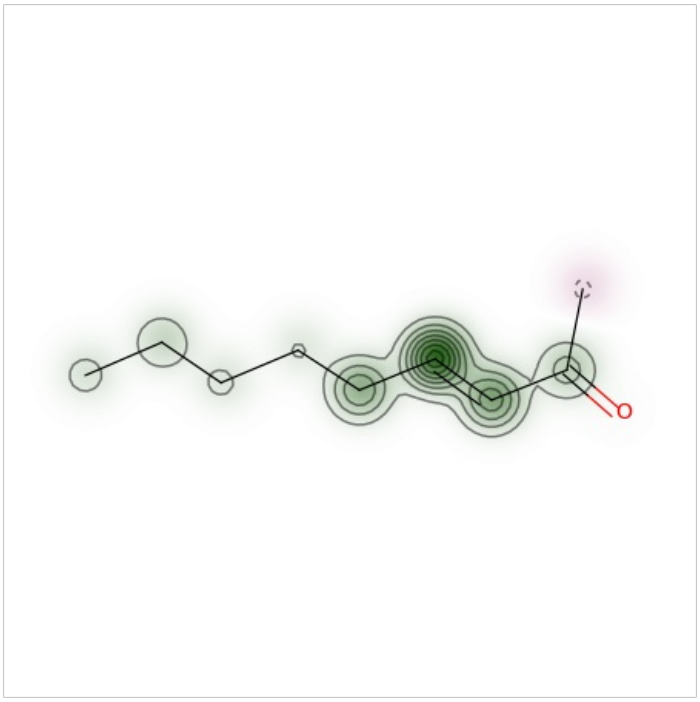


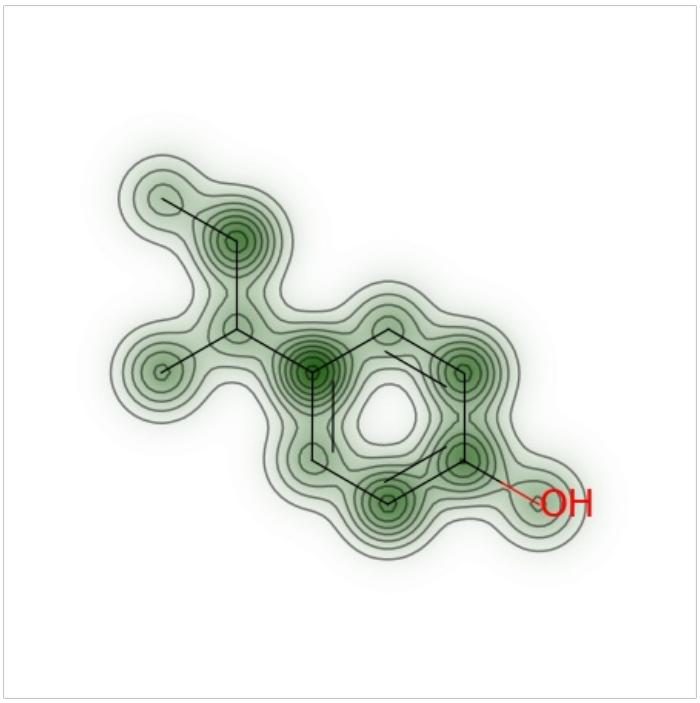

OH

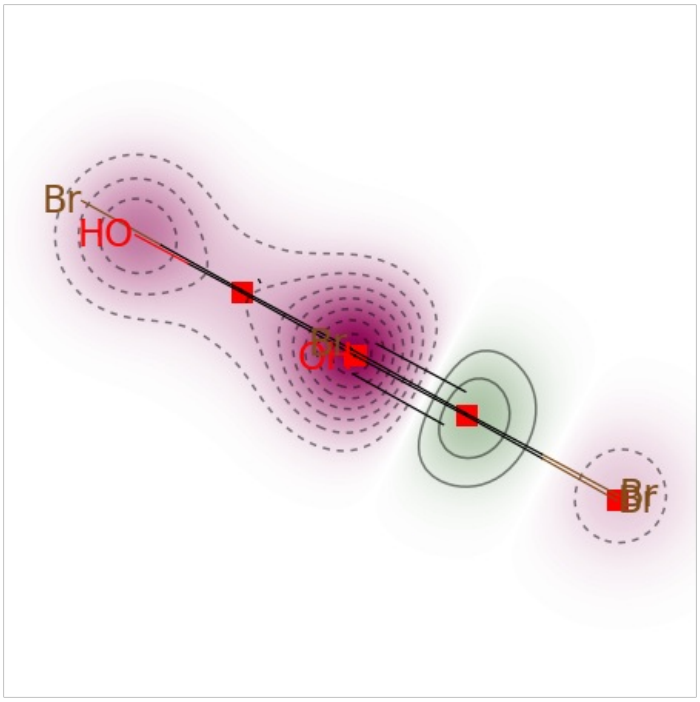

S
O
O

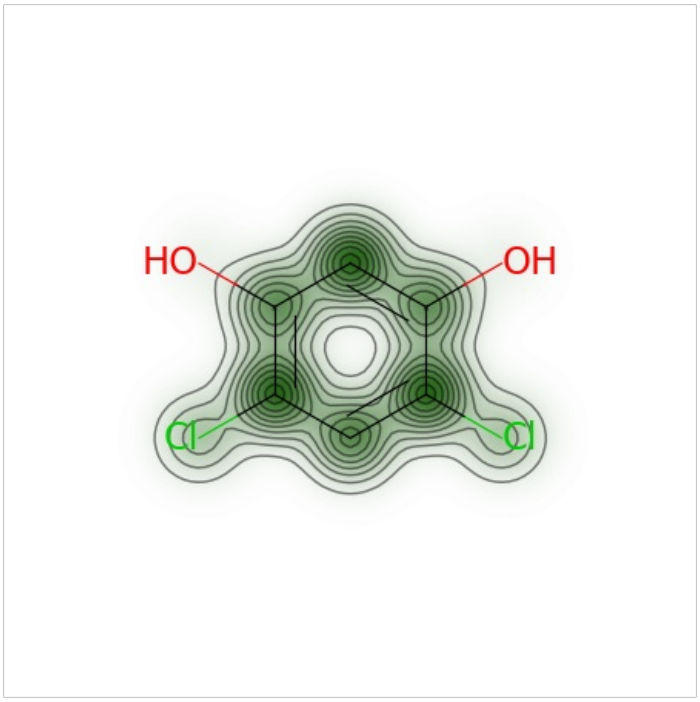

OH


O
O
O
O


O


OH





We can see that known toxicophores are in green, namely nitro-aromatics, halo-aromatics, long alkyl chains, and aldehyde; while carboxylic groups, alcohols, and aminos are detoxyfying, as is consistent with literature [3]

Appendix

In this tutorial we operated on SDF files. However, if we use CSV files with SMILES as input, the order of the atoms in the dataframe DOES NOT correspond to the original atom order. If we want to recover the original atom order for each molecule (to have it in our main dataframe), we need to use RDKit's Chem.rdmolfiles.CanonicalRankAtoms. Here are some utilities to do this.

We can add a column with atom ids (as in input molecules) and use the resulting dataframe for analysis with any other software, outside the "python-rdkit-deepchem" environment.

In [27]:
```
def get_mapping(mols, mol_names):
    """perform mapping:
    atom number original <-> atom number(position)
    after ranking (both 1-based)"""
    # mols - RDKit mols
    # names  - any seq of strings
    # return list of nested lists: [[molecule, [atom , atom, ..], [...]]
    assert(len(mols)==len(mol_names))
    mapping = []
    for m,n in zip(mols, mol_names):
        atom_ids = [i+1 for i in list(Chem.rdmolfiles.CanonicalRankAtoms(m))]
        mapping.append([n, atom_ids])
    return mapping
```

In [28]:
```
def append_atomid_col(df, mapping):
    # add column with CORRECT atom number(position)
    for i in mapping:
        df.loc[i[0],"AtomID"] = i[1]
    return df
```

Bibliography:

1. Polishchuk, P., O. Tinkov, T. Khristova, L. Ognichenko, A. Kosinskaya, A. Varnek & V. Kuz'min (2016) Structural and Physico-Chemical Interpretation (SPCI) of QSAR Models and Its Comparison with Matched Molecular Pair Analysis. Journal of Chemical Information and Modeling, 56, 1455-1469.
2. Riniker, S. & G. Landrum (2013) Similarity maps - a visualization strategy for molecular fingerprints and machine-learning methods. Journal of Cheminformatics, 5, 43.
3. M. Matveieva, M. T. D. Cronin, P. Polishchuk, Mol. Inf. 2019, 38, 1800084.
4. Matveieva, M., Polishchuk, P. Benchmarks for interpretation of QSAR models. J Cheminform 13, 41 (2021). https://doi.org/10.1186/s13321-021-00519-x

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Interactive model evaluation with Trident Chemwidgets

In this tutorial we'll build on the *Introduction to Graph Covolutions* tutorial to show how you can use the Trident Chemwidgets (TCW) package to interact with and test the model you've trained.

Evaluating models on new data, including corner cases, is a critical step toward model deployment. However, generating new molecules to test in an interactive way is rarely straightforward. TCW provides several tools to help subset larger datasets and draw new molecules to test against your models. You can find the full documentation for the Trident Chemwidgets library here.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Installing the prerequisites

In [ ]:
```
!pip install tensorflow deepchem trident-chemwidgets seaborn
```

For this tutorial, you'll need Trident Chemwidgets version 0.2.0 or greater. We can check the installed version with the following command:

In [1]:
```
import trident_chemwidgets as tcw
print(tcw.__version__)
```

```
0.2.1
```

Throughout this tutorial, we'll use the convention `tcw` to call the classes from the Trident Chemwidgets package.

Exploring the data

We'll start out by loading the Tox21 dataset and extracting the predefined train, validation, and test splits.

In [2]:
```
import deepchem as dc

tasks, datasets, transformers = dc.molnet.load_tox21(featurizer='GraphConv')
train_dataset, valid_dataset, test_dataset = datasets
```

We can then use RDKit to calculate some additional features for each of the training examples. Specifically, we'll compute the logP and molecular weight of each molecule and return this new data in a dataframe.

In [3]:
```
import rdkit.Chem as Chem
from rdkit.Chem.Crippen import MolLogP
from rdkit.Chem.Descriptors import MolWt
import pandas as pd

data = []

for dataset, split in zip(datasets, ['train', 'valid', 'test']):
    for smiles in dataset.ids:
        mol = Chem.MolFromSmiles(smiles)
        logp = MolLogP(mol)
        mwt = MolWt(mol)
        data.append({
            'smiles': smiles,
            'logp': logp,
            'mwt': mwt,
            'split': split
        })

mol_data = pd.DataFrame(data)

mol_data.head()
```

```
[15:36:55] WARNING: not removing hydrogen atom without neighbors
```

Out[3]:

	smiles	logp	mwt	split
0	CC(O)(P(=O)(O)O)P(=O)(O)O	-0.9922	206.027	train
1	CC(C)(C)OOC(C)(C)CCC(C)(C)OOC(C)(C)C	4.8172	290.444	train
2	OC[C@H](O)[C@@H](O)[C@H](O)CO	-2.9463	152.146	train
3	CCCCCCCC(=O)[O-].CCCCCCCC(=O)[O-].[Zn+2]	2.1911	351.802	train
4	CC(C)COC(=O)C(C)C	1.8416	144.214	train

One-dimensional distributions

We can examine one-dimensional distributions using a histogram. Unlike histograms from static plotting libraries like Matplotlib or Seaborn, the TCW Histogram provides interactive functionality. TCW enables subsetting of the data, plotting chemical structures in a gallery next to the plot, and saving a reference to the subset portion of the dataframe. Unfortunately, this interactivity comes at the price of portability, so we have included screenshots for this tutorial in addition to providing the code to generate the interactive visuals. If you run this tutorial yourself (either locally or on Colab), you'll be able to display and interact with full demo plots.

In the plot below, you can see the histogram of the molecular weight distribution from the combined dataset on the left. If you click and drag within plot area in the live widget, you can subset a portion of the distribution for further examination. The background of the selected portion will turn gray and the selected data points will be shown in teal within the bars of the plot. The x axis of the Histogram widget is compatible with either numeric or date data types, which makes it a convenient choice for splitting your ML datasets based on a property or the date the experimental data were collected.

Histogram example

To generate an interactive example of the widget, run the next cell:

In [4]:
```
hist = tcw.Histogram(data=mol_data, smiles='smiles', x='mwt')
hist
```

```
Histogram(data={'points': [{'smiles': 'CC(O)(P(=O)(O)O)P(=O)(O)O', 'x': 206.027, 'index': 0}, {'smiles': 'CC(C…
```

If you select subset of the data by clicking and dragging, you can view the selected structures in the gallery to the right by pressing the `SHOW STRUCTURES` button beneath the plot. You can extract this subset of the original dataframe by pressing `SAVE SELECTION` and accessing the `hist.selection` property as shown in the next cell. This workflow is convenient for applications like data splitting based on a single dimension.

In [5]:
```
hist.selection
```

Out[5]:

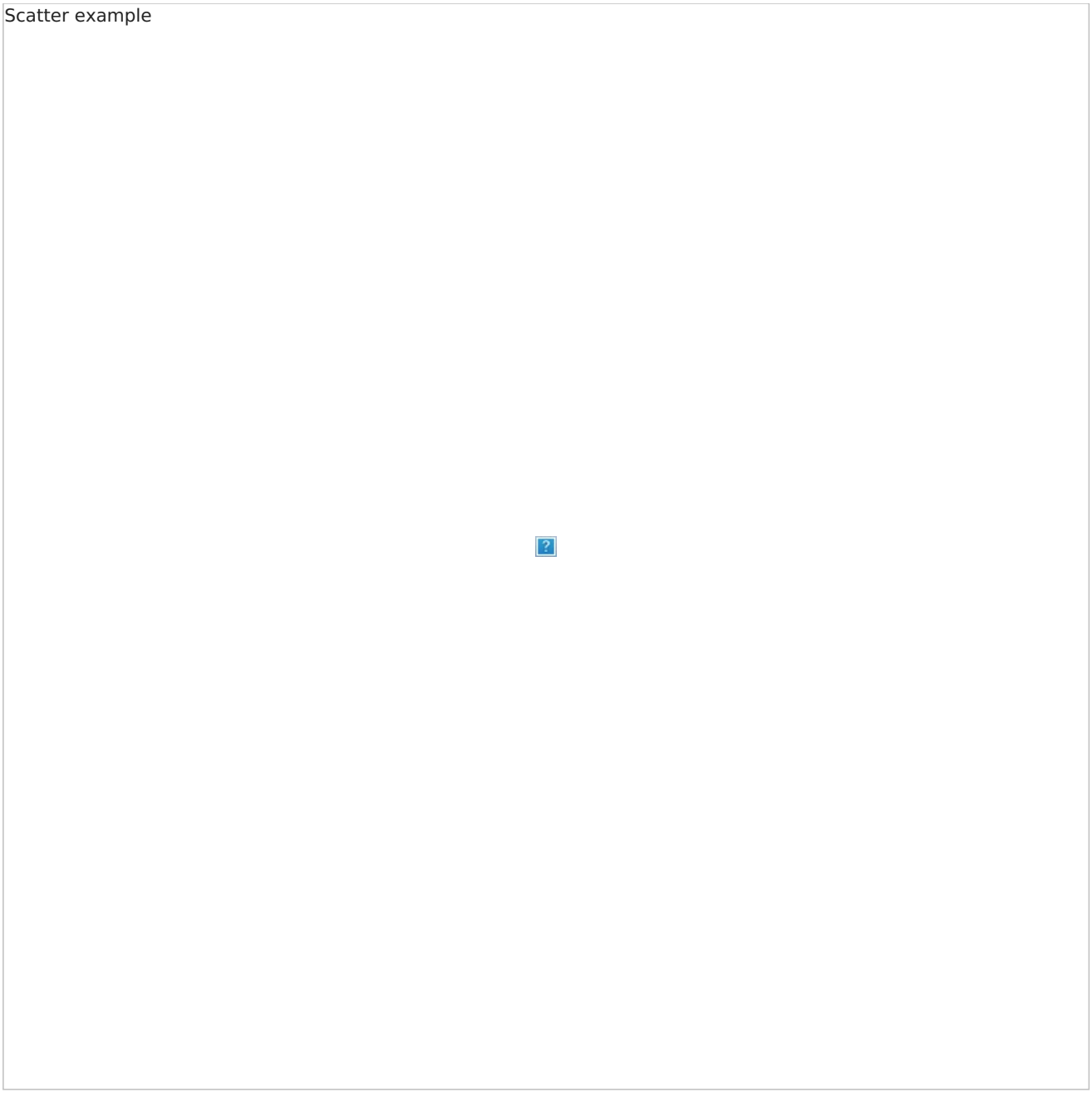
	smiles	logp	mwt	split

Two- or three-dimensional distributions

In addition to histograms, TCW also has provides a class for scatter plots. The Scatter class is useful when comparing two or three dimensions or your data. As of v0.2.0, TCW Scatter supports the use of the x and y axes as well as the color of each point (`hue` keyword) to represent either continuous or discrete variables. Just like in the Histogram example, you can click and drag within the plot area to subset along the x and y axes. The Scatter widget also supports dates along the x, y, and hue axes.

In the image below, we have selected a portion of dataset with large molecular weight values, but minimal training examples (displayed points in orange), to demonstrate how the Scatter widget can be useful for outlier identification. In addition to selection by bounding box, you can also hover over individual points to display a drawing of the underlying structure.

Scatter example



To generate an interactive example of the widget, run the next cell:

In [6]:
```
scatter = tcw.Scatter(data=mol_data, smiles='smiles', x='mwt', y='logp', hue='split')
scatter
```

```
Scatter(data={'points': [{'smiles': 'CC(O)(P(=O)(O)O)P(=O)(O)O', 'x': 206.027, 'y': -0.9922000000000002, 'hue'…
```

If you select subset of the data by clicking and dragging, you can view the selected structures in the gallery to the right by pressing the `SHOW STRUCTURES` button beneath the plot. You can extract this subset of the original dataframe by pressing `SAVE SELECTION` and accessing the `scatter.selection` property as shown in the next cell.

In [7]:
```
scatter.selection
```

Out[7]:

	smiles	logp	mwt	split

Training a GraphConvModel

Now that we've had a look at the training data, we can train a GraphConvModel to predict the 12 Tox21 classes. We'll replicate the training procedure exactly from the *Introduction to Graph Covolutions* tutorial. We'll train for 50 epochs, just as in the original tutorial.

In [8]:
```
# The next line filters tensorflow warnings relevant to memory consumption.
# To see these warnings, comment the next line.
import warnings; warnings.filterwarnings('ignore')

# Now we'll set the tensorflow seed to make sure the results of this notebook are reproducible
import tensorflow as tf; tf.random.set_seed(27)
```

```
n_tasks = len(tasks)
model = dc.models.GraphConvModel(n_tasks, mode='classification')
model.fit(train_dataset, nb_epoch=50)
```

```
2022-06-29 15:37:03.915828: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optim
ized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

Out[8]: 0.2661594772338867

Now that we have a trained model, we can check AUROC values for the training and test datasets:

In [9]:
```
metric = dc.metrics.Metric(dc.metrics.roc_auc_score)
print(f'Training set score: {model.evaluate(train_dataset, [metric], transformers)["roc_auc_score"]:.2f}')
print(f'Test set score: {model.evaluate(test_dataset, [metric], transformers)["roc_auc_score"]:.2f}')
```

```
Training set score: 0.97
Test set score: 0.68
```

Just as in the original tutorial, we see that the model performs reasonably well on the predefined train/test splits. Now we'll use this model to evaluate compounds that are outside the training distribution, just as we might in a real-world drug discovery scenario.

Evaluating the model on new data

One of the challenging first steps toward deploying an ML model in production is evaluating it on new data. Here, new data refers to both data outside the initial train/val/test distributions and also data that may not be already processed for use with the model.

We can use the JSME widget provided by TCW to quickly test our model again some molecules of interest. We'll start with a known therapeutic molecule: ibuprofen. We can see that ibuprofen is not included in any of the datasets that we have evaluated our model against so far:

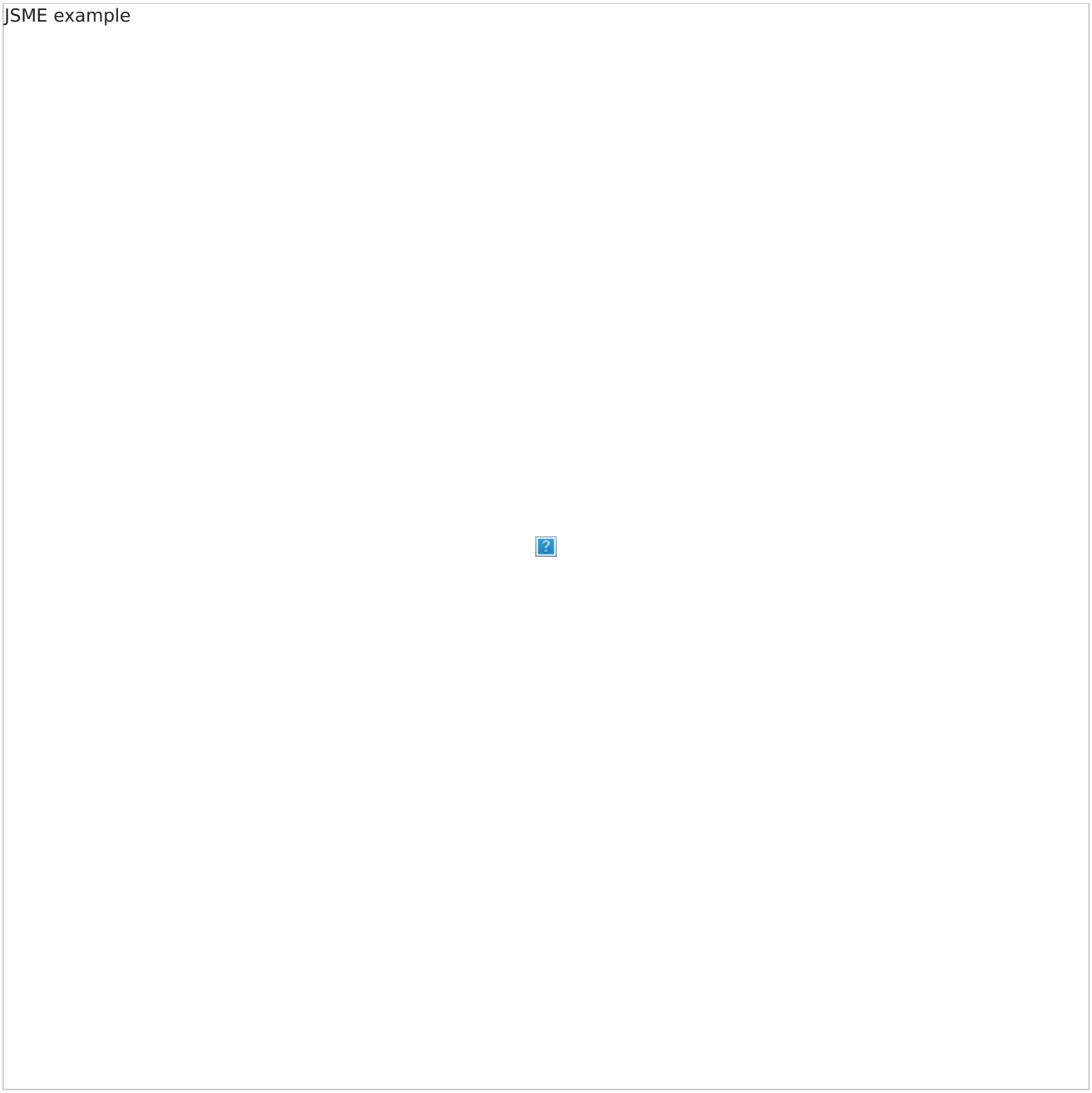
In [10]:
```
print(f"Ibuprofen structure in Tox21 dataset: {'CC(C)CC1=CC=C(C=C1)C(C)C(=O)O' in mol_data['smiles']}")
```

```
Ibuprofen structure in Tox21 dataset: False
```

To simulate a drug discovery application, let's say you're a chemist tasked with identifying potential new therapeutics derived from ibuprofen. Ideally, the molecules you test would have limited toxicity. You've just developed the model above to predict the tox outcomes from Tox21 data and now you want to use it to do some first-pass screening of your derivatives. The standard workflow for a task like this might include drawing the molecules in a program like ChemDraw, exporting to SMILES format, importing into the notebook, then prepping the data and running it through your model.

With TCW, we can shortcut the first few steps of that workflow by using the JSME widget to draw molecules and convert to SMILES directly in the notebook. We can even use the `base_smiles` argument to specify a base molecular structure, which is great for generating derivatives. Here we'll set the base_smiles value to `'CC(C)CC1=CC=C(C=C1)C(C)C(=O)O'`, the SMILES string for ibuprofen. Below is a screenshot using JSME to generate a few derivative molecules to test against our toxicity model.

JSME example



To generate your own set of derivatives, run the cell below. To add a SMILES string to the saved set, click the `ADD TO SMILES LIST` button below the interface. If you want to regenerate the original base molecule, in this case ibuprofen, click the `RESET TO BASE SMILES` button below the interface. By using this button, it's easy to generate distinct derivatives from a shared starting structure. Go ahead and create some ibuprofen derivatives to test against the tox model:

In [11]:
```
jsme = tcw.JSME(base_smiles='CC(C)CC1=CC=C(C=C1)C(C)C(=O)O')
jsme
```

```
JSME(base_smiles='CC(C)CC1=CC=C(C=C1)C(C)C(=O)O')
```

You can access the smiles using the `jsme.smiles` property. This call will return a list of the SMILES strings that have been added to the SMILES list of the widget (the ones shown in the molecule gallery to the right of the JSME interface).

In [12]:
```
print(jsme.smiles)
```

```
[]
```

To ensure the rest of this notebook runs correctly, the following cell sets the new test SMILES set to the ones from the screenshot above in the case that you have not defined your own set using the widget. Otherwise, it will use the molecules you have drawn.

In [13]:
```
# This cell will provide a preset list of SMILES strings in case you did not create your own.
if len(jsme.smiles) > 1:
    drawn_smiles = jsme.smiles
else:
    drawn_smiles = [
        'CC(C)Cc1ccc(C(C)C(=O)O)cc1',
        'CC(C)C(S)c1ccc(C(C)C(=O)O)cc1',
        'CCSC(c1ccc(C(C)C(=O)O)cc1)C(C)CC',
```

```
        'CCSC(c1ccc(C(C)C(=O)O)cc1)C(C)C(=O)O',
        'CC(C(=O)O)c1ccc(C(S)C(C)C(=O)O)cc1'
    ]
```

Next we have to create a dataset that is compatible with our model to test these new molecules.

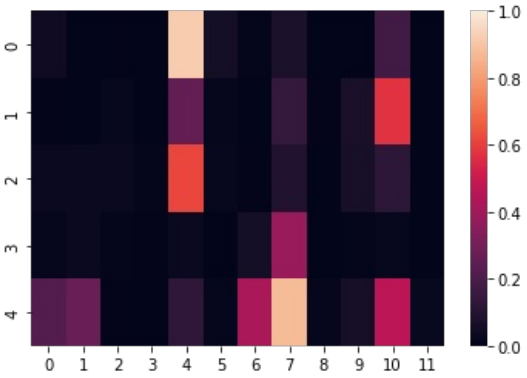
In [14]:
```
featurizer = dc.feat.ConvMolFeaturizer()
loader = dc.data.InMemoryLoader(tasks=list(train_dataset.tasks), featurizer=featurizer)
dataset = loader.create_dataset(drawn_smiles, shard_size=1)
```

Finally, we can generate our predictions of positive results here and plot them.

In [15]:
```
predictions = model.predict(dataset, transformers)[:, :, 1]

import seaborn as sns
sns.heatmap(predictions, vmin=0, vmax=1)
```

Out[15]: <AxesSubplot:>



Now we can get the predicted most toxic compound/assay result for further inspection. Below we extract the highest predicted positive hit (most toxic) and display the assay name, SMILES string, and an image of the structure.

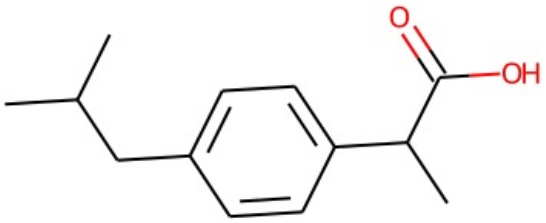
In [16]:
```
import numpy as np

mol_idx, assay_idx = np.unravel_index(predictions.argmax(), predictions.shape)
smiles = drawn_smiles[mol_idx]

print(f'Most toxic result (predicted): {train_dataset.tasks[assay_idx]}, {smiles}')
mol = Chem.MolFromSmiles(smiles)
mol
```

```
Most toxic result (predicted): NR-ER, CC(C)Cc1ccc(C(C)C(=O)O)cc1
```

Out[16]:



Interpreting the model's predictions

Often predictions alone are insufficient to decide whether to move forward with costly experiments. We might also want some metric or metrics that allow us to interpret the model's output.

Building on the tutorial Calculating Atomic Contributions for Molecules Based on a Graph Convolutional QSAR Model, we can calculate the relative contribution of each atom in a molecule to the predicted output value. This attribution strategy enables us to determine whether the molecular features that a chemist may identify as important and those most affecting the predictions are in alignment. If the chemist's interpretation and the model's interpretation metrics are consistent, that may indicate that the model is a good fit for the task at hand. However, the inverse is not necessarily true either. A model may have the capacity to make accurate predictions that a trained chemist cannot fully understand. This is just one tool in a machine learning practitioner's toolbox.

We'll start by using the built-in `per_atom_fragmentation` argument for the `ConvMolFeaturizer`. This will generate a list of ConvMol objects that have each had a single atom removed.

In [17]:
```
featurizer = dc.feat.ConvMolFeaturizer(per_atom_fragmentation=True)
mol_list = featurizer(smiles)
loader = dc.data.InMemoryLoader(tasks=list(train_dataset.tasks),
                                featurizer=dc.feat.DummyFeaturizer())
```

```
dataset = loader.create_dataset(mol_list[0], shard_size=1)
```

We can then run these predictions through the model and retrieve the predicted values for the molecule and assay specified in the last section.

In [18]:
```
full_molecule_prediction = predictions[mol_idx, assay_idx]
fragment_predictions = model.predict(dataset, transformers)[:, assay_idx, 0]
contributions = pd.DataFrame({
    'Change in predicted toxicity':
    (full_molecule_prediction - fragment_predictions).round(3)
})
```

We can use the InteractiveMolecule widget from TCW to superimpose the contribution scores on the molecule itself, allowing us to easily asses the relative importance of each atom to the final prediction. If you click on one of the atoms, you can retrieve the contribution data in a card shown to the right of the structure. In this panel you can also select a variable by which to color the atoms in the plot.

InteractiveMolecule example



You can generate the interactive widget by running the cell below.

In [19]:
```
tcw.InteractiveMolecule(smiles, data = contributions)
```

```
InteractiveMolecule(data=[{'Change in predicted toxicity': 0.5529999732971191}, {'Change in predicted toxicity…
```

Wrapping up

In this tutorial, we learned how to incorporate Trident Chemwidgets into your DeepChem-based ML workflow. While TCW was built with molecular ML workflows in mind, the library also works well for general cheminformatics notebooks as

well.

Star Trident Chemwidgets on GitHub

If you find the Trident Chemwidgets package helpful please give it a on GitHub. Starring the project helps it grow and find new audiences.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Tutorial: ChemBERTa: Large-Scale Self-Supervised Pretraining for Molecular Property Prediction using a Smiles Tokenization Strategy



By Seyone Chithrananda (Twitter)

Deep learning for chemistry and materials science remains a novel field with lots of potiential. However, the popularity of transfer learning based methods in areas such as natural language processing (NLP) and computer vision have not yet been effectively developed in computational chemistry + machine learning. Using HuggingFace's suite of models and the ByteLevel tokenizer, we are able to train a large-transformer model, RoBERTa, on a large corpus of 10,000,000 SMILES strings from a commonly known benchmark chemistry dataset, PubChem.

Training RoBERTa over 10 epochs, the model achieves a pretty good loss of 0.198, and may likely continue to converge if trained for a larger number of epochs. The model can predict masked/corrupted tokens within a SMILES sequence/molecule, allowing for variants of a molecule within discoverable chemical space to be predicted.

By applying the representations of functional groups and atoms learned by the model, we can try to tackle problems of toxicity, solubility, drug-likeness, and synthesis accessibility on smaller datasets using the learned representations as features for graph convolution and attention models on the graph structure of molecules, as well as fine-tuning of BERT. Finally, we propose the use of attention visualization as a helpful tool for chemistry practitioners and students to quickly identify important substructures in various chemical properties.

Additionally, visualization of the attention mechanism have been seen through previous research as incredibly valuable towards chemical reaction classification. The applications of open-sourcing large-scale transformer models such as RoBERTa with HuggingFace may allow for the acceleration of these individual research directions.

A link to a repository which includes the training, uploading and evaluation notebook (with sample predictions on compounds such as Remdesivir) can be found here. All of the notebooks can be copied into a new Colab runtime for easy execution. This repository will be updated with new features, such as attention visualization, easier benchmarking infrastructure, and more. The work behind this tutorial has been published on Arxiv, and was accepted for a **poster presentation at NeurIPS 2020's ML for Molecules Workshop**.

For the sake of this tutorial, we'll be fine-tuning a pre-trained ChemBERTa on a small-scale molecule dataset, Clintox, to show the potiential and effectiveness of HuggingFace's NLP-based transfer learning applied to computational chemistry. Output for some cells are purposely cleared for readability, so do not worry if some output messages for your cells differ!

In short, there are three major components we'll be going over in this notebook.

1. Masked token inference predictions on SMILES strings
2. Attention visualizaiton of the PubChem-10M model
3. Fine-tuninhg BPE-ChemBERTa and Smiles-Tokenizer ChemBERTa model's on the CLintox toxicity dataset.

Don't worry if you aren't familiar with some of these terms. We will explain them later in the tutorial!

If you're looking to dive deeper, check out the poster here.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

Open in Colab

Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5

minutes to run to completion and install your environment.

In [1]:
```
!curl -Lo conda_installer.py https://raw.githubusercontent.com/deepchem/deepchem/master/scripts/colab_install.py
import conda_installer
conda_installer.install()
!/root/miniconda/bin/conda info -e
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3501  100  3501    0     0  16995      0 --:--:-- --:--:-- --:--:-- 16995
add /root/miniconda/lib/python3.7/site-packages to PYTHONPATH
python version: 3.7.10
remove current miniconda
fetching installer from https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
done
installing miniconda to /root/miniconda
done
installing rdkit, openmm, pdbfixer
added omnia to channels
added conda-forge to channels
done
conda packages installation finished!
# conda environments:
#
base                  *  /root/miniconda
```

In [2]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

```
Requirement already satisfied: deepchem in /usr/local/lib/python3.7/dist-packages (2.5.0)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.4.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem) (0.22.2.po
st1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.19.5)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.0.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem) (2
018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
pandas->deepchem) (1.15.0)
wandb: WARNING W&B installed but not logged in.  Run `wandb login` or set the WANDB_API_KEY env variable.
wandb: WARNING W&B installed but not logged in.  Run `wandb login` or set the WANDB_API_KEY env variable.
```

Out[2]:
```
'2.5.0'
```

In [3]:
```
from rdkit import Chem
```

We want to install NVIDIA's Apex tool, for the training pipeline used by `simple-transformers` and Weights and Biases. This package enables us to use 16-bit training, mixed precision, and distributed training without any changes to our code. Generally GPUs are good at doing 32-bit(single precision) math, not at 16-bit(half) nor 64-bit(double precision). Therefore traditionally deep learning model trainings are done in 32-bit. By switching to 16-bit, we'll be using half the memory and theoretically less computation at the expense of the available number range and precision. However, pure 16-bit training creates a lot of problems for us (imprecise weight updates, gradient underflow and overflow). **Mixed precision training, with Apex, alleviates these problems**.

We will be installing `simple-transformers`, a library which builds ontop of HuggingFace's `transformers` package specifically for fine-tuning ChemBERTa.

In [ ]:
```
!git clone https://github.com/NVIDIA/apex
!cd /content/apex
!pip install -v --no-cache-dir /content/apex
!pip install transformers
!pip install simpletransformers
!pip install wandb
!cd ..
```

In [5]:
```
import sys
!test -d bertviz_repo && echo "FYI: bertviz_repo directory already exists, to pull latest version uncomment thi
# !rm -r bertviz_repo # Uncomment if you need a clean pull from repo
!test -d bertviz_repo || git clone https://github.com/jessevig/bertviz bertviz_repo
if not 'bertviz_repo' in sys.path:
  sys.path += ['bertviz_repo']
!pip install regex
```

```
FYI: bertviz_repo directory already exists, to pull latest version uncomment this line: !rm -r bertviz_repo
Requirement already satisfied: regex in /usr/local/lib/python3.7/dist-packages (2019.12.20)
```

We're going to clone an auxillary repository, bert-loves-chemistry, which will enable us to use the MolNet dataloader for ChemBERTa, which automatically generates scaffold splits on any MoleculeNet dataset!

In [6]:
```
!git clone https://github.com/seyonechithrananda/bert-loves-chemistry.git
```

```
fatal: destination path 'bert-loves-chemistry' already exists and is not an empty directory.
```

In [7]:
```
!nvidia-smi
```

```
Thu Mar 18 14:48:19 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.56       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   31C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Now, to ensure our model demonstrates an understanding of chemical syntax and molecular structure, we'll be testing it on predicting a masked token/character within the SMILES molecule for benzene.

In [8]:
```
# Test if NVIDIA apex training tool works
from apex import amp
```

What is a tokenizer?

A tokenizer is in charge of preparing the inputs for a natural language processing model. For many scientific applications, it is possible to treat inputs as “words”/”sentences” and use NLP methods to make meaningful predictions. For example, SMILES strings or DNA sequences have grammatical structure and can be usefully modeled with NLP techniques. DeepChem provides some scientifically relevant tokenizers for use in different applications. These tokenizers are based on those from the Huggingface transformers library (which DeepChem tokenizers inherit from).

The base classes PreTrainedTokenizer and PreTrainedTokenizerFast in HuggingFace implements the common methods for encoding string inputs in model inputs and instantiating/saving python tokenizers either from a local file or directory or from a pretrained tokenizer provided by the library (downloaded from HuggingFace’s AWS S3 repository).

PreTrainedTokenizer (transformers.PreTrainedTokenizer)) thus implements the main methods for using all the tokenizers:

- Tokenizing (spliting strings in sub-word token strings), converting tokens strings to ids and back, and encoding/decoding (i.e. tokenizing + convert to integers),
- Adding new tokens to the vocabulary in a way that is independant of the underlying structure (BPE, SentencePiece...),
- Managing special tokens like mask, beginning-of-sentence, etc tokens (adding them, assigning them to attributes in the tokenizer for easy access and making sure they are not split during tokenization)

The default tokenizer used by ChemBERTa, is a Byte-Pair-Encoder (BPE). It is a hybrid between character and word-level representations, which allows for the handling of large vocabularies in natural language corpora. Motivated by the intuition that rare and unknown words can often be decomposed into multiple known subwords, BPE finds the best word segmentation by iteratively and greedily merging frequent pairs of characters.

First, lets load the model's Byte-Pair Encoding tokenizer, and model, and setup a Huggingface pipeline for masked tokeni prediction.

In [9]:
```
from transformers import AutoModelForMaskedLM, AutoTokenizer, pipeline, RobertaModel, RobertaTokenizer
from bertviz import head_view

model = AutoModelForMaskedLM.from_pretrained("seyonec/PubChem10M_SMILES_BPE_450k")
tokenizer = AutoTokenizer.from_pretrained("seyonec/PubChem10M_SMILES_BPE_450k")

fill_mask = pipeline('fill-mask', model=model, tokenizer=tokenizer)
```

```
Downloading:   0%|          | 0.00/515 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/336M [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/165k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/101k [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/772 [00:00<?, ?B/s]
Downloading:   0%|          | 0.00/62.0 [00:00<?, ?B/s]
```

What is a transformer model?

Previously, we spoke about the attention mechanism in modern deep learning models. Attention is a concept that helped improve the performance of neural machine translation applications. The **Transformer** is a model that uses attention to boost the speed with which these models can be trained.

With the emergence of BERT by Google AI in 2018, transformers have quickly shot to the top of emerging deep learning methods, outperforming Neural Machine Translation models such as seq2seq and recurrent neural networks at dozens of tasks.

The biggest benefit, however, comes from how The Transformer lends itself to efficient **pre-training**. Using the same pre-training procedure used by RoBERTa, a follow-up work of BERT, which masks 15% of the tokens, we mask 15% of the tokens in each SMILES string and assign a maximum sequence length of 256 characters.

The model then learns to **predict masked tokens consisting of atoms and functional groups**, or specific groups of atoms within molecules which have their own characteristic properties. Through this, the model learns the relevant molecular context for transferable tasks, such as property prediction.

ChemBERTa employs a bidirectional training context to learn context-aware representations of the PubChem 10M dataset, downloadable through MoleculeNet for self-supervised pre-training (link). Our variant of the BERT transformer uses 12 attention heads and 6 layers, resulting in 72 distinct attention mechanisms.

The Transformer was proposed in the paper Attention is All You Need.

Now, to ensure our the ChemBERTa model demonstrates an understanding of chemical syntax and molecular structure, we'll be testing it on predicting a masked token/character within the SMILES molecule for benzene. Using the Huggingface pipeline we initialized earlier we can fetch a list of the model's predictions by confidence score:

In [10]:
```
smiles_mask = "C1=CC=CC<mask>C1"
smiles = "C1=CC=CC=C1"

masked_smi = fill_mask(smiles_mask)

for smi in masked_smi:
  print(smi)
```

```
{'sequence': 'C1=CC=CC=C1', 'score': 0.9755934476852417, 'token': 33, 'token_str': '='}
{'sequence': 'C1=CC=CC#C1', 'score': 0.020923888310790062, 'token': 7, 'token_str': '#'}
{'sequence': 'C1=CC=CC1C1', 'score': 0.0007658962858840823, 'token': 21, 'token_str': '1'}
{'sequence': 'C1=CC=CC2C1', 'score': 0.0004129768058191985, 'token': 22, 'token_str': '2'}
{'sequence': 'C1=CC=CC=[C1', 'score': 0.00025319133419543505, 'token': 352, 'token_str': '=['}
```

Here, we get some interesting results. The final branch, `C1=CC=CC=C1`, is a benzene ring. Since its a pretty common molecule, the model is easily able to predict the final double carbon bond with a score of 0.98. Let's get a list of the top 5 predictions (including the target, Remdesivir), and visualize them (with a highlighted focus on the beginning of the final benzene-like pattern). To visualize them, we'll be using the RDKit cheminoformatics package we installed earlier, specifically the `rdkit.chem.Draw` module.

In [11]:
```
import torch
import rdkit
import rdkit.Chem as Chem
from rdkit.Chem import rdFMCS
from matplotlib import colors
from rdkit.Chem import Draw
from rdkit.Chem.Draw import MolToImage
from PIL import Image


def get_mol(smiles):
    mol = Chem.MolFromSmiles(smiles)
    if mol is None:
        return None
    Chem.Kekulize(mol)
    return mol


def find_matches_one(mol,submol):
    #find all matching atoms for each submol in submol_list in mol.
    match_dict = {}
```

```
    mols = [mol,submol] #pairwise search
    res=rdFMCS.FindMCS(mols) #,ringMatchesRingOnly=True)
    mcsp = Chem.MolFromSmarts(res.smartsString)
    matches = mol.GetSubstructMatches(mcsp)
    return matches

#Draw the molecule
def get_image(mol,atomset):
    hcolor = colors.to_rgb('green')
    if atomset is not None:
        #highlight the atoms set while drawing the whole molecule.
        img = MolToImage(mol, size=(600, 600),fitImage=True, highlightAtoms=atomset,highlightColor=hcolor)
    else:
        img = MolToImage(mol, size=(400, 400),fitImage=True)
    return img
```

In [12]:
```
sequence = f"C1=CC=CC={tokenizer.mask_token}1"
substructure = "CC=CC"
image_list = []

input = tokenizer.encode(sequence, return_tensors="pt")
mask_token_index = torch.where(input == tokenizer.mask_token_id)[1]

token_logits = model(input)[0]
mask_token_logits = token_logits[0, mask_token_index, :]

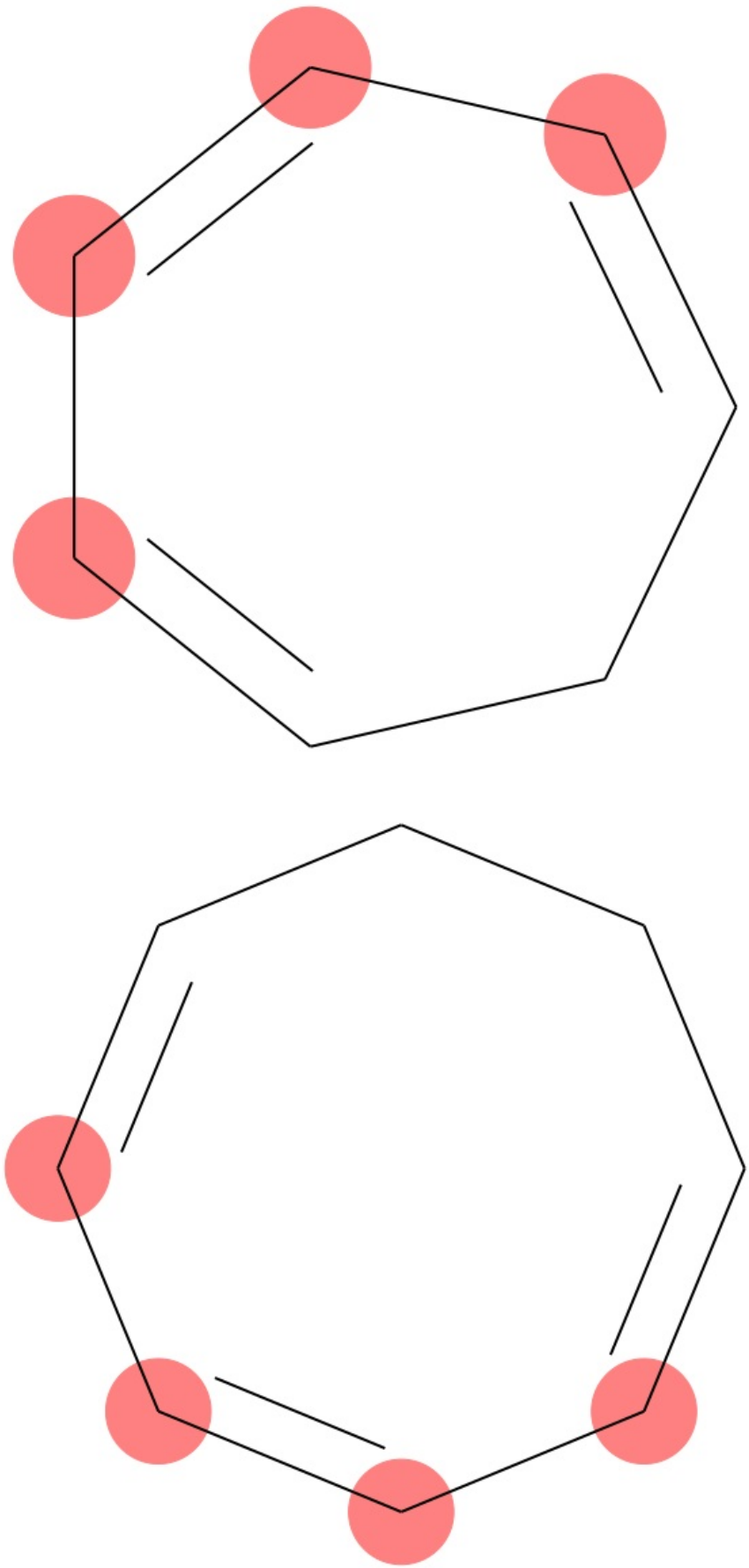
top_5_tokens = torch.topk(mask_token_logits, 5, dim=1).indices[0].tolist()

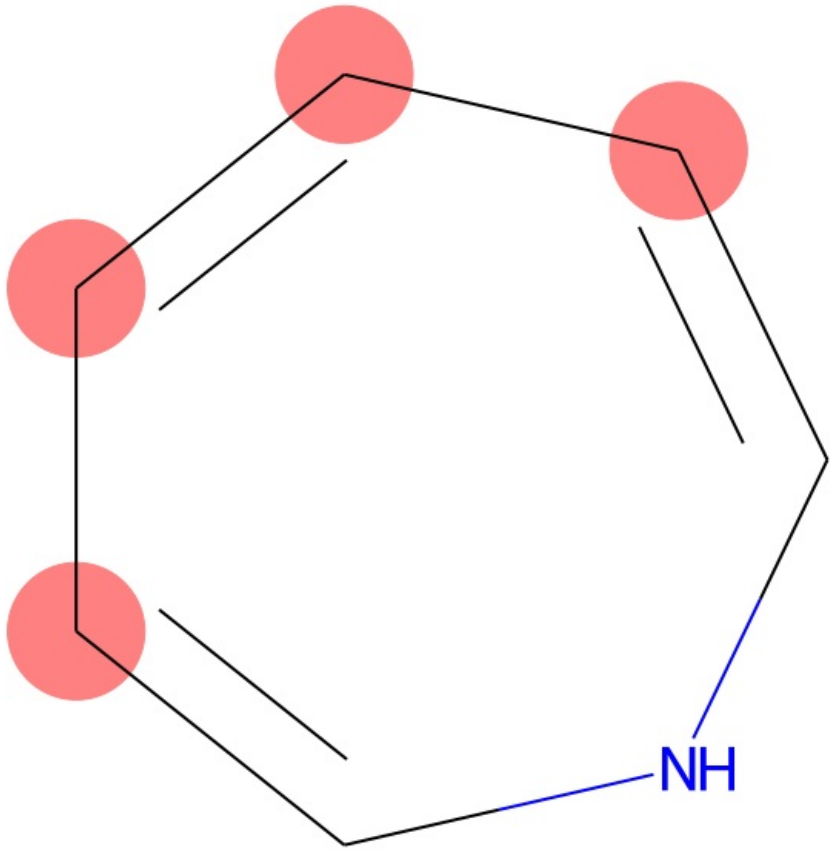
for token in top_5_tokens:
  smi = (sequence.replace(tokenizer.mask_token, tokenizer.decode([token])))
  print (smi)
  smi_mol = get_mol(smi)
  substructure_mol = get_mol(substructure)
  if smi_mol is None: # if the model's token prediction isn't chemically feasible
    continue
  Draw.MolToFile(smi_mol, smi+".png")
  matches = find_matches_one(smi_mol, substructure_mol)
  atomset = list(matches[0])
  img = get_image(smi_mol, atomset)
  img.format="PNG"
  image_list.append(img)
```

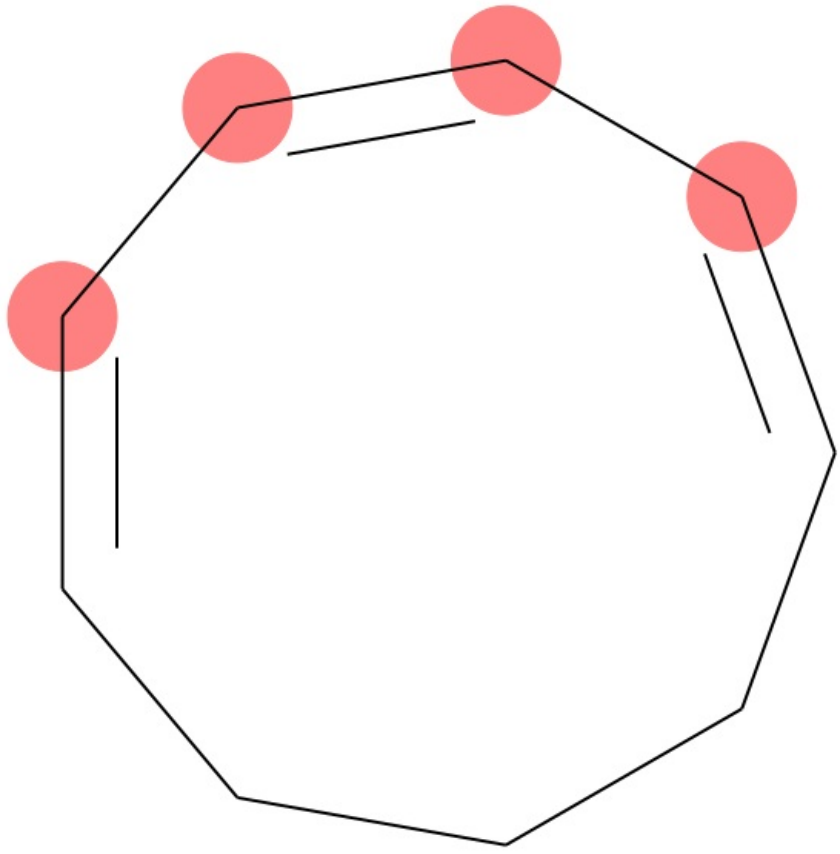
```
C1=CC=CC=CC1
C1=CC=CC=CCC1
C1=CC=CC=CN1
C1=CC=CC=CCCC1
C1=CC=CC=CCO1
```

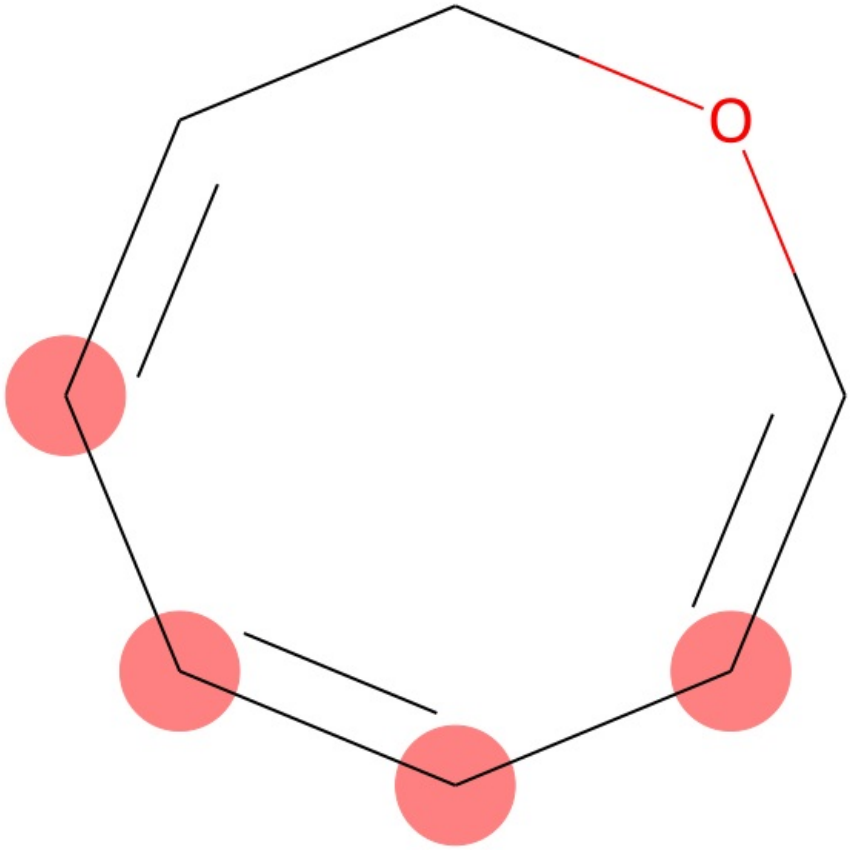
In [13]:
```
from IPython.display import Image

for img in image_list:
  display(img)
```




NH





As we can see above, 5 out of 5 of the model's MLM predictions are chemically valid. Overall, the model seems to understand syntax with a pretty decent degree of certainity.

However, further training on a more specific dataset (say leads for a specific target) may generate a stronger chemical transformer model. Let's now fine-tune our model on a dataset of our choice, ClinTox. You can run ChemBERTa on any MoleculeNet dataset, but for the sake of convinience, we will use ClinTox as it is small and trains quickly.

What is attention?

Previously, recurrent models struggled with generating a fixed-length vector for large sequences, leading to deteriorating performance as the length of an input sequence increased.

Attention is, to some extent, motivated by how we pay visual attention to different regions of our vision or how we correlate words in a sentence. Human visual attention allows us to focus on a certain subregion with a higher focus while perceiving the surrounding image in with a lower focus, and then adjust the focal point.

Similarly, we can explain the relationship between words in one sentence or close context. When we see “eating”, we expect to read a food word very soon. The color term describes the food, but probably not as directly as “eating” does:



The attention mechanism extends on the encoder-decoder model, by taking in three values for a SMILES sequence: a value vector (V), a query vector (Q) and a key vector (K).

Each vector is similar to a type of word embedding, specifically for determining the compatibility of neighbouring tokens. From these vectors, a dot production attention is derived from the dot product of the query vector of one word, and the key vector of the other.

A scaling factor of $\frac{1}{\sqrt{d_k}}$ is added to the dot product attention such that the value doesn't grow too large in respect to d_k, the dimension of the key. The softmax normalization function is applied to return a score between 0 to 1 for each individual token:

[Math Processing Error]

Visualizing the Attention Mechanism in ChemBERTa using BertViz

BertViz is a tool for visualizing attention in the Transformer model, supporting all models from the transformers library (BERT, GPT-2, XLNet, RoBERTa, XLM, CTRL, etc.). It extends the Tensor2Tensor visualization tool by Llion Jones and the transformers library from HuggingFace.

Using this tool, we can easily plug in ChemBERTa from the HuggingFace model hub and visualize the attention patterns produced by one or more attention heads in a given transformer layer. This is known as the attention-head view.

Lets start by obtaining a Javascript object for d3.js and jquery to create interactive visualizations:

In [14]:
```
%%javascript
require.config({
  paths: {
      d3: '//cdnjs.cloudflare.com/ajax/libs/d3/3.4.8/d3.min',
      jquery: '//ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min',
  }
});
```

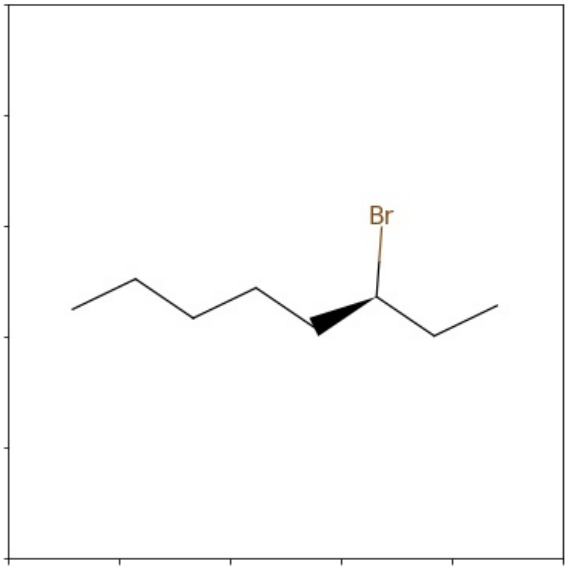
In [15]:
```
def call_html():
  import IPython
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
        <script>
          requirejs.config({
            paths: {
              base: '/static/base',
              "d3": "https://cdnjs.cloudflare.com/ajax/libs/d3/3.5.8/d3.min",
              jquery: '//ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min',
            },
          });
        </script>
        '''))
```

Now, we create an instance of ChemBERTa, tokenize a set of SMILES strings, and compute the attention for each head in the transformer. There are two available models hosted by DeepChem on HuggingFace's model hub, one being `seyonec/ChemBERTa-zinc-base-v1` which is the ChemBERTa model trained via masked lagnuage modelling (MLM) on the ZINC100k dataset, and the other being `seyonec/ChemBERTa-zinc250k-v1`, which is trained via MLM on the larger ZINC250k dataset.

In the following example, we take two SMILES molecules from the ZINC database with nearly identical chemical structure, the only difference being rooted in chiral specification (hence the additional `'@'` symbol). This is a feature of molecules which indicates that there exists tetrahedral centres. `'@'` tells us whether the neighbours of a molecule appear in a counter-clockwise order, whereas `'@@'` indicates that the neighbours are ordered in a clockwise direction. The model should ideally refer to similar substructures in each SMILES string with a higher attention weightage.

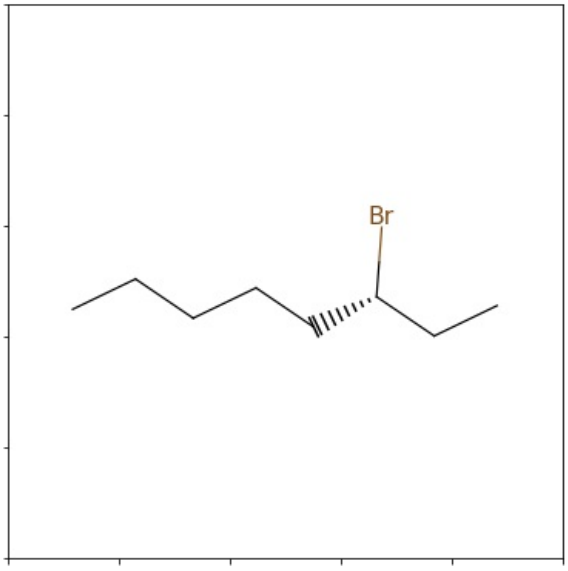
Lets look at the first SMILES string: `CCCCC[C@@H](Br)CC`:

In [16]:
```
m = Chem.MolFromSmiles('CCCCC[C@@H](Br)CC')
fig = Draw.MolToMPL(m, size=(200, 200))
```



And the second SMILES string, `CCCCC[C@H](Br)CC` :

In [17]:
```
m = Chem.MolFromSmiles('CCCCC[C@H](Br)CC')
fig = Draw.MolToMPL(m, size=(200,200))
```



The visualization below shows the attention induced by a sample input SMILES. This view visualizes attention as lines connecting the tokens being updated (left) with the tokens being attended to (right), following the design of the figures above. Color intensity reflects the attention weight; weights close to one show as very dark lines, while weights close to zero appear as faint lines or are not visible at all. The user may highlight a particular SMILES character to see the attention from that token only. This visualization is called the attention-head view. It is based on the excellent Tensor2Tensor visualization tool, and are all generated by the Bertviz library.

In [18]:
```
from transformers import RobertaModel, RobertaTokenizer
from bertviz import head_view

model_version = 'seyonec/PubChem10M_SMILES_BPE_450k'
model = RobertaModel.from_pretrained(model_version, output_attentions=True)
tokenizer = RobertaTokenizer.from_pretrained(model_version)

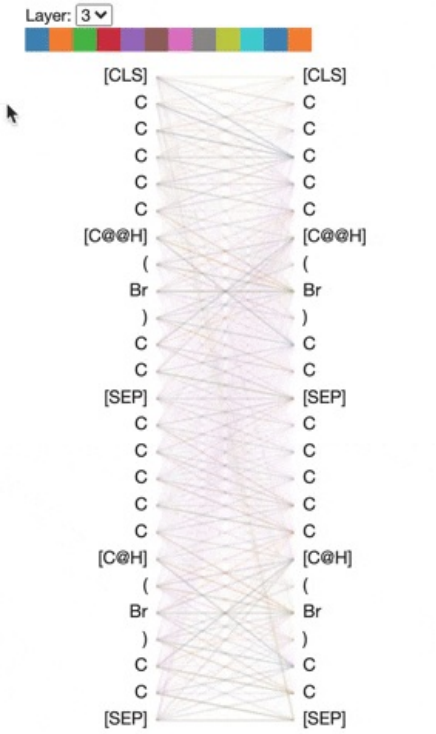
sentence_a = "CCCCC[C@@H](Br)CC"
sentence_b = "CCCCC[C@H](Br)CC"
inputs = tokenizer.encode_plus(sentence_a, sentence_b, return_tensors='pt', add_special_tokens=True)
input_ids = inputs['input_ids']
attention = model(input_ids)[-1]
input_id_list = input_ids[0].tolist() # Batch index 0
tokens = tokenizer.convert_ids_to_tokens(input_id_list)

call_html()

head_view(attention, tokens)
```

Layer:

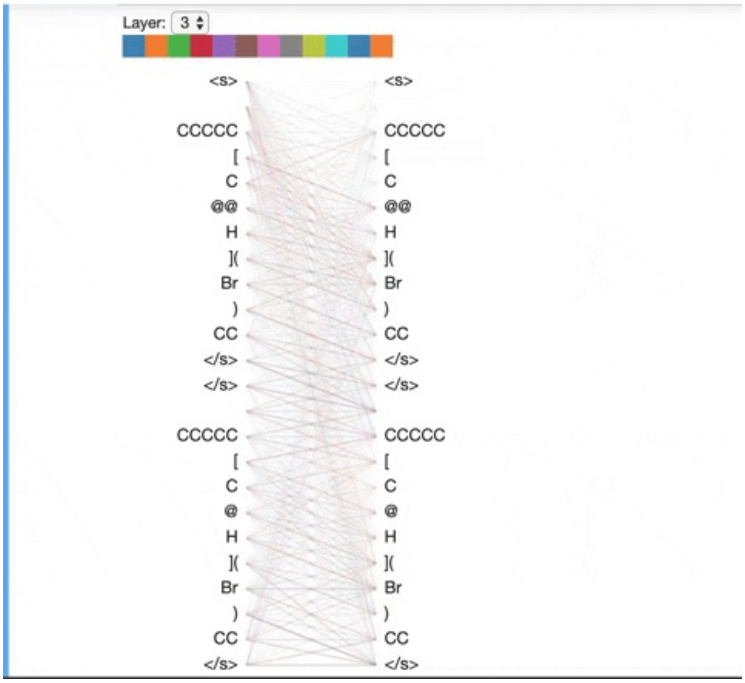
Smiles-Tokenizer Attention by Head View



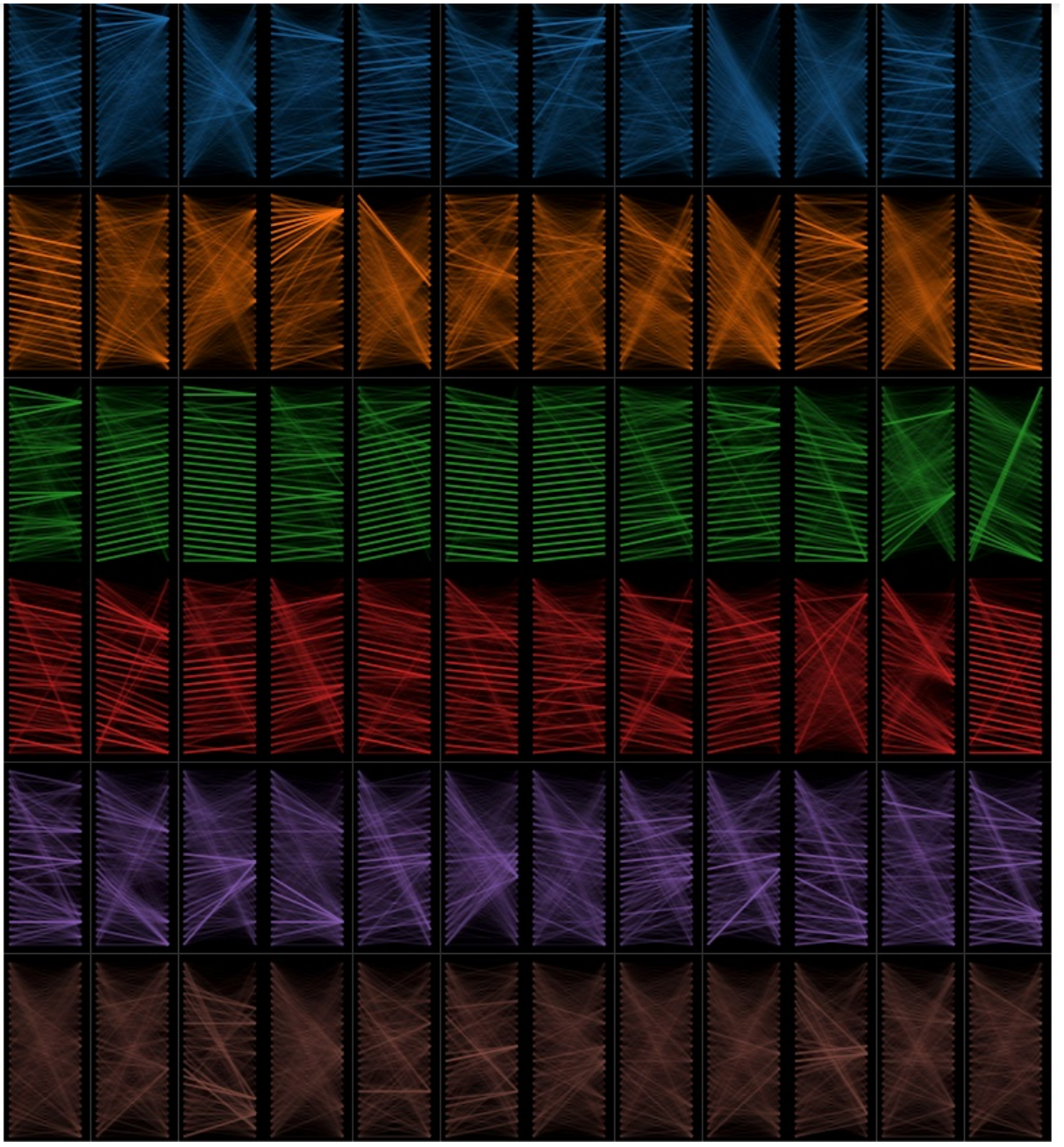
The visualization shows that attention is highest between words that don't cross a boundary between the two SMILES strings; the model seems to understand that it should relate tokens to other tokens in the same molecule in order to best understand their context.

There are many other fascinating visualizations we can do, such as a neuron-by neuron analysis of attention or a model overview that visualizes all of the heads at once:

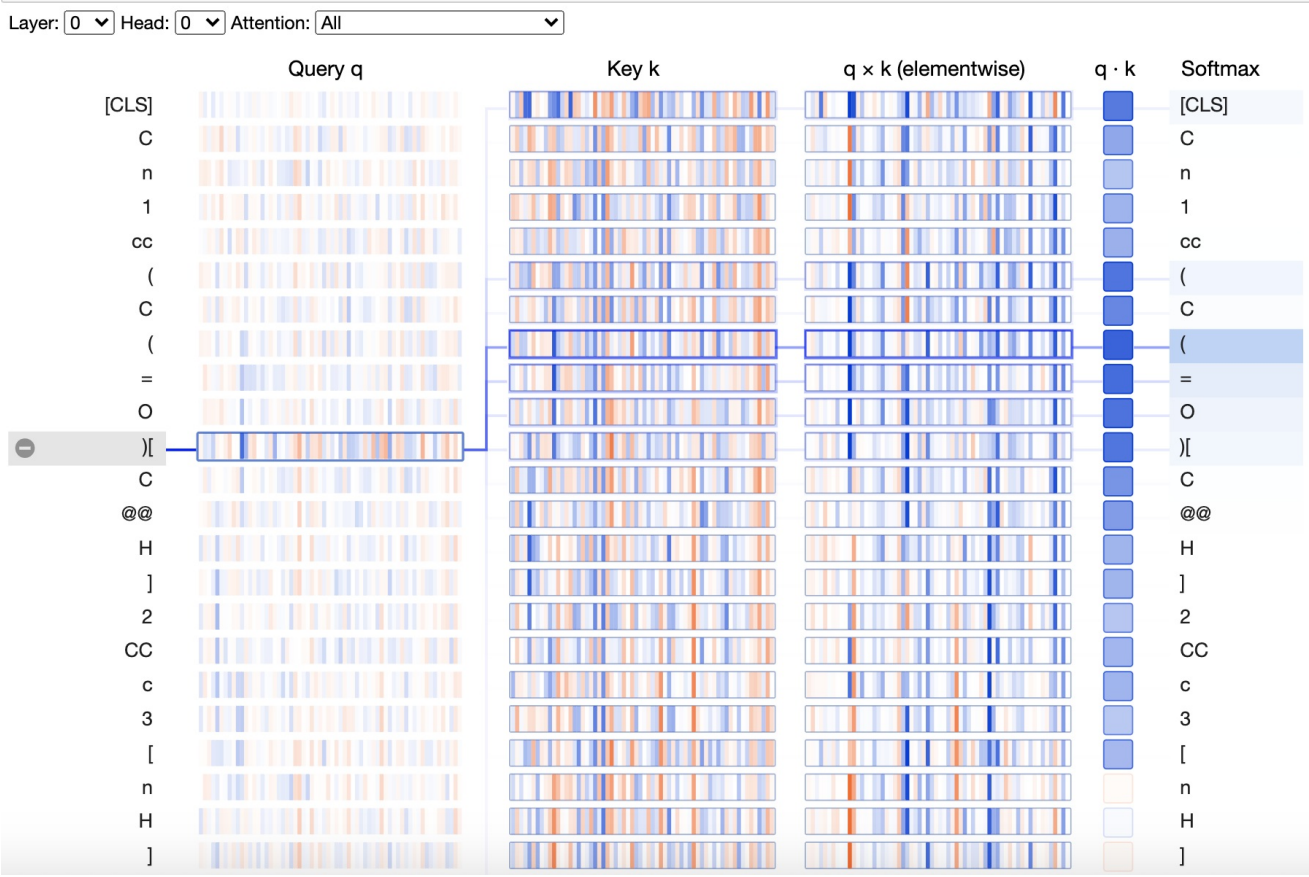
Attention by Head View:



Model View:



Neuron-by-neuron view:



You can try out the ChemBERTa attention visualization demo's in more detail, with custom SMILES/SELFIES strings, tokenizers, and more in the public library, here.

What is Transfer Learning, and how does ChemBERTa utilize it?

Transfer learning is a research problem in machine learning that focuses on **storing knowledge gained while solving one problem and applying it to a different but related problem**.

By pre-training directly on SMILES strings, and teaching ChemBERTa to recognize masked tokens in each string, the model learns a strong molecular representation. We then can take this model, trained on a structural chemistry task, and apply it to a suite of classification tasks in the MoleculeNet suite, from Tox21 to BBBP!

Fine-tuning ChemBERTa on a Small Mollecular Dataset

Our fine-tuning dataset, ClinTox, consists of qualitative data of drugs approved by the FDA and those that have failed clinical trials for toxicity reasons.

The ClinTox dataset consists of 1478 binary labels for toxicity, using the SMILES representations for identifying molecules. The computational models produced from the dataset could become decision-making tools for government agencies in determining which drugs are of the greatest potential concern to human health. Additionally, these models can act as drug screening tools in the drug discovery pipelines for toxicity.

Let's start by importing the MolNet dataloder from `bert-loves-chemistry`, before importing apex and transformers, the tool which will allow us to import the ChemBERTA language model (LM) trained on PubChem-10M.

In [19]:
```
%cd /content/bert-loves-chemistry
```

```
/content/bert-loves-chemistry
```

In [20]:
```
!pwd
```

```
/content/bert-loves-chemistry
```

In [21]:
```
import os

import numpy as np
import pandas as pd

from typing import List
```

```
# import molnet loaders from deepchem
from deepchem.molnet import load_bbbp, load_clearance, load_clintox, load_delaney, load_hiv, load_qm7, load_tox
from rdkit import Chem


# import MolNet dataloder from bert-loves-chemistry fork
from chemberta.utils.molnet_dataloader import load_molnet_dataset, write_molnet_dataset_for_chemprop
```

But why use custom Smiles-Tokenizer's over BPE?

In this tutorial, we will be comparing the BPE tokenization algorithm with a **custom SmilesTokenizer** based on a regex pattern, which we have released as part of DeepChem. To compare tokenizers, we pretrained an identical model tokenized using this novel tokenizer, on the PubChem-1M set. The pretrained model was evaluated on the BBBP and Tox21 in the paper. We found that the SmilesTokenizer narrowly outperformed the BPE algorithm by ΔPRC-AUC = $+0.021$
.

Though this result suggests that a more semantically relevant tokenization may provide performance benefits, further benchmarking on additional datasets is needed to validate this finding. **In this tutorial, we aim to do so, by testing this alternate model on the ClinTox dataset.**

Let's fetch the Smiles Tokenizer's character per line vocabulary file, which can bve loaded from the DeepChem S3 data bucket:

In [22]:
```
!wget https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/vocab.txt
```

```
--2021-03-18 14:48:45--  https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/vocab.txt
Resolving deepchemdata.s3-us-west-1.amazonaws.com (deepchemdata.s3-us-west-1.amazonaws.com)... 52.219.113.41
Connecting to deepchemdata.s3-us-west-1.amazonaws.com (deepchemdata.s3-us-west-1.amazonaws.com)|52.219.113.41|:4
43... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3524 (3.4K) [text/plain]
Saving to: ‘vocab.txt’

vocab.txt           100%[===================>]   3.44K  --.-KB/s    in 0s

2021-03-18 14:48:46 (62.2 MB/s) - ‘vocab.txt’ saved [3524/3524]
```

Lets use the MolNet dataloader to generate scaffold splits from the ClinTox dataset.

In [23]:
```
tasks, (train_df, valid_df, test_df), transformers = load_molnet_dataset("clintox", tasks_wanted=None)
```

```
'split' is deprecated.  Use 'splitter' instead.
Failed to featurize datapoint 7, None. Appending empty array
Exception message: Python argument types in
    rdkit.Chem.rdmolfiles.CanonicalRankAtoms(NoneType)
did not match C++ signature:
    CanonicalRankAtoms(RDKit::ROMol mol, bool breakTies=True, bool includeChirality=True, bool includeIsotopes=T
rue)
Failed to featurize datapoint 302, None. Appending empty array
Exception message: Python argument types in
    rdkit.Chem.rdmolfiles.CanonicalRankAtoms(NoneType)
did not match C++ signature:
    CanonicalRankAtoms(RDKit::ROMol mol, bool breakTies=True, bool includeChirality=True, bool includeIsotopes=T
rue)
Failed to featurize datapoint 983, None. Appending empty array
Exception message: Python argument types in
    rdkit.Chem.rdmolfiles.CanonicalRankAtoms(NoneType)
did not match C++ signature:
    CanonicalRankAtoms(RDKit::ROMol mol, bool breakTies=True, bool includeChirality=True, bool includeIsotopes=T
rue)
Failed to featurize datapoint 984, None. Appending empty array
Exception message: Python argument types in
    rdkit.Chem.rdmolfiles.CanonicalRankAtoms(NoneType)
did not match C++ signature:
    CanonicalRankAtoms(RDKit::ROMol mol, bool breakTies=True, bool includeChirality=True, bool includeIsotopes=T
rue)
Failed to featurize datapoint 1219, None. Appending empty array
Exception message: Python argument types in
    rdkit.Chem.rdmolfiles.CanonicalRankAtoms(NoneType)
did not match C++ signature:
    CanonicalRankAtoms(RDKit::ROMol mol, bool breakTies=True, bool includeChirality=True, bool includeIsotopes=T
rue)
Failed to featurize datapoint 1220, None. Appending empty array
Exception message: Python argument types in
    rdkit.Chem.rdmolfiles.CanonicalRankAtoms(NoneType)
did not match C++ signature:
    CanonicalRankAtoms(RDKit::ROMol mol, bool breakTies=True, bool includeChirality=True, bool includeIsotopes=T
rue)
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray
from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
Using tasks ['CT_TOX'] from available tasks for clintox: ['FDA_APPROVED', 'CT_TOX']
```

If you're only running the toxicity prediction portion of this tutorial, make sure you install transformers here. If you've ran all the cells before, you can ignore this install as we've already done `pip install transformers` before.

In [24]:
```
!pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (4.4.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.59.0)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in /usr/local/lib/python3.7/dist-packa
ges (from transformers) (3.7.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (
2019.12.20)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.19.5
)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
Requirement already satisfied: tokenizers<0.11,>=0.10.1 in /usr/local/lib/python3.7/dist-packages (from transfor
mers) (0.10.1)
Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-packages (from transformers) (0.0.43)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.0.12)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from transformers) (20.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata; pyt
hon_version < "3.8"->transformers) (3.4.1)
Requirement already satisfied: typing-extensions>=3.6.4; python_version < "3.8" in /usr/local/lib/python3.7/dist
-packages (from importlib-metadata; python_version < "3.8"->transformers) (3.7.4.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->trans
formers) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transforme
rs) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages
(from requests->transformers) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->tran
sformers) (2020.12.5)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (
7.1.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1.
15.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers)
(1.0.1)
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->trans
formers) (2.4.7)
```

In [25]:
```
train_df
```

Out[25]:

	text	labels
0	CC(C)C[C@H](NC(=O)CNC(=O)c1cc(Cl)ccc1Cl)B(O)O	0
1	O=C(NCC(O)CO)c1c(I)c(C(=O)NCC(O)CO)c(I)c(N(CCO...	1
2	Clc1cc(Cl)c(OCC#Cl)cc1Cl	1
3	N#Cc1cc(NC(=O)C(=O)[O-])c(Cl)c(NC(=O)C(=O)[O-])c1	1
4	NS(=O)(=O)c1cc(Cl)c(Cl)c(S(N)(=O)=O)c1	1
...	...	...
1177	CC(C[NH2+]C1CCCCC1)OC(=O)c1ccccc1	1
1178	CC(C(=O)[O-])c1ccc(C(=O)c2cccs2)cc1	1
1179	CC(c1cc2ccccc2s1)N(O)C(N)=O	1
1180	CC(O)C(CO)NC(=O)C1CSSCC(NC(=O)C([NH3+])Cc2cccc...	1
1181	CC(C)OC(=O)CCC/C=C\C[C@H]1[C@@H](O)C[C@@H](O)[...	1

1182 rows × 2 columns

In [26]:
```
valid_df
```

Out[26]:

	text	labels
0	CC(C)OC(=O)CCC/C=C\C[C@H]1[C@@H](O)C[C@@H](O)[...	1
1	CC(C)Nc1cccnc1N1CCN(C(=O)c2cc3cc(NS(C)(=O)=O)c...	1
2	CC(C)n1c(/C=C/[C@H](O)C[C@H](O)CC(=O)[O-])c(-c...	1
3	CC(C)COCC(CN(Cc1ccccc1)c1ccccc1)[NH+]1CCCC1	1
4	CSCC[C@H](NC(=O)[C@H](Cc1c[nH]c2ccccc12)NC(=O)...	1
...	...	...
143	C[C@H](OC(=O)c1ccccc1)C1=CCC23OCC[NH+](C)CC12C...	1
144	C[C@@H](c1ncncc1F)[C@](O)(Cn1cncn1)c1ccc(F)cc1F	1
145	CC(C)C[C@@H](NC(=O)[C@H](C)NC(=O)CNC(=O)[C@@H]...	1
146	C[C@H](O)[C@H](O)[C@H]1CNc2[nH]c(N)nc(=O)c2N1	1
147	C[NH+]1C[C@H](C(=O)N[C@]2(C)O[C@@]3(O)[C@@H]4C...	1

148 rows × 2 columns

In [27]:
```
test_df
```

Out[27]:

	text	labels
0	C[NH+]1C[C@H](C(=O)N[C@]2(C)O[C@@]3(O)[C@@H]4C...	1
1	C[C@]1(Cn2ccnn2)[C@H](C(=O)[O-])N2C(=O)C[C@H]2...	1
2	C[NH+]1CCC[C@@H]1CCO[C@](C)(c1ccccc1)c1ccc(Cl)cc1	1
3	Nc1nc(NC2CC2)c2ncn([C@H]3C=C[C@@H](CO)C3)c2n1	1
4	OC[C@H]1O[C@@H](n2cnc3c2NC=[NH+]C[C@H]3O)C[C@@...	1
...	...	...
143	O=C1O[C@H]([C@@H](O)CO)C([O-])=C1O	1
144	C#CCC(Cc1cnc2nc(N)nc(N)c2n1)c1ccc(C(=O)N[C@@H]...	1
145	C#CC[NH2+][C@@H]1CCc2ccccc21	1
146	[H]/[NH+]=C(\N)c1ccc(OCCCCCOc2ccc(/C(N)=[NH+]/...	1
147	[H]/[NH+]=C(\N)C1=CC(=O)/C(=C\C=c2ccc(=C(N)[NH...	1

148 rows × 2 columns

From here, lets set up a logger to record if any issues occur, and notify us if there are any problems with the arguments we've set for the model.

In [28]:
```
from simpletransformers.classification import ClassificationModel
import logging
logging.basicConfig(level=logging.INFO)
transformers_logger = logging.getLogger("transformers")
transformers_logger.setLevel(logging.WARNING)
```

Now, using `simple-transformer`, let's load the pre-trained model from HuggingFace's useful model-hub. We'll set the number of epochs to 10 in the arguments, but you can train for longer, and pass early-stopping as an argument to prevent overfitting. Also make sure that `auto_weights` is set to True to do automatic weight balancing, as we are dealing with imbalanced toxicity datasets.

In [29]:
```
from simpletransformers.classification import ClassificationModel, ClassificationArgs

model = ClassificationModel('roberta', 'seyonec/PubChem10M_SMILES_BPE_396_250', args={'evaluate_each_epoch': Tru
```

```
INFO:filelock:Lock 139908324261648 acquired on /root/.cache/huggingface/transformers/fac1cb3c26e15ed0ea455cf8111
5189edfd28b0cfa0ad7dca9922b8319475530.6662bce220e70bb69e1cc10c236b68e778001c010a6880b624c2159a235be52d.lock
Downloading:   0%|          | 0.00/515 [00:00<?, ?B/s]
INFO:filelock:Lock 139908324261648 released on /root/.cache/huggingface/transformers/fac1cb3c26e15ed0ea455cf8111
5189edfd28b0cfa0ad7dca9922b8319475530.6662bce220e70bb69e1cc10c236b68e778001c010a6880b624c2159a235be52d.lock
INFO:filelock:Lock 139908246375248 acquired on /root/.cache/huggingface/transformers/fca63b78d86d5e1ceec66e1d9f3
ff8ec0d078055e0ba387926cf9baf6b86ce79.93843c462ba2f6d2fecf01338be4b448f0b6f8f7dfed6535b7ffbd3e4203f223.lock
Downloading:   0%|          | 0.00/336M [00:00<?, ?B/s]
INFO:filelock:Lock 139908246375248 released on /root/.cache/huggingface/transformers/fca63b78d86d5e1ceec66e1d9f3
ff8ec0d078055e0ba387926cf9baf6b86ce79.93843c462ba2f6d2fecf01338be4b448f0b6f8f7dfed6535b7ffbd3e4203f223.lock
Some weights of the model checkpoint at seyonec/PubChem10M_SMILES_BPE_396_250 were not used when initializing Ro
bertaForSequenceClassification: ['lm_head.bias', 'lm_head.dense.weight', 'lm_head.dense.bias', 'lm_head.layer_no
rm.weight', 'lm_head.layer_norm.bias', 'lm_head.decoder.weight', 'lm_head.decoder.bias']
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model train
ed on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model t
hat you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenc
eClassification model).
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at seyonec/PubCh
em10M_SMILES_BPE_396_250 and are newly initialized: ['classifier.dense.weight', 'classifier.dense.bias', 'classi
fier.out_proj.weight', 'classifier.out_proj.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
INFO:filelock:Lock 139908246200848 acquired on /root/.cache/huggingface/transformers/3df58ba3fcca472da48db1fb3a6
69ebed9808cae886e8f7b99e6aed197a808cb.98d8cf992f31bc68994648ce3120c3cb14bf75e4e60a70a06cc61cce44b902f0.lock
Downloading:   0%|          | 0.00/165k [00:00<?, ?B/s]
INFO:filelock:Lock 139908246200848 released on /root/.cache/huggingface/transformers/3df58ba3fcca472da48db1fb3a6
69ebed9808cae886e8f7b99e6aed197a808cb.98d8cf992f31bc68994648ce3120c3cb14bf75e4e60a70a06cc61cce44b902f0.lock
INFO:filelock:Lock 139908246331344 acquired on /root/.cache/huggingface/transformers/3aa7993e4d850d3abfc9b05959c
762d864591d3d5d450310e9ccb1ef0b2c339c.07b622242dcc6c7fd6a5f356d2e200c4f44be0279b767b85afcb24e778809d3c.lock
Downloading:   0%|          | 0.00/101k [00:00<?, ?B/s]
INFO:filelock:Lock 139908246331344 released on /root/.cache/huggingface/transformers/3aa7993e4d850d3abfc9b05959c
762d864591d3d5d450310e9ccb1ef0b2c339c.07b622242dcc6c7fd6a5f356d2e200c4f44be0279b767b85afcb24e778809d3c.lock
INFO:filelock:Lock 139908246331344 acquired on /root/.cache/huggingface/transformers/4b86306d6b8b22c548d737ae242
68401236ad7a42564ddb028bf193f485c55f2.cb2244924ab24d706b02fd7fcedaea4531566537687a539ebb94db511fd122a0.lock
Downloading:   0%|          | 0.00/772 [00:00<?, ?B/s]
INFO:filelock:Lock 139908246331344 released on /root/.cache/huggingface/transformers/4b86306d6b8b22c548d737ae242
68401236ad7a42564ddb028bf193f485c55f2.cb2244924ab24d706b02fd7fcedaea4531566537687a539ebb94db511fd122a0.lock
INFO:filelock:Lock 139908112777168 acquired on /root/.cache/huggingface/transformers/10b820db140011d86e29dad69ed
31c58db810e5f85a13982ec9457a63da1bb17.1788df22ba1a6817edb607a56efa931ee13ebad3b3500e58029a8f4e6d799a29.lock
Downloading:   0%|          | 0.00/62.0 [00:00<?, ?B/s]
INFO:filelock:Lock 139908112777168 released on /root/.cache/huggingface/transformers/10b820db140011d86e29dad69ed
31c58db810e5f85a13982ec9457a63da1bb17.1788df22ba1a6817edb607a56efa931ee13ebad3b3500e58029a8f4e6d799a29.lock
```

In [30]:
```
print(model.tokenizer)
```

```
PreTrainedTokenizer(name_or_path='seyonec/PubChem10M_SMILES_BPE_396_250', vocab_size=7924, model_max_len=1000000
000000000019884624838656, is_fast=False, padding_side='right', special_tokens={'bos_token': AddedToken("<s>", rs
trip=False, lstrip=False, single_word=False, normalized=True), 'eos_token': AddedToken("</s>", rstrip=False, lst
rip=False, single_word=False, normalized=True), 'unk_token': AddedToken("<unk>", rstrip=False, lstrip=False, sin
gle_word=False, normalized=True), 'sep_token': AddedToken("</s>", rstrip=False, lstrip=False, single_word=False,
normalized=True), 'pad_token': AddedToken("<pad>", rstrip=False, lstrip=False, single_word=False, normalized=Tru
e), 'cls_token': AddedToken("<s>", rstrip=False, lstrip=False, single_word=False, normalized=True), 'mask_token'
: AddedToken("<mask>", rstrip=False, lstrip=True, single_word=False, normalized=True)})
```

In [31]:
```
# check if our train and evaluation dataframes are setup properly. There should only be two columns for the SMI
print("Train Dataset: {}".format(train_df.shape))
print("Eval Dataset: {}".format(valid_df.shape))
print("TEST Dataset: {}".format(test_df.shape))
```

```
Train Dataset: (1182, 2)
Eval Dataset: (148, 2)
TEST Dataset: (148, 2)
```

Now that we've set everything up, lets get to the fun part: training the model! We use Weights and Biases, which is optional (simply remove `wandb_project` from the list of `args`). Its a really useful tool for monitering the model's

training results (such as accuracy, learning rate and loss), alongside custom visualizations of attention and gradients.

When you run this cell, Weights and Biases will ask for an account, which you can setup through a Github account, giving you an authorization API key which you can paste into the output of the cell. Again, this is completely optional and it can be removed from the list of arguments.

In [32]:
```
!wandb login
```

```
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter:
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
```

Finally, the moment we've been waiting for! Let's train the model on the train scaffold set of ClinTox, and monitor our runs using W&B. We will evaluate the performance of our model each epoch using the validation set.

In [33]:
```
# Create directory to store model weights (change path accordingly to where you want!)
!mkdir BPE_PubChem_10M_ClinTox_run

# Train the model
model.train_model(train_df, eval_df=valid_df, output_dir='/content/BPE_PubChem_10M_ClinTox_run', args={'wandb_p
```

```
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
  0%|          | 0/2 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_t
rain_roberta_128_2_1182
Epoch:   0%|          | 0/10 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_model: Initializing WandB run for training.
wandb: Currently logged in as: seyonec (use `wandb login --relogin` to force relogin)
```

Tracking run with wandb version 0.10.22

Syncing run **snowy-firefly-250** to Weights & Biases (Documentation).

Project page: https://wandb.ai/seyonec/project-name

Run page: https://wandb.ai/seyonec/project-name/runs/1t7dyfs4

Run data is saved locally in `/content/bert-loves-chemistry/wandb/run-20210318_145336-1t7dyfs4`

```
Running Epoch 0 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/nn/modules/module.py:760: UserWarning: Using non-full backward hook
s on a Module that does not return a single Tensor or a tuple of Tensors is deprecated and will be removed in fu
ture versions. This hook will be missing some of the grad_output. Please use register_full_backward_hook to get
the documented behavior.
  warnings.warn("Using non-full backward hooks on a Module that does not return a "
/usr/local/lib/python3.7/dist-packages/torch/nn/modules/module.py:795: UserWarning: Using a non-full backward ho
ok when the forward contains multiple autograd Nodes is deprecated and will be removed in future versions. This
hook will be missing some grad_input. Please use register_full_backward_hook to get the documented behavior.
  warnings.warn("Using a non-full backward hook when the forward contains multiple autograd Nodes "
Running Epoch 1 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 2 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 3 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 4 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 5 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 6 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 7 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 8 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
Running Epoch 9 of 10:   0%|          | 0/148 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_model: Training of roberta model complete. Saved to /conte
nt/BPE_PubChem_10M_ClinTox_run.
```

Out[33]:
```
(1480, 0.10153530814545406)
```

Let's install scikit-learn now, to evaluate the model we've trained. We will be using the accuracy and PRC-AUC metrics (average precision score).

In [34]:
```
import sklearn

# accuracy
result, model_outputs, wrong_predictions = model.eval_model(test_df, acc=sklearn.metrics.accuracy_score)

# ROC-PRC
result, model_outputs, wrong_predictions = model.eval_model(test_df, acc=sklearn.metrics.average_precision_scor
```

```
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_d
ev_roberta_128_2_148
Running Evaluation:   0%|          | 0/19 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_model: Initializing WandB run for evaluation.
Finishing last run (ID:1t7dyfs4) before initializing another...
```

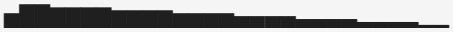
Waiting for W&B process to finish, PID 4627

Program ended successfully.
VBox(children=(Label(value=' 0.01MB of 0.01MB uploaded (0.00MB deduped)\r'), FloatProgress(value=1.0, max=1.0)…
Find user logs for this run at: /content/bert-loves-chemistry/wandb/run-20210318_145336-1t7dyfs4/logs/debug.log
Find internal logs for this run at: /content/bert-loves-chemistry/wandb/run-20210318_145336-1t7dyfs4/logs/debug-internal.log

Run summary:

Training loss	0.0003
lr	0.0
global_step	1450
_runtime	116
_timestamp	1616079332
_step	28

Run history:

Training loss	
lr	
global_step	
_runtime	
_timestamp	
_step	

Synced 5 W&B file(s), 1 media file(s), 0 artifact file(s) and 0 other file(s)

Synced **snowy-firefly-250**: https://wandb.ai/seyonec/project-name/runs/1t7dyfs4
...Successfully finished last run (ID:1t7dyfs4). Initializing new run:

Tracking run with wandb version 0.10.22
Syncing run **summer-pyramid-251** to Weights & Biases (Documentation).
Project page: https://wandb.ai/seyonec/project-name
Run page: https://wandb.ai/seyonec/project-name/runs/a6brkv9i
Run data is saved locally in /content/bert-loves-chemistry/wandb/run-20210318_145535-a6brkv9i

INFO:simpletransformers.classification.classification_model:{'mcc': 0.664470436990577, 'tp': 138, 'tn': 5, 'fp': 4, 'fn': 1, 'auroc': 0.8281374900079936, 'auprc': 0.9855371861072479, 'acc': 0.9662162162162162, 'eval_loss': 0.2469284737426757}
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_dev_roberta_128_2_148
Running Evaluation: 0%| | 0/19 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_model: Initializing WandB run for evaluation.
Finishing last run (ID:a6brkv9i) before initializing another...

Waiting for W&B process to finish, PID 4677
Program ended successfully.
VBox(children=(Label(value=' 0.00MB of 0.00MB uploaded (0.00MB deduped)\r'), FloatProgress(value=1.0, max=1.0)…
Find user logs for this run at: /content/bert-loves-chemistry/wandb/run-20210318_145535-a6brkv9i/logs/debug.log
Find internal logs for this run at: /content/bert-loves-chemistry/wandb/run-20210318_145535-a6brkv9i/logs/debug-internal.log

Run summary:

_runtime	3
_timestamp	1616079341
_step	2

Run history:

_runtime	▁▁▁▁
_timestamp	▁▁▁▁
_step	▁▅▅▅

Synced 5 W&B file(s), 3 media file(s), 0 artifact file(s) and 0 other file(s)

Synced **summer-pyramid-251**: https://wandb.ai/seyonec/project-name/runs/a6brkv9i
...Successfully finished last run (ID:a6brkv9i). Initializing new run:

Tracking run with wandb version 0.10.22
Syncing run **vivid-morning-252** to Weights & Biases (Documentation).
Project page: https://wandb.ai/seyonec/project-name
Run page: https://wandb.ai/seyonec/project-name/runs/7bl6wyef
Run data is saved locally in `/content/bert-loves-chemistry/wandb/run-20210318_145541-7bl6wyef`

```
INFO:simpletransformers.classification.classification_model:{'mcc': 0.664470436990577, 'tp': 138, 'tn': 5, 'fp': 4, 'fn': 1, 'auroc': 0.8281374900079936, 'auprc': 0.9855371861072479, 'acc': 0.9715961528455196, 'eval_loss': 0.2469284737426757}
```

The model performs pretty well, averaging above 97% ROC-PRC after training on only ~1400 data samples and 150 positive leads in a couple of minutes! We can clearly see the predictive power of transfer learning, and approaches like these are becoming increasing popular in the pharmaceutical industry where larger datasets are scarce. By training on more epochs and tasks, we can probably boost the accuracy as well!

Lets evaluate the model on one last string from ClinTox's test set for toxicity. The model should predict 1, meaning the drug failed clinical trials for toxicity reasons and wasn't approved by the FDA.

In [35]:
```
# Lets input a molecule with a toxicity value of 1
predictions, raw_outputs = model.predict(['C1=C(C(=O)NC(=O)N1)F'])
```

```
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_dev_roberta_128_2_1
  0%|          | 0/1 [00:00<?, ?it/s]
```

In [36]:
```
print(predictions)
print(raw_outputs)
```

```
[1]
[[-4.51171875  4.58203125]]
```

The model predicts the sample correctly! Some future tasks may include using the same model on multiple tasks (Tox21 provides multiple tasks relating to different biochemical pathways for toxicity, as an example), through multi-task classification, as well as training on a larger dataset such as HIV, one of the other harder tasks in molecular machine learning. This will be expanded on in future work!

Benchmarking Smiles-Tokenizer ChemBERTa models on ClinTox

Now lets compare how this model performs to a similar variant of ChemBERTa, that utilizes a different tokenizer, the SmilesTokenizer which is built-in to DeepChem! Let see if using a tokenizer which splits SMILES sequences into syntatically relevant chemical tokens performs differently, especially on molecular property prediction.

First off, lets initialize this variant model:

In [37]:
```
from simpletransformers.classification import ClassificationModel, ClassificationArgs
```

```
model = ClassificationModel('roberta', 'seyonec/SMILES_tokenized_PubChem_shard00_160k', args={'evaluate_each_ep
```

```
INFO:filelock:Lock 139908321724944 acquired on /root/.cache/huggingface/transformers/30ac96f427325ec13c51dfd4507
636207bdb9be77521b77ad334279cf1f5c184.f6ebc79ab803ca349ef7b469b0fbe6aa40d053e3c1c2da0501521c46c2a51bb7.lock
Downloading:   0%|          | 0.00/515 [00:00<?, ?B/s]
INFO:filelock:Lock 139908321724944 released on /root/.cache/huggingface/transformers/30ac96f427325ec13c51dfd4507
636207bdb9be77521b77ad334279cf1f5c184.f6ebc79ab803ca349ef7b469b0fbe6aa40d053e3c1c2da0501521c46c2a51bb7.lock
INFO:filelock:Lock 139908321724944 acquired on /root/.cache/huggingface/transformers/3a95725b53b9958c41159cd19bb
de8dad8e5988ff0a6971189ef3b6b625e5f5b.ae1cdbb61878f3444ee2c5aa28dfc4577a642a31729bf0b477ccd4d948ad9081.lock
Downloading:   0%|          | 0.00/336M [00:00<?, ?B/s]
INFO:filelock:Lock 139908321724944 released on /root/.cache/huggingface/transformers/3a95725b53b9958c41159cd19bb
de8dad8e5988ff0a6971189ef3b6b625e5f5b.ae1cdbb61878f3444ee2c5aa28dfc4577a642a31729bf0b477ccd4d948ad9081.lock
Some weights of the model checkpoint at seyonec/SMILES_tokenized_PubChem_shard00_160k were not used when initial
izing RobertaForSequenceClassification: ['lm_head.bias', 'lm_head.dense.weight', 'lm_head.dense.bias', 'lm_head.
layer_norm.weight', 'lm_head.layer_norm.bias', 'lm_head.decoder.weight', 'lm_head.decoder.bias']
- This IS expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model train
ed on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing RobertaForSequenceClassification from the checkpoint of a model t
hat you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenc
eClassification model).
Some weights of RobertaForSequenceClassification were not initialized from the model checkpoint at seyonec/SMILE
S_tokenized_PubChem_shard00_160k and are newly initialized: ['classifier.dense.weight', 'classifier.dense.bias',
'classifier.out_proj.weight', 'classifier.out_proj.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
INFO:filelock:Lock 139906415180368 acquired on /root/.cache/huggingface/transformers/b5b8a0f3afd321810f8ab3864fd
3b562ac78b45cffd986e1fe33b0dae85e4149.dcb6a95ce7ba1c00e125887fcabb2ed5074718e901096d78d86a6d720f57db60.lock
Downloading:   0%|          | 0.00/8.14k [00:00<?, ?B/s]
INFO:filelock:Lock 139906415180368 released on /root/.cache/huggingface/transformers/b5b8a0f3afd321810f8ab3864fd
3b562ac78b45cffd986e1fe33b0dae85e4149.dcb6a95ce7ba1c00e125887fcabb2ed5074718e901096d78d86a6d720f57db60.lock
INFO:filelock:Lock 139908023274512 acquired on /root/.cache/huggingface/transformers/e994bb60d8301b04451980e779a
f11e3fd55dfc1a97545f7ed9f25c4bb0144f8.0d2bc617dafe1551d37a1ee810476c86b8fcb92acede8e1ee6faf97e76000351.lock
Downloading: 0.00B [00:00, ?B/s]
INFO:filelock:Lock 139908023274512 released on /root/.cache/huggingface/transformers/e994bb60d8301b04451980e779a
f11e3fd55dfc1a97545f7ed9f25c4bb0144f8.0d2bc617dafe1551d37a1ee810476c86b8fcb92acede8e1ee6faf97e76000351.lock
INFO:filelock:Lock 139906414851088 acquired on /root/.cache/huggingface/transformers/186e51d9d044b8d234c30b286f5
8a87c44409db18948a3fd9b40fa795a4b89ad.dd8bd9bfd3664b530ea4e645105f557769387b3da9f79bdb55ed556bdd80611d.lock
Downloading:   0%|          | 0.00/112 [00:00<?, ?B/s]
INFO:filelock:Lock 139906414851088 released on /root/.cache/huggingface/transformers/186e51d9d044b8d234c30b286f5
8a87c44409db18948a3fd9b40fa795a4b89ad.dd8bd9bfd3664b530ea4e645105f557769387b3da9f79bdb55ed556bdd80611d.lock
INFO:filelock:Lock 139906414851152 acquired on /root/.cache/huggingface/transformers/c9175af31705aea512d539a7e6d
96803af809ba0d307eb762cb4b6a1c1af5ced.444225800184b0dbd3b86bfd798c4195c0af90f2b3b1540552cacd505c3f7c60.lock
Downloading:   0%|          | 0.00/327 [00:00<?, ?B/s]
INFO:filelock:Lock 139906414851152 released on /root/.cache/huggingface/transformers/c9175af31705aea512d539a7e6d
96803af809ba0d307eb762cb4b6a1c1af5ced.444225800184b0dbd3b86bfd798c4195c0af90f2b3b1540552cacd505c3f7c60.lock
Special tokens have been added in the vocabulary, make sure the associated word embedding are fine-tuned or trai
ned.
```

In [38]:
```
print(model.tokenizer)
```

```
PreTrainedTokenizer(name_or_path='seyonec/SMILES_tokenized_PubChem_shard00_160k', vocab_size=591, model_max_len=
514, is_fast=False, padding_side='right', special_tokens={'bos_token': AddedToken("<s>", rstrip=False, lstrip=Fa
lse, single_word=False, normalized=True), 'eos_token': AddedToken("</s>", rstrip=False, lstrip=False, single_wor
d=False, normalized=True), 'unk_token': '[UNK]', 'sep_token': '[SEP]', 'pad_token': '[PAD]', 'cls_token': '[CLS]
', 'mask_token': '[MASK]'})
```

In [39]:
```
# check if our train and evaluation dataframes are setup properly. There should only be two columns for the SMI
print("Train Dataset: {}".format(train_df.shape))
print("Eval Dataset: {}".format(valid_df.shape))
print("TEST Dataset: {}".format(test_df.shape))
```

```
Train Dataset: (1182, 2)
Eval Dataset: (148, 2)
TEST Dataset: (148, 2)
```

Now that we've set everything up, lets get to the fun part: training the model! We use Weights and Biases, which is optional (simply remove `wandb_project` from the list of `args`). Its a really useful tool for monitering the model's training results (such as accuracy, learning rate and loss), alongside custom visualizations of attention and gradients.

When you run this cell, Weights and Biases will ask for an account, which you can setup through a Github account, giving you an authorization API key which you can paste into the output of the cell. Again, this is completely optional and it can be removed from the list of arguments.

In [40]:
```
!wandb login
```

```
wandb: Currently logged in as: seyonec (use `wandb login --relogin` to force relogin)
```

In [41]:
```
# Create directory to store model weights (change path accordingly to where you want!)
!mkdir SmilesTokenizer_PubChem_10M_ClinTox_run
```

```
# Train the model
model.train_model(train_df, eval_df=valid_df, output_dir='/content/SmilesTokenizer_PubChem_10M_ClinTox_run', ar
```

```
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
  0%|          | 0/2 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_t
rain_roberta_128_2_1182
Epoch:   0%|          | 0/15 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_model: Initializing WandB run for training.
```

Finishing last run (ID:7bl6wyef) before initializing another...

Waiting for W&B process to finish, PID 4711

Program ended successfully.

```
VBox(children=(Label(value=' 0.00MB of 0.00MB uploaded (0.00MB deduped)\r'), FloatProgress(value=1.0, max=1.0)…
```

Find user logs for this run at: `/content/bert-loves-chemistry/wandb/run-20210318_145541-7bl6wyef/logs/debug.log`

Find internal logs for this run at: `/content/bert-loves-chemistry/wandb/run-20210318_145541-7bl6wyef/logs/debug-internal.log`

Run summary:

_runtime	3
_timestamp	1616079348
_step	2

Run history:

_runtime	▁▅▅▅

Out[41]: (2220, 0.09892498987772685)

Let's install scikit-learn now, to evaluate the model we've trained. We will be using the accuracy and PRC-AUC metrics (average precision score).

In [42]:
```
import sklearn

# accuracy
result, model_outputs, wrong_predictions = model.eval_model(test_df, acc=sklearn.metrics.accuracy_score)

# ROC-PRC
result, model_outputs, wrong_predictions = model.eval_model(test_df, acc=sklearn.metrics.average_precision_scor
```

```
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_d
ev_roberta_128_2_148
Running Evaluation:   0%|          | 0/19 [00:00<?, ?it/s]
INFO:simpletransformers.classification.classification_model: Initializing WandB run for evaluation.
```

Finishing last run (ID:v04qi4gi) before initializing another...

Waiting for W&B process to finish, PID 4767
Program ended successfully.

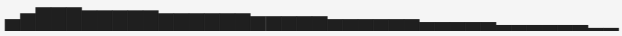
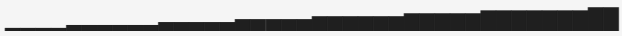
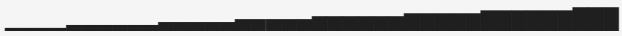
```
VBox(children=(Label(value=' 0.01MB of 0.01MB uploaded (0.00MB deduped)\r'), FloatProgress(value=1.0, max=1.0)…
```

Find user logs for this run at: `/content/bert-loves-chemistry/wandb/run-20210318_145608-v04qi4gi/logs/debug.log`
Find internal logs for this run at: `/content/bert-loves-chemistry/wandb/run-20210318_145608-v04qi4gi/logs/debug-internal.log`

Run summary:

Training loss	0.11875
lr	0.0
global_step	2200
_runtime	175
_timestamp	1616079546
_step	43

Run history:

Training loss	
lr	
global_step	
_runtime	
_timestamp	
_step	

Synced 5 W&B file(s), 1 media file(s), 0 artifact file(s) and 0 other file(s)

Synced **revived-armadillo-253**: https://wandb.ai/seyonec/project-name/runs/v04qi4gi
...Successfully finished last run (ID:v04qi4gi). Initializing new run:

Tracking run with wandb version 0.10.22
Syncing run **pleasant-wave-254** to Weights & Biases (Documentation).
Project page: https://wandb.ai/seyonec/project-name
Run page: https://wandb.ai/seyonec/project-name/runs/3ti3lfl8
Run data is saved locally in `/content/bert-loves-chemistry/wandb/run-20210318_145908-3ti3lfl8`

```
INFO:simpletransformers.classification.classification_model:{'mcc': 0.3646523331752495, 'tp': 138, 'tn': 2, 'fp'
: 7, 'fn': 1, 'auroc': 0.8073541167066347, 'auprc': 0.984400271563181, 'acc': 0.9459459459459459, 'eval_loss': 0
.3173560830033047}
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_d
ev_roberta_128_2_148
```

```
Running Evaluation:   0%|          | 0/19 [00:00<?, ?it/s]
```

```
INFO:simpletransformers.classification.classification_model: Initializing WandB run for evaluation.
```

Finishing last run (ID:3ti3lfl8) before initializing another...

Waiting for W&B process to finish, PID 4826
Program ended successfully.

```
VBox(children=(Label(value=' 0.00MB of 0.00MB uploaded (0.00MB deduped)\r'), FloatProgress(value=0.76956648239…
```

Find user logs for this run at: `/content/bert-loves-chemistry/wandb/run-20210318_145908-3ti3lfl8/logs/debug.log`
Find internal logs for this run at: `/content/bert-loves-chemistry/wandb/run-20210318_145908-3ti3lfl8/logs/debug-internal.log`

Run summary:

_runtime	3
_timestamp	1616079554
_step	2

Run history:

_runtime	▁▁▁▁
_timestamp	▁▁▁▁
_step	▁▄▄▄

Synced 5 W&B file(s), 3 media file(s), 0 artifact file(s) and 0 other file(s)

Synced **pleasant-wave-254**: https://wandb.ai/seyonec/project-name/runs/3ti3lfl8
...Successfully finished last run (ID:3ti3lfl8). Initializing new run:

Tracking run with wandb version 0.10.22
Syncing run **dulcet-shadow-255** to Weights & Biases (Documentation).
Project page: https://wandb.ai/seyonec/project-name
Run page: https://wandb.ai/seyonec/project-name/runs/17769dhr
Run data is saved locally in `/content/bert-loves-chemistry/wandb/run-20210318_145914-17769dhr`

```
INFO:simpletransformers.classification.classification_model:{'mcc': 0.3646523331752495, 'tp': 138, 'tn': 2, 'fp': 7, 'fn': 1, 'auroc': 0.8073541167066347, 'auprc': 0.984400271563181, 'acc': 0.951633958443683, 'eval_loss': 0.3173560830033047}
```

The model performs incredibly well, averaging above 96% PRC-AUC after training on only ~1400 data samples and 150 positive leads in a couple of minutes! This model was also trained on 1/10th the amount of pre-training data as the PubChem-10M BPE model we used previously, but it still showcases robust performance. We can clearly see the predictive power of transfer learning, and approaches like these are becoming increasing popular in the pharmaceutical industry where larger datasets are scarce. By training on more epochs and tasks, we can probably boost the accuracy as well!

Lets evaluate the model on one last string from ClinTox's test set for toxicity. The model should predict 1, meaning the drug failed clinical trials for toxicity reasons and wasn't approved by the FDA.

In [43]:
```
# Lets input a molecule with a toxicity value of 1
predictions, raw_outputs = model.predict(['C1=C(C(=O)NC(=O)N1)F'])
```

```
INFO:simpletransformers.classification.classification_utils: Converting to features started. Cache is not used.
INFO:simpletransformers.classification.classification_utils: Saving features into cached file cache_dir/cached_dev_roberta_128_2_1
  0%|          | 0/1 [00:00<?, ?it/s]
```

In [44]:
```
print(predictions)
print(raw_outputs)
```

```
[1]
[[-4.546875    4.83984375]]
```

The model predicts the sample correctly! Some future tasks may include using the same model on multiple tasks (Tox21 provides multiple tasks relating to different biochemical pathways for toxicity, as an example), through multi-task classification, as well as training on a larger dataset such as HIV, one of the other harder tasks in molecular machine

learning. This will be expanded on in future work!

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on Github

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Training a Normalizing Flow on QM9

By Nathan C. Frey | Twitter

In this tutorial, we will train a Normalizing Flow (NF) on the QM9 dataset. The dataset comprises 133,885 stable small organic molecules made up of CHNOF atoms. We will try to train a network that is an invertible transformation between a simple base distribution and the distribution of molecules in QM9. One of the key advantages of normalizing flows is that they can be constructed to efficiently sample from a distribution (generative modeling) and do probability density calculations (exactly compute log-likelihoods), whereas other models make tradeoffs between the two or can only approximate probability densities. This work has been published and considered as FastFlows see reference.

NFs are useful whenever we need a probabilistic model with one or both of these capabilities. Note that because NFs are completely invertible, there is no "latent space" in the sense used when referring to generative adversarial networks or variational autoencoders. For more on NFs, we refer to this review paper.

To encode the QM9 dataset, we'll make use of the SELFIES (SELF-referencIng Embedded Strings) representation, which is a 100% robust molecular string representation. SMILES strings produced by generative models are often syntactically invalid (they do not correspond to a molecular graph), or they violate chemical rules like the maximum number of bonds between atoms. SELFIES are designed so that even totally random SELFIES strings correspond to valid molecular graphs, so they are a great framework for generative modeling. For more details about SELFIES, see the GitHub repo and the associated paper.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5 minutes to run to completion and install your environment.

In [1]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

```
Requirement already satisfied: deepchem in /usr/local/lib/python3.7/dist-packages (2.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.3.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.0.2)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.21.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.4.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.0)
Requirement already satisfied: rdkit-pypi in /usr/local/lib/python3.7/dist-packages (from deepchem) (2021.9.5.1)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem) (2
018.9)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
pandas->deepchem) (1.15.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from rdkit-pypi->deepchem) (7.1
.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn
->deepchem) (3.1.0)
```

Out[1]:
```
'2.6.1'
```

Install the SELFIES library to translate SMILES strings.

In [2]:
```
!pip install selfies
```

```
Requirement already satisfied: selfies in /usr/local/lib/python3.7/dist-packages (2.0.0)
```

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import os

import deepchem as dc
from deepchem.models.normalizing_flows import NormalizingFlow, NormalizingFlowModel
```

```
from deepchem.models.optimizers import Adam
from deepchem.data import NumpyDataset
from deepchem.splits import RandomSplitter
from deepchem.molnet import load_tox21

import rdkit
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw

from IPython.display import Image, display

import selfies as sf

import tensorflow as tf
import tensorflow_probability as tfp

tfd = tfp.distributions
tfb = tfp.bijectors
tfk = tf.keras

tfk.backend.set_floatx('float64')
```

First, let's get a dataset of 2500 small organic molecules from the QM9 dataset. We'll then convert the molecules to SELFIES, one-hot encode them, and dequantize the inputs so they can be processed by a normalizing flow. 2000 molecules will be used for training, while the remaining 500 will be split into validation and test sets. We'll use the validation set to see how our architecture is doing at learning the underlying the distribution, and leave the test set alone. You should feel free to experiment with this notebook to get the best model you can and evaluate it on the test set when you're done!

In [4]:
```
# Download from MolNet
tasks, datasets, transformers = dc.molnet.load_qm9(featurizer='ECFP')
df = pd.DataFrame(data={'smiles': datasets[0].ids})
```

In [5]:
```
data = df[['smiles']].sample(2500, random_state=42)
```

SELFIES defines a dictionary called `bond_constraints` that enforces how many bonds every atom or ion can make. E.g., 'C': 4, 'H': 1, etc. The `?` symbol is used for any atom or ion that isn't defined in the dictionary, and it defaults to 8 bonds. Because QM9 contains ions and we don't want to allow those ions to form up to 8 bonds, we'll constrain them to 3. This will really improve the percentage of valid molecules we generate. You can read more about setting constraints in the SELFIES documentation.

In [6]:
```
sf.set_semantic_constraints()  # reset constraints
constraints = sf.get_semantic_constraints()
constraints['?'] = 3

sf.set_semantic_constraints(constraints)
constraints
```

Out[6]:
```
{'?': 3,
 'B': 3,
 'B+1': 2,
 'B-1': 4,
 'Br': 1,
 'C': 4,
 'C+1': 5,
 'C-1': 3,
 'Cl': 1,
 'F': 1,
 'H': 1,
 'I': 1,
 'N': 3,
 'N+1': 4,
 'N-1': 2,
 'O': 2,
 'O+1': 3,
 'O-1': 1,
 'P': 5,
 'P+1': 6,
 'P-1': 4,
 'S': 6,
 'S+1': 7,
 'S-1': 5}
```

In [7]:
```
def preprocess_smiles(smiles):
  return sf.encoder(smiles)

def keys_int(symbol_to_int):
  d={}
```

```
  i=0
  for key in symbol_to_int.keys():
    d[i]=key
    i+=1
  return d

data['selfies'] = data['smiles'].apply(preprocess_smiles)
```

Let's take a look at some short SMILES strings and their corresponding SELFIES representations. We can see right away that there is a key difference in how the two representations deal with Rings and Branches. SELFIES is designed so that branch length and ring size are stored locally with the `Branch` and `Ring` identifiers, and the SELFIES grammar prevents invalid strings.

In [8]:
```
data['len'] = data['smiles'].apply(lambda x: len(x))
data.sort_values(by='len').head()
```

Out[8]:

	smiles	selfies	len
6728	[H]c1nnc([H])o1	[H][C][=N][N][=C][Branch1][C][H][O][Ring1][=Br...	15
72803	[H]c1nnnc([H])n1	[H][C][=N][N][=N][C][Branch1][C][H][=N][Ring1]...	16
97670	[H]c1onnc1C(F)(F)F	[H][C][O][N][=N][C][=Ring1][Branch1][C][Branch...	18
25487	[H]n1nnc(C#CC#N)n1	[H][N][N][=N][C][Branch1][Branch1][C][#C][C][#...	18
32004	[H]C#Cc1nnc(F)nc1[H]	[H][C][#C][C][=N][N][=C][Branch1][C][F][N][=C]...	20

To convert SELFIES to a one-hot encoded representation, we need to construct an `alphabet` of all the characters that occur in the list of SELFIES strings. We also have to know what the longest SELFIES string is, so that all the shorter SELFIES can be padded with `'[nop]'` to be equal length.

In [9]:
```
selfies_list = np.asanyarray(data.selfies)
selfies_alphabet = sf.get_alphabet_from_selfies(selfies_list)
selfies_alphabet.add('[nop]')  # Add the "no operation" symbol as a padding character
selfies_alphabet.add('.')
selfies_alphabet = list(sorted(selfies_alphabet))
largest_selfie_len = max(sf.len_selfies(s) for s in selfies_list)
symbol_to_int = dict((c, i) for i, c in enumerate(selfies_alphabet))
int_mol=keys_int(symbol_to_int)
```

`selfies` has a handy utility function to translate SELFIES strings into one-hot encoded vectors.

In [10]:
```
onehots=sf.batch_selfies_to_flat_hot(selfies_list, symbol_to_int,largest_selfie_len)
```

Next, we "dequantize" the inputs by adding random noise from the interval `[0, 1)` to every input in the encodings. This allows the normalizing flow to operate on continuous inputs (rather than discrete), and the original inputs can easily be recovered by applying a floor function.

In [11]:
```
input_tensor = tf.convert_to_tensor(onehots, dtype='float64')
noise_tensor = tf.random.uniform(shape=input_tensor.shape, minval=0, maxval=1, dtype='float64')
dequantized_data = tf.add(input_tensor, noise_tensor)
```

The dequantized data is ready to be processed as a DeepChem dataset and split into training, validation, and test sets. We'll also keep track of the SMILES strings for the training set so we can compare the training data to our generated molecules later on.

In [12]:
```
ds = NumpyDataset(dequantized_data)  # Create a DeepChem dataset
splitter = RandomSplitter()
train, val, test = splitter.train_valid_test_split(dataset=ds, seed=42)
train_idx, val_idx, test_idx = splitter.split(dataset=ds, seed=42)

dim = len(train.X[0])  # length of one-hot encoded vectors
train.X.shape  # 2000 samples, N-dimensional one-hot vectors that represent molecules
```

Out[12]:
```
(2000, 2596)
```

In [13]:
```
# SMILES strings of training data
train_smiles = data['smiles'].iloc[train_idx].values
```

Next we'll set up the normalizing flow model. The base distribution is a multivariate Normal distribution. The `permutation` layer permutes the dimensions of the input so that the normalizing flow layers will operate along multiple dimensions of the inputs. To understand why the permutation is needed, we need to know a bit about how the normalizing flow architecture works.

In [14]:
```
base_dist = tfd.MultivariateNormalDiag(loc=np.zeros(dim), scale_diag=np.ones(dim))
```

```
if dim % 2 == 0:
    permutation = tf.cast(np.concatenate((np.arange(dim / 2, dim), np.arange(0, dim / 2))),
                                  tf.int32)
else:
    permutation = tf.cast(np.concatenate((np.arange(dim / 2 + 1, dim), np.arange(0, dim / 2))), tf.int32)
```

For this simple example, we'll set up a flow of repeating Masked Autoregressive Flow layers. The autoregressive property is enforced by using the Masked Autoencoder for Distribution Estimation architecture. The layers of the flow are a bijector, an invertible mapping between the base and target distributions.

MAF takes the inputs from the base distribution and transforms them with a simple scale-and-shift (affine) operation, but crucially the scale-and-shift for each dimension of the output *depends on the previously generated dimensions of the output*. That independence of future dimensions preserves the *autoregressive* property and ensures that the normalizing flow is invertible. Now we can see that we need permutations to change the ordering of the inputs, or else the normalizing flow would only transform certain dimensions of the inputs.

Batch Normalization layers can be added for additional stability in training, but may have strange effects on the outputs and require some input reshaping to work properly. Increasing `num_layers` and `hidden_units` can make more expressive flows capable of modeling more complex target distributions.

In [15]:
```
num_layers = 8
flow_layers = []

Made = tfb.AutoregressiveNetwork(params=2,
                                 hidden_units=[512, 512], activation='relu')

for i in range(num_layers):
    flow_layers.append(
        (tfb.MaskedAutoregressiveFlow(shift_and_log_scale_fn=Made)
    ))

    permutation = tf.cast(np.random.permutation(np.arange(0, dim)), tf.int32)

    flow_layers.append(tfb.Permute(permutation=permutation))

#     if (i + 1) % int(2) == 0:
#         flow_layers.append(tfb.BatchNormalization())
```

We can draw samples from the untrained distribution, but for now they don't have any relation to the QM9 dataset distribution.

In [16]:
```
%%time
nf = NormalizingFlow(base_distribution=base_dist,
                     flow_layers=flow_layers)
```

```
CPU times: user 280 ms, sys: 10.2 ms, total: 290 ms
Wall time: 289 ms
```

A `NormalizingFlowModel` takes a `NormalizingFlow` and any parameters used by `deepchem.models.KerasModel`.

In [17]:
```
nfm = NormalizingFlowModel(nf, learning_rate=1e-4, batch_size=128)
```

Now to train the model! We'll try to minimize the negative log likelihood loss, which measures the likelihood that generated samples are drawn from the target distribution, i.e. as we train the model, it should get better at modeling the target distribution and it will generate samples that look like molecules from the QM9 dataset.

In [18]:
```
losses = []
val_losses = []
```

In [19]:
```
%%time
max_epochs = 10 # maximum number of epochs of the training

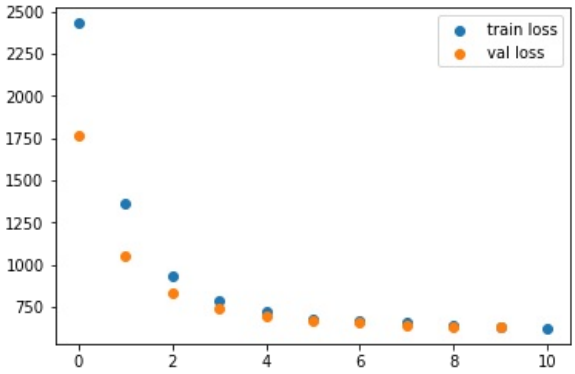
for epoch in range(max_epochs):
  loss = nfm.fit(train, nb_epoch=1, all_losses=losses)
  val_loss = nfm.create_nll(val.X)
  val_losses.append(val_loss.numpy())
```

```
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
```

e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shap
e=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), bu
t it was called on an input with incompatible shape (1, 128, 2596).

```
WARNING:tensorflow:Model was constructed with shape (None, 2596) for input KerasTensor(type_spec=TensorSpec(shape=(None, 2596), dtype=tf.float64, name='input_1'), name='input_1', description="created by layer 'input_1'"), but it was called on an input with incompatible shape (1, 128, 2596).
CPU times: user 13min 40s, sys: 20.9 s, total: 14min 1s
Wall time: 7min 27s
```

In [20]:
```
f, ax = plt.subplots()
ax.scatter(range(len(losses)), losses, label='train loss')
ax.scatter(range(len(val_losses)), val_losses, label='val loss')
plt.legend(loc='upper right');
```



The normalizing flow is learning a mapping between the multivariate Gaussian and the target distribution! We can see this by visualizing the loss on the validation set. We can now use `nfm.flow.sample()` to generate new QM9-like molecules and `nfm.flow.log_prob()` to evaluate the likelihood that a molecule was drawn from the underlying distribution.

In [21]:
```
generated_samples = nfm.flow.sample(10)  # generative modeling
log_probs = nfm.flow.log_prob(generated_samples)  # probability density estimation
```

Now we transform the generated samples back into SELFIES. We have to quantize the outputs and add padding characters to any one-hot encoding vector that has all zeros.

In [22]:
```
mols = tf.math.floor(generated_samples)  # quantize data
mols = tf.clip_by_value(mols, 0, 1)  # Set negative values to 0 and values > 1 to 1
mols_list = mols.numpy().tolist()

# Add padding characters if needed
for mol in mols_list:
  for i in range(largest_selfie_len):
    row = mol[len(selfies_alphabet) * i: len(selfies_alphabet) * (i + 1)]
    if all(elem == 0 for elem in row):
      mol[len(selfies_alphabet) * (i+1) - 1] = 1
```

`selfies` has another utility function to translate one-hot encoded representations back to SELFIES strings.

In [23]:
```
mols=sf.batch_flat_hot_to_selfies(mols_list, int_mol)
```

We can use RDKit to find valid generated molecules. Some have unphysical valencies and should be discarded. If you've ever tried to generate valid SMILES strings, you'll notice right away that this model is doing much better than we would expect! Using SELFIES, 90% of the generated molecules are valid, even though our normalizing flow architecture doesn't know any rules that govern chemical validity.

In [24]:
```
from rdkit import RDLogger
from rdkit import Chem
RDLogger.DisableLog('rdApp.*')  # suppress error messages

valid_count = 0
valid_selfies, invalid_selfies = [], []
for idx, selfies in enumerate(mols):
  try:
    if Chem.MolFromSmiles(sf.decoder(mols[idx]), sanitize=True) is not None:
        valid_count += 1
        valid_selfies.append(selfies)
    else:
      invalid_selfies.append(selfies)
  except Exception:
    pass
print('%.2f' % (valid_count / len(mols)),  '% of generated samples are valid molecules.')
```

```
1.00 % of generated samples are valid molecules.
```

Let's take a look at some of the generated molecules! We'll borrow some helper functions from the Modeling Solubility tutorial to display molecules with RDKit.

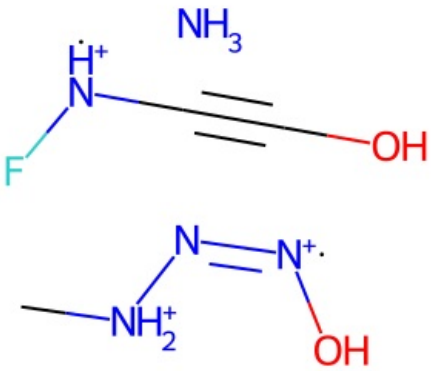
In [25]:
```
gen_mols = [Chem.MolFromSmiles(sf.decoder(vs)) for vs in valid_selfies]
```

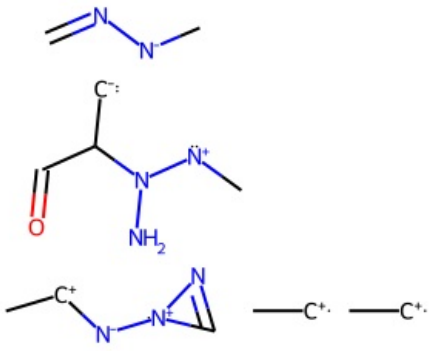
In [26]:
```
def display_images(filenames):
    """Helper to pretty-print images."""
    for file in filenames:
      display(Image(file))

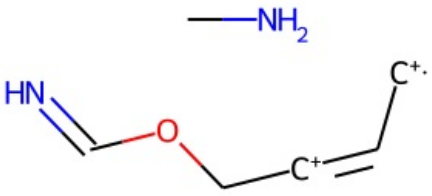
def mols_to_pngs(mols, basename="generated_mol"):
    """Helper to write RDKit mols to png files."""
    filenames = []
    for i, mol in enumerate(mols):
        filename = "%s%d.png" % (basename, i)
        Draw.MolToFile(mol, filename)
        filenames.append(filename)
    return filenames
```

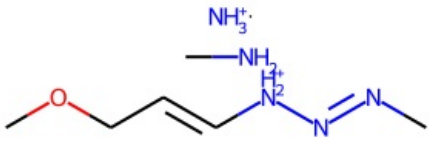
In [27]:
```
display_mols = []
for i in range(10):
  display_mols.append(gen_mols[i])

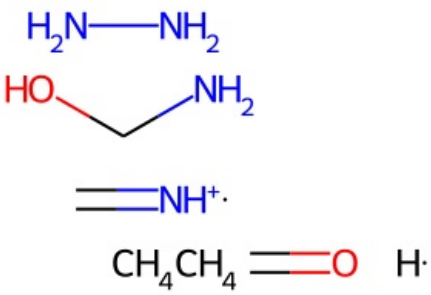
display_images(mols_to_pngs(display_mols))
```

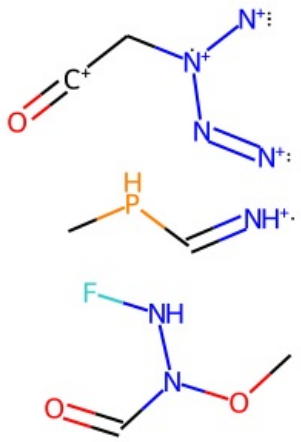


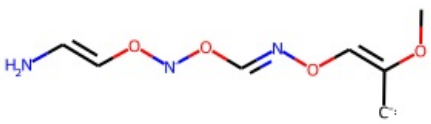


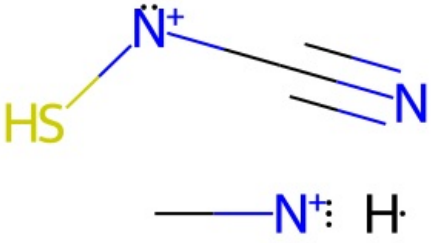


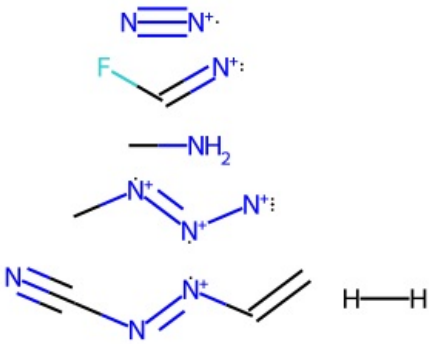














Finally, we can compare generated molecules with our training data via a similarity search with Tanimoto similarity. This gives an indication of how "original" the generated samples are, versus simply producing samples that are extremely similar to molecules the model has already seen. We have to keep in mind that QM9 contains *all* stable small molecules with up to 9 heavy atoms (CONF). So anything new we generate either exists in the full QM9 dataset, or else will not obey the charge neutrality and stability criteria used to generated QM9.

In [28]:
```
from rdkit.Chem.Fingerprints.FingerprintMols import FingerprintMol
from rdkit.DataStructs import FingerprintSimilarity
from IPython.display import display

def tanimoto_similarity(database_mols, query_mol):
    """Compare generated molecules to database by Tanimoto similarity."""
    # convert Mol to datastructure type
    fps = [FingerprintMol(m) for m in database_mols]

    # set a query molecule to compare against database
    query = FingerprintMol(query_mol)

    similarities = []

    # loop through to find Tanimoto similarity
    for idx, f in enumerate(fps):
        # tuple: (idx, similarity)
```

```
        similarities.append((idx, FingerprintSimilarity(query, f)))

    # sort sim using the similarities
    similarities.sort(key=lambda x:x[1], reverse=True)

    return similarities
```

We'll consider our generated molecules and look at the top 3 most similar molecules from the training data by Tanimoto similarity. Here's an example where the Tanimoto similarity scores are medium. There are molecules in our training set that are similar to our generated sample. This might be interesting, or it might mean that the generated molecule is unrealistic.

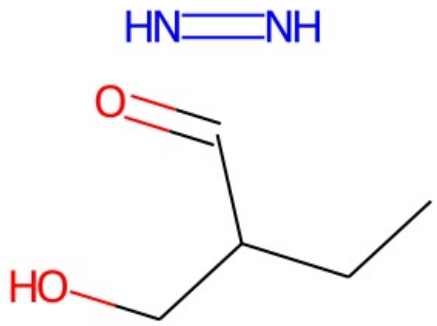
In [29]:
```
train_mols = [Chem.MolFromSmiles(smiles) for smiles in train_smiles]
```

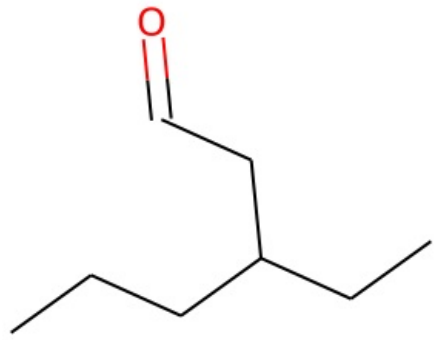
In [30]:
```
# change the second argument to compare different generated molecules to QM9
tanimoto_scores = tanimoto_similarity(train_mols, gen_mols[3])
similar_mols = []
```

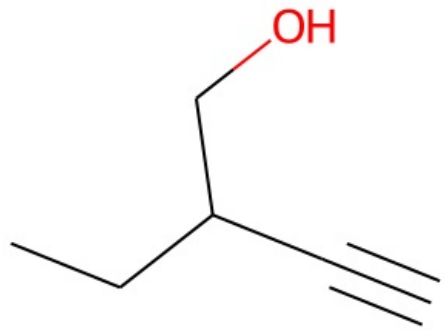
In [31]:
```
for idx, ts in tanimoto_scores[:3]:
    print(round(ts, 3))
    similar_mols.append(train_mols[idx])

display_images(mols_to_pngs(similar_mols, 'qm9_mol'))
```

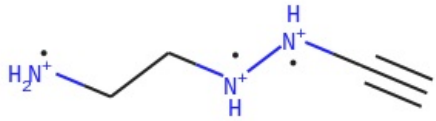
```
0.521
0.471
0.468
```

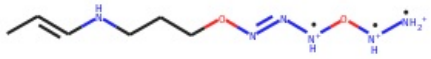


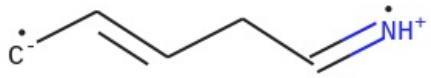




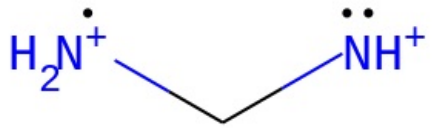
Molecules of the previous tutorial:

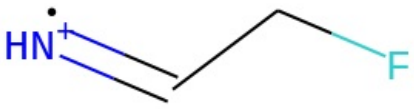


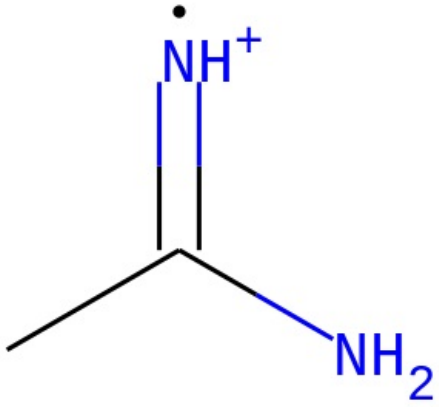


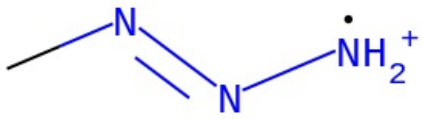


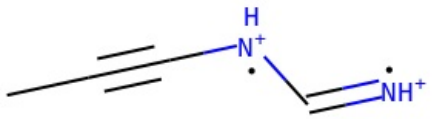


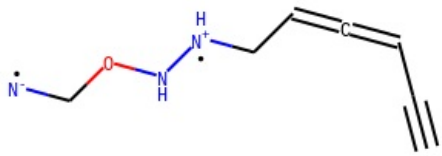

H2N+
NH+


HN+
F


NH+
NH2

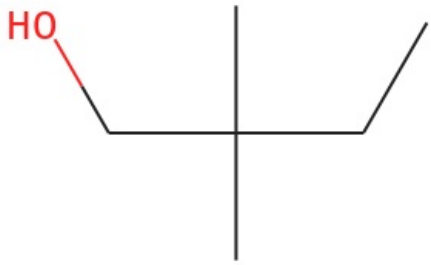

N
NH2+
N

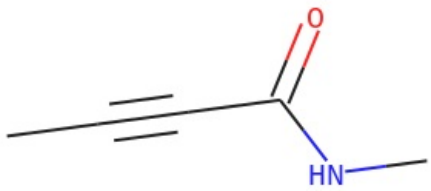


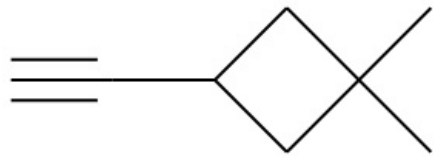


These molecules were obteined through sampling.

Comparing with the tanimoto similarity:







With scores of:

- 0.243
- 0.243
- 0.241

Further reading

So far we have looked at a measure of validity and done a bit of investigation into the novelty of the generated compounds. There are more dimensions along which we can and should evaluate the performance of a generative model. For an example of some standard benchmarks, see the GuacaMol evaluation framework.

For more information about FastFlows look at this paper where the workflow is crearly explained.

For examples of normalizing flow-based molecular graph generation frameworks, check out the MoFlow, GraphAF, and GraphNVP papers.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Screening Zinc For HIV Inhibition

In this tutorial we will walk through how to efficiently screen a large compound library with DeepChem (ZINC). Screening a large compound library using machine learning is a CPU bound pleasingly parrellel problem. The actual code examples I will use assume the resources available are a single very big machine (like an AWS c5.18xlarge), but should be readily swappable for other systmes (like a super computing cluster). At a high level what we will do is...

1. Create a Machine Learning Model Over Labeled Data
2. Transform ZINC into "Work-Units"
3. Create an inference script which runs predictions over a "Work-Unit"
4. Load "Work-Unit" into a "Work Queue"
5. Consume work units from "Work Queue"
6. Gather Results

This tutorial is unlike the previous tutorials in that it's designed to be run on AWS rather than on Google Colab. That's because we'll need access to a large machine with many cores to do this computation efficiently. We'll try to provide details about how to do this throughout the tutorial.

1. Train Model On Labelled Data

We are just going to knock out a simple model here. In a real world problem you will probably try several models and do a little hyper parameter searching.

In [1]:
```
from deepchem.molnet.load_function import hiv_datasets
```

```
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/sklearn/externals/joblib/__init__.py:15:
FutureWarning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.23. Please import this fu
nctionality directly from joblib, which can be installed with: pip install joblib. If this warning is raised whe
n loading pickled models, you may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=FutureWarning)
RDKit WARNING: [18:15:24] Enabling RDKit 2019.09.3 jupyter extensions
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:516
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:517
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:518
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:519
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:520
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/framework/dtypes.py:525
: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of numpy,
it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes
.py:541: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes
.py:542: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes
.py:543: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes
.py:544: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes
.py:545: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorboard/compat/tensorflow_stub/dtypes
.py:550: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
```

In [2]:

```
from deepchem.models import GraphConvModel
from deepchem.data import NumpyDataset
from sklearn.metrics import average_precision_score
import numpy as np

tasks, all_datasets, transformers = hiv_datasets.load_hiv(featurizer="GraphConv")
train, valid, test = [NumpyDataset.from_DiskDataset(x) for x in all_datasets]
model = GraphConvModel(1, mode="classification")
model.fit(train)
```

```
Loading raw samples now.
shard_size: 8192
About to start loading CSV from /var/folders/st/ds45jcqj2232lvhr0y9qt5sc0000gn/T/HIV.csv
Loading shard 1 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 0 took 12.479 s
Loading shard 2 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 1 took 13.668 s
Loading shard 3 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 2 took 13.550 s
Loading shard 4 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 3 took 13.173 s
Loading shard 5 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
RDKit WARNING: [18:16:53] WARNING: not removing hydrogen atom without neighbors
RDKit WARNING: [18:16:53] WARNING: not removing hydrogen atom without neighbors
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 4 took 13.362 s
Loading shard 6 of size 8192.
Featurizing sample 0
TIMING: featurizing shard 5 took 0.355 s
TIMING: dataset construction took 80.394 s
Loading dataset from disk.
TIMING: dataset construction took 16.676 s
Loading dataset from disk.
TIMING: dataset construction took 7.529 s
Loading dataset from disk.
TIMING: dataset construction took 7.796 s
Loading dataset from disk.
TIMING: dataset construction took 17.521 s
Loading dataset from disk.
TIMING: dataset construction took 7.770 s
Loading dataset from disk.
TIMING: dataset construction took 7.873 s
Loading dataset from disk.
TIMING: dataset construction took 15.495 s
Loading dataset from disk.
TIMING: dataset construction took 1.959 s
Loading dataset from disk.
TIMING: dataset construction took 1.949 s
Loading dataset from disk.
```

```
WARNING:tensorflow:From /Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python
/ops/init_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is depr
ecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the constructor
WARNING:tensorflow:Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c0
48>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing
the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
converting <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>>: Attribute
Error: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>> could
not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, s
et the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting
<bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>>: AttributeError: modu
le 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e
80>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing
the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
converting <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>>: Attribute
Error: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>> could
not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, s
et the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting
<bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>>: AttributeError: modu
le 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa
90>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing
the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
converting <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>>: Attribute
Error: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>> could
not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, s
et the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting
<bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>>: AttributeError: modu
le 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d1
98>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing
the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
converting <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>>: Attribute
Error: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>> could
not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, s
et the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting
<bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>>: AttributeError: modu
le 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43
f3a940>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When fi
ling the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Ca
use: converting <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>>:
AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>> c
ould not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the b
ug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: conve
rting <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>>: AttributeE
rror: module 'gast' has no attribute 'Num'
WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/layers.py:222: The name tf.unsorted_segment
_sum is deprecated. Please use tf.math.unsorted_segment_sum instead.

WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/layers.py:224: The name tf.unsorted_segment
_max is deprecated. Please use tf.math.unsorted_segment_max instead.

WARNING:tensorflow:Entity <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput ob
ject at 0x1a41a9ecf8>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph
team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the fu
ll output. Cause: converting <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput
object at 0x1a41a9ecf8>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput object at 0x
1a41a9ecf8>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. Whe
n filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output
. Cause: converting <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput object a
t 0x1a41a9ecf8>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/keras_model.py:169: The name tf.Session is
deprecated. Please use tf.compat.v1.Session instead.

WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/optimizers.py:76: The name tf.train.AdamOpt
imizer is deprecated. Please use tf.compat.v1.train.AdamOptimizer instead.

WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/keras_model.py:258: The name tf.global_vari
ables is deprecated. Please use tf.compat.v1.global_variables instead.
```

```
WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/keras_model.py:260: The name tf.variables_initializer is deprecated. Please use tf.compat.v1.variables_initializer instead.

WARNING:tensorflow:Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a3e35c048>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a41856e80>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphConv.call of <deepchem.models.layers.GraphConv object at 0x1a49f5aa90>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphPool.call of <deepchem.models.layers.GraphPool object at 0x1a43f5d198>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method GraphGather.call of <deepchem.models.layers.GraphGather object at 0x1a43f3a940>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:Entity <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput object at 0x1a41a9ecf8>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput object at 0x1a41a9ecf8>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING: Entity <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput object at 0x1a41a9ecf8>> could not be transformed and will be executed as-is. Please report this to the AutgoGraph team. When filing the bug, set the verbosity to 10 (on Linux, `export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause: converting <bound method TrimGraphOutput.call of <deepchem.models.graph_models.TrimGraphOutput object at 0x1a41a9ecf8>>: AttributeError: module 'gast' has no attribute 'Num'
WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/losses.py:108: The name tf.losses.softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.softmax_cross_entropy instead.

WARNING:tensorflow:From /Users/bharath/Code/deepchem/deepchem/models/losses.py:109: The name tf.losses.Reduction is deprecated. Please use tf.compat.v1.losses.Reduction instead.

WARNING:tensorflow:From /Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/ops/math_grad.py:318: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

```
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/ops/gradients_util.py:93: UserWarning: Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/ops/gradients_util.py:93: UserWarning: Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
/Users/bharath/opt/anaconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/ops/gradients_util.py:93: UserWarning: Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may consume a large amount of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

Out[2]: 0.0

In [3]:
```
y_true = np.squeeze(valid.y)
y_pred = model.predict(valid)[:,0,1]
print("Average Precision Score:%s" % average_precision_score(y_true, y_pred))
sorted_results = sorted(zip(y_pred, y_true), reverse=True)
hit_rate_100 = sum(x[1] for x in sorted_results[:100]) / 100
print("Hit Rate Top 100: %s" % hit_rate_100)
```

```
Average Precision Score:0.19783388433313015
Hit Rate Top 100: 0.37
```

Retrain Model Over Full Dataset For The Screen

In [29]:
```
tasks, all_datasets, transformers = hiv_datasets.load_hiv(featurizer="GraphConv", split=None)

model = GraphConvModel(1, mode="classification", model_dir="/tmp/zinc/screen_model")
model.fit(all_datasets[0])
```

```
Loading raw samples now.
shard_size: 8192
About to start loading CSV from /tmp/HIV.csv
Loading shard 1 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 0 took 15.701 s
Loading shard 2 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 1 took 15.869 s
Loading shard 3 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 2 took 19.106 s
Loading shard 4 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 3 took 16.267 s
Loading shard 5 of size 8192.
Featurizing sample 0
Featurizing sample 1000
Featurizing sample 2000
Featurizing sample 3000
Featurizing sample 4000
Featurizing sample 5000
Featurizing sample 6000
Featurizing sample 7000
Featurizing sample 8000
TIMING: featurizing shard 4 took 16.754 s
Loading shard 6 of size 8192.
Featurizing sample 0
TIMING: featurizing shard 5 took 0.446 s
TIMING: dataset construction took 98.214 s
Loading dataset from disk.
TIMING: dataset construction took 21.127 s
Loading dataset from disk.
/home/leswing/miniconda3/envs/deepchem/lib/python3.6/site-packages/tensorflow/python/ops/gradients_impl.py:100:
UserWarning: Converting sparse IndexedSlices to a dense Tensor of unknown shape. This may consume a large amount
of memory.
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

2. Create Work-Units

1. Download All of ZINC15.

Go to http://zinc15.docking.org/tranches/home and download all non-empty tranches in .smi format. I found it easiest to download the wget script and then run the wget script. For the rest of this tutorial I will assume zinc was downloaded to /tmp/zinc.

The way zinc downloads the data isn't great for inference. We want "Work-Units" which a single CPU can execute that takes a resonable amount of time (10 minutes to an hour). To accomplish this we are going to split the zinc data into

files each with 500 thousand lines.

```
mkdir /tmp/zinc/screen
find /tmp/zinc -name '*.smi' -exec cat {} \; | grep -iv "smiles" \
     | split -l 500000 /tmp/zinc/screen/segment
```

This bash command

1. Finds all smi files
2. prints to stdout the contents of the file
3. removes header lines
4. splits into multiple files in /tmp/zinc/screen that are 500k molecules long

3. Creat Inference Script

Now that we have work unit we need to construct a program which ingests a work unit and logs the result. It is important that the logging mechanism is thread safe! For this example we will get the work unit via a file-path, and log the result to a file. An easy extensions to distribute over multiple computers would be to get the work unit via a url, and log the results to a distributed queue.

Here is what mine looks like

inference.py

```
import sys
import deepchem as dc
import numpy as np
from rdkit import Chem
import pickle
import os


def create_dataset(fname, batch_size=50000):
    featurizer = dc.feat.ConvMolFeaturizer()
    fin = open(fname)
    mols, orig_lines = [], []
    for line in fin:
        line = line.strip().split()
        try:
            mol = Chem.MolFromSmiles(line[0])
            if mol is None:
                continue
            mols.append(mol)
            orig_lines.append(line)
        except:
            pass
        if len(mols) > 0 and len(mols) % batch_size == 0:
            features = featurizer.featurize(mols)
            y = np.ones(shape=(len(mols), 1))
            ds = dc.data.NumpyDataset(features, y)
            yield ds, orig_lines
            mols, orig_lines = [], []
    if len(mols) > 0:
        features = featurizer.featurize(mols)
        y = np.ones(shape=(len(mols), 1))
        ds = dc.data.NumpyDataset(features, y)
        yield ds, orig_lines


def evaluate(fname):
    fout_name = "%s_out.smi" % fname
    model = dc.models.TensorGraph.load_from_dir('screen_model')
    for ds, lines in create_dataset(fname):
        y_pred = np.squeeze(model.predict(ds), axis=1)
        with open(fout_name, 'a') as fout:
            for index, line in enumerate(lines):
                line.append(y_pred[index][1])
                line = [str(x) for x in line]
                line = "\t".join(line)
                fout.write("%s\n" % line)


if __name__ == "__main__":
    evaluate(sys.argv[1])
```

4. Load "Work-Unit" into a "Work Queue"

We are going to use a flat file as our distribution mechanism. It will be a bash script calling our inference script for every work unit. If you are at an academic institution this would be queing your jobs in pbs/qsub/slurm. An option for cloud computing would be rabbitmq or kafka.

In [ ]:
```
import os
work_units = os.listdir('/tmp/zinc/screen')
with open('/tmp/zinc/work_queue.sh', 'w') as fout:
    fout.write("#!/bin/bash\n")
    for work_unit in work_units:
        full_path = os.path.join('/tmp/zinc', work_unit)
        fout.write("python inference.py %s" % full_path)
```

5. Consume work units from "distribution mechanism"

We will consume work units from our work queue using a very simple Process Pool. It takes lines from our "Work Queue" and runs them, running as many processes in parrallel as we have cpus. If you are using a supercomputing cluster system like pbs/qsub/slurm it will take care of this for you. The key is to use one CPU per work unit to get highest throughput. We accomplish that here using the linux utility "taskset".

Using an c5.18xlarge on aws this will finish overnight.

process_pool.py

```
import multiprocessing
import sys
from multiprocessing.pool import Pool

import delegator


def run_command(args):
  q, command = args
  cpu_id = q.get()
  try:
    command = "taskset -c %s %s" % (cpu_id, command)
    print("running %s" % command)
    c = delegator.run(command)
    print(c.err)
    print(c.out)
  except Exception as e:
    print(e)
  q.put(cpu_id)


def main(n_processors, command_file):
  commands = [x.strip() for x in open(command_file).readlines()]
  commands = list(filter(lambda x: not x.startswith("#"), commands))
  q = multiprocessing.Manager().Queue()
  for i in range(n_processors):
    q.put(i)
  argslist = [(q, x) for x in commands]
  pool = Pool(processes=n_processors)
  pool.map(run_command, argslist)


if __name__ == "__main__":
  processors = multiprocessing.cpu_count()
  main(processors, sys.argv[1])
>> python process_pool.py /tmp/zinc/work_queue.sh
```

6. Gather Results

Since we logged our results to *_out.smi we now need to gather all of them up and sort them by our predictions. The resulting file wile be > 40GB. To analyze the data further you can use dask, or put the data in a rdkit postgres cartridge.

Here I show how to join the and sort the data to get the "best" results.

```
find /tmp/zinc -name '*_out.smi' -exec cat {} \; > /tmp/zinc/screen/results.smi
sort -rg -k 3,3 /tmp/zinc/screen/results.smi > /tmp/zinc/screen/sorted_results.smi
# Put the top 100k scoring molecules in their own file
head -n 50000 /tmp/zinc/screen/sorted_results.smi > /tmp/zinc/screen/top_100k.smi
```

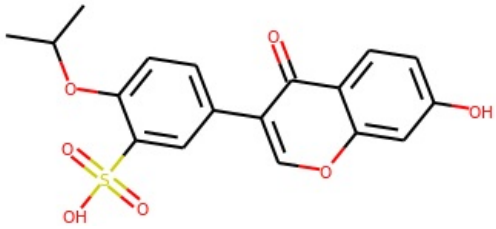
/tmp/zinc/screen/top_100k.smi is now a small enough file to investigate using standard tools like pandas.

In [9]:
```
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from IPython.display import SVG
from rdkit.Chem.Draw import rdMolDraw2D
best_mols = [Chem.MolFromSmiles(x.strip().split()[0]) for x in open('/tmp/zinc/screen/top_100k.smi').readlines(
best_scores = [x.strip().split()[2] for x in open('/tmp/zinc/screen/top_100k.smi').readlines()[:100]]
```

In [10]:
```
print(best_scores[0])
best_mols[0]
```

```
0.98874843
```

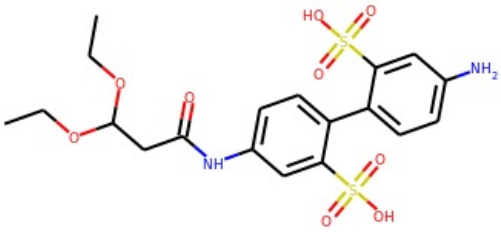
Out[10]:



In [11]:
```
print(best_scores[0])
best_mols[1]
```

```
0.98874843
```

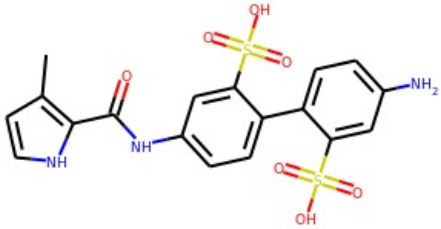
Out[11]:



In [12]:
```
print(best_scores[0])
best_mols[2]
```

```
0.98874843
```

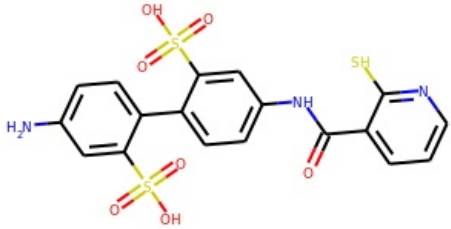
Out[12]:



In [13]:
```
print(best_scores[0])
best_mols[3]
```

```
0.98874843
```

Out[13]:



The screen seems to favor molecules with one or multiple sulfur trioxides. The top scoring molecules also have low diversity. When creating a "buy list" we want to optimize for more things than just activity, for instance diversity and drug like MPO.

In [ ]:
```
#We use the code from https://github.com/PatWalters/rd_filters, detailed explanation is here: http://practicalc
#We will run the PAINS filter on best_mols as suggested by Issue 1355 (https://github.com/deepchem/deepchem/issu
import os

import pandas as pd
from rdkit import Chem
from rdkit.Chem.Descriptors import MolWt, MolLogP, NumHDonors, NumHAcceptors, TPSA
from rdkit.Chem.rdMolDescriptors import CalcNumRotatableBonds

#First we get the rules from alert_collection.csv and then filter to get PAINS filter
rule_df = pd.read_csv(os.path.join(os.path.abspath(''), 'assets', 'alert_collection.csv'))
rule_df = rule_df[rule_df['rule_set_name']=='PAINS']
rule_list = []
for rule_id, smarts, max_val, desc in rule_df[["rule_id", "smarts", "max", "description"]].values.tolist():
```

```
    smarts_mol = Chem.MolFromSmarts(smarts)
    if smarts_mol:
        rule_list.append((smarts_mol, max_val, desc))

def evaluate(smile):
    mol = Chem.MolFromSmiles(smile)
    if mol is None:
        return [smile, "INVALID", -999, -999, -999, -999, -999, -999]
    desc_list = [MolWt(mol), MolLogP(mol), NumHDonors(mol), NumHAcceptors(mol), TPSA(mol), CalcNumRotatableBond
    for patt, max_val, desc in rule_list:
        if len(mol.GetSubstructMatches(patt)) > max_val:
            return [smiles, desc + " > %d" % (max_val)] +desc_list
    return [smiles, "OK"]+desc_list

smiles = [x.strip().split()[0] for x in open('/tmp/zinc/screen/top_100k.smi').readlines()[:100]] # obtain the s
res = list(map(evaluate, smiles)) # here we apply the PAINS filter

df = pd.DataFrame(res, columns=["SMILES", "FILTER", "MW", "LogP", "HBD", "HBA", "TPSA", "Rot"])
df_ok = df[
    (df.FILTER == "OK") &
    df.MW.between(*[0, 500]) & # MW
    df.LogP.between(*[-5, 5]) & #LogP
    df.HBD.between(*[0, 5]) & #HBD
    df.HBA.between(*[0, 10]) & #HBA
    df.TPSA.between(*[0, 200]) & #TPSA
    df.Rot.between(*[0, 10]) #Rot
    ]
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Introduction to the Molecular Attention Transformer.

In this tutorial we will learn more about the Molecular Attention Transformer, or MAT. MAT is a model based on transformers, aimed towards performing molecular prediction tasks. MAT is easy to tune and performs quite well relative to other molecular prediction tasks. The weights from MAT are chemically interpretable, thus making the model quite useful.

Reference Paper: Molecular Attention Transformer, Maziarka et. al.

In this tutorial, we will explore how to train MAT, and predict hydration enthalpy values for molecules from the freesolv hydration enthalpy dataset with MAT.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
```

Import required modules

In [1]:
```
import deepchem as dc
from deepchem.models.torch_models import MATModel
from deepchem.feat import MATFeaturizer
import matplotlib.pyplot as plt
```

```
wandb: WARNING W&B installed but not logged in.  Run `wandb login` or set the WANDB_API_KEY env variable.
wandb: WARNING W&B installed but not logged in.  Run `wandb login` or set the WANDB_API_KEY env variable.
```

Molecule Featurization using MATFeaturizer

MATFeaturizer is the featurizer intended to be used with the Molecular Attention Transformer, or MAT in short. MATFeaturizer takes a smile string or molecule as input and returns a MATEncoding dataclass object, which contains 3 numpy arrays: Node features matrix, adjacency matrix and distance matrix.

In [2]:
```
featurizer = dc.feat.MATFeaturizer()
# Let us now take an example array of smile strings and featurize it.
smile_string = ["CCC"]
output = featurizer.featurize(smile_string)
print(type(output[0]))
print(output[0].node_features)
print(output[0].adjacency_matrix)
print(output[0].distance_matrix)
```

```
<class 'deepchem.feat.molecule_featurizers.mat_featurizer.MATEncoding'>
[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
  0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
  0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0.
  0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]]
[[0. 0. 0. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 1.]
 [0. 1. 1. 0.]]
[[1.e+06 1.e+06 1.e+06 1.e+06]
 [1.e+06 0.e+00 2.e+00 1.e+00]
 [1.e+06 2.e+00 0.e+00 1.e+00]
 [1.e+06 1.e+00 1.e+00 0.e+00]]
```

Getting the Freesolv Hydration enthalpy dataset

We will now acquire the Freesolv Hydration enthalpy dataset from MoleculeNet. If it already exists in the directory, the file will be used. Else, deepchem will automatically download the dataset from its AWS bucket.

In [3]:
```
tasks, dataset, transformers = dc.molnet.load_freesolv()
```

```
train_dataset, val_dataset, test_dataset = dataset
train_smiles = train_dataset.ids
val_smiles = val_dataset.ids
```

In [4]:
```
train_dataset
```

Out[4]:
```
<DiskDataset X.shape: (513,), y.shape: (513, 1), w.shape: (513, 1), ids: ['CCCCNCCCC' 'CCOC=O' 'CCCCCCCCC' ...
'COC' 'CCCCCCCCBr'
 'CCCc1ccc(c(c1)OC)O'], task_names: ['y']>
```

Training the model

Now that we have acquired the dataset to be used and made the necessary imports, we will be invoking the Molecular Attention Transformer in deepchem, called MATModel, and we will be training it. We will be using default parameters for the purposes of this tutorial, however they can be changed anytime according to the user's preferences.

In [5]:
```
device = 'cpu'
model = MATModel(device = device)
```

In [6]:
```
losses, val_losses = [], []
```

In [7]:
```
%%time
max_epochs = 10

for epoch in range(max_epochs):
    loss = model.fit(train_dataset, nb_epoch = 1, max_checkpoints_to_keep = 1, all_losses = losses)
    metric = dc.metrics.Metric(dc.metrics.score_function.rms_score)
    val_losses.append(model.evaluate(val_dataset, metrics = [metric])['rms_score']**2)

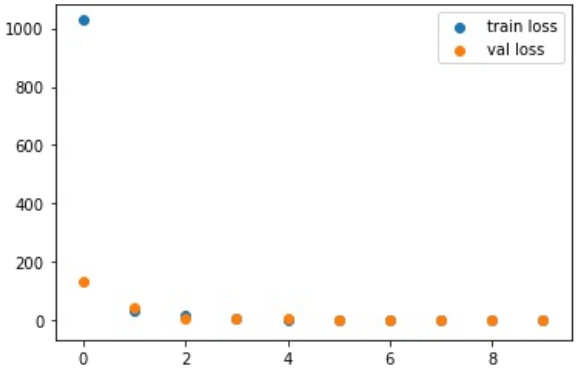
# The warnings are not relevant to this tutorial thus we can safely skip them.
```

```
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
```

```
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
```

```
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
CPU times: user 20min 48s, sys: 10.7 s, total: 20min 58s
Wall time: 2min 41s
```

In [8]:
```
f, ax = plt.subplots()
ax.scatter(range(len(losses)), losses, label='train loss')
ax.scatter(range(len(val_losses)), val_losses, label='val loss')
plt.legend(loc='upper right');
```



Testing the model

Optimally, MAT should be trained for a lot more epochs with a GPU. Due to computational constraints, we train this model for very few epochs in this tutorial. Let us now see how to predict the hydration enthalpy values for molecues now with MAT.

In [9]:
```
# We will be predicting the enthalpy value for the smile string we featurized earlier in the MATFeaturizer sect
model.predict_on_batch(output)
```

```
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:165: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  node_features = torch.tensor(data[0]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:166: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  adjacency_matrix = torch.tensor(data[1]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/mat.py:167: UserWarning: To copy construct from a
tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(T
rue), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(data[2]).float()
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:171: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  torch.sum(torch.tensor(adj_matrix), dim=-1).unsqueeze(2) + eps)
/home/atreyamaj/Desktop/deepchem/deepchem/models/torch_models/layers.py:178: UserWarning: To copy construct from
a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_
(True), rather than torch.tensor(sourceTensor).
  distance_matrix = torch.tensor(distance_matrix).squeeze().masked_fill(
```

Out[9]:
```
array([[-0.32668447]], dtype=float32)
```

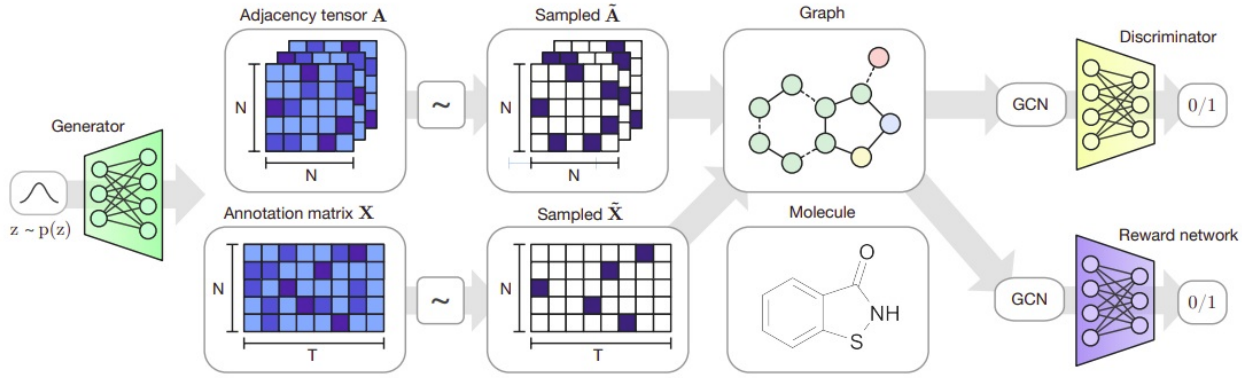
Generating Molecules with MolGAN

In this tutorial, we will train a MolGAN network on the tox21 dataset which is a dataset of 12,060 training samples and 647 test samples of chemical compounds. The MolGAN network was first introduced in "MolGAN: An implicit generative model for small molecular graphs" by Cao and Kipf. It uses a GAN directly on graph data and a reinforcement learning objective to induce the network to generate molecules with certain chemical properties.

The architecture consits of 3 main sections: a generator, a discriminator, and a reward network.

The generator takes a sample (z) from a standard normal distribution to generate an a graph using a MLP (this limits the network to a fixed maximum size) to generate the graph at once. Sepcifically a dense adjacency tensor A (bond types) and an annotation matrix X (atom types) are produced. Since these are probabilities, a discrete, sparse x and a are generated through categorical sampling.

The discriminator and reward network have the same architectures and recieve graphs as inputs. A Relational-GCN and MLPs are used to produce the singular output.



Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

Open in Colab

Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands.

In [1]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

```
Requirement already satisfied: deepchem in /usr/local/lib/python3.10/dist-packages (2.8.1.dev20240603202041)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.4.2)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from deepchem) (2.0.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.2.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.12.1)
Requirement already satisfied: scipy>=1.10.1 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.11.4)
Requirement already satisfied: rdkit in /usr/local/lib/python3.10/dist-packages (from deepchem) (2023.9.6)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem) (
2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem)
(2024.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from rdkit->deepchem) (9.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n->deepchem) (3.5.0)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->deep
chem) (1.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->deepchem) (1.16.0)
```

```
WARNING:deepchem.feat.molecule_featurizers.rdkit_descriptors:No normalization for SPS. Feature removed!
WARNING:deepchem.feat.molecule_featurizers.rdkit_descriptors:No normalization for AvgIpc. Feature removed!
WARNING:tensorflow:From /usr/local/lib/python3.10/dist-packages/tensorflow/python/util/deprecation.py:588: calli
ng function (from tensorflow.python.eager.polymorphic_function.polymorphic_function) with experimental_relax_sha
pes is deprecated and will be removed in a future version.
Instructions for updating:
experimental_relax_shapes is deprecated, use reduce_retracing instead
WARNING:deepchem.models.torch_models:Skipped loading modules with pytorch-geometric dependency, missing a depend
ency. No module named 'torch_geometric'
WARNING:deepchem.models:Skipped loading modules with pytorch-geometric dependency, missing a dependency. cannot
import name 'DMPNN' from 'deepchem.models.torch_models' (/usr/local/lib/python3.10/dist-packages/deepchem/models
/torch_models/__init__.py)
WARNING:deepchem.models:Skipped loading modules with pytorch-lightning dependency, missing a dependency. No modu
le named 'lightning'
WARNING:deepchem.models:Skipped loading some Jax models, missing a dependency. No module named 'haiku'
```

Out[1]: '2.8.1.dev'

Import the packages you'll need.

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os
from collections import OrderedDict

import deepchem as dc
import deepchem.models
import torch
from deepchem.models.torch_models import BasicMolGANModel as MolGAN
from deepchem.models.optimizers import ExponentialDecay
from torch.nn.functional import one_hot
from rdkit import Chem
from rdkit.Chem.Draw import IPythonConsole
from rdkit.Chem import Draw

from deepchem.feat.molecule_featurizers.molgan_featurizer import GraphMatrix
```

Download, load, and extract the SMILES strings from the tox21 dataset. The original paper used the QM9 dataset, however we use the tox21 dataset here to save time.

In [3]:
```
# Download from MolNet
# Try tox21 or LIPO dataset
tasks, datasets, transformers = dc.molnet.load_tox21()
df = pd.DataFrame(data={'smiles': datasets[0].ids})
```

Specify the maximum number of atoms to enocde for the featurizer and the MolGAN network. The higher the number of atoms, the more data you'll have in the dataset. However, this also increases the model complexity as the input dimensions become higher.

In [4]:
```
num_atoms = 12
```

In [5]:
```
df
```

Out[5]:

	smiles
0	CC(O)(P(=O)(O)O)P(=O)(O)O
1	CC(C)(C)OOC(C)(C)CCC(C)(C)OOC(C)(C)C
2	OC[C@H](O)[C@@H](O)[C@H](O)CO
3	CCCCCCCC(=O)[O-].CCCCCCCC(=O)[O-].[Zn+2]
4	CC(C)COC(=O)C(C)C
...	...
6259	CC1CCCCN1CCCOC(=O)c1ccc(OC2CCCCC2)cc1
6260	Cc1cc(CCCOc2c(C)cc(-c3noc(C(F)(F)F)n3)cc2C)on1
6261	O=C1OC(OC(=O)c2cccnc2Nc2cccc(C(F)(F)F)c2)c2ccc...
6262	CC(=O)C1(C)CC2=C(CCCC2(C)C)CC1C
6263	CC(C)CCC[C@@H](C)[C@H]1CC(=O)C2=C3CC[C@H]4C[C@...

6264 rows × 1 columns

Uncomment the first line if you want to subsample from the full dataset.

In [6]:
```
#data = df[['smiles']].sample(4000, random_state=42)
data = df
```

Initialize the featurizer with the maxmimum number of atoms per molecule. atom_labels is a parameter to pass the atomic number of atoms you want to be able to parse. Similar to the num_atoms parameter above, more atom_labels means more data, though the model gets more complex/unstable.

In [7]:
```
# create featurizer
feat = dc.feat.MolGanFeaturizer(max_atom_count=num_atoms, atom_labels=[0, 5, 6, 7, 8, 9, 11, 12, 13, 14]) #15, .
```

Extract the smiles from the dataframe as a list of strings

In [8]:
```
smiles = data['smiles'].values
```

Filter out the molecules with too many atoms to reduce the number of unnecessary error messages in later steps.

In [9]:
```
filtered_smiles = [x for x in smiles if Chem.MolFromSmiles(x).GetNumAtoms() < num_atoms]
```

```
[13:29:08] WARNING: not removing hydrogen atom without neighbors
```

The next cell featurizes the filtered molecules, however, since we have limited the atomic numbers to `[5, 6, 7, 8, 9, 11, 12, 13, 14]` which is B, C, N, O, F, Na, Mg, Al and Si, the featurizer fails to featurize several molecules in the dataset. Feel free to experiment with more atomic numbers!

In [10]:
```
# featurize molecules
features = feat.featurize(filtered_smiles)
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 0, CC(O)(P(=O)(O)O)P(=O)(O)O. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 11, O=[N+]([O-])[O-].O=[N+]([O-])[O-].[Ca+2]. A
ppending empty array
WARNING:deepchem.feat.base_classes:Exception message: 20
[13:29:09] WARNING: not removing hydrogen atom without neighbors
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 12, F[B-](F)(F)F.[H+]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 1
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 14, [I-].[K+]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 19
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 17, C=CC(=O)OCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 25, ClCOCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 27, [Cu]I. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 29
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 32, BrCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 33, CCC(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 39, NC(=S)NNC(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 40, NC(=S)C(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 41, C[Hg]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 42, [Hg+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 80
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 44, O=[Cr](=O)([O-])O[Cr](=O)(=O)[O-]. Appendin
g empty array
WARNING:deepchem.feat.base_classes:Exception message: 24
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 45, O=P(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 49, O=C(O)[C@@H](S)[C@H](S)C(=O)O. Appending em
pty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 54, CO[Si](CCCS)(OC)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 55, C[N+](C)=CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 57, O=[Zr](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 61, CC(C)OS(C)(=O)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 66, CC(Cl)(Cl)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 67, O=C(CCl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 68, CC(=O)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 71, ClCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 75, CCOP(=O)(CC)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 81, CSCCC=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 82, CC(C)(C)C(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 87, CNC(=O)NC(O)C(Cl)(Cl)Cl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 90, BrCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 91, O=S(=O)([O-])[O-].[Na+].[Na+]. Appending em
pty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 98, [O-][Cl+3]([O-])([O-])[O-]. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 106, OCCSSCCO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 107, ClCCCCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 108, [Ba+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 56
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 116, N#CSCSC#N. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 117, CCCCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 118, C[S+](C)CCC(N)C(=O)O. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 120, CC[Sn](Br)(CC)CC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 122, CCOP(=O)(OCC)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 133, CCOP(=O)([O-])C(N)=O. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 136, COC(=O)CCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 141, CCCCCCCCC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 142, COC(=O)C(Cl)C(=O)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 146, NCCOS(=O)(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 147, COP(=O)(OC)OC=C(Cl)Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 148, CC(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 151, Cl[Pd]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 152, ClC=CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 153, O=S(=O)([O-])CCCS(=O)(=O)[O-]. Appending e
mpty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 154, [TlH2+]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 81
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 155, CC(C)OP(C)(=O)OC(C)C. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 156, COS(=O)(=O)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 158, O=P([O-])([O-])OC(CO)CO. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 160, C=C(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 161, OC(CCl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 162, Cl[Ba]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 163, O=C(O)CP(=O)(O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 166, ClC(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 167, ClC(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 168, CC(C)(CCl)C(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 175, ClC(Cl)=C(Cl)C(Cl)=C(Cl)Cl. Appending empt
y array
```

```
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 180, ClC(Cl)(Cl)C(Cl)(Cl)Cl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 183, [Na]I. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 185, BrCCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 187, CCNC(=S)NCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 190, CC(C)(C)S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 193, Cl/C=C/Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 194, Cl/C=C\Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 195, OCC(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 200, O=[Mo](=O)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 42
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 201, CCOP(=S)(S)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 203, O=[Cr]O[Cr]=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 24
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 204, [Cr+3]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 24
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 210, CCN(CC)C(=S)[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 211, C[Sn](C)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 212, CCCCC(CC)C(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 214, OCC(O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 217, O=S(=O)(O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 218, [Fe+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 26
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 223, N#C[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 224, CC(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 228, ClC(Cl)(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 230, CSCCCN=C=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 238, CCCC[Sn](Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 50
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 241, C=CCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 242, C=CCSSCCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 243, C=CCNC(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 245, Cl[Nd](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 246, [Co+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 27
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 250, CC(C)C(Br)C(=O)NC(N)=O. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 251, O=S(=O)([O-])CO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 253, CC(=O)C(Cl)C(=O)N(C)C. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 259, O=C/C(Cl)=C(/Cl)C(=O)O. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 260, O=C(O)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 263, C=CCSSCC=C. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 267, Cl[Yb](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 271, NS(=O)(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 272, [PbH2+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 82
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 276, OCC(S)CS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 279, CCCCNP(N)(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 281, N[C@@H](CS)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 282, CC(C)(O)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 284, NC(CS)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 290, ClCCN(CCCl)CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 291, O=C(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 293, [Fe+3]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 26
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 294, CCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 295, ClC(Cl)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 305, CC(=O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 310, [Cu+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 29
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 311, ClCC(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 316, Cl[In](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 322, [As]#[In]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 33
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 323, COS(=O)(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 332, O=S(=O)([O-])[O-].[Li+].[Li+]. Appending e
mpty array
WARNING:deepchem.feat.base_classes:Exception message: 3
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 333, CCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 334, CS(=O)(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 339, S=C([S-])NCCNC(=S)[S-]. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 341, C[N+]([O-])(CCCl)CCCl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 350, CCC(C)S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 355, O=C(O)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 356, FC(F)(Cl)C(F)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 357, BrC(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 360, COP(=O)(NC(C)=O)SC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 362, ClP(Cl)(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 370, OCC(CBr)(CBr)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 371, O=C(O)C(Cl)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 372, O=C(O)C(Cl)(Cl)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 374, OCCCCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 384, CSSC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 387, CC(Cl)(Cl)[N+](=O)[O-]. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 390, O=[N+]([O-])C(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 395, CNC(=S)NN. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 397, O=S(=O)(O)C(F)(F)F. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 400, CCSCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 401, CCCC[Sn](Cl)(Cl)CCCC. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 50
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 403, [C-]#N.[Cu+]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 29
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 404, CC(C)(C)C(=O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 405, CC(S)C(=O)NCC(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 406, [Cd+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 48
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 409, CC(CC(=O)Cl)CC(C)(C)C. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 411, CCCCCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 413, C[N+](C)(C)CC(O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 417, CNC(=S)[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 418, [SnH2+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 50
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 420, COC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 422, C=CC(Cl)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 423, O=[Se]([O-])[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 34
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 424, CCOC(=S)[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 429, Cl[Au](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 436, N#CCCC(Br)(C#N)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 440, C=CC(=O)[O-].C=CC(=O)[O-].[Zn+2]. Appendin
g empty array
WARNING:deepchem.feat.base_classes:Exception message: 30
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 441, CCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 442, OCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 444, CCCCCCI. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 445, CCOC(=O)CS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 448, CC(C)S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 449, COP(OC)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 450, C[Se]CC[C@H](N)C(=O)O. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 34
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 454, CN(C)C(=S)SC(=S)N(C)C. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 455, OC[P+](CO)(CO)CO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 457, CCOS(C)(=O)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 458, COP(=O)(O)OP(=O)(O)OC. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 463, OCC(Br)=C(Br)CO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 464, BrC/C=C/CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 465, N#CC(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 467, O=S(=O)([O-])CCO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 471, COP(=O)(O)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 473, Cl[Dy](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 474, CCCSSCCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 477, N#CC(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 478, NC(=O)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 479, CC(=O)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 481, CCCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 487, OC(O)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 491, ClC(Cl)=C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 492, ClC(Cl)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 493, ClCC(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 495, OCCSCSCCO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 497, CC(=O)[O-].[K+]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 19
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 498, O=C(Cl)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 501, CCSC(=O)N(CC)CC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 504, ClCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 506, CC(Br)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 507, S=C([S-])NCCNC(=S)[S-].[Mn+2]. Appending e
mpty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 511, COS(=O)(=O)C(F)(F)F. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 515, O=C(O)C(Br)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 516, O=C([O-])CS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 518, O=S(=O)([O-])SSS(=O)(=O)[O-]. Appending em
pty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 520, BrCC(Br)(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 521, BrC(Br)C(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 523, CS(C)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 532, CNC(=S)N(C)C. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 534, CSC(C)(C)/C=N\O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 535, N#CCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 536, ClCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 537, O=C([O-])C(=O)[O-].[Ca+2]. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 20
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 539, Br[Ca]Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 543, CN(C)C(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 544, C#CC(O)(/C=C\Cl)CC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 545, BrCCCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 546, O=P([O-])(O)C(Cl)(Cl)P(=O)([O-])O. Appendi
ng empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 548, CCOP(OCC)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 551, CCCCCCC(C)(C)S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 559, CC(CCl)OC(C)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 560, OCC(CO)(CBr)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 561, ClCCOCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 563, OCCSCCO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 564, P#[In]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 565, O=C(O)CC(Cl)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 566, N#CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 567, [Mn+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 25
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 568, CCC(C)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 571, FC(F)OC(Cl)C(F)(F)F. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 576, CC(O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 578, O=C([O-])[O-].[Sr+2]. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 38
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 583, CNC(=S)NC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 586, ClCCOCCOCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 588, CC(C)(C)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 593, N#CC(Br)(Br)C(N)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 597, CCCC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 598, CCCCCCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 599, CC(Cl)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 600, O=C(O)CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 601, O=[Cd]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 48
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 604, S=C=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 608, ClC(Cl)(Cl)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 611, O=S(=O)([O-])CCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 613, OCC(O)CS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 614, [Be+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 4
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 615, CC(C)(C)C(=O)C(Cl)Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 617, O=S(=O)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 618, O=S(=O)([O-])[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 621, O=C([O-])CC(S[Au])C(=O)[O-]. Appending emp
ty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 629, CN(C)CCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 637, O=P(O)(O)CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 638, N#CC(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 639, NC(=O)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 640, CC(Cl)C(C)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 644, CC=C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 648, CC(=O)N[C@@H](CS)C(=O)O. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 650, [Zn+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 30
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 651, O=C(O)C(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 653, O=C([O-])P(=O)([O-])[O-]. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 654, C=CS(=O)(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 660, CCCCCCCCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 666, O=S(=O)([O-])O.[Na+]. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 668, ClSC(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 671, N[Pt](N)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 678, CCCCSCCCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 681, COS(C)(=O)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 686, O=C(O)CS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 688, NCCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 692, CN(C)P(=O)(N(C)C)N(C)C. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 694, CC(C)OC(=S)[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 695, CNC(=O)ON=C(C)SC. Appending empty array
```

```
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 698, CCOC(=O)CC(=O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 699, BrC(Br)C(Br)(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 700, CCOC(=O)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 702, O=[Se](O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 34
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 703, NC(N)=[Se]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 34
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 704, CCOS(=O)(=O)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 707, C=C(C)CS(=O)(=O)[O-]. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 708, CCCCOC(=O)CCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 712, CNCCS(=O)(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 713, FC(F)(F)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 715, CCSCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 716, ClCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 718, OCC(Br)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 721, C[Si](C)(C)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 722, O=[N+]([O-])C(Cl)(Cl)Cl. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 724, O=C([O-])CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 728, CCCCCSCCCCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 729, CC(S)C(C)S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 732, O=[Se]=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 34
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 733, ClC/C=C\CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 735, COC(=O)CS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 737, ClCC(Br)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 738, BrCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 740, IC(I)=C(I)I. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 741, O=P([O-])([O-])OP(=O)([O-])[O-]. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 742, CCCCCCCCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 743, N#CCCSCCC#N. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 745, CC(Cl)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 748, CC(C)(Cl)[N+](=O)[O-]. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 749, CCC(Cl)[N+](=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 751, N#CSCC(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 755, CCN(CC)CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 757, C=CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 758, C=CCN=C=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 761, O=S([O-])CO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 762, CCSC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 764, CCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 765, OCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 767, C=CCOCC(O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 770, C/C(Cl)=C/CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 771, BrCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 773, O=S(=O)([O-])OOS(=O)(=O)[O-]. Appending em
pty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 774, O=[N+]([O-])[O-].[K+]. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 19
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 778, CS(=O)(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 780, Cl[Zn]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 781, ClC/C=C/CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 782, CC(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 783, ClCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 786, NC(=S)NC(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 788, CC(Cl)(Cl)C(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 790, C[Sn](Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 791, C=CCS(=O)(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 792, O=P([O-])(O)OP(=O)([O-])O. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 799, CN(CCCl)CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 800, C=C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 801, N=C(N)S(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 803, COP(=O)(OC)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 810, Cl[Ni]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 812, CCOP(=S)(Cl)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 813, COP(=S)(Cl)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 814, Cl[Sb](Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 823, O=[Bi]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 824, [Br-].[Na+]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 839, N[C@@H](CSCCCO)C(=O)O. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 843, O=C(C(Cl)(Cl)Cl)C(Cl)(Cl)Cl. Appending emp
ty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 849, II. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 853, O=C([O-])C(Cl)(Cl)Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 857, O=C(CCl)NCO. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 858, SCCCCCCCCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 861, O=C(Cl)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 863, COC(=O)C(Br)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 864, ClC=C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 865, ClCC(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 866, CC(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 867, FC(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 873, ClCC=CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 877, CCOC(=O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 879, O=S(=O)([O-])CC(S)CS. Appending empty arra
```

```
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 880, O=[N+]([O-])C(Br)(CO)CO. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 881, OCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 888, SCCSCCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 889, CCCCNC(C)(C)[PH](=O)O. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 892, CCCCCCCCSCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 893, O=S(O)CO[Na]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 901, COP(C)(=O)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 906, [Ca+2].[Cl-].[Cl-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 911, CCCCCCCCCI. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 918, CC(C)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 919, [SbH6+3]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 51
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 920, ClC(Cl)C(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 924, O=P(O)(O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 927, NC(CSCC(=O)O)C(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 935, [Ni+2]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 28
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 940, O=[N+]([O-])O[Cd]O[N+](=O)[O-]. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 48
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 941, CCOC(=O)C(Cl)C(C)=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 943, N[C@@H](COP(=O)(O)O)C(=O)O. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 947, NNC(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 948, CCCCS(=O)(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 949, NC(N)=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 952, CCOC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 953, NS(=O)(=O)[O-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 954, COCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 955, C[N+](C)(C)CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 956, CCOP(=S)(OCC)SCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 958, CC[Ge](Cl)(CC)CC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 960, CCOP(=O)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 967, O=C(O)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 968, CCC(C)SSC(C)CC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 970, O=CC(Br)(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 971, O=C(O)C(Br)(Br)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 978, C=C(C)CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 979, CCCCCCCCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 981, CCCCC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 982, CCC(C)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 983, N#CCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 984, O=C(O)CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 985, O=S(=O)(O)F. Appending empty array
```

```
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 986, O=C(Cl)/C=C/C(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 987, O=C(O)CCSCCC(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 989, FC(F)(F)C(Cl)Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 991, O=C(O)CCS. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 997, CO[PH](=O)OC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1003, CC(C)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1008, O=C(CCl)C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1009, N#C[Au-]C#N. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 79
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1010, ClC(Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1013, CCCSC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1017, ClC=CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1018, NCCS(=O)(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1020, CCOP(O)OCC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1022, FC(Cl)(Cl)C(F)(Cl)Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1024, CN(C)C(=S)[S-]. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1027, ClCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1028, CCCCCCCCCBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1029, Cl[Sn](Cl)(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1030, CC(O)(P(=O)([O-])O)P(=O)([O-])O. Appendin
g empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1041, CN=C=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1043, ClCCOCOCCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1044, CC(C)=CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1046, C[As](C)(=O)O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 33
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1049, COP(N)(=O)SC. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1053, O=S(=O)([O-])CC(O)CCl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1054, CC(C)CP(=S)([S-])CC(C)C. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1060, O=C([O-])C(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1061, NNC(=S)NN. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1065, O=CCCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1067, CCCCCCCC(=O)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1070, NC(=O)CI. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1071, IC(I)I. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1073, FC(F)OC(F)(F)C(F)Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1074, O=S(=O)(Cl)c1ccccc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1077, Nc1ccccc1S(=O)(=O)O. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1087, ClCc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1090, Nc1cc(Cl)cc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1093, ClCc1ccccc1Cl. Appending empty array
```

```
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1094, COc1cccc(Br)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1099, Cc1cc(O)cc(C)c1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1100, S=C=NCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1101, ClCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1107, O=C=Nc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1109, O=Cc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1111, O=Cc1cccc(Br)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1117, O=C(CBr)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1120, Cc1cc(C)c(N)c(Cl)c1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1122, NC(=S)Nc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1127, O=C(Cl)c1cc(Cl)cc(Cl)c1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1128, COC(=O)c1cccc(Cl)c1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1129, O=[N+]([O-])c1ccc(Cl)cc1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1135, [S-]c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1137, O=[N+]([O-])c1cccc(Cl)c1Cl. Appending emp
ty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1138, O=[N+]([O-])c1ccc(Cl)cc1Cl. Appending emp
ty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1139, O=[N+]([O-])c1ccc(Cl)c(Cl)c1. Appending e
mpty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1140, Oc1ccc(Cl)c(Cl)c1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1141, Oc1cc(Cl)cc(Cl)c1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1142, Oc1c(Cl)ccc(Cl)c1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1143, ClC(Cl)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1146, O=C(Cl)OCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1151, BrCCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1153, Cc1c(Cl)cccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1159, Cc1ccccc1CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1164, Nc1ccc(Cl)cc1[N+](=O)[O-]. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1167, Oc1cccc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1170, Cc1c(N)cccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1171, Cc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1172, Nc1ccc(Br)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1177, NC(=S)c1c(Cl)cccc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1178, Cc1ccccc1Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1184, BrCCOc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1185, Brc1ccc(Br)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1191, CCc1ccc(Br)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1196, Cc1ccccc1S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1199, O=S(=O)([O-])c1ccc(O)cc1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1205, Nc1cc(Cl)c(N)c(Cl)c1. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1206, Oc1cc(Cl)c(Cl)cc1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1210, O=[N+]([O-])c1ccccc1CCl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1211, O=[N+]([O-])c1ccc(CCl)cc1. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1212, O=[N+]([O-])c1cccc(CCl)c1. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1213, Nc1ccc(S(=O)(=O)[O-])cc1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1221, CC(=O)O[Hg]c1ccccc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 80
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1222, NC(=O)c1c(Cl)cccc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1223, Fc1cccc(Cl)c1CCl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1231, O=C(CCl)Nc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1232, COc1ccc(OC)c(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1233, Cc1cc(O)c(Cl)cc1C. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1237, O=C(Cl)c1c(Cl)cccc1Cl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1241, Cc1ccc(S(=O)(=O)O)cc1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1246, ICCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1247, O=[N+]([O-])c1cccc(I)c1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1248, Nc1c(Cl)cc(Cl)cc1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1249, Nc1cc(Cl)ccc1O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1250, O=Cc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1251, N#Cc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1252, COC(=O)c1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1255, N#CSCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1260, O=P(O)(O)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1263, Clc1ccc(Cl)c(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1274, O=[N+]([O-])c1ccc(CBr)cc1. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1275, Cc1ccc(Br)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1277, COc1ccc(Br)c(C)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1278, Oc1c(Cl)c(Cl)cc(Cl)c1Cl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1287, C=Cc1cccc(CCl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1288, Cc1cc(O)c(Cl)c(C)c1Cl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1290, Cc1ccc(S(N)(=O)=O)cc1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1291, CCC(=O)c1ccc(Cl)cc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1293, CC(C)(C)c1ccc(S)cc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1294, O=C(CS)Nc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1306, Nc1cc(Cl)ccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1307, Nc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1308, O=[PH](O)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1313, O=C(CBr)c1ccc(O)cc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1314, FC(F)(F)c1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1315, O=C(O)c1cccc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1316, O=C(O)c1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1317, O=C(O)c1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1318, FC(F)(F)c1cccc(Cl)c1. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1322, Cc1ccccc1S(=O)(=O)[O-]. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1324, O=Cc1cc(O)ccc1Br. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1326, Clc1ccc(C(Cl)(Cl)Cl)cc1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1328, Cc1cc(Cl)c(C)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1329, Clc1ccc(Cl)c(Cl)c1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1330, [O-]c1cc(Cl)c(Cl)cc1Cl. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1332, COc1cc(Cl)ccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1333, CSc1ccc(C=O)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1338, Cc1ccc(Cl)c(O)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1344, Cc1cc(O)ccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1345, Oc1c(Cl)cc(Cl)c(Cl)c1Cl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1347, COc1ccc(Cl)cc1C. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1348, Cc1cc(Cl)ccc1N. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1351, Cc1ccc(Cl)cc1N. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1352, Cc1ccc(N)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1353, Nc1ccc(N)c(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1354, Nc1ccc(Cl)cc1N. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1355, Nc1ccc(Cl)c(N)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1356, O=[N+]([O-])c1ccccc1Cl. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1359, O=S(=O)(O)c1ccccc1O. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1361, N#Cc1c(Cl)cccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1363, ClC(Cl)(Cl)c1ccccc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1367, Sc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1368, Nc1cc(Cl)c(O)c(Cl)c1. Appending empty arr
```

```
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1369, O=C(O)c1ccc([Hg]Cl)cc1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1370, Oc1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1374, SCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1379, Oc1cc(Cl)c(Cl)c(Cl)c1Cl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1380, Nc1cc(C(=O)O)ccc1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1381, N#CCc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1383, Cc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1384, Oc1cc(O)c(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1391, Oc1ccccc1[Hg]Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 80
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1394, O=S(=O)(O)c1ccc(Cl)cc1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1395, O=C(O)c1cc(Cl)cc(Cl)c1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1396, O=C(O)c1ccc(Cl)c(Cl)c1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1398, Clc1c(Cl)cc(Cl)c(Cl)c1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1402, Oc1ccc(Cl)c(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1403, Oc1c(Cl)cccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1405, Oc1cc(Cl)cc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1406, Oc1cccc(Cl)c1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1407, Clc1cccc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1408, Oc1cc(Cl)ccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1411, CC(C)(CCl)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1419, Nc1ccc([As](=O)([O-])O)cc1. Appending emp
ty array
WARNING:deepchem.feat.base_classes:Exception message: 33
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1421, O=Cc1ccc(F)c(Br)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1422, O=C(O)c1ccccc1S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1423, O=P(O)(O)Oc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1429, Oc1cc(Cl)c(Cl)c(Cl)c1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1436, Nc1ccc(Cl)c(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1437, Nc1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1438, FC(F)(F)c1ccc(Cl)cc1. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1440, Nc1ccc([As](=O)(O)O)cc1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 33
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1443, OCCSc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1446, O=S(=O)(O)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1450, Sc1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1451, ClCc1cccc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1453, Fc1cc(Br)ccc1CBr. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1455, Oc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1457, Cc1c(Cl)cccc1[N+](=O)[O-]. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1460, Cc1cc(Cl)ccc1O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1462, Clc1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1463, Clc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1464, Oc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1465, Brc1ccc(Br)c(Br)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1471, N#Cc1cc(Br)c(O)c(Br)c1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1472, N#Cc1cc(I)c(O)c(I)c1. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 53
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1477, O=S(=O)(O)c1ccc(O)cc1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1480, Clc1cc(Cl)cc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1481, Clc1cccc(Cl)c1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1483, Clc1cc(Cl)c(Cl)c(Cl)c1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1484, Nc1ccc(S(N)(=O)=O)cc1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1487, Nc1cccc(S(=O)(=O)O)c1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1493, Brc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1498, Oc1c(Br)cc(Br)cc1Br. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1499, Oc1ccc([Hg]Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1500, Oc1c(Cl)cc(Cl)cc1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1501, O=C(Cl)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1508, Clc1cc(Cl)c(Cl)c(Cl)c1Cl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1509, Cc1cccc(Br)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1510, BrCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1513, O=[N+]([O-])c1cccc(Cl)c1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1517, Cl[Hg]c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1521, Cc1ccccc1S(N)(=O)=O. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1523, Cc1ccc(Cl)c(N)c1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1525, O=C(Cl)c1ccccc1F. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1530, Fc1ccc(Br)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1533, OCc1ccc(Cl)cc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1536, S=C=NCCc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1539, S=C=Nc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1541, Cc1ccc(S)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1542, CCCCc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1562, ClP(Cl)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1571, O=Cc1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1572, Nc1cccc(Cl)c1. Appending empty array
```

```
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1580, CCC(=O)c1cccc(Cl)c1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1581, O=[N+]([O-])c1ccc(F)c(Cl)c1. Appending em
pty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1585, OB(O)O[Hg]c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 80
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1588, BrCc1ccc(Br)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1589, O=C(O)c1cc(Cl)ccc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1590, O=C(O)c1c(Cl)cccc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1591, O=C(O)c1cccc(Cl)c1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1592, O=C(O)c1ccc(Cl)cc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1593, O=Cc1c(Cl)cccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1594, O=Cc1ccc(Cl)c(Cl)c1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1595, O=C(Cl)c1ccc(F)c(Cl)c1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1598, Cc1ccc([N+](=O)[O-])cc1Cl. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1599, Cc1cccc(Cl)c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1600, NS(=O)(=O)c1ccccc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1603, O=C=Nc1ccc(Cl)c(Cl)c1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1605, NC(=O)Nc1ccc(Cl)cc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1607, O=C(CCl)c1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1608, Nc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1609, CSc1ccc(Cl)cc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1610, Cc1ccccc1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1611, CS(=O)(=O)c1ccc(Cl)cc1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1612, Clc1ccccc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1614, Oc1nc(Cl)c(Cl)cc1Cl. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1616, O=C(O)c1cccc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1619, Clc1ccccn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1620, ClCc1ccccn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1621, ClCc1cccnc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1626, Clc1nc(Cl)c(Cl)c(Cl)c1Cl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1629, Clc1cccc(C(Cl)(Cl)Cl)n1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1637, CCc1cc(C(N)=S)ccn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1643, Clc1ccc(C(Cl)(Cl)Cl)cn1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1648, Cc1nc(C)c(Cl)c(O)c1Cl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
```

```
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1649, O=C(O)c1nc(Cl)ccc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1659, Clc1cc(Cl)c(Cl)nc1Cl. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1665, Clc1cccc2ccccc12. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1668, Clc1ccc2ccccc2c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1674, O=S(=O)(O)NC1CCCCC1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1683, SC1CCCCC1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1694, O=S(=O)([O-])NC1CCCCC1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1706, CCn1cc[n+](C)c1.N#C[S-]. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1729, [O-][n+]1ccccc1[S-]. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1740, [O-][n+]1ccc(Cl)cc1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1748, Nc1nc(N)nc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1750, Clc1nc(Cl)nc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1751, CCNc1nc(N)nc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1754, CC(C)SCc1ccco1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1758, Cc1c(S)cco1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1759, CSSc1c(C)occ1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1764, CC(=O)SCc1ccco1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1777, C[C@@H]1O[C@@H]1P(=O)([O-])[O-]. Appendin
g empty array
WARNING:deepchem.feat.base_classes:Exception message: 15
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1781, BrCC1CO1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1784, ClCC1CO1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1787, Cc1nc2ccccc2s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1788, Nc1nc2c(Cl)cccc2s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1789, Sc1nc2ccccc2s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1790, [S-]c1nc2ccccc2s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1791, c1ccc2scnc2c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1792, Nc1nc2ccccc2s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1793, Brc1nc2ccccc2s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1794, Cc1cccc2c1nc(N)s2. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1795, Nc1nc(Cl)cc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1796, Clc1cc(Cl)nc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1797, Clc1ccnc(Cl)n1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1808, Cc1c(CCCl)scn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1809, Cc1cscn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1810, CC(C)Cc1nccs1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1811, Nc1nc(CC(=O)O)cs1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1812, Nc1nccs1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1813, CC(=O)c1nccs1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1814, Nc1ncc([N+](=O)[O-])s1. Appending empty a
```

```
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1815, Cc1c(CCO)scn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1816, CCCc1ncc(C(=O)O)s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1817, CC(=O)c1c(C)nc(C)s1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1849, Cc1nc2ccc(Cl)cc2[nH]1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1870, O=C1CCCCC1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1892, CSc1cnccn1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1896, ClCCN1CCCC1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1902, CC1(C)C(=O)N(Cl)C(=O)N1Br. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1903, CC1(C)C(=O)N(Br)C(=O)N1Cl. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1905, CC1(C)C(=O)N(Br)C(=O)N1Br. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1906, CC1(C)C(=O)N(Cl)C(=O)N1Cl. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1920, SC1CCCC1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1924, O=C1OC(O)C(C(Cl)Cl)=C1Cl. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1925, Clc1c(Cl)c(Cl)c(Cl)s1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1926, c1cscc1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1927, Cc1ccc(C=O)s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1928, Cc1c(Br)ccs1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1933, O=c1[nH]c(=O)n(Cl)c(=O)n1Cl. Appending em
pty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1937, O=[N+]([O-])C1(Br)COCOC1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 35
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1952, Clc1ccc2[nH]nnc2c1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1954, Oc1ccc(Cl)c2c1CCC2. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1962, O=c1[nH]c2cc(Cl)ccc2o1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1963, O=c1[nH]c2ccc(Cl)cc2o1. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1974, [Fe+2].c1c[cH-]cc1.c1c[cH-]cc1. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 26
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1975, [Ni+2].c1c[cH-]cc1.c1c[cH-]cc1. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 28
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1976, [Cr+2].c1c[cH-]cc1.c1c[cH-]cc1. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 24
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1977, [Co+2].c1c[cH-]cc1.c1c[cH-]cc1. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 27
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1982, Cn1sccc1=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1983, Cn1sc(Cl)cc1=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1984, Cc1c[nH]c(=S)[nH]c1=O. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1985, Cc1cc(=O)[nH]c(=S)[nH]1. Appending empty
array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1986, O=c1cc[nH]c(=S)[nH]1. Appending empty arr
```

```
ay
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1987, CCCc1cc(=O)[nH]c(=S)[nH]1. Appending empt
y array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 1998, O=C1CCC(=O)N1Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2008, Cl[C@H]1OCCO[C@@H]1Cl. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2011, Sc1nnc(S)s1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2024, O=c1[nH]sc2ccccc12. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2025, O=c1[nH]sc2cc(Cl)ccc12. Appending empty a
rray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2033, Cn1cc[nH]c1=S. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2040, O=C1Cc2cc(Cl)ccc2N1. Appending empty arra
y
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2058, ClC1=C(Cl)C(Cl)(Cl)C(Cl)=C1Cl. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2061, C1CSCCS1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2062, CC1(O)CSC(C)(O)CS1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2063, O=S1(=O)CCCC1. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2064, O=S1(=O)CC(Cl)(Cl)C(Cl)(Cl)C1. Appending
empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2065, CC(=O)NC1CCSC1=O. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2069, S=C1NC=NC2N=CNC12. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2070, S=c1[nH]c2ccccc2[nH]1. Appending empty ar
ray
WARNING:deepchem.feat.base_classes:Exception message: 16
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2072, O=c1cc(O)c(Cl)c[nH]1. Appending empty arr
ay
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Failed to featurize datapoint 2079, ClC1=C(Cl)C1(Cl)Cl. Appending empty array
WARNING:deepchem.feat.base_classes:Exception message: 17
WARNING:deepchem.feat.base_classes:Exception message: setting an array element with a sequence. The requested ar
ray has an inhomogeneous shape after 1 dimensions. The detected shape was (2081,) + inhomogeneous part.
```

Remove more invalid molecules.

In [11]:
```
indices = [ i for i, data in enumerate(features) if type(data) is GraphMatrix ]
print(indices)
features = [features[i] for i in indices]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 15, 16, 18, 19, 20, 21, 22, 23, 24, 26, 28, 29, 30, 31, 34, 35, 36, 37, 38,
43, 46, 47, 48, 50, 51, 52, 53, 56, 58, 59, 60, 62, 63, 64, 65, 69, 70, 72, 73, 74, 76, 77, 78, 79, 80, 83, 84,
85, 86, 88, 89, 92, 93, 94, 95, 96, 97, 99, 100, 101, 102, 103, 104, 105, 109, 110, 111, 112, 113, 114, 115, 119
, 121, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 134, 135, 137, 138, 139, 140, 143, 144, 145, 149, 150,
157, 159, 164, 165, 169, 170, 171, 172, 173, 174, 176, 177, 179, 181, 182, 184, 186, 188, 189, 191, 192, 196, 19
7, 198, 199, 202, 205, 206, 207, 208, 209, 213, 215, 216, 219, 220, 221, 222, 225, 226, 227, 229, 231, 232, 233,
234, 235, 236, 237, 239, 240, 244, 247, 248, 249, 252, 254, 255, 256, 257, 258, 261, 262, 264, 265, 266, 268, 26
9, 270, 273, 274, 275, 277, 278, 280, 283, 285, 286, 287, 288, 289, 292, 296, 297, 298, 299, 300, 301, 302, 303,
304, 306, 307, 308, 309, 312, 313, 314, 315, 317, 318, 319, 320, 321, 324, 325, 326, 327, 328, 329, 330, 331, 33
5, 336, 337, 338, 340, 342, 343, 344, 345, 346, 347, 348, 349, 351, 352, 353, 354, 358, 359, 361, 363, 364, 365,
366, 367, 368, 369, 373, 375, 376, 377, 378, 379, 380, 381, 382, 383, 385, 386, 388, 389, 391, 392, 393, 394, 39
6, 398, 399, 402, 407, 408, 410, 412, 414, 415, 416, 419, 421, 425, 426, 427, 428, 430, 431, 432, 433, 434, 437,
438, 439, 443, 446, 447, 451, 452, 453, 456, 459, 460, 461, 462, 466, 468, 469, 470, 472, 475, 476, 480, 482, 48
3, 484, 485, 486, 488, 489, 490, 494, 499, 500, 502, 503, 505, 508, 509, 510, 512, 513, 514, 517, 519, 522, 524,
525, 526, 527, 528, 529, 530, 531, 533, 538, 540, 541, 542, 547, 549, 550, 552, 553, 554, 555, 556, 557, 558, 56
2, 569, 570, 572, 573, 574, 575, 577, 579, 580, 581, 582, 584, 585, 587, 589, 590, 591, 592, 594, 595, 596, 602,
603, 605, 606, 607, 609, 610, 612, 616, 619, 620, 622, 623, 624, 625, 626, 627, 628, 630, 631, 632, 633, 634, 63
5, 636, 641, 642, 643, 645, 646, 647, 649, 652, 655, 656, 657, 658, 659, 661, 662, 663, 664, 665, 667, 669, 670,
672, 673, 674, 675, 676, 677, 679, 680, 682, 683, 684, 687, 689, 690, 691, 693, 696, 697, 701, 705, 706, 709, 71
0, 711, 714, 717, 719, 720, 723, 725, 726, 727, 730, 731, 734, 736, 739, 744, 746, 747, 750, 752, 753, 754, 756,
759, 760, 763, 766, 768, 769, 772, 775, 776, 777, 779, 784, 785, 787, 789, 793, 794, 795, 796, 797, 798, 802, 80
4, 805, 806, 807, 808, 809, 811, 815, 816, 817, 818, 819, 820, 821, 822, 825, 826, 827, 828, 829, 830, 831, 832,
833, 834, 835, 836, 837, 838, 840, 841, 842, 844, 845, 846, 847, 848, 850, 851, 852, 854, 855, 856, 859, 860, 86
2, 868, 869, 870, 871, 872, 874, 875, 876, 878, 882, 883, 884, 885, 886, 887, 890, 891, 894, 895, 896, 897, 898,
899, 900, 902, 903, 904, 905, 907, 908, 909, 910, 912, 913, 914, 915, 916, 917, 921, 922, 923, 925, 926, 928, 92
9, 930, 931, 932, 933, 934, 936, 937, 938, 939, 942, 944, 945, 946, 950, 951, 957, 959, 961, 962, 963, 964, 965,
966, 969, 972, 973, 974, 975, 976, 977, 980, 988, 990, 992, 993, 994, 995, 996, 998, 999, 1000, 1001, 1002, 1004
, 1005, 1006, 1007, 1011, 1012, 1014, 1015, 1016, 1019, 1021, 1023, 1025, 1026, 1031, 1032, 1033, 1034, 1035, 10
36, 1037, 1038, 1039, 1040, 1042, 1045, 1047, 1048, 1050, 1051, 1052, 1055, 1056, 1057, 1058, 1059, 1062, 1063,
1064, 1066, 1068, 1069, 1072, 1075, 1076, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1088, 1089, 1091
, 1092, 1095, 1096, 1097, 1098, 1102, 1103, 1104, 1105, 1106, 1108, 1110, 1112, 1113, 1114, 1115, 1116, 1118, 11
19, 1121, 1123, 1124, 1125, 1126, 1130, 1131, 1132, 1133, 1134, 1136, 1144, 1145, 1147, 1148, 1149, 1150, 1152,
1154, 1155, 1156, 1157, 1158, 1160, 1161, 1162, 1163, 1165, 1166, 1168, 1169, 1173, 1174, 1175, 1176, 1179, 1180
, 1181, 1182, 1183, 1186, 1187, 1188, 1189, 1190, 1192, 1193, 1194, 1195, 1197, 1198, 1200, 1201, 1202, 1203, 12
04, 1207, 1208, 1209, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1234,
1235, 1236, 1238, 1239, 1240, 1242, 1243, 1244, 1245, 1253, 1254, 1256, 1257, 1258, 1259, 1261, 1262, 1264, 1265
, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1276, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1289, 12
92, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1309, 1310, 1311, 1312, 1319, 1320, 1321,
1323, 1325, 1327, 1331, 1334, 1335, 1336, 1337, 1339, 1340, 1341, 1342, 1343, 1346, 1349, 1350, 1357, 1358, 1360
, 1362, 1364, 1365, 1366, 1371, 1372, 1373, 1375, 1376, 1377, 1378, 1382, 1385, 1386, 1387, 1388, 1389, 1390, 13
92, 1393, 1397, 1399, 1400, 1401, 1404, 1409, 1410, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1420, 1424, 1425,
1426, 1427, 1428, 1430, 1431, 1432, 1433, 1434, 1435, 1439, 1441, 1442, 1444, 1445, 1447, 1448, 1449, 1452, 1454
, 1456, 1458, 1459, 1461, 1466, 1467, 1468, 1469, 1470, 1473, 1474, 1475, 1476, 1478, 1479, 1482, 1485, 1486, 14
88, 1489, 1490, 1491, 1492, 1494, 1495, 1496, 1497, 1502, 1503, 1504, 1505, 1506, 1507, 1511, 1512, 1514, 1515,
1516, 1518, 1519, 1520, 1522, 1524, 1526, 1527, 1528, 1529, 1531, 1532, 1534, 1535, 1537, 1538, 1540, 1543, 1544
, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1563, 15
64, 1565, 1566, 1567, 1568, 1569, 1570, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1582, 1583, 1584, 1586, 1587,
1596, 1597, 1601, 1602, 1604, 1606, 1613, 1615, 1617, 1618, 1622, 1623, 1624, 1625, 1627, 1628, 1630, 1631, 1632
, 1633, 1634, 1635, 1636, 1638, 1639, 1640, 1641, 1642, 1644, 1645, 1646, 1647, 1650, 1651, 1652, 1653, 1654, 16
55, 1656, 1657, 1658, 1660, 1661, 1662, 1663, 1664, 1666, 1667, 1669, 1670, 1671, 1672, 1673, 1675, 1676, 1677,
1678, 1679, 1680, 1681, 1682, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1695, 1696, 1697, 1698
, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 17
18, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737,
1738, 1739, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1749, 1752, 1753, 1755, 1756, 1757, 1760, 1761, 1762, 1763
, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1778, 1779, 1780, 1782, 1783, 1785, 17
86, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825,
1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844
, 1845, 1846, 1847, 1848, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 18
64, 1865, 1866, 1867, 1868, 1869, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883,
1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1893, 1894, 1895, 1897, 1898, 1899, 1900, 1901, 1904, 1907, 1908
, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1921, 1922, 1923, 1929, 1930, 1931, 1932, 19
34, 1935, 1936, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1953, 1955,
1956, 1957, 1958, 1959, 1960, 1961, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1978, 1979, 1980
, 1981, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 20
06, 2007, 2009, 2010, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2026, 2027, 2028,
2029, 2030, 2031, 2032, 2034, 2035, 2036, 2037, 2038, 2039, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049
, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2059, 2060, 2066, 2067, 2068, 2071, 2073, 2074, 2075, 2076, 20
77, 2078, 2080]
```

Instantiate the MolGAN model and set the learning rate and maximum number of atoms as the size of the vertices. Then, we create the dataset in the format of the input to MolGAN.

In [12]:
```
# create model
gan = MolGAN(learning_rate=ExponentialDecay(0.001, 0.9, 5000), vertices=num_atoms)
dataset = dc.data.NumpyDataset([x.adjacency_matrix for x in features],[x.node_features for x in features])
```

Define the iterbatches function because the gan_fit function requires an iterable for the batches.

In [13]:
```
def iterbatches(epochs):
    for i in range(epochs):
        for batch in dataset.iterbatches(batch_size=gan.batch_size, pad_batches=True):
            flattened_adjacency = torch.from_numpy(batch[0]).view(-1).to(dtype=torch.int64) # flatten the input
```

```
            invalid_mask = (flattened_adjacency < 0) | (flattened_adjacency >= gan.edges) # edge type cannot be
            clamped_adjacency = torch.clamp(flattened_adjacency, 0, gan.edges-1) # clamp the input so it can be
            adjacency_tensor = one_hot(clamped_adjacency, num_classes=gan.edges) # actual one_hot
            adjacency_tensor[invalid_mask] = torch.zeros(gan.edges, dtype=torch.long) # make the invalid entries
            adjacency_tensor = adjacency_tensor.view(*batch[0].shape, -1) # reshape to original shape and change

            flattened_node = torch.from_numpy(batch[1]).view(-1).to(dtype=torch.int64)
            invalid_mask = (flattened_node < 0) | (flattened_node >= gan.nodes)
            clamped_node = torch.clamp(flattened_node, 0, gan.nodes-1)
            node_tensor = one_hot(clamped_node, num_classes=gan.nodes)
            node_tensor[invalid_mask] = torch.zeros(gan.nodes, dtype=torch.long)
            node_tensor = node_tensor.view(*batch[1].shape, -1)

            yield {gan.data_inputs[0]: adjacency_tensor, gan.data_inputs[1]:node_tensor}
```

Train the model with the fit_gan function and generate molecules with the predict_gan_generator function.

In [14]:
```
gan.fit_gan(iterbatches(25), generator_steps=0.2, checkpoint_interval=5000)
generated_data = gan.predict_gan_generator(1000)
```

```
/usr/local/lib/python3.10/dist-packages/torch/autograd/graph.py:744: UserWarning: Attempting to run cuBLAS, but
there was no current CUDA context! Attempting to set the primary context... (Triggered internally at ../aten/src
/ATen/cuda/CublasHandlePool.cpp:135.)
  return Variable._execution_engine.run_backward(  # Calls into the C++ engine to run the backward pass
Ending global_step 349: generator average loss -3.55075, discriminator average loss -5.23049
TIMING: model fitting took 8.770 s
Generating 1000 samples
```

Convert the generated graphs to RDKit molecules.

In [15]:
```
nmols = feat.defeaturize(generated_data)
print("{} molecules generated".format(len(nmols)))
```

```
[13:29:24] Explicit valence for atom # 2 O, 30, is greater than permitted
[13:29:24] Explicit valence for atom # 0 C, 26, is greater than permitted
[13:29:24] non-ring atom 0 marked aromatic
1000 molecules generated
[13:29:24] Explicit valence for atom # 1 B, 7, is greater than permitted
[13:29:24] non-ring atom 2 marked aromatic
[13:29:24] Explicit valence for atom # 1 C, 11, is greater than permitted
```

Remove invalid molecules from list.

In [16]:
```
nmols = list(filter(lambda x: x is not None, nmols))
```

Print out the number of valid molecules, but training can be unstable so some the number can vary significantly.

In [17]:
```
# currently training is unstable so 0 is a common outcome
print ("{} valid molecules".format(len(nmols)))
```

```
411 valid molecules
```

Remove duplicate generated molecules.

In [18]:
```
nmols_smiles = [Chem.MolToSmiles(m) for m in nmols]
nmols_smiles_unique = list(OrderedDict.fromkeys(nmols_smiles))
nmols_viz = [Chem.MolFromSmiles(x) for x in nmols_smiles_unique]
print ("{} unique valid molecules".format(len(nmols_viz)))
```

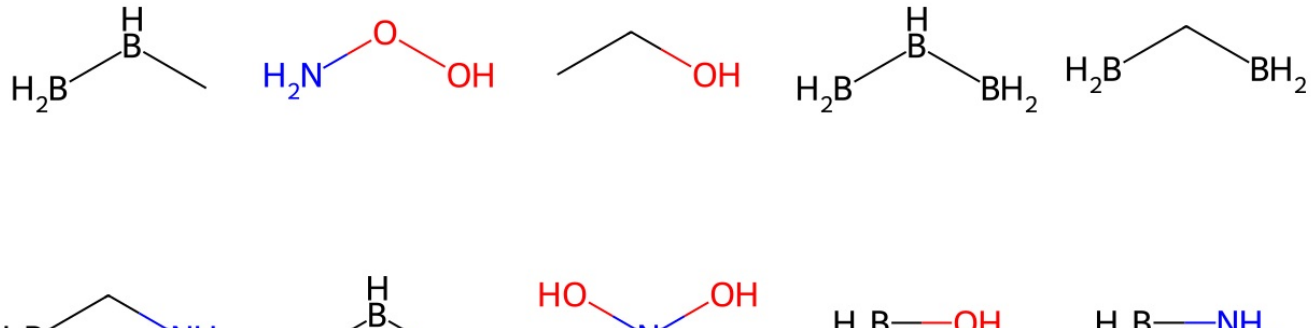
```
48 unique valid molecules
```

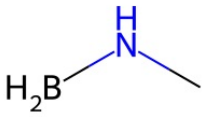
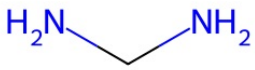
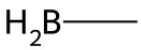
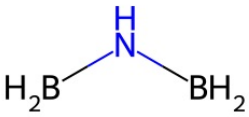
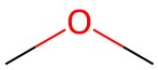
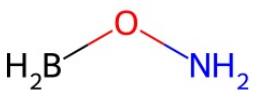
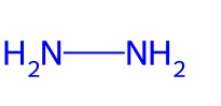
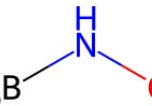
Print out up to 100 unique valid molecules.

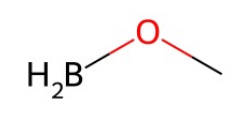
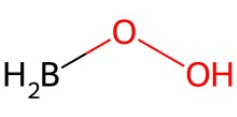
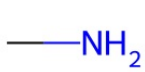
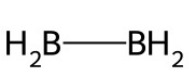
In [19]:
```
img = Draw.MolsToGridImage(nmols_viz[0:100], molsPerRow=5, subImgSize=(250, 250), maxMols=100, legends=None, re
```

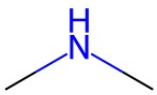
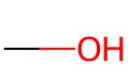
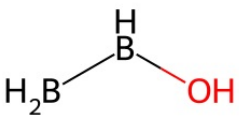
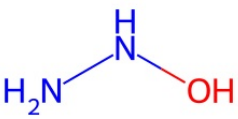
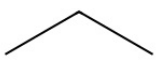
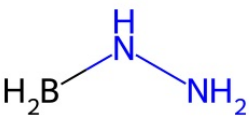
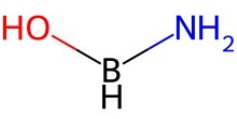
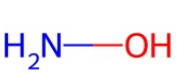
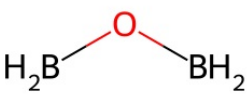
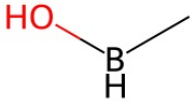
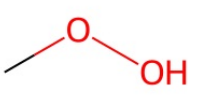
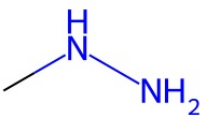
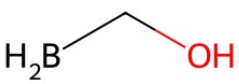
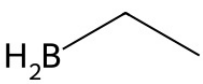
In [20]:
```
img
```

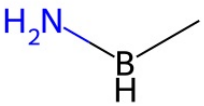
Out[20]:



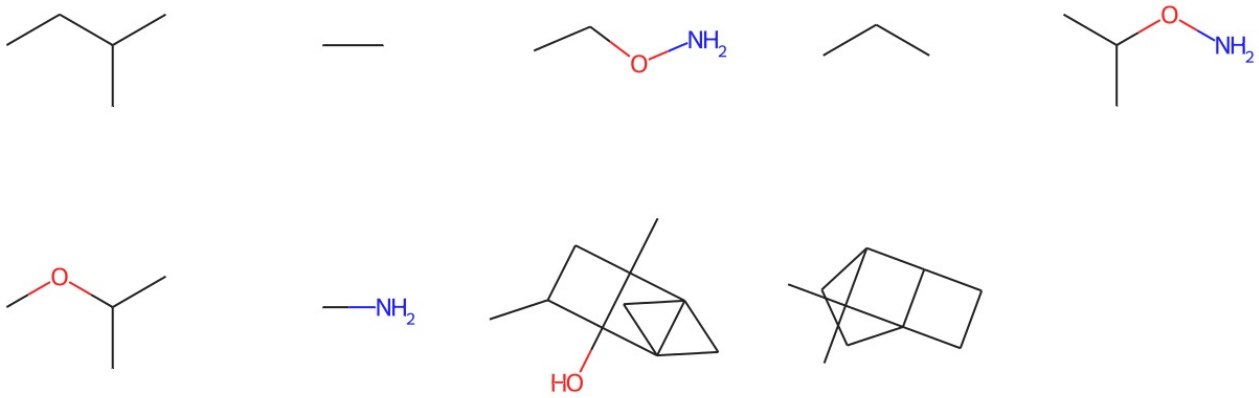












This is an example of what the molecules should look like.



Introduction to GROVER

In this tutorial, we will go over what Grover is, and how to get it up and running.

GROVER, or, Graph Representation frOm selfsuperVised mEssage passing tRansformer, is a novel framework proposed by Tencent AI Lab. GROVER utilizes self-supervised tasks in the node, edge and graph level in order to learn rich structural and semantic information of molecules from large unlabelled molecular datasets. GROVER integrates Message Passing Networks into a Transformer-style architecture to deliver more expressive molecular encoding.

Reference Paper: Rong, Yu, et al. "Grover: Self-supervised message passing transformer on large-scale molecular data." Advances in Neural Information Processing Systems (2020).

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following installation commands. This will take about 5 minutes to run to completion and install your environment. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install Anaconda on your local machine.

Import and Setup required modules.

We will first clone the repository onto the preferred platform, then install it as a library. We will also import deepchem and install descriptastorus.

NOTE: The original GROVER repository does not contain a `setup.py` file, thus we are currently using a fork which does.

In [1]:
```
# Clone the forked repository.
%cd drive/MyDrive
!git clone https://github.com/atreyamaj/grover.git
```

```
/content/drive/MyDrive
fatal: destination path 'grover' already exists and is not an empty directory.
```

In [2]:
```
# Navigate to the working folder.
%cd grover
```

```
/content/drive/MyDrive/grover
```

In [3]:
```
# Install the forked repository.
!pip install -e ./
```

```
Obtaining file:///content/drive/MyDrive/grover
Installing collected packages: grover
  Running setup.py develop for grover
Successfully installed grover-1.0.0
```

In [4]:
```
# Install deepchem and descriptastorus.
!pip install deepchem
!pip install git+https://github.com/bp-kelley/descriptastorus
```

```
Collecting deepchem
  Downloading deepchem-2.6.1-py3-none-any.whl (608 kB)
     |▌                               | 10 kB 29.8 MB/s eta 0:00:01
     |▉                               | 20 kB 34.5 MB/s eta 0:00:01
     |█▎                              | 30 kB 37.0 MB/s eta 0:00:01
     |█▋                              | 40 kB 20.6 MB/s eta 0:00:01
     |██                              | 51 kB 23.0 MB/s eta 0:00:01
     |██▎                             | 61 kB 25.9 MB/s eta 0:00:01
     |██▋                             | 71 kB 23.6 MB/s eta 0:00:01
     |███                             | 81 kB 24.8 MB/s eta 0:00:01
     |███▎                            | 92 kB 26.6 MB/s eta 0:00:01
     |███▋                            | 102 kB 28.3 MB/s eta 0:00:01
     |████                            | 112 kB 28.3 MB/s eta 0:00:01
     |████▎                           | 122 kB 28.3 MB/s eta 0:00:01
     |████▋                           | 133 kB 28.3 MB/s eta 0:00:01
     |█████                           | 143 kB 28.3 MB/s eta 0:00:01
```

```
     |▏                                | 153 kB 28.3 MB/s eta 0:00:01
     |▏                                | 163 kB 28.3 MB/s eta 0:00:01
     |▏                                | 174 kB 28.3 MB/s eta 0:00:01
     |▏                                | 184 kB 28.3 MB/s eta 0:00:01
     |▏                                | 194 kB 28.3 MB/s eta 0:00:01
     |▏                                | 204 kB 28.3 MB/s eta 0:00:01
     |▏                                | 215 kB 28.3 MB/s eta 0:00:01
     |▏                                | 225 kB 28.3 MB/s eta 0:00:01
     |▏                                | 235 kB 28.3 MB/s eta 0:00:01
     |▏                                | 245 kB 28.3 MB/s eta 0:00:01
     |▏                                | 256 kB 28.3 MB/s eta 0:00:01
     |▏                                | 266 kB 28.3 MB/s eta 0:00:01
     |▏                                | 276 kB 28.3 MB/s eta 0:00:01
     |▏                                | 286 kB 28.3 MB/s eta 0:00:01
     |▏                                | 296 kB 28.3 MB/s eta 0:00:01
     |▏                                | 307 kB 28.3 MB/s eta 0:00:01
     |▏                                | 317 kB 28.3 MB/s eta 0:00:01
     |▏                                | 327 kB 28.3 MB/s eta 0:00:01
     |▏                                | 337 kB 28.3 MB/s eta 0:00:01
     |▏                                | 348 kB 28.3 MB/s eta 0:00:01
     |▏                                | 358 kB 28.3 MB/s eta 0:00:01
     |▏                                | 368 kB 28.3 MB/s eta 0:00:01
     |▏                                | 378 kB 28.3 MB/s eta 0:00:01
     |▏                                | 389 kB 28.3 MB/s eta 0:00:01
     |▏                                | 399 kB 28.3 MB/s eta 0:00:01
     |▏                                | 409 kB 28.3 MB/s eta 0:00:01
     |▏                                | 419 kB 28.3 MB/s eta 0:00:01
     |▏                                | 430 kB 28.3 MB/s eta 0:00:01
     |▏                                | 440 kB 28.3 MB/s eta 0:00:01
     |▏                                | 450 kB 28.3 MB/s eta 0:00:01
     |▏                                | 460 kB 28.3 MB/s eta 0:00:01
     |▏                                | 471 kB 28.3 MB/s eta 0:00:01
     |▏                                | 481 kB 28.3 MB/s eta 0:00:01
     |▏                                | 491 kB 28.3 MB/s eta 0:00:01
     |▏                                | 501 kB 28.3 MB/s eta 0:00:01
     |▏                                | 512 kB 28.3 MB/s eta 0:00:01
     |▏                                | 522 kB 28.3 MB/s eta 0:00:01
     |▏                                | 532 kB 28.3 MB/s eta 0:00:01
     |▏                                | 542 kB 28.3 MB/s eta 0:00:01
     |▏                                | 552 kB 28.3 MB/s eta 0:00:01
     |▏                                | 563 kB 28.3 MB/s eta 0:00:01
     |▏                                | 573 kB 28.3 MB/s eta 0:00:01
     |▏                                | 583 kB 28.3 MB/s eta 0:00:01
     |▏                                | 593 kB 28.3 MB/s eta 0:00:01
     |▏                                | 604 kB 28.3 MB/s eta 0:00:01
     |▏                                | 608 kB 28.3 MB/s
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.0.2)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.21.6)
Collecting rdkit-pypi
  Downloading rdkit_pypi-2022.3.1-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (22.5 MB)
     |▏                                | 22.5 MB 1.4 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.4.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.3.5)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem) (2
022.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
pandas->deepchem) (1.15.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from rdkit-pypi->deepchem) (7.1
.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn
->deepchem) (3.1.0)
Installing collected packages: rdkit-pypi, deepchem
Successfully installed deepchem-2.6.1 rdkit-pypi-2022.3.1
Collecting git+https://github.com/bp-kelley/descriptastorus
  Cloning https://github.com/bp-kelley/descriptastorus to /tmp/pip-req-build-_462lldf
  Running command git clone -q https://github.com/bp-kelley/descriptastorus /tmp/pip-req-build-_462lldf
Collecting pandas_flavor
  Downloading pandas_flavor-0.3.0-py3-none-any.whl (6.3 kB)
Requirement already satisfied: xarray in /usr/local/lib/python3.7/dist-packages (from pandas_flavor->descriptast
orus==2.3.0.6) (0.18.2)
  Downloading pandas_flavor-0.2.0-py2.py3-none-any.whl (6.6 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from pandas_flavor->descriptast
orus==2.3.0.6) (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->pa
ndas_flavor->descriptastorus==2.3.0.6) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->pandas_flavo
r->descriptastorus==2.3.0.6) (2022.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (from pandas->pandas_flav
or->descriptastorus==2.3.0.6) (1.21.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
```

```
pandas->pandas_flavor->descriptastorus==2.3.0.6) (1.15.0)
Requirement already satisfied: setuptools>=40.4 in /usr/local/lib/python3.7/dist-packages (from xarray->pandas_f
lavor->descriptastorus==2.3.0.6) (57.4.0)
Building wheels for collected packages: descriptastorus
  Building wheel for descriptastorus (setup.py) ... done
  Created wheel for descriptastorus: filename=descriptastorus-2.3.0.6-py3-none-any.whl size=60704 sha256=10872f9
972ee502829c712449b7dbd8d54717461dce2fdffe495f21e10044446
  Stored in directory: /tmp/pip-ephem-wheel-cache-k9kvyu6l/wheels/f9/c3/4f/e7d01f4f2f1a89aef8f0ef088beb4a9497632
4f3ee21410b10
Successfully built descriptastorus
Installing collected packages: pandas-flavor, descriptastorus
Successfully installed descriptastorus-2.3.0.6 pandas-flavor-0.2.0
```

Extracting semantic motif labels

The semantic motif label is extracted by `scripts/save_feature.py` with feature generator `fgtasklabel`.

In [5]:
```
!python scripts/save_features.py --data_path exampledata/pretrain/tryout.csv  \
                                 --save_path exampledata/pretrain/tryout.npz   \
                                 --features_generator fgtasklabel \
                                 --restart
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
100% 5970/5970 [00:09<00:00, 620.91it/s]
```

Extracting atom/bond contextual properties (vocabulary)

The atom/bond Contextual Property (Vocabulary) is extracted by `scripts/build_vocab.py`.

In [6]:
```
!python scripts/build_vocab.py --data_path exampledata/pretrain/tryout.csv  \
                               --vocab_save_folder exampledata/pretrain  \
                               --dataset_name tryout
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
Building atom vocab from file: exampledata/pretrain/tryout.csv
50000it [00:04, 10946.14it/s]
atom vocab size 324
Building bond vocab from file: exampledata/pretrain/tryout.csv
50000it [00:16, 3094.21it/s]
bond vocab size 353
```

Splitting the data

To accelerate the data loading and reduce the memory cost in the multi-gpu pretraining scenario, the unlabelled molecular data need to be spilt into several parts using `scripts/split_data.py`.

In [7]:
```
!python scripts/split_data.py --data_path exampledata/pretrain/tryout.csv  \
                              --features_path exampledata/pretrain/tryout.npz  \
                              --sample_per_file 100  \
                              --output_path exampledata/pretrain/tryout
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
Number of files: 60
```

Running Pretraining on Single GPU

In [8]:
```
!python main.py pretrain \
```

```
                --data_path exampledata/pretrain/tryout \
                --save_dir model/tryout \
                --atom_vocab_path exampledata/pretrain/tryout_atom_vocab.pkl \
                --bond_vocab_path exampledata/pretrain/tryout_bond_vocab.pkl \
                --batch_size 32 \
                --dropout 0.1 \
                --depth 5 \
                --num_attn_head 1 \
                --hidden_size 100 \
                --epochs 3 \
                --init_lr 0.0002 \
                --max_lr 0.0004 \
                --final_lr 0.0001 \
                --weight_decay 0.0000001 \
                --activation PReLU \
                --backbone gtrans \
                --embedding_output_type both
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
[WARNING] Horovod cannot be imported; multi-GPU training is unsupported
Namespace(activation='PReLU', atom_vocab_path='exampledata/pretrain/tryout_atom_vocab.pkl', backbone='gtrans', b
atch_size=32, bias=False, bond_drop_rate=0, bond_vocab_path='exampledata/pretrain/tryout_bond_vocab.pkl', cuda=T
rue, data_path='exampledata/pretrain/tryout', dense=False, depth=5, dist_coff=0.1, dropout=0.1, embedding_output
_type='both', enable_multi_gpu=False, epochs=3, fg_label_path=None, final_lr=0.0001, fine_tune_coff=1, hidden_si
ze=100, init_lr=0.0002, max_lr=0.0004, no_cache=True, num_attn_head=1, num_mt_block=1, parser_name='pretrain', s
ave_dir='model/tryout', save_interval=9999999999, undirected=False, warmup_epochs=2.0, weight_decay=1e-07)
Loading data
Loading data:
Number of files: 60
Number of samples: 5970
Samples/file: 100
Splitting data with seed 0.
Total size = 5,970 | train size = 5,400 | val size = 570
atom vocab size: 324, bond vocab size: 353, Number of FG tasks: 85
Pre-loaded test data: 6
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 12 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Restore checkpoint, current epoch: 2
GROVEREmbedding(
  (encoders): GTransEncoder(
    (edge_blocks): ModuleList(
      (0): MTBlock(
        (heads): ModuleList(
          (0): Head(
            (mpn_q): MPNEncoder(
              (dropout_layer): Dropout(p=0.1, inplace=False)
              (act_func): PReLU(num_parameters=1)
              (W_h): Linear(in_features=100, out_features=100, bias=False)
            )
            (mpn_k): MPNEncoder(
              (dropout_layer): Dropout(p=0.1, inplace=False)
              (act_func): PReLU(num_parameters=1)
              (W_h): Linear(in_features=100, out_features=100, bias=False)
            )
            (mpn_v): MPNEncoder(
              (dropout_layer): Dropout(p=0.1, inplace=False)
              (act_func): PReLU(num_parameters=1)
              (W_h): Linear(in_features=100, out_features=100, bias=False)
            )
          )
        )
        (act_func): PReLU(num_parameters=1)
        (dropout_layer): Dropout(p=0.1, inplace=False)
        (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (W_i): Linear(in_features=165, out_features=100, bias=False)
        (attn): MultiHeadedAttention(
          (linear_layers): ModuleList(
            (0): Linear(in_features=100, out_features=100, bias=True)
            (1): Linear(in_features=100, out_features=100, bias=True)
```

```
        (2): Linear(in_features=100, out_features=100, bias=True)
      )
      (output_linear): Linear(in_features=100, out_features=100, bias=False)
      (attention): Attention()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (W_o): Linear(in_features=100, out_features=100, bias=False)
    (sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(node_blocks): ModuleList(
  (0): MTBlock(
    (heads): ModuleList(
      (0): Head(
        (mpn_q): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_k): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_v): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
      )
    )
    (act_func): PReLU(num_parameters=1)
    (dropout_layer): Dropout(p=0.1, inplace=False)
    (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
    (W_i): Linear(in_features=151, out_features=100, bias=False)
    (attn): MultiHeadedAttention(
      (linear_layers): ModuleList(
        (0): Linear(in_features=100, out_features=100, bias=True)
        (1): Linear(in_features=100, out_features=100, bias=True)
        (2): Linear(in_features=100, out_features=100, bias=True)
      )
      (output_linear): Linear(in_features=100, out_features=100, bias=False)
      (attention): Attention()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (W_o): Linear(in_features=100, out_features=100, bias=False)
    (sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(ffn_atom_from_atom): PositionwiseFeedForward(
  (W_1): Linear(in_features=251, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_atom_from_bond): PositionwiseFeedForward(
  (W_1): Linear(in_features=251, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_bond_from_atom): PositionwiseFeedForward(
  (W_1): Linear(in_features=265, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_bond_from_bond): PositionwiseFeedForward(
  (W_1): Linear(in_features=265, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(atom_from_atom_sublayer): SublayerConnection(
  (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
  (dropout): Dropout(p=0.1, inplace=False)
)
```

```
    (atom_from_bond_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (bond_from_atom_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (bond_from_bond_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (act_func_node): PReLU(num_parameters=1)
    (act_func_edge): PReLU(num_parameters=1)
    (dropout_layer): Dropout(p=0.1, inplace=False)
  )
)
Total parameters: 768614
EP:3 Model Saved on: model/tryout/model.ep3
Total Time: 14.828
```

Training and Finetuning

Extracting Molecular Features

Given a labelled molecular dataset, it is possible to extract the additional molecular features in order to train & finetune the model from the existing pretrained model. The feature matrix is stored as `.npz`.

In [9]:
```
!python scripts/save_features.py --data_path exampledata/finetune/bbbp.csv \
                                 --save_path exampledata/finetune/bbbp.npz \
                                 --features_generator rdkit_2d_normalized \
                                 --restart
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
[21:04:21] WARNING: not removing hydrogen atom without neighbors
[21:04:21] WARNING: not removing hydrogen atom without neighbors
[21:04:21] WARNING: not removing hydrogen atom without neighbors
```

```
[21:04:21] WARNING: not removing hydrogen atom without neighbors
  0% 6/2039 [00:01<06:41,  5.07it/s][21:04:23] WARNING: not removing hydrogen atom without neighbors
  3% 53/2039 [00:02<00:56, 35.31it/s][21:04:24] WARNING: not removing hydrogen atom without neighbors
  5% 95/2039 [00:04<01:22, 23.62it/s][21:04:26] WARNING: not removing hydrogen atom without neighbors
  6% 115/2039 [00:04<00:50, 38.29it/s][21:04:27] WARNING: not removing hydrogen atom without neighbors
  7% 152/2039 [00:06<01:23, 22.62it/s][21:04:28] WARNING: not removing hydrogen atom without neighbors
  8% 169/2039 [00:07<01:46, 17.53it/s][21:04:29] WARNING: not removing hydrogen atom without neighbors
[21:04:29] WARNING: not removing hydrogen atom without neighbors
 10% 198/2039 [00:08<01:02, 29.58it/s][21:04:30] WARNING: not removing hydrogen atom without neighbors
 13% 268/2039 [00:11<01:06, 26.67it/s][21:04:33] WARNING: not removing hydrogen atom without neighbors
[21:04:33] WARNING: not removing hydrogen atom without neighbors
[21:04:33] WARNING: not removing hydrogen atom without neighbors
 15% 306/2039 [00:12<01:08, 25.31it/s][21:04:34] WARNING: not removing hydrogen atom without neighbors
[21:04:34] WARNING: not removing hydrogen atom without neighbors
[21:04:34] WARNING: not removing hydrogen atom without neighbors
 18% 372/2039 [00:16<03:04,  9.03it/s][21:04:38] WARNING: not removing hydrogen atom without neighbors
 21% 424/2039 [00:16<00:46, 34.78it/s][21:04:39] WARNING: not removing hydrogen atom without neighbors
[21:04:39] WARNING: not removing hydrogen atom without neighbors
 23% 471/2039 [00:18<00:51, 30.55it/s][21:04:40] WARNING: not removing hydrogen atom without neighbors
 26% 528/2039 [00:20<00:56, 26.79it/s][21:04:42] WARNING: not removing hydrogen atom without neighbors
 27% 556/2039 [00:21<00:58, 25.56it/s]WARNING: not removing hydrogen atom without neighbors
 28% 571/2039 [00:22<01:09, 21.01it/s][21:04:44] WARNING: not removing hydrogen atom without neighbors
 29% 598/2039 [00:23<01:02, 23.22it/s][21:04:45] WARNING: not removing hydrogen atom without neighbors
 30% 619/2039 [00:24<00:50, 28.33it/s][21:04:45] WARNING: not removing hydrogen atom without neighbors
[21:04:45] WARNING: not removing hydrogen atom without neighbors
 32% 645/2039 [00:25<00:52, 26.57it/s][21:04:47] WARNING: not removing hydrogen atom without neighbors
[21:04:47] WARNING: not removing hydrogen atom without neighbors
 34% 689/2039 [00:27<00:53, 25.23it/s][21:04:48] WARNING: not removing hydrogen atom without neighbors
 35% 713/2039 [00:28<01:02, 21.14it/s][21:04:50] WARNING: not removing hydrogen atom without neighbors
 39% 800/2039 [00:32<02:30,  8.23it/s]WARNING: not removing hydrogen atom without neighbors
 40% 813/2039 [00:33<01:26, 14.10it/s][21:04:55] WARNING: not removing hydrogen atom without neighbors
 44% 901/2039 [00:36<00:45, 24.82it/s][21:04:58] WARNING: not removing hydrogen atom without neighbors
 44% 905/2039 [00:36<00:46, 24.15it/s][21:04:58] WARNING: not removing hydrogen atom without neighbors
 45% 908/2039 [00:37<01:08, 16.55it/s][21:04:59] WARNING: not removing hydrogen atom without neighbors
[21:04:59] WARNING: not removing hydrogen atom without neighbors
[21:04:59] WARNING: not removing hydrogen atom without neighbors
[21:05:00] WARNING: not removing hydrogen atom without neighbors
 50% 1010/2039 [00:40<00:30, 33.63it/s][21:05:02] WARNING: not removing hydrogen atom without neighbors
 51% 1039/2039 [00:41<00:34, 28.94it/s][21:05:03] WARNING: not removing hydrogen atom without neighbors
 56% 1134/2039 [00:45<00:33, 26.85it/s][21:05:07] WARNING: not removing hydrogen atom without neighbors
 56% 1150/2039 [00:46<00:37, 23.89it/s][21:05:07] WARNING: not removing hydrogen atom without neighbors
 57% 1161/2039 [00:46<00:29, 29.93it/s][21:05:08] WARNING: not removing hydrogen atom without neighbors
 57% 1168/2039 [00:46<00:35, 24.50it/s]WARNING: not removing hydrogen atom without neighbors
 58% 1186/2039 [00:47<00:24, 34.22it/s][21:05:09] WARNING: not removing hydrogen atom without neighbors
 58% 1192/2039 [00:47<00:21, 39.24it/s][21:05:09] WARNING: not removing hydrogen atom without neighbors
 61% 1235/2039 [00:49<00:34, 23.35it/s][21:05:11] WARNING: not removing hydrogen atom without neighbors
 62% 1264/2039 [00:50<00:32, 23.94it/s][21:05:12] WARNING: not removing hydrogen atom without neighbors
 62% 1268/2039 [00:50<00:29, 25.95it/s][21:05:12] WARNING: not removing hydrogen atom without neighbors
 62% 1273/2039 [00:50<00:25, 29.91it/s][21:05:12] WARNING: not removing hydrogen atom without neighbors
 63% 1289/2039 [00:51<00:35, 21.19it/s][21:05:13] WARNING: not removing hydrogen atom without neighbors
 63% 1294/2039 [00:51<00:30, 24.51it/s][21:05:13] WARNING: not removing hydrogen atom without neighbors
 64% 1297/2039 [00:51<00:30, 24.42it/s][21:05:13] WARNING: not removing hydrogen atom without neighbors
 64% 1308/2039 [00:52<00:29, 24.50it/s]
 65% 1318/2039 [00:52<00:31, 23.25it/s][21:05:14] WARNING: not removing hydrogen atom without neighbors
 66% 1339/2039 [00:53<00:19, 35.84it/s][21:05:14] WARNING: not removing hydrogen atom without neighbors
 66% 1354/2039 [00:53<00:27, 25.00it/s][21:05:15] WARNING: not removing hydrogen atom without neighbors
 68% 1384/2039 [00:55<00:30, 21.15it/s][21:05:16] WARNING: not removing hydrogen atom without neighbors
 71% 1443/2039 [00:57<00:26, 22.41it/s][21:05:18] WARNING: not removing hydrogen atom without neighbors
 71% 1451/2039 [00:57<00:22, 26.27it/s][21:05:19] WARNING: not removing hydrogen atom without neighbors
 72% 1467/2039 [00:58<00:29, 19.33it/s][21:05:20] WARNING: not removing hydrogen atom without neighbors
 72% 1471/2039 [00:58<00:26, 21.24it/s][21:05:20] WARNING: not removing hydrogen atom without neighbors
 77% 1568/2039 [01:02<00:20, 23.17it/s][21:05:23] WARNING: not removing hydrogen atom without neighbors
 78% 1591/2039 [01:02<00:20, 22.13it/s][21:05:24] WARNING: not removing hydrogen atom without neighbors[21:05:24
]
WARNING: not removing hydrogen atom without neighbors
[21:05:24] WARNING: not removing hydrogen atom without neighbors
 79% 1616/2039 [01:03<00:15, 26.82it/s][21:05:25] WARNING: not removing hydrogen atom without neighbors
 79% 1620/2039 [01:04<00:19, 21.82it/s][21:05:25] WARNING: not removing hydrogen atom without neighbors
 82% 1665/2039 [01:05<00:11, 31.94it/s][21:05:27] WARNING: not removing hydrogen atom without neighbors
 82% 1671/2039 [01:05<00:12, 29.68it/s][21:05:27] WARNING: not removing hydrogen atom without neighbors
 83% 1685/2039 [01:06<00:17, 19.79it/s][21:05:28] WARNING: not removing hydrogen atom without neighbors
[21:05:28] WARNING: not removing hydrogen atom without neighbors
 83% 1689/2039 [01:06<00:17, 20.30it/s][21:05:28] WARNING: not removing hydrogen atom without neighbors
 84% 1713/2039 [01:07<00:14, 22.54it/s][21:05:29] WARNING: not removing hydrogen atom without neighbors
 85% 1731/2039 [01:08<00:12, 25.18it/s][21:05:30] WARNING: not removing hydrogen atom without neighbors
 87% 1766/2039 [01:09<00:08, 32.14it/s][21:05:31] WARNING: not removing hydrogen atom without neighbors
[21:05:31] WARNING: not removing hydrogen atom without neighbors
 88% 1792/2039 [01:10<00:09, 25.06it/s][21:05:32] WARNING: not removing hydrogen atom without neighbors
 92% 1870/2039 [01:13<00:05, 30.02it/s][21:05:35] WARNING: not removing hydrogen atom without neighbors
 93% 1896/2039 [01:14<00:05, 27.70it/s][21:05:36] WARNING: not removing hydrogen atom without neighbors
 95% 1947/2039 [01:16<00:03, 24.29it/s][21:05:38] WARNING: not removing hydrogen atom without neighbors
 97% 1976/2039 [01:17<00:02, 29.55it/s][21:05:39] WARNING: not removing hydrogen atom without neighbors
100% 2039/2039 [01:19<00:00, 25.67it/s]
```

Finetuning with existing data

Given the labelled dataset and the molecular features, we can use `finetune` function to finetune the pretrained model.

In [10]:
```
!python main.py finetune --data_path exampledata/finetune/bbbp.csv \
                         --features_path exampledata/finetune/bbbp.npz \
                         --save_dir model/finetune/bbbp/ \
                         --checkpoint_path model/tryout/model.ep3 \
                         --dataset_type classification \
                         --split_type scaffold_balanced \
                         --ensemble_size 1 \
                         --num_folds 3 \
                         --no_features_scaling \
                         --ffn_hidden_size 200 \
                         --batch_size 32 \
                         --epochs 10 \
                         --init_lr 0.00015
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
[WARNING] Horovod cannot be imported; multi-GPU training is unsupported
Fold 0
Loading data
Number of tasks = 1
Splitting data with seed 0
100% 2039/2039 [00:00<00:00, 3681.51it/s]
Total scaffolds = 1,025 | train scaffolds = 764 | val scaffolds = 123 | test scaffolds = 138
Label averages per scaffold, in decreasing order of scaffold frequency,capped at 10 scaffolds and 20 labels: [(a
rray([0.72992701]), array([137])), (array([1.]), array([1])), (array([0.]), array([1])), (array([1.]), array([1]
)), (array([1.]), array([1])), (array([0.]), array([1])), (array([1.]), array([1])), (array([1.]), array([2])),
(array([0.]), array([2])), (array([1.]), array([1]))]
Class sizes
p_np 0: 23.49%, 1: 76.51%
Total size = 2,039 | train size = 1,631 | val size = 203 | test size = 205
Loading model 0 from model/tryout/model.ep3
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
```

```
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Pretrained parameter "av_task_atom.linear.weight" cannot be found in model parameters.
Pretrained parameter "av_task_atom.linear.bias" cannot be found in model parameters.
Pretrained parameter "av_task_bond.linear.weight" cannot be found in model parameters.
Pretrained parameter "av_task_bond.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear_rev.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear_rev.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear_rev.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear_rev.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.readout.cached_zero_vector" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_atom.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_atom.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_bond.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_bond.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_atom.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_atom.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_bond.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_bond.bias" cannot be found in model parameters.
GroverFinetuneTask(
  (grover): GROVEREmbedding(
    (encoders): GTransEncoder(
      (edge_blocks): ModuleList(
        (0): MTBlock(
          (heads): ModuleList(
            (0): Head(
              (mpn_q): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_k): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_v): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
            )
          )
          (act_func): PReLU(num_parameters=1)
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
          (W_i): Linear(in_features=165, out_features=100, bias=False)
          (attn): MultiHeadedAttention(
            (linear_layers): ModuleList(
              (0): Linear(in_features=100, out_features=100, bias=True)
              (1): Linear(in_features=100, out_features=100, bias=True)
              (2): Linear(in_features=100, out_features=100, bias=True)
```

```
      )
      (output_linear): Linear(in_features=100, out_features=100, bias=False)
      (attention): Attention()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (W_o): Linear(in_features=100, out_features=100, bias=False)
    (sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(node_blocks): ModuleList(
  (0): MTBlock(
    (heads): ModuleList(
      (0): Head(
        (mpn_q): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_k): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_v): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
      )
    )
    (act_func): PReLU(num_parameters=1)
    (dropout_layer): Dropout(p=0.1, inplace=False)
    (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
    (W_i): Linear(in_features=151, out_features=100, bias=False)
    (attn): MultiHeadedAttention(
      (linear_layers): ModuleList(
        (0): Linear(in_features=100, out_features=100, bias=True)
        (1): Linear(in_features=100, out_features=100, bias=True)
        (2): Linear(in_features=100, out_features=100, bias=True)
      )
      (output_linear): Linear(in_features=100, out_features=100, bias=False)
      (attention): Attention()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (W_o): Linear(in_features=100, out_features=100, bias=False)
    (sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(ffn_atom_from_atom): PositionwiseFeedForward(
  (W_1): Linear(in_features=251, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_atom_from_bond): PositionwiseFeedForward(
  (W_1): Linear(in_features=251, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_bond_from_atom): PositionwiseFeedForward(
  (W_1): Linear(in_features=265, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_bond_from_bond): PositionwiseFeedForward(
  (W_1): Linear(in_features=265, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(atom_from_atom_sublayer): SublayerConnection(
  (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
  (dropout): Dropout(p=0.1, inplace=False)
)
(atom_from_bond_sublayer): SublayerConnection(
```

```
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (bond_from_atom_sublayer): SublayerConnection(
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (bond_from_bond_sublayer): SublayerConnection(
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (act_func_node): PReLU(num_parameters=1)
      (act_func_edge): PReLU(num_parameters=1)
      (dropout_layer): Dropout(p=0.1, inplace=False)
    )
  )
  (readout): Readout()
  (mol_atom_from_atom_ffn): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=300, out_features=200, bias=True)
    (2): PReLU(num_parameters=1)
    (3): Dropout(p=0.1, inplace=False)
    (4): Linear(in_features=200, out_features=1, bias=True)
  )
  (mol_atom_from_bond_ffn): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=300, out_features=200, bias=True)
    (2): PReLU(num_parameters=1)
    (3): Dropout(p=0.1, inplace=False)
    (4): Linear(in_features=200, out_features=1, bias=True)
  )
  (sigmoid): Sigmoid()
)
Number of parameters = 889,418
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0000 loss_train: 1.027524 loss_val: 0.494863 auc_val: 0.8744 cur_lr: 0.00059 t_time: 5.5550s v_time: 0.73
79s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0001 loss_train: 0.855072 loss_val: 0.488093 auc_val: 0.8805 cur_lr: 0.00098 t_time: 5.4703s v_time: 0.74
35s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0002 loss_train: 0.802001 loss_val: 0.488020 auc_val: 0.8953 cur_lr: 0.00073 t_time: 5.5585s v_time: 0.73
17s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0003 loss_train: 0.743282 loss_val: 0.483438 auc_val: 0.8804 cur_lr: 0.00055 t_time: 5.5933s v_time: 0.73
05s
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0004 loss_train: 0.705130 loss_val: 0.473394 auc_val: 0.9043 cur_lr: 0.00041 t_time: 5.5970s v_time: 0.75
58s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0005 loss_train: 0.682583 loss_val: 0.473367 auc_val: 0.8962 cur_lr: 0.00030 t_time: 5.5740s v_time: 0.72
44s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0006 loss_train: 0.659755 loss_val: 0.477886 auc_val: 0.8939 cur_lr: 0.00023 t_time: 5.5852s v_time: 0.72
88s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0007 loss_train: 0.658016 loss_val: 0.476979 auc_val: 0.8923 cur_lr: 0.00017 t_time: 5.5050s v_time: 0.72
80s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0008 loss_train: 0.647427 loss_val: 0.470443 auc_val: 0.9020 cur_lr: 0.00013 t_time: 5.5287s v_time: 0.72
95s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0009 loss_train: 0.646125 loss_val: 0.474078 auc_val: 0.8938 cur_lr: 0.00010 t_time: 5.7616s v_time: 0.72
85s
Model 0 best validation auc = 0.904320 on epoch 4
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
```

```
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Loading pretrained parameter "readout.cached_zero_vector".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_atom_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.bias".
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Model 0 test auc = 0.921247
Ensemble test auc = 0.921247
Fold 1
Loading data
Number of tasks = 1
Splitting data with seed 1
```

```
100% 2039/2039 [00:00<00:00, 3551.50it/s]
Total scaffolds = 1,025 | train scaffolds = 768 | val scaffolds = 132 | test scaffolds = 125
Label averages per scaffold, in decreasing order of scaffold frequency,capped at 10 scaffolds and 20 labels: [(a
rray([0.72992701]), array([137])), (array([1.]), array([2])), (array([1.]), array([3])), (array([0.8]), array([5
])), (array([1.]), array([9])), (array([1.]), array([1])), (array([1.]), array([1])), (array([1.]), array([1])),
(array([1.]), array([1])), (array([1.]), array([1]))]
Class sizes
p_np 0: 23.49%, 1: 76.51%
Total size = 2,039 | train size = 1,631 | val size = 203 | test size = 205
Loading model 0 from model/tryout/model.ep3
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Pretrained parameter "av_task_atom.linear.weight" cannot be found in model parameters.
Pretrained parameter "av_task_atom.linear.bias" cannot be found in model parameters.
Pretrained parameter "av_task_bond.linear.weight" cannot be found in model parameters.
```

```
Pretrained parameter "av_task_bond.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear_rev.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear_rev.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear_rev.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear_rev.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.readout.cached_zero_vector" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_atom.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_atom.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_bond.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_bond.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_atom.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_atom.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_bond.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_bond.bias" cannot be found in model parameters.
GroverFinetuneTask(
  (grover): GROVEREmbedding(
    (encoders): GTransEncoder(
      (edge_blocks): ModuleList(
        (0): MTBlock(
          (heads): ModuleList(
            (0): Head(
              (mpn_q): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_k): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_v): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
            )
          )
          (act_func): PReLU(num_parameters=1)
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
          (W_i): Linear(in_features=165, out_features=100, bias=False)
          (attn): MultiHeadedAttention(
            (linear_layers): ModuleList(
              (0): Linear(in_features=100, out_features=100, bias=True)
              (1): Linear(in_features=100, out_features=100, bias=True)
              (2): Linear(in_features=100, out_features=100, bias=True)
            )
            (output_linear): Linear(in_features=100, out_features=100, bias=False)
            (attention): Attention()
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (W_o): Linear(in_features=100, out_features=100, bias=False)
          (sublayer): SublayerConnection(
            (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
      )
      (node_blocks): ModuleList(
        (0): MTBlock(
          (heads): ModuleList(
            (0): Head(
              (mpn_q): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_k): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_v): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
            )
```

```
        )
        (act_func): PReLU(num_parameters=1)
        (dropout_layer): Dropout(p=0.1, inplace=False)
        (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (W_i): Linear(in_features=151, out_features=100, bias=False)
        (attn): MultiHeadedAttention(
          (linear_layers): ModuleList(
            (0): Linear(in_features=100, out_features=100, bias=True)
            (1): Linear(in_features=100, out_features=100, bias=True)
            (2): Linear(in_features=100, out_features=100, bias=True)
          )
          (output_linear): Linear(in_features=100, out_features=100, bias=False)
          (attention): Attention()
          (dropout): Dropout(p=0.1, inplace=False)
        )
        (W_o): Linear(in_features=100, out_features=100, bias=False)
        (sublayer): SublayerConnection(
          (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (ffn_atom_from_atom): PositionwiseFeedForward(
      (W_1): Linear(in_features=251, out_features=400, bias=True)
      (W_2): Linear(in_features=400, out_features=100, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (act_func): PReLU(num_parameters=1)
    )
    (ffn_atom_from_bond): PositionwiseFeedForward(
      (W_1): Linear(in_features=251, out_features=400, bias=True)
      (W_2): Linear(in_features=400, out_features=100, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (act_func): PReLU(num_parameters=1)
    )
    (ffn_bond_from_atom): PositionwiseFeedForward(
      (W_1): Linear(in_features=265, out_features=400, bias=True)
      (W_2): Linear(in_features=400, out_features=100, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (act_func): PReLU(num_parameters=1)
    )
    (ffn_bond_from_bond): PositionwiseFeedForward(
      (W_1): Linear(in_features=265, out_features=400, bias=True)
      (W_2): Linear(in_features=400, out_features=100, bias=True)
      (dropout): Dropout(p=0.1, inplace=False)
      (act_func): PReLU(num_parameters=1)
    )
    (atom_from_atom_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (atom_from_bond_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (bond_from_atom_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (bond_from_bond_sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (act_func_node): PReLU(num_parameters=1)
    (act_func_edge): PReLU(num_parameters=1)
    (dropout_layer): Dropout(p=0.1, inplace=False)
  )
)
(readout): Readout()
(mol_atom_from_atom_ffn): Sequential(
  (0): Dropout(p=0.1, inplace=False)
  (1): Linear(in_features=300, out_features=200, bias=True)
  (2): PReLU(num_parameters=1)
  (3): Dropout(p=0.1, inplace=False)
  (4): Linear(in_features=200, out_features=1, bias=True)
)
(mol_atom_from_bond_ffn): Sequential(
  (0): Dropout(p=0.1, inplace=False)
  (1): Linear(in_features=300, out_features=200, bias=True)
  (2): PReLU(num_parameters=1)
  (3): Dropout(p=0.1, inplace=False)
  (4): Linear(in_features=200, out_features=1, bias=True)
)
(sigmoid): Sigmoid()
```

```
)
Number of parameters = 889,418
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0000 loss_train: 1.016377 loss_val: 0.492704 auc_val: 0.8791 cur_lr: 0.00059 t_time: 6.3182s v_time: 0.77
94s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0001 loss_train: 0.822924 loss_val: 0.487600 auc_val: 0.8680 cur_lr: 0.00098 t_time: 5.5121s v_time: 0.79
89s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0002 loss_train: 0.752341 loss_val: 0.470391 auc_val: 0.8893 cur_lr: 0.00073 t_time: 5.5443s v_time: 0.76
47s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0003 loss_train: 0.709847 loss_val: 0.468552 auc_val: 0.8863 cur_lr: 0.00055 t_time: 5.6165s v_time: 0.81
04s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0004 loss_train: 0.682037 loss_val: 0.463301 auc_val: 0.8895 cur_lr: 0.00041 t_time: 5.5689s v_time: 0.77
95s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0005 loss_train: 0.659133 loss_val: 0.464382 auc_val: 0.8914 cur_lr: 0.00030 t_time: 5.5949s v_time: 0.80
20s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
```

```
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0006 loss_train: 0.630823 loss_val: 0.463676 auc_val: 0.8871 cur_lr: 0.00023 t_time: 5.5311s v_time: 0.75
48s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0007 loss_train: 0.613836 loss_val: 0.460376 auc_val: 0.8912 cur_lr: 0.00017 t_time: 5.5768s v_time: 0.75
11s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0008 loss_train: 0.604636 loss_val: 0.464385 auc_val: 0.8900 cur_lr: 0.00013 t_time: 5.5764s v_time: 0.78
48s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0009 loss_train: 0.600993 loss_val: 0.461464 auc_val: 0.8902 cur_lr: 0.00010 t_time: 5.6025s v_time: 0.77
36s
Model 0 best validation auc = 0.891352 on epoch 5
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
```

```
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Loading pretrained parameter "readout.cached_zero_vector".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_atom_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.bias".
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Model 0 test auc = 0.920000
Ensemble test auc = 0.920000
Fold 2
Loading data
Number of tasks = 1
Splitting data with seed 2
100% 2039/2039 [00:00<00:00, 3569.05it/s]
Total scaffolds = 1,025 | train scaffolds = 766 | val scaffolds = 125 | test scaffolds = 134
Label averages per scaffold, in decreasing order of scaffold frequency,capped at 10 scaffolds and 20 labels: [(a
rray([0.72992701]), array([137])), (array([1.]), array([1])), (array([1.]), array([1])), (array([1.]), array([1]
)), (array([0.]), array([1])), (array([1.]), array([1])), (array([1.]), array([1])), (array([1.]), array([1])),
(array([0.]), array([5])), (array([1.]), array([1]))]
Class sizes
p_np 0: 23.49%, 1: 76.51%
Total size = 2,039 | train size = 1,631 | val size = 203 | test size = 205
Loading model 0 from model/tryout/model.ep3
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
```

```
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Pretrained parameter "av_task_atom.linear.weight" cannot be found in model parameters.
Pretrained parameter "av_task_atom.linear.bias" cannot be found in model parameters.
Pretrained parameter "av_task_bond.linear.weight" cannot be found in model parameters.
Pretrained parameter "av_task_bond.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear_rev.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_atom.linear_rev.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear.bias" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear_rev.weight" cannot be found in model parameters.
Pretrained parameter "bv_task_bond.linear_rev.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.readout.cached_zero_vector" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_atom.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_atom.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_bond.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_atom_from_bond.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_atom.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_atom.bias" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_bond.weight" cannot be found in model parameters.
Pretrained parameter "fg_task_all.linear_bond_from_bond.bias" cannot be found in model parameters.
GroverFinetuneTask(
  (grover): GROVEREmbedding(
    (encoders): GTransEncoder(
      (edge_blocks): ModuleList(
        (0): MTBlock(
          (heads): ModuleList(
            (0): Head(
              (mpn_q): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
                (act_func): PReLU(num_parameters=1)
                (W_h): Linear(in_features=100, out_features=100, bias=False)
              )
              (mpn_k): MPNEncoder(
                (dropout_layer): Dropout(p=0.1, inplace=False)
```

```
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_v): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
      )
    )
    (act_func): PReLU(num_parameters=1)
    (dropout_layer): Dropout(p=0.1, inplace=False)
    (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
    (W_i): Linear(in_features=165, out_features=100, bias=False)
    (attn): MultiHeadedAttention(
      (linear_layers): ModuleList(
        (0): Linear(in_features=100, out_features=100, bias=True)
        (1): Linear(in_features=100, out_features=100, bias=True)
        (2): Linear(in_features=100, out_features=100, bias=True)
      )
      (output_linear): Linear(in_features=100, out_features=100, bias=False)
      (attention): Attention()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (W_o): Linear(in_features=100, out_features=100, bias=False)
    (sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(node_blocks): ModuleList(
  (0): MTBlock(
    (heads): ModuleList(
      (0): Head(
        (mpn_q): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_k): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
        (mpn_v): MPNEncoder(
          (dropout_layer): Dropout(p=0.1, inplace=False)
          (act_func): PReLU(num_parameters=1)
          (W_h): Linear(in_features=100, out_features=100, bias=False)
        )
      )
    )
    (act_func): PReLU(num_parameters=1)
    (dropout_layer): Dropout(p=0.1, inplace=False)
    (layernorm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
    (W_i): Linear(in_features=151, out_features=100, bias=False)
    (attn): MultiHeadedAttention(
      (linear_layers): ModuleList(
        (0): Linear(in_features=100, out_features=100, bias=True)
        (1): Linear(in_features=100, out_features=100, bias=True)
        (2): Linear(in_features=100, out_features=100, bias=True)
      )
      (output_linear): Linear(in_features=100, out_features=100, bias=False)
      (attention): Attention()
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (W_o): Linear(in_features=100, out_features=100, bias=False)
    (sublayer): SublayerConnection(
      (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
  )
)
(ffn_atom_from_atom): PositionwiseFeedForward(
  (W_1): Linear(in_features=251, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
  (act_func): PReLU(num_parameters=1)
)
(ffn_atom_from_bond): PositionwiseFeedForward(
  (W_1): Linear(in_features=251, out_features=400, bias=True)
  (W_2): Linear(in_features=400, out_features=100, bias=True)
  (dropout): Dropout(p=0.1, inplace=False)
```

```
        (act_func): PReLU(num_parameters=1)
      )
      (ffn_bond_from_atom): PositionwiseFeedForward(
        (W_1): Linear(in_features=265, out_features=400, bias=True)
        (W_2): Linear(in_features=400, out_features=100, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
        (act_func): PReLU(num_parameters=1)
      )
      (ffn_bond_from_bond): PositionwiseFeedForward(
        (W_1): Linear(in_features=265, out_features=400, bias=True)
        (W_2): Linear(in_features=400, out_features=100, bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
        (act_func): PReLU(num_parameters=1)
      )
      (atom_from_atom_sublayer): SublayerConnection(
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (atom_from_bond_sublayer): SublayerConnection(
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (bond_from_atom_sublayer): SublayerConnection(
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (bond_from_bond_sublayer): SublayerConnection(
        (norm): LayerNorm((100,), eps=1e-05, elementwise_affine=True)
        (dropout): Dropout(p=0.1, inplace=False)
      )
      (act_func_node): PReLU(num_parameters=1)
      (act_func_edge): PReLU(num_parameters=1)
      (dropout_layer): Dropout(p=0.1, inplace=False)
    )
  )
  (readout): Readout()
  (mol_atom_from_atom_ffn): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=300, out_features=200, bias=True)
    (2): PReLU(num_parameters=1)
    (3): Dropout(p=0.1, inplace=False)
    (4): Linear(in_features=200, out_features=1, bias=True)
  )
  (mol_atom_from_bond_ffn): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=300, out_features=200, bias=True)
    (2): PReLU(num_parameters=1)
    (3): Dropout(p=0.1, inplace=False)
    (4): Linear(in_features=200, out_features=1, bias=True)
  )
  (sigmoid): Sigmoid()
)
Number of parameters = 889,418
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0000 loss_train: 1.000364 loss_val: 0.507716 auc_val: 0.8434 cur_lr: 0.00059 t_time: 5.7976s v_time: 0.78
02s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0001 loss_train: 0.824395 loss_val: 0.504539 auc_val: 0.8560 cur_lr: 0.00098 t_time: 5.5894s v_time: 0.77
79s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
```

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0002 loss_train: 0.735137 loss_val: 0.493423 auc_val: 0.8539 cur_lr: 0.00073 t_time: 5.5191s v_time: 0.76
10s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0003 loss_train: 0.687535 loss_val: 0.487282 auc_val: 0.8597 cur_lr: 0.00055 t_time: 5.5613s v_time: 0.75
95s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0004 loss_train: 0.681197 loss_val: 0.489330 auc_val: 0.8702 cur_lr: 0.00041 t_time: 5.5513s v_time: 0.75
01s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0005 loss_train: 0.647608 loss_val: 0.488870 auc_val: 0.8618 cur_lr: 0.00030 t_time: 5.6565s v_time: 0.77
39s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0006 loss_train: 0.638494 loss_val: 0.488281 auc_val: 0.8729 cur_lr: 0.00023 t_time: 5.5400s v_time: 0.75
84s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0007 loss_train: 0.626862 loss_val: 0.490144 auc_val: 0.8702 cur_lr: 0.00017 t_time: 5.6183s v_time: 0.78
14s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0008 loss_train: 0.619776 loss_val: 0.484179 auc_val: 0.8782 cur_lr: 0.00013 t_time: 5.9662s v_time: 0.75
96s
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 10 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller th
an what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader
running slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
```

```
  cpuset_checked))
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Epoch: 0009 loss_train: 0.613262 loss_val: 0.486484 auc_val: 0.8789 cur_lr: 0.00010 t_time: 6.3030s v_time: 0.79
31s
Model 0 best validation auc = 0.878887 on epoch 9
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Loading pretrained parameter "readout.cached_zero_vector".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_atom_ffn.2.weight".
```

```
Loading pretrained parameter "mol_atom_from_atom_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.bias".
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
Model 0 test auc = 0.888635
Ensemble test auc = 0.888635
3-fold cross validation
Seed 0 ==> test auc = 0.921247
Seed 1 ==> test auc = 0.920000
Seed 2 ==> test auc = 0.888635
overall_scaffold_balanced_test_auc=0.909961
std=0.015088
```

Predicting output

Extracting molecular features

If the finetuned model uses the molecular feature as input, we need to generate the molecular feature for the target molecules as well.

In [11]:
```
!python scripts/save_features.py --data_path exampledata/finetune/bbbp.csv \
                                 --save_path exampledata/finetune/bbbp.npz \
                                 --features_generator rdkit_2d_normalized \
                                 --restart
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
[21:09:19] WARNING: not removing hydrogen atom without neighbors
[21:09:19] WARNING: not removing hydrogen atom without neighbors
[21:09:19] WARNING: not removing hydrogen atom without neighbors
```

```
[21:09:19] WARNING: not removing hydrogen atom without neighbors
[21:09:19] WARNING: not removing hydrogen atom without neighbors
  0% 6/2039 [00:01<06:32,  5.18it/s][21:09:21] WARNING: not removing hydrogen atom without neighbors
  2% 47/2039 [00:02<01:03, 31.22it/s][21:09:22] WARNING: not removing hydrogen atom without neighbors
  4% 91/2039 [00:04<01:14, 26.15it/s][21:09:24] WARNING: not removing hydrogen atom without neighbors
  6% 120/2039 [00:05<01:03, 30.16it/s][21:09:25] WARNING: not removing hydrogen atom without neighbors
  8% 161/2039 [00:07<01:14, 25.26it/s][21:09:27] WARNING: not removing hydrogen atom without neighbors
  8% 169/2039 [00:08<02:01, 15.42it/s][21:09:28] WARNING: not removing hydrogen atom without neighbors
[21:09:28] WARNING: not removing hydrogen atom without neighbors
 10% 198/2039 [00:08<01:00, 30.28it/s][21:09:29] WARNING: not removing hydrogen atom without neighbors
 13% 265/2039 [00:10<00:55, 32.07it/s][21:09:31] WARNING: not removing hydrogen atom without neighbors
[21:09:31] WARNING: not removing hydrogen atom without neighbors
[21:09:31] WARNING: not removing hydrogen atom without neighbors
 15% 305/2039 [00:12<01:07, 25.66it/s][21:09:32] WARNING: not removing hydrogen atom without neighbors
[21:09:32] WARNING: not removing hydrogen atom without neighbors
[21:09:32] WARNING: not removing hydrogen atom without neighbors
 20% 417/2039 [00:16<00:46, 35.03it/s][21:09:37] WARNING: not removing hydrogen atom without neighbors
[21:09:37] WARNING: not removing hydrogen atom without neighbors
[21:09:38] WARNING: not removing hydrogen atom without neighbors
 23% 464/2039 [00:18<00:48, 32.41it/s][21:09:39] WARNING: not removing hydrogen atom without neighbors
 26% 529/2039 [00:21<00:52, 28.80it/s][21:09:41] WARNING: not removing hydrogen atom without neighbors
 27% 552/2039 [00:22<00:56, 26.22it/s][21:09:42] WARNING: not removing hydrogen atom without neighbors
 28% 570/2039 [00:22<00:58, 25.26it/s][21:09:42] WARNING: not removing hydrogen atom without neighbors
 29% 600/2039 [00:23<00:55, 25.71it/s][21:09:43] WARNING: not removing hydrogen atom without neighbors
 30% 619/2039 [00:24<00:55, 25.71it/s][21:09:44] WARNING: not removing hydrogen atom without neighbors
[21:09:44] WARNING: not removing hydrogen atom without neighbors
 32% 645/2039 [00:25<00:49, 28.14it/s][21:09:45] WARNING: not removing hydrogen atom without neighbors
[21:09:45] WARNING: not removing hydrogen atom without neighbors
 34% 687/2039 [00:27<00:57, 23.64it/s][21:09:47] WARNING: not removing hydrogen atom without neighbors
 35% 714/2039 [00:28<00:57, 23.01it/s][21:09:48] WARNING: not removing hydrogen atom without neighbors
 39% 798/2039 [00:31<00:58, 21.09it/s][21:09:53] WARNING: not removing hydrogen atom without neighbors
 40% 813/2039 [00:33<01:37, 12.61it/s][21:09:54] WARNING: not removing hydrogen atom without neighbors
 44% 897/2039 [00:36<00:49, 23.10it/s][21:09:56] WARNING: not removing hydrogen atom without neighbors
 44% 903/2039 [00:36<00:49, 22.76it/s][21:09:57] WARNING: not removing hydrogen atom without neighbors
 45% 919/2039 [00:37<00:44, 25.44it/s][21:09:58] WARNING: not removing hydrogen atom without neighbors
[21:09:58] [21:09:58] WARNING: not removing hydrogen atom without neighbors
WARNING: not removing hydrogen atom without neighbors
[21:09:59] WARNING: not removing hydrogen atom without neighbors
 50% 1013/2039 [00:41<00:33, 30.69it/s][21:10:01] WARNING: not removing hydrogen atom without neighbors
 51% 1041/2039 [00:42<00:34, 29.09it/s][21:10:02] WARNING: not removing hydrogen atom without neighbors
 56% 1135/2039 [00:45<00:36, 24.64it/s][21:10:05] WARNING: not removing hydrogen atom without neighbors
 57% 1153/2039 [00:46<00:35, 24.77it/s][21:10:06] WARNING: not removing hydrogen atom without neighbors
 57% 1161/2039 [00:46<00:28, 30.81it/s][21:10:07] WARNING: not removing hydrogen atom without neighbors
 57% 1168/2039 [00:47<00:36, 23.87it/s][21:10:07] WARNING: not removing hydrogen atom without neighbors
 58% 1186/2039 [00:47<00:25, 33.33it/s][21:10:07] WARNING: not removing hydrogen atom without neighbors
 59% 1194/2039 [00:48<00:26, 32.07it/s][21:10:08] WARNING: not removing hydrogen atom without neighbors
 61% 1235/2039 [00:49<00:38, 21.11it/s][21:10:10] WARNING: not removing hydrogen atom without neighbors
 62% 1263/2039 [00:51<00:32, 24.19it/s][21:10:11] WARNING: not removing hydrogen atom without neighbors
[21:10:11] WARNING: not removing hydrogen atom without neighbors
 62% 1268/2039 [00:51<00:32, 24.06it/s][21:10:11] WARNING: not removing hydrogen atom without neighbors
 63% 1292/2039 [00:52<00:28, 26.37it/s][21:10:12] WARNING: not removing hydrogen atom without neighbors
[21:10:12] WARNING: not removing hydrogen atom without neighbors
 64% 1296/2039 [00:52<00:30, 24.41it/s][21:10:12] WARNING: not removing hydrogen atom without neighbors
 64% 1308/2039 [00:52<00:27, 26.72it/s][21:10:12] WARNING: not removing hydrogen atom without neighbors
 65% 1318/2039 [00:53<00:31, 22.61it/s][21:10:13] WARNING: not removing hydrogen atom without neighbors
 65% 1334/2039 [00:53<00:22, 31.22it/s][21:10:13] WARNING: not removing hydrogen atom without neighbors
 67% 1356/2039 [00:54<00:36, 18.90it/s][21:10:14] WARNING: not removing hydrogen atom without neighbors
 68% 1384/2039 [00:55<00:36, 17.78it/s][21:10:15] WARNING: not removing hydrogen atom without neighbors
 71% 1442/2039 [00:57<00:27, 22.02it/s][21:10:18] WARNING: not removing hydrogen atom without neighbors
 71% 1448/2039 [00:58<00:27, 21.47it/s][21:10:18] WARNING: not removing hydrogen atom without neighbors
 72% 1465/2039 [00:58<00:19, 29.81it/s][21:10:19] WARNING: not removing hydrogen atom without neighbors
 72% 1472/2039 [00:59<00:26, 21.40it/s][21:10:19] WARNING: not removing hydrogen atom without neighbors
 77% 1567/2039 [01:02<00:22, 21.35it/s][21:10:22] WARNING: not removing hydrogen atom without neighbors
 78% 1587/2039 [01:03<00:13, 33.18it/s][21:10:23] WARNING: not removing hydrogen atom without neighbors
 78% 1591/2039 [01:03<00:20, 21.92it/s][21:10:23] WARNING: not removing hydrogen atom without neighbors
 78% 1594/2039 [01:03<00:20, 21.43it/s]WARNING: not removing hydrogen atom without neighbors
 79% 1616/2039 [01:04<00:15, 26.99it/s][21:10:24] WARNING: not removing hydrogen atom without neighbors
 79% 1620/2039 [01:04<00:19, 21.72it/s][21:10:24] WARNING: not removing hydrogen atom without neighbors
 81% 1659/2039 [01:06<00:16, 23.11it/s][21:10:26] WARNING: not removing hydrogen atom without neighbors
 82% 1671/2039 [01:06<00:13, 28.20it/s][21:10:26] WARNING: not removing hydrogen atom without neighbors
 83% 1686/2039 [01:07<00:14, 23.67it/s][21:10:27] WARNING: not removing hydrogen atom without neighbors
[21:10:27] WARNING: not removing hydrogen atom without neighbors
 83% 1689/2039 [01:07<00:17, 19.68it/s][21:10:27] WARNING: not removing hydrogen atom without neighbors
 84% 1712/2039 [01:08<00:14, 22.21it/s][21:10:28] WARNING: not removing hydrogen atom without neighbors
 85% 1736/2039 [01:09<00:09, 31.18it/s][21:10:29] WARNING: not removing hydrogen atom without neighbors
 87% 1766/2039 [01:10<00:08, 32.87it/s][21:10:30] WARNING: not removing hydrogen atom without neighbors
[21:10:30] WARNING: not removing hydrogen atom without neighbors
 88% 1797/2039 [01:11<00:08, 29.30it/s][21:10:31] WARNING: not removing hydrogen atom without neighbors
 92% 1870/2039 [01:14<00:06, 26.35it/s][21:10:34] WARNING: not removing hydrogen atom without neighbors
 93% 1896/2039 [01:15<00:05, 26.27it/s][21:10:35] WARNING: not removing hydrogen atom without neighbors
 96% 1948/2039 [01:17<00:03, 27.22it/s][21:10:37] WARNING: not removing hydrogen atom without neighbors
 97% 1971/2039 [01:18<00:02, 26.46it/s][21:10:38] WARNING: not removing hydrogen atom without neighbors
100% 2039/2039 [01:20<00:00, 25.38it/s]
```

Predicting output with the finetuned model

In [12]:
```
!python main.py predict --data_path exampledata/finetune/bbbp.csv \
               --features_path exampledata/finetune/bbbp.npz \
               --checkpoint_dir ./model \
               --no_features_scaling \
               --output data_pre.csv
```

```
WARNING:root:No normalization for BCUT2D_MWHI
WARNING:root:No normalization for BCUT2D_MWLOW
WARNING:root:No normalization for BCUT2D_CHGHI
WARNING:root:No normalization for BCUT2D_CHGLO
WARNING:root:No normalization for BCUT2D_LOGPHI
WARNING:root:No normalization for BCUT2D_LOGPLOW
WARNING:root:No normalization for BCUT2D_MRHI
WARNING:root:No normalization for BCUT2D_MRLOW
[WARNING] Horovod cannot be imported; multi-GPU training is unsupported
Loading training args
Loading data
Validating SMILES
Test size = 2,039
Predicting...
  0% 0/3 [00:00<?, ?it/s]Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_func.weig
ht".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
```

```
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Loading pretrained parameter "readout.cached_zero_vector".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_atom_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.bias".
Moving model to cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will cre
ate 4 worker processes in total. Our suggested max number of worker in current system is 2, which is smaller tha
n what this DataLoader is going to create. Please be aware that excessive worker creation might get DataLoader r
unning slow or even freeze, lower the worker number to avoid potential slowness/freeze if necessary.
  cpuset_checked))
 33% 1/3 [00:08<00:17,  8.86s/it]Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_f
unc.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
```

```
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Loading pretrained parameter "readout.cached_zero_vector".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_atom_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.bias".
Moving model to cuda
 67% 2/3 [00:13<00:06,  6.60s/it]Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.act_f
unc.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.edge_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.edge_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_q.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_k.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.heads.0.mpn_v.W_h.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.act_func.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.layernorm.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_i.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.0.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.1.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.linear_layers.2.bias".
Loading pretrained parameter "grover.encoders.node_blocks.0.attn.output_linear.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.W_o.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.node_blocks.0.sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_atom.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_atom_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_atom.act_func.weight".
```

```
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_1.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.weight".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.W_2.bias".
Loading pretrained parameter "grover.encoders.ffn_bond_from_bond.act_func.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.atom_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_atom_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.weight".
Loading pretrained parameter "grover.encoders.bond_from_bond_sublayer.norm.bias".
Loading pretrained parameter "grover.encoders.act_func_node.weight".
Loading pretrained parameter "grover.encoders.act_func_edge.weight".
Loading pretrained parameter "readout.cached_zero_vector".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_atom_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_atom_ffn.4.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.1.bias".
Loading pretrained parameter "mol_atom_from_bond_ffn.2.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.weight".
Loading pretrained parameter "mol_atom_from_bond_ffn.4.bias".
Moving model to cuda
100% 3/3 [00:18<00:00,  6.29s/it]
Saving predictions to data_pre.csv
```

Output

The output will be saved in a file called `data_pre.csv`.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on Github

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

An Introduction to PROTACs

David Zhang and David Figueroa

PROTACs represent an emerging wave of new therapeutics capable of modulating proteins once thought nearly impossible to target. By eliminating rather than merely inhibiting disease-causing proteins, PROTACs address the limitations of existing drug modalities. As PROTACs progress through clinical trials with promising results, there is growing anticipation surrounding their therapeutic potential.

This DeepChem tutorial serves as a starting point for exploring the world of PROTACs and the exciting field of targeted protein degradation. The tutorial is divided into five partitions:

1. Background literature
2. Data extraction
3. Featurization
4. Model deployment
5. References

With that in mind, let's jump into how we can predict efficacy of PROTAC degraders!

1. Background literature

Traditional drug modalities, such as small-molecule drugs or monoclonal antibodies, are limited to certain modes of action, like targeting specific receptors or blocking particular pathways. Targeted protein degradation (TPD) represents a promising new approach to modulate proteins that have been traditionally difficult to target. TPD has given rise to major classes of molecules that have emerged as promising therapeutic approaches against various disease contexts.

1.1 Targeted protein degradation

Targeted protein degradation represents a way of leveraging the cell's natural degradation mechanisms to target disease causing proteins. Typically, the cell maintains protein homeostasis through clearance by proteasomes or lysosomes. By leveraging these intrinsic cellular mechanisms, TPD methods can target a variety of proteins throughout the cell. This has given rise to a collection of TPD methods aimed at degrading proteins that may play roles across many disease states. One of these major class of molecules is proteolysis-targeting chimera (PROTACs).

1.2 How do PROTACs work?

PROTAC molecules are ternary structures consisting of a linker, a ligand to recruit and bind to the target protein, and a ligand to recruit the E3 ubiquitin ligase. Before we dive into how PROTACs mediate this degradation mechanism, it is crucial to understand the underlying biological pathway that makes this all possible.

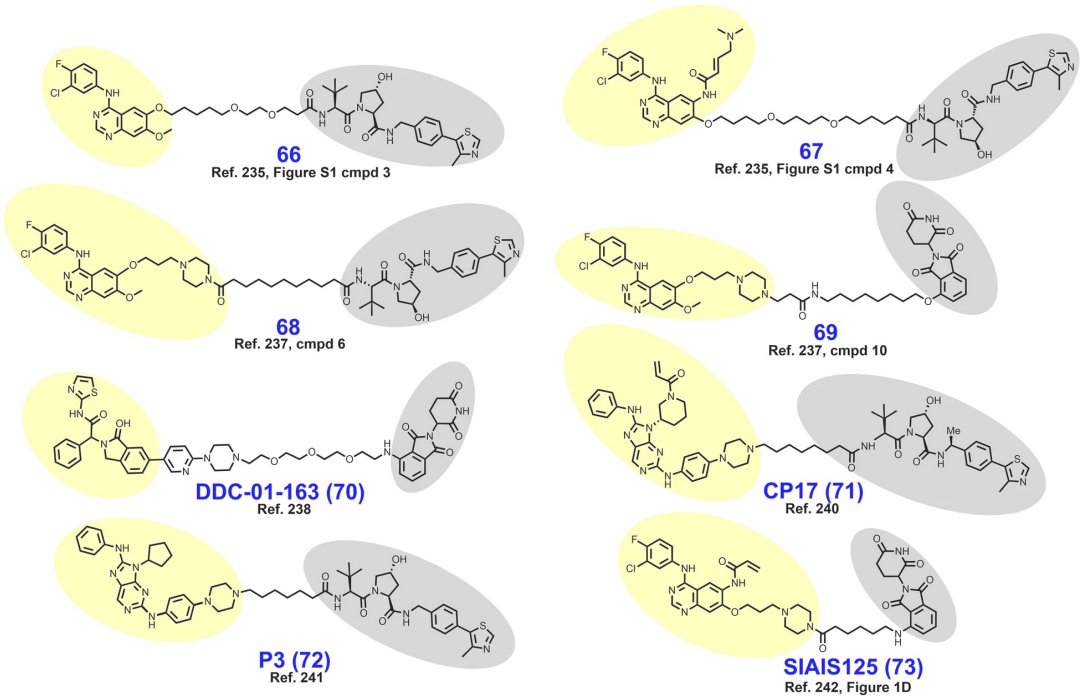


Figure 1: Molecular structure of PROTACs molecules designed to inhibit epidermal growth factor receptor (EGFR). The

PROTAC linker connects the EGFR ligand and E3 ligase which are highlighted in yellow and gray, respectively [1].

1.1.1 Ubiquitin Proteasome System

The ubiquitin proteasome system, or UPS for short, is a crucial cellular maintenance mechanism. Ubiquitin-dependent proteolysis is a three-step process which involves ubiquitin-activating enzymes (E1), ubiquitin-conjugate enzymes (E2), and ubiquitin-protein ligases (E3). In general, E1 activates ubiquitin, priming it for transfer to E2 which interacts with E3 at which point E3 ligases are responsible for binding of the target protein substrate for subsequent ubiquitination by E2. Once the protein is tagged with a polyubiquitin chain, it is recognized by the proteasome, a large protease complex that degrades that protein into peptides.

Figure 2: The ubiquitin proteasome system is one of the cell's internal degradation mechanism crucial for targetting dysfunctional proteins. Naturally this opens up opportunities to leverage this in a therapeutic context [2].

1.1.2 Connection to PROTACs

The realization that the UPS could be leveraged for therapeutic purposes was initially made through early studies of viruses and plants. The underlying idea involves design of small molecules capable of recruiting the E3 ligase and inducing degradation of a protein of interest (POI). This general idea naturally extended itself to the case of PROTACs. Together, the POI ligand, linker, and E3 ligase ligand make up the PROTAC complex responsible for protein degradation. Note that the presence of two ligands enables simultaneous recruitment of the E3 ligase and POI, hence its heterobifunctionality property.

Furthermore, after the POI is degraded by the proteasome, PROTACs can disassociate and continue to induce further degradation, enabling low concentrations to be efficacious. This catalytic mechanism of action and event-drive pharmacology prevents PROTACs from suffering the same limitations as conventional therapeutic strategies such as drug resistance and off-target effects.

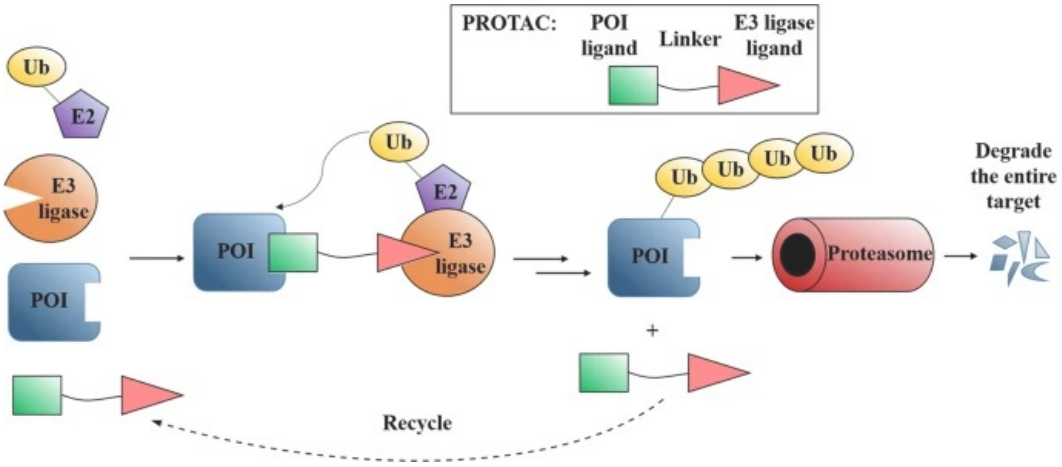


Figure 3: The mechanism of action of PROTACs center around the UPS. In a heterobifunctional manner, recruiting both a target protein of interest and an E3 ligase, PROTACs are able to promote protein degradation in diseases [3].

1.3 Molecular Glues

It is worth noting that PROTACs are not the only TPD method. Another major class which which also leverages the UPS to elicit degradation are molecular glues. As implied by its name, molecular glues stabilize protein-protein interactions between the target protein and E3 ligase. Notice how this is different than PROTACs which consists of two separate binding ligands connected by a linker. Consequently, molecular glues may be less sterically hindered without the need for a linker. However, identifying binding sites to induce new protein-protein interactions is typically harder than accomodating for existing ligands, as in the case for PROTACs, making it harder to design molecular glues.

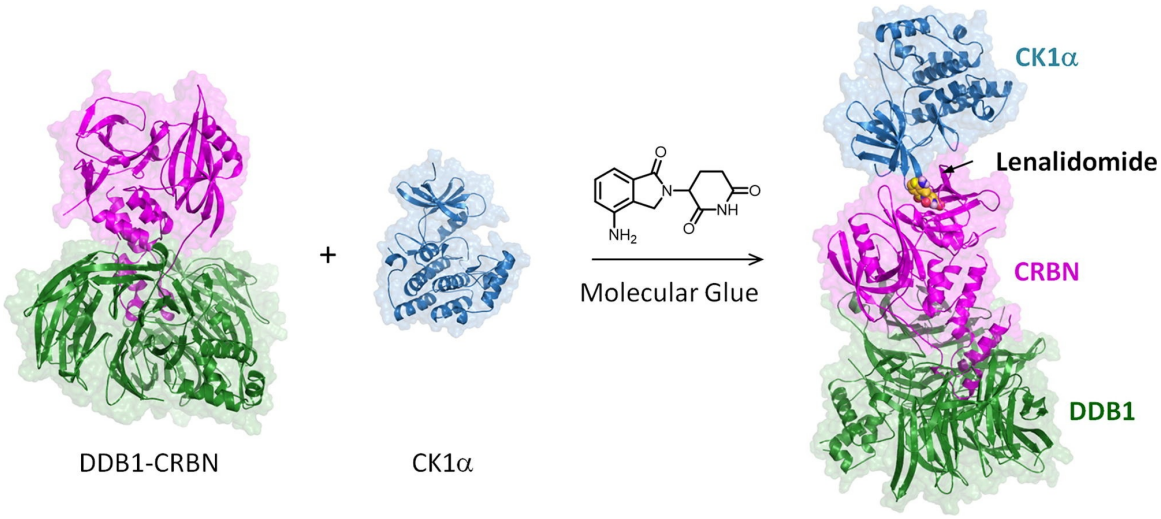


Figure 4: Lenalidomide is a molecular glue which mediates the interaction between CRBN, an E3 ligase, and CK1α, resulting in subsequent ubiquitination. [4]

1.4 How can we leverage machine learning?

As a novel and promising technique, PROTACs have demonstrated positive clinical results thus far. However, much of the clinical validation has been against classically drugged targets. In order for PROTACs to reach their full potential, their clinical efficacy against novel or hard to reach targets must be demonstrated. Consequently, there has been growing research in designing PROTAC molecules capable of elicting an effective response. However, much of the current work is empirical and requires extensive trial-and-error processes. Machine learning could potentially revolutionize this. By correlating molecular structure with physiochemical properties and biological activity, we could potentially streamline the discovery process, significantly reducing the time and cost associated with validation. With that in mind, let's jump into this tutorial to predict efficacy of PROTAC degraders!

For a more in-depth dive into PROTACs, ubiquitin proteasome system, and targeted protein degradation, readers are referred to [5] and [6].

2. Data extraction

Before we proceed, let's install deepchem into our colab environment.

In [1]:
```
!pip install deepchem
```

```
Collecting deepchem
  Downloading deepchem-2.8.0-py3-none-any.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 5.8 MB/s eta 0:00:00
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.4.2)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from deepchem) (2.0.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.2.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.12.1)
Requirement already satisfied: scipy>=1.10.1 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.11.4)
Collecting rdkit (from deepchem)
  Downloading rdkit-2023.9.6-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 34.9/34.9 MB 12.0 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem) (
2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem)
(2024.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from rdkit->deepchem) (9.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n->deepchem) (3.5.0)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->deep
chem) (1.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas->deepchem) (1.16.0)
Installing collected packages: rdkit, deepchem
Successfully installed deepchem-2.8.0 rdkit-2023.9.6
```

Now let's download this dataset on PROTACs, curated by [7], which includes 3270 PROTACs.

In [67]:
```
# Python library imports
import os

# DeepChem and scientific library imports
```

```
import deepchem as dc
import rdkit
from rdkit import Chem
from rdkit.Chem import Draw

# Third-party library imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [3]:
```
os.system('wget https://deepchemdata.s3.us-west-1.amazonaws.com/datasets/protac_10_06_24.csv')
```

Out[3]:
```
0
```

In [4]:
```
protac_db = pd.read_csv('protac_10_06_24.csv')
```

Note that there exists a many-to-many mapping between PROTAC compounds and target proteins. A single PROTAC compound can be designed to target multiple proteins, and conversely, multiple PROTAC compounds can be developed to target the same protein. This many-to-many relationship allows for greater flexibility and adaptability in the design and application of PROTACs.

In [5]:
```
print('''In this dataset, there are {} unique PROTAC compounds, targeting {} unique proteins for a total of {} 
                                                                                    len(protac_db['Target'].uniq

protac_db
```

```
In this dataset, there are 3270 unique PROTAC compounds, targeting 323 unique proteins for a total of 5388 combi
nations
```

Out[5]:

	Compound ID	Uniprot	Target	E3 ligase	PDB	Name	Smiles	DC (n
0	1	Q9NPI1	BRD7	VHL	NaN	NaN	COC1=CC(C2=CN(C)C(=O)C3=CN=CC=C23)=CC(OC)=C1CN...	N
1	1	Q9H8M2	BRD9	VHL	NaN	NaN	COC1=CC(C2=CN(C)C(=O)C3=CN=CC=C23)=CC(OC)=C1CN...	N
2	2	Q9NPI1	BRD7	VHL	NaN	NaN	COC1=CC(C2=CN(C)C(=O)C3=CN=CC=C23)=CC(OC)=C1CN...	N
3	2	Q9H8M2	BRD9	VHL	NaN	NaN	COC1=CC(C2=CN(C)C(=O)C3=CN=CC=C23)=CC(OC)=C1CN...	N
4	3	Q9H8M2	BRD9	CRBN	NaN	NaN	COC1=CC(C2=CN(C)C(=O)C3=CN=CC=C23)=CC(OC)=C1CN...	N
...	...	...	...	...	...	...	...	
5383	3266	O60885	BRD4	FEM1B	NaN	NaN	CC1=C(C)C2=C(S1)N1C(C)=NN=C1[C@H](CC(=O)NCCOCC...	16
5384	3267	NaN	BCR-ABL	FEM1B	NaN	NaN	CC1=NC(NC2=NC=C(C(=O)NC3=C(C)C=CC=C3Cl)S2)=CC(...	N
5385	3268	NaN	BCR-ABL	FEM1B	NaN	NaN	CC1=NC(NC2=NC=C(C(=O)NC3=C(C)C=CC=C3Cl)S2)=CC(...	N
5386	3269	P03372	ER	CRBN	NaN	ARV-471	O=C1CC[C@H](N2CC3=CC(N4CCN(CC5CCN(C6=CC=C([C@@...	
5387	3270	P10275	AR	CRBN	NaN	ARV-110	N#CC1=CC=C(O[C@H]2CC[C@H](NC(=O)C3=CC=C(N4CCC(...	

5388 rows × 89 columns

Taking a closer look at the dataset, each PROTAC compound has a SMILEs representation along with its target protein of interest and E3 ligase. For reference, here is an example:

In [14]:
```
example = protac_db.iloc[0]
print('''Here is the SMILEs representation of a PROTAC compound: {}
designed to target {} protein through ubiquitination by {} E3 ligase.'''.format(example['Smiles'], example['Tar
```

Here is the SMILEs representation of a PROTAC compound: COC1=CC(C2=CN(C)C(=O)C3=CN=CC=C23)=CC(OC)=C1CN1CCN(CCOCCOCC(=O)N[C@H](C(=O)N2C[C@H](O)C[C@H]2C(=O)NCC2=CC=C(C3=C(C)N=CS3)C=C2)C(C)(C)C)CC1
designed to target BRD7 protein through ubiquitination by VHL E3 ligase.

In [ ]:
```
protac_db.columns
```

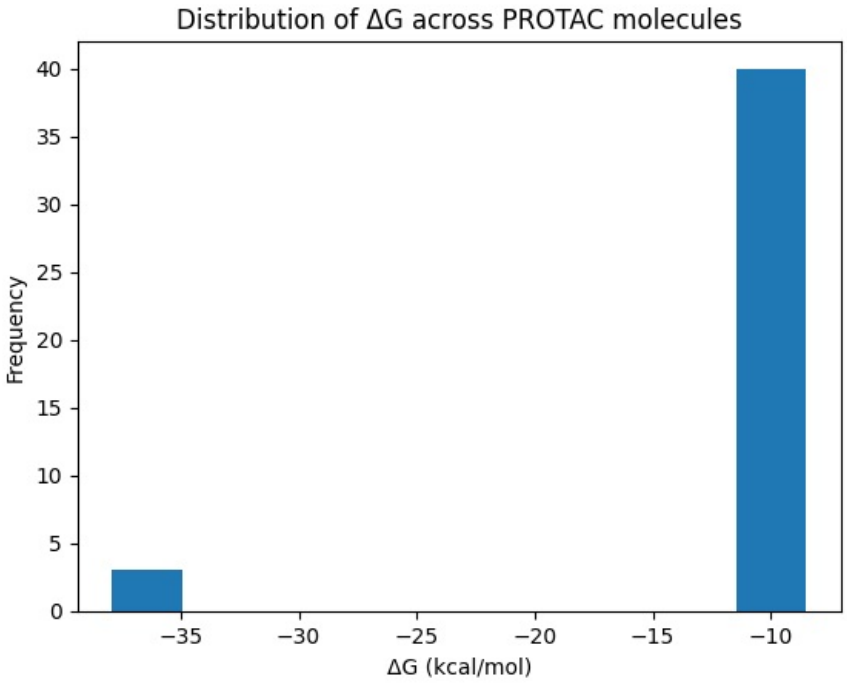
Out[ ]:
```
Index(['Compound ID', 'Uniprot', 'Target', 'E3 ligase', 'PDB', 'Name',
       'Smiles', 'DC50 (nM)', 'Dmax (%)', 'Assay (DC50/Dmax)',
       'Percent degradation (%)', 'Assay (Percent degradation)',
       'IC50 (nM, Protac to Target)', 'Assay (Protac to Target, IC50)',
       'EC50 (nM, Protac to Target)', 'Assay (Protac to Target, EC50)',
       'Kd (nM, Protac to Target)', 'Assay (Protac to Target, Kd)',
       'Ki (nM, Protac to Target)', 'Assay (Protac to Target, Ki)',
       'delta G (kcal/mol, Protac to Target)',
       'delta H (kcal/mol, Protac to Target)',
       '-T*delta S (kcal/mol, Protac to Target)',
       'Assay (Protac to Target, G/H/-TS)', 'kon (1/Ms, Protac to Target)',
       'koff (1/s, Protac to Target)', 't1/2 (s, Protac to Target)',
       'Assay (Protac to Target, kon/koff/t1/2)', 'IC50 (nM, Protac to E3)',
       'Assay (Protac to E3, IC50)', 'EC50 (nM, Protac to E3)',
       'Assay (Protac to E3, EC50)', 'Kd (nM, Protac to E3)',
       'Assay (Protac to E3, Kd)', 'Ki (nM, Protac to E3)',
       'Assay (Protac to E3, Ki)', 'delta G (kcal/mol, Protac to E3)',
       'delta H (kcal/mol, Protac to E3)',
       '-T*delta S (kcal/mol, Protac to E3)', 'Assay (Protac to E3, G/H/-TS)',
       'kon (1/Ms, Protac to E3)', 'koff (1/s, Protac to E3)',
       't1/2 (s, Protac to E3)', 'Assay (Protac to E3, kon/koff/t1/2)',
       'IC50 (nM, Ternary complex)', 'Assay (Ternary complex, IC50)',
       'EC50 (nM, Ternary complex)', 'Assay (Ternary complex, EC50)',
       'Kd (nM, Ternary complex)', 'Assay (Ternary complex, Kd)',
       'Ki (nM, Ternary complex)', 'Assay (Ternary complex, Ki)',
       'delta G (kcal/mol, Ternary complex)',
       'delta H (kcal/mol, Ternary complex)',
       '-T*delta S (kcal/mol, Ternary complex)',
       'Assay (Ternary complex, G/H/-TS)', 'kon (1/Ms, Ternary complex)',
       'koff (1/s, Ternary complex)', 't1/2 (s, Ternary complex)',
       'Assay (Ternary complex, kon/koff/t1/2)',
       'IC50 (nM, Cellular activities)', 'Assay (Cellular activities, IC50)',
       'EC50 (nM, Cellular activities)', 'Assay (Cellular activities, EC50)',
       'GI50 (nM, Cellular activities)', 'Assay (Cellular activities, GI50)',
       'ED50 (nM, Cellular activities)', 'Assay (Cellular activities, ED50)',
       'GR50 (nM, Cellular activities)', 'Assay (Cellular activities, GR50)',
       'PAMPA Papp (nm/s, Permeability)', 'Assay (Permeability, PAMPA Papp)',
       'Caco-2 A2B Papp (nm/s, Permeability)',
       'Assay (Permeability, Caco-2 A2B Papp)',
       'Caco-2 B2A Papp (nm/s, Permeability)',
       'Assay (Permeability, Caco-2 B2A Papp)', 'Article DOI',
       'Molecular Weight', 'Exact Mass', 'XLogP3', 'Heavy Atom Count',
       'Ring Count', 'Hydrogen Bond Acceptor Count',
       'Hydrogen Bond Donor Count', 'Rotatable Bond Count',
       'Topological Polar Surface Area', 'Molecular Formula', 'InChI',
       'InChI Key'],
      dtype='object')
```

In general, the PROTAC-DB dataset contains information for a variety of different physiochemical and biochemical properties of PROTAC structures. Several useful ones to point out are
ΔG
, which describes the spontaneity of a chemical reaction,
K_d
which measures the concentration of a ligand to achieve 50% occupancy of the protein binding sites, and
$XLogP3$
which measures a compound's solubility, an indication of its absorption and distribution characteristics.

Before we proceed, let's plot the distribution of each of these properties to get a better sense of our PROTAC dataset starting with ΔG values.

In [ ]:
```
delta_G = protac_db['delta G (kcal/mol, Protac to E3)']
delta_G = delta_G.dropna()
delta_G = delta_G.astype(float)
plt.hist(delta_G, bins=10)
plt.xlabel('ΔG (kcal/mol)')
plt.ylabel('Frequency')
plt.title(f'Distribution of ΔG across PROTAC molecules')
plt.plot()
```

Out[ ]:
```
[]
```



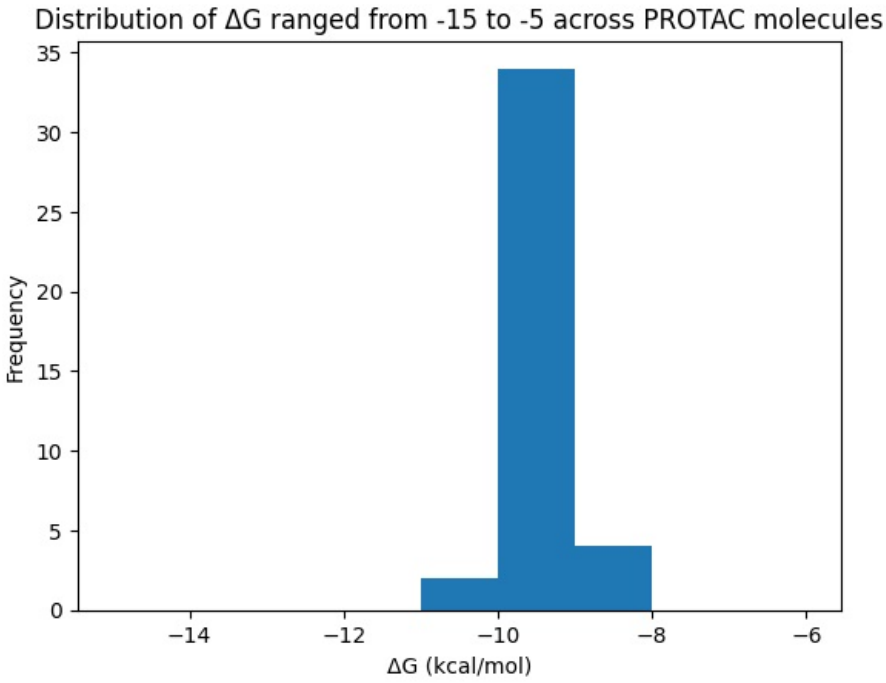
Let's take a closer look at the distribution of PROTAC molecules around the -10 range of ΔG values.

In [34]:
```
delta_G = protac_db['delta G (kcal/mol, Protac to E3)']
delta_G = delta_G.dropna()
delta_G = delta_G.astype(float)

x_min = -15
x_max = -5
bin_size = 1
bins = np.arange(x_min, x_max, bin_size)

plt.hist(delta_G, bins=bins)
plt.xlabel('ΔG (kcal/mol)')
plt.ylabel('Frequency')
plt.title('Distribution of ΔG ranged from -15 to -5 across PROTAC molecules')
plt.plot()
```

Out[34]:
```
[]
```

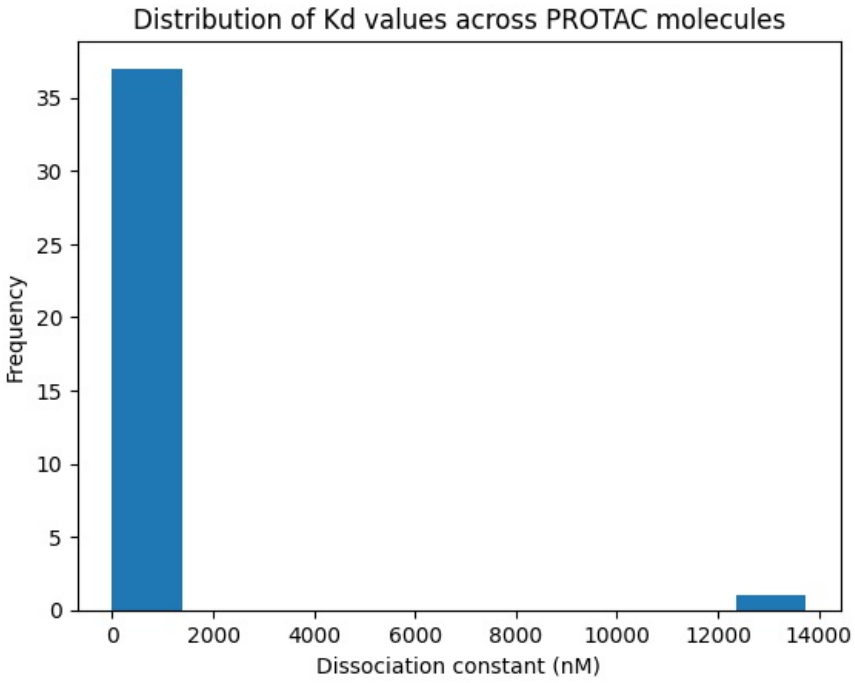


There does not appear to be a lot of information on the spontaneity of PROTAC reactions but it is worth noting that the ones with recorded ΔGs appear energetically favorable, as expected.

Let's now take a look at the
K_d
values.

In [ ]:
```
kd_data = protac_db['Kd (nM, Ternary complex)']
kd_data = kd_data.dropna()
kd_data = kd_data[~kd_data.str.contains('/')]
kd_data = kd_data.astype(float)
plt.hist(kd_data)
plt.xlabel('Dissociation constant (nM)')
plt.ylabel('Frequency')
plt.title('Distribution of Kd values across PROTAC molecules')
plt.plot()
```

Out[ ]:
```
[]
```



Similar to ΔG values, there does not appear to be a lot of information on the affinity of formed PROTAC complexes. Since the range is so large, let's plot a second histogram focused on the PROTACs with low
K_d
.

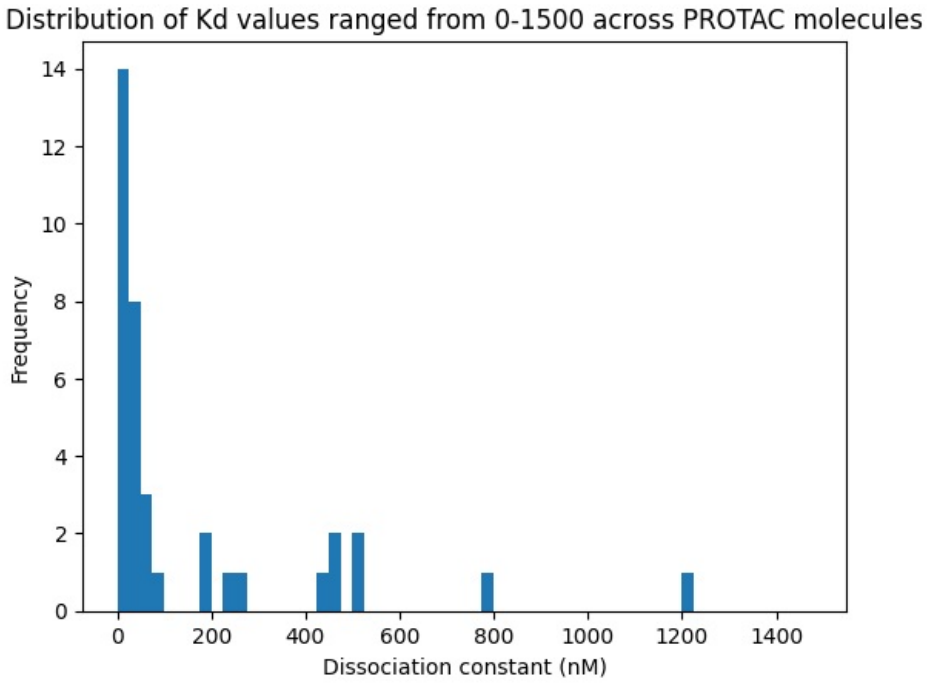
In [31]:
```
kd_data = protac_db['Kd (nM, Ternary complex)']
kd_data = kd_data.dropna()
kd_data = kd_data[~kd_data.str.contains('/')]
kd_data = kd_data.astype(float)

# limit range
x_max = 1500
x_min = 0
bin_size = 25
bins = np.arange(x_min, x_max, bin_size)

plt.hist(kd_data, bins=bins)

plt.xlabel('Dissociation constant (nM)')
plt.ylabel('Frequency')
plt.title('Distribution of Kd values ranged from 0-1500 across PROTAC molecules')
plt.plot()
```

Out[31]:
```
[]
```



The improved resolution of values illustrates a much cleaner distribution of K_d values. We do see that a number of them have low, favorable K_d values indicating that the PROTAC linker can form a strong connection with the E3 ligase and target protein.

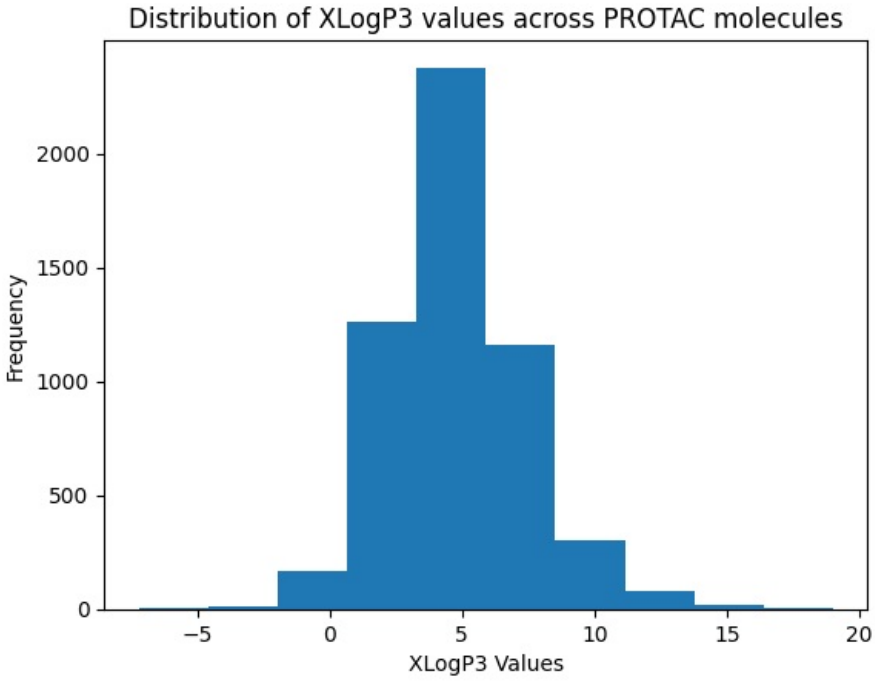
Let's now take a look at XLogP3 values. Note that this is slightly different than the typical LogP partition coefficient. Recall that LogP is defined

$$LogP = log(\frac{[solute]_{oct}}{[solute]_{H_2O}})$$

In other words, LogP is the measured ratio of the concentration of a compound in the organic phase to the its concentration in the aqueous phase, measuring the compound's solubility. XLogP3 is a knowledge-based method for calculating the partition coefficient by accounting for the molecular structure, presence of functional groups, and bonding [8]. Both properties estimate a compound's liphophilicity, giving insight into how a compound may behave in biological systems.

In [ ]:
```
plt.hist(protac_db['XLogP3'])
plt.xlabel('XLogP3 Values')
plt.ylabel('Frequency')
plt.title('Distribution of XLogP3 values across PROTAC molecules')
plt.plot()
```

Out[ ]:
```
[]
```



All PROTAC compounds have a recorded XLogP3 value. The distribution looks normally distributed with few molecules with extreme logP profiles.

Now, let's take a look at the PROTAC degradation properties. "DC50 (nM)" and "Dmax (%)" represent the half maximal degradation concentration and maximal degradation of the target protein of interest, respectively. Let's take a quick look at their distributions.

Let's first do some data cleaning

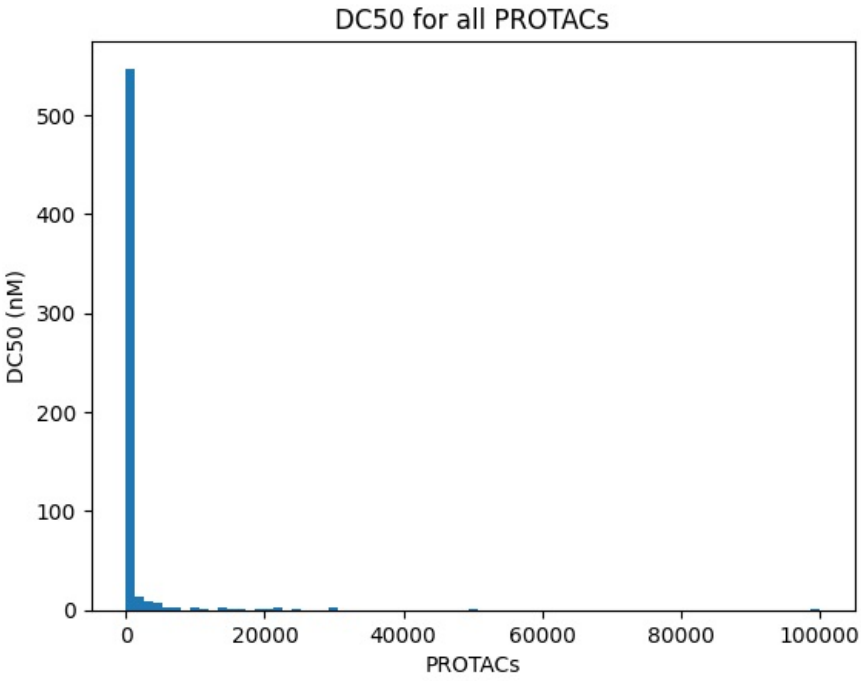
In [5]:
```
# Let's first drop all the NaN values
raw_dc50 = protac_db['DC50 (nM)']
raw_dc50 = raw_dc50.dropna()
```

Notice that the values are all in string format with non-numerical characters such as '<', '/', and '>'. For the time being, let's remove these values.

In [6]:
```
raw_dc50 = raw_dc50[~raw_dc50.str.contains('<|>|/|~|-')]
raw_dc50 = raw_dc50.astype(float)
```

In [137…
```
plt.hist(raw_dc50.values, bins=75)
plt.xlabel('PROTACs')
plt.ylabel('DC50 (nM)')
plt.title('DC50 for all PROTACs')
plt.plot()
```

Out[137… []

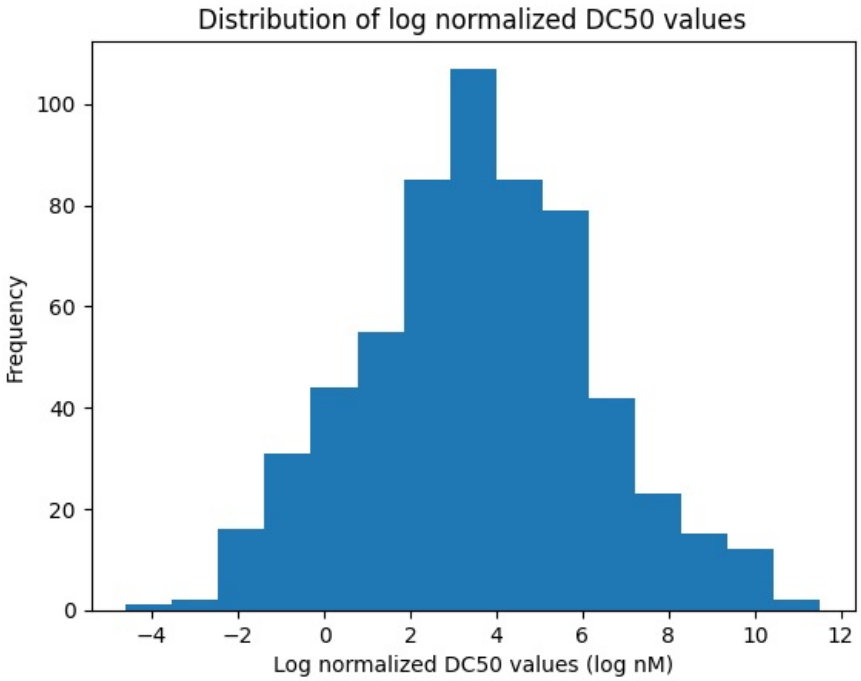


The distribution is certainly skewed and has a few outliers. Let's log normalize.

In [7]:
```
lognorm_dc50 = np.log(raw_dc50)
```

In [15]:
```
plt.hist(lognorm_dc50, bins=15)
plt.xlabel('Log normalized DC50 values (log nM)')
plt.ylabel('Frequency')
plt.title('Distribution of log normalized DC50 values')
plt.plot()
```

Out[15]:
```
[]
```



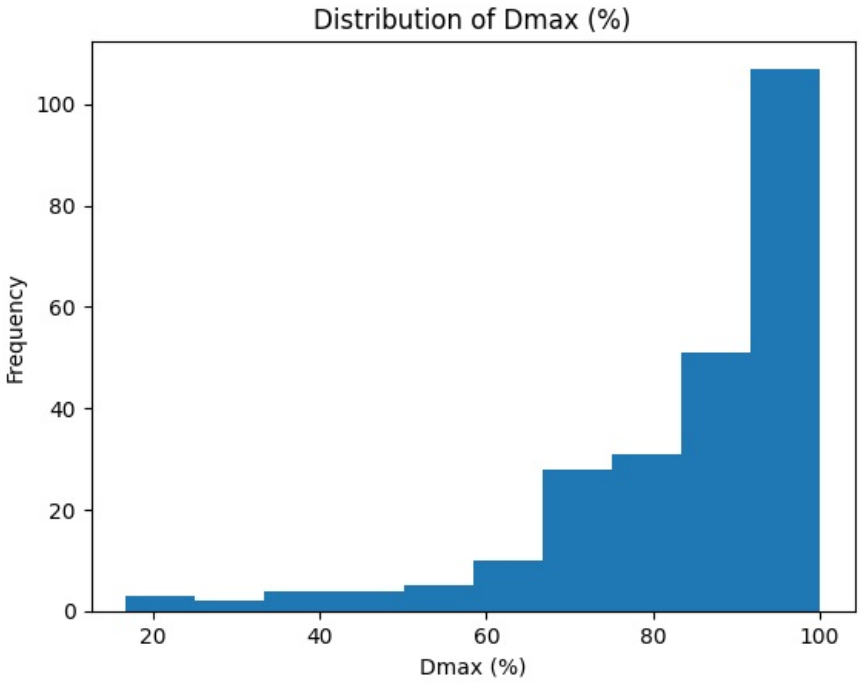
Now, let's take a look at Dmax percentage which represents the maximal degradation a PROTAC can elicit relative to the total activity of the target protein of interest [7].

In [28]:
```
# Using the same row indices as our cleaned DC50 data
dmax = protac_db.iloc[lognorm_dc50.index]['Dmax (%)']
```

In [ ]:
```
# Following the same data cleaning procedure:
dmax = dmax.dropna()
dmax = dmax[~dmax.str.contains('<|>|/|~|-')]
dmax = dmax.astype(float)
```

In [ ]:
```
plt.hist(dmax.values, bins=10)
plt.xlabel('Dmax (%)')
plt.ylabel('Frequency')
plt.title('Distribution of Dmax (%)')
plt.plot()
```

Out[ ]:
```
[]
```



Notice that Dmax is represented as a percentage. For now, let's continue with regressing on DC50. We are now ready to featurize!

In [8]:
```
# Let's predict DC50 properties for the time being
cleaned_data = protac_db.iloc[lognorm_dc50.index]
print('There are {} PROTAC samples.'.format(cleaned_data.shape[0]))
```

```
There are 599 PROTAC samples.
```

In [9]:
```
protac_smiles = cleaned_data['Smiles']
dc_vals = lognorm_dc50
```

3. Featurization

Let's featurize using CircularFingerprint which is incorporated in DeepChem! CircularFingerprint is a common featurizer for molecules that encode local information about each atom and their neighborhood. For more information, the reader is refered to [9].

In [10]:
```
from rdkit import Chem
featurizer = dc.feat.CircularFingerprint(radius=4, chiral=True)
```

In [11]:
```
features = featurizer.featurize(protac_smiles)
```

In [12]:
```
# Let's initialize our dataset and perform splits
dataset = dc.data.NumpyDataset(X = features, y = dc_vals, ids=list(protac_smiles))
```

In [65]:
```
splitter = dc.splits.RandomSplitter()
train_random, val_random, test_random = splitter.train_valid_test_split(dataset, seed=42)
```

Along with a random split, let's also use a scaffold split which ensures that the split contain a structurally diverse array of compounds. Scaffold split groups molecules according to presence of rings, linkers, combinations of rings and linkers, as well as atomic properties. In general, scaffold splits are a good way of ensuring generalizability of our models.

In [14]:
```
# Scaffold split
splitter = dc.splits.ScaffoldSplitter()
train_scaffold, val_scaffold, test_scaffold = splitter.train_valid_test_split(dataset, seed=42)
```

To see the scaffold split in action, let's visualize the chosen compounds across the splits.

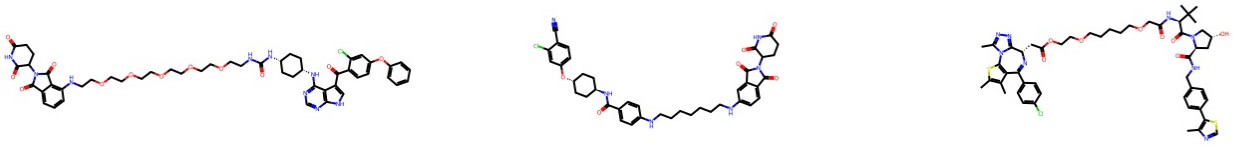
In [59]:
```
print("Three molecules from the train set:")
np.random.seed(42)
indices = np.random.choice(len(train_scaffold), size=3, replace=False)

smiles = [train_scaffold.ids[index] for index in indices]
mols = [Chem.MolFromSmiles(smile) for smile in smiles]

Draw.MolsToGridImage(mols, molsPerRow=3, subImgSize=(450, 350))
```

```
Three molecules from the train set:
```

Out[59]:



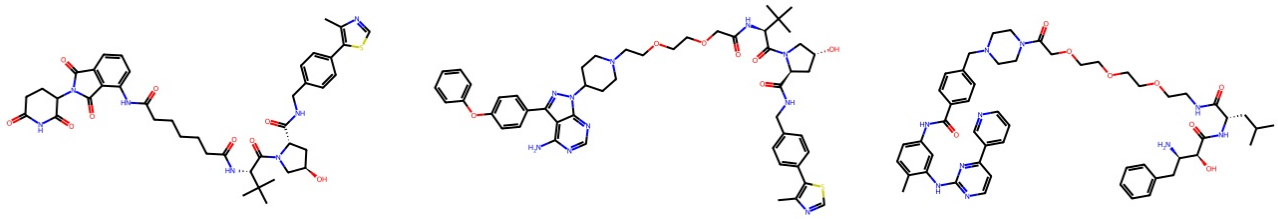
In [60]:
```
print("Three molecules from the validation set:")
indices = np.random.choice(len(val_scaffold), size=3, replace=False)

smiles = [val_scaffold.ids[index] for index in indices]
mols = [Chem.MolFromSmiles(smile) for smile in smiles]

Draw.MolsToGridImage(mols, molsPerRow=3, subImgSize=(450, 350))
```

```
Three molecules from the validation set:
```

Out[60]:



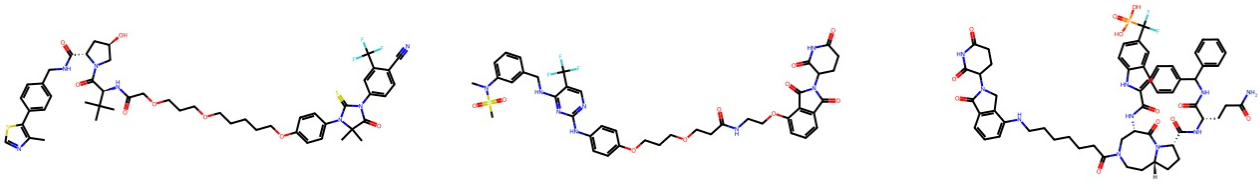
In [61]:
```
print("Five molecules from the test set:")
indices = np.random.choice(len(test_scaffold), size=3, replace=False)

smiles = [test_scaffold.ids[index] for index in indices]
mols = [Chem.MolFromSmiles(smile) for smile in smiles]

Draw.MolsToGridImage(mols, molsPerRow=3, subImgSize=(450, 350))
```

```
Five molecules from the test set:
```

Out[61]:



There are certainly functional group differences spread throughout the splits. Notice the presence of the nitrile group in the train set, amine group in the validation set, as well as the sulfonamide group in the test set.

Additionally, notice the structural and conformational differences among the various data splits. It will be interesting to see how well our model generalizes.

4. Model deployment

We have successfully generated our train and test datasets. Let's now create a simple MLP model to predict PROTAC degradation properties!

In [93]:
```
# Initialize a 2 layer FCN
import torch
import torch.nn as nn

n_tasks = 1
n_features = train_random.X.shape[1]
layer_sizes = [256, 32, 1]
dropouts = [0.0, 0.2, 0]
activation_fns = [nn.ReLU(), nn.ReLU(), nn.Identity()]

optimizer = dc.models.optimizers.Adam()

# Let's log every train epoch
batch_size = 10
log_freq = int(len(train_random) / batch_size +1)

# L2 loss is default
protac_model_random = dc.models.MultitaskRegressor(n_tasks, n_features, layer_sizes, dropouts=dropouts, activati
                                                   optimizer=optimizer, batch_size=10, log_frequency=log_freq)
protac_model_scaffold = dc.models.MultitaskRegressor(n_tasks, n_features, layer_sizes, dropouts=dropouts, activa
                                                   optimizer=optimizer, batch_size=10, log_frequency=log_freq)
```

Let's now wrap everything together to instantiate a DeepChem model! Note that due to the small sample size, a smaller batch size actually helps performance.

In [94]:
```
# protac_model = dc.models.torch_models.TorchModel(protac_model, loss=criterion, optimizer=optimizer, batch_siz
param_count = sum(p.numel() for p in protac_model_random.model.parameters() if p.requires_grad)
print("There are {} trainable parameters".format(param_count))
protac_model_random.model
```

```
There are 532805 trainable parameters
```

Out[94]:
```
PytorchImpl(
  (layers): ModuleList(
    (0): Linear(in_features=2048, out_features=256, bias=True)
    (1): Linear(in_features=256, out_features=32, bias=True)
    (2): Linear(in_features=32, out_features=1, bias=True)
  )
  (output_layer): Linear(in_features=1, out_features=1, bias=True)
  (uncertainty_layer): Linear(in_features=1, out_features=1, bias=True)
)
```

Let's define the validation function to prevent overfitting.

In [95]:
```
train_losses_random = []
val_losses_random = []

train_losses_scaffold = []
val_losses_scaffold = []

metric = [dc.metrics.Metric(dc.metrics.mean_squared_error)]

n_epochs=100
for i in range(n_epochs):
  protac_model_random.fit(train_random, nb_epoch=1, all_losses=train_losses_random)

  protac_model_scaffold.fit(train_scaffold, nb_epoch=1, all_losses=train_losses_scaffold)

  # Validate on every other epoch
  if i % 2 == 0:
    loss = protac_model_random.evaluate(val_random, metrics=metric)
    val_losses_random.append(loss['mean_squared_error'])

    loss = protac_model_scaffold.evaluate(val_scaffold, metrics=metric)
    val_losses_scaffold.append(loss['mean_squared_error'])
```

We can easily look at how the training went through plotting the recorded losses.

In [99]:
```
train_steps = [(i+1)*log_freq for i in range(len(train_losses_random))]
val_steps = [(i+1)*(log_freq*2) for i in range(len(val_losses_random))]

fig, ax = plt.subplots(1, 2, figsize=(12, 5))

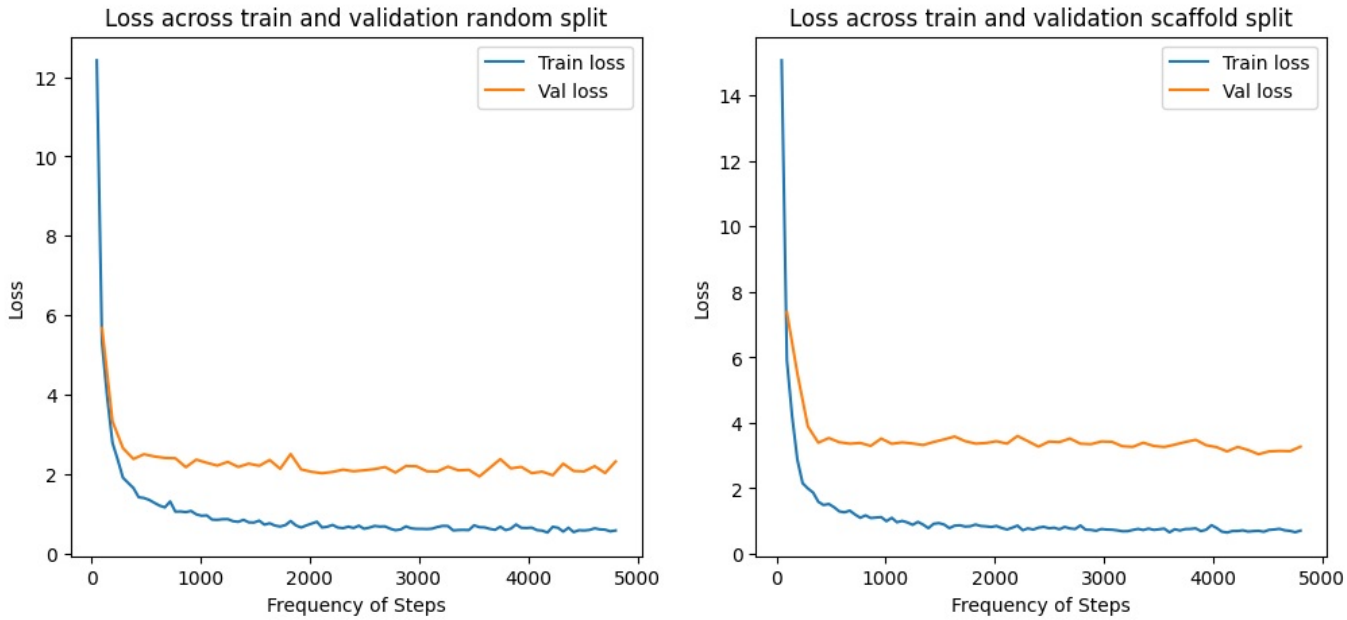
ax[0].plot(train_steps, train_losses_random, label='Train loss')
ax[0].plot(val_steps, val_losses_random, label='Val loss')
```

```
ax[0].legend()
ax[0].set_xlabel('Frequency of Steps')
ax[0].set_ylabel('Loss')
ax[0].set_title('Loss across train and validation random split')

ax[1].plot(train_steps, train_losses_scaffold, label='Train loss')
ax[1].plot(val_steps, val_losses_scaffold, label='Val loss')
ax[1].legend()
ax[1].set_xlabel('Frequency of Steps')
ax[1].set_ylabel('Loss')
ax[1].set_title('Loss across train and validation scaffold split')

plt.plot()
```

Out[99]: []



We can see that the model performs less well on the scaffold validation set which makes sense as the scaffold splits ensures that more validation molecules are out of distribution relative to the train distribution.

Let's now perform some inference on our test set to evaluate our models!

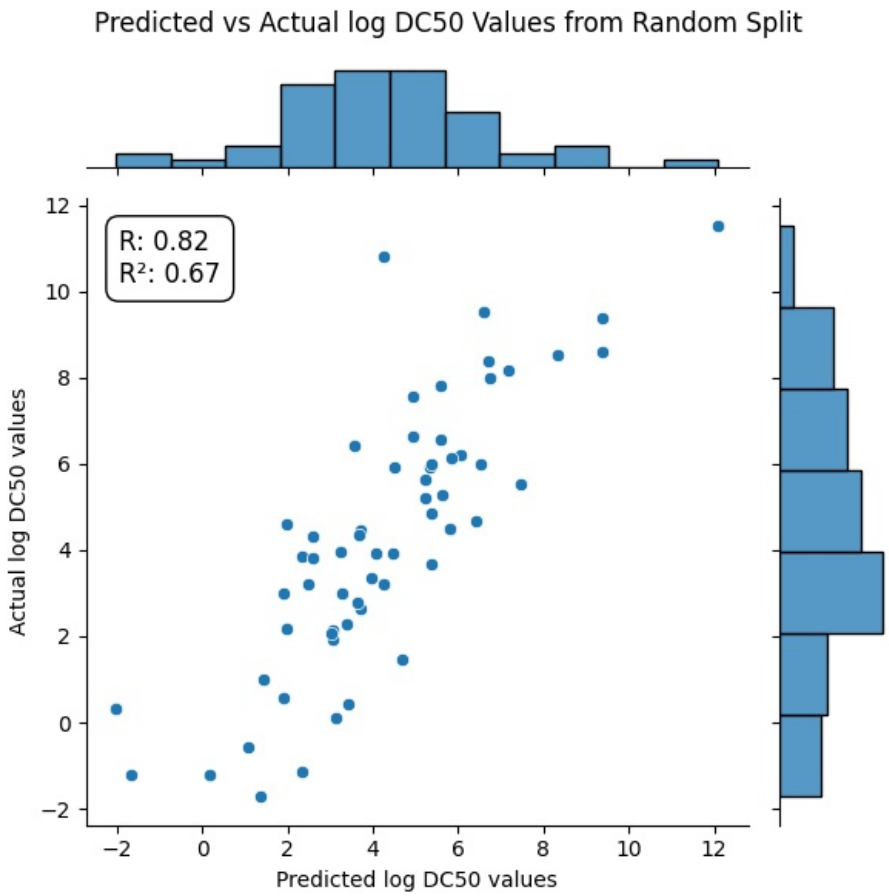
In [114…
```
metrics = [dc.metrics.Metric(dc.metrics.mean_squared_error), dc.metrics.Metric(dc.metrics.pearsonr), dc.metrics
eval_metrics = protac_model_random.evaluate(test_random, metrics)
preds = protac_model_random.predict(test_random)
```

In [115…
```
for k, v in eval_metrics.items():
  print('{}: {}'.format(k, v))
```

```
mean_squared_error: 3.074001339280645
pearsonr: 0.818568671566446
pearson_r2_score: 0.6700546700700561
```

In [132…
```
import seaborn as sns
preds_and_labels = np.concatenate((test_random.y.reshape([60, 1]), preds.reshape([60, 1])), axis=1)
pred_df = pd.DataFrame(preds_and_labels, columns=['Actual log DC50 values', 'Predicted log DC50 values'])
sns.jointplot(pred_df, x='Predicted log DC50 values', y='Actual log DC50 values')
plt.annotate(f"R: {eval_metrics['pearsonr']:.2f}\nR²: {eval_metrics['pearson_r2_score']:.2f}",
             xy=(0.05, 0.95),
             xycoords='axes fraction',
             ha='left',
             va='top',
             fontsize=12,
             bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))
# Set the title
plt.suptitle('Predicted vs Actual log DC50 Values from Random Split')

# Adjust the position of the title to avoid overlap with the plot
plt.tight_layout()
plt.show()
```



The random split appears to do fairly well. Let's see how well our model does on the scaffold split.

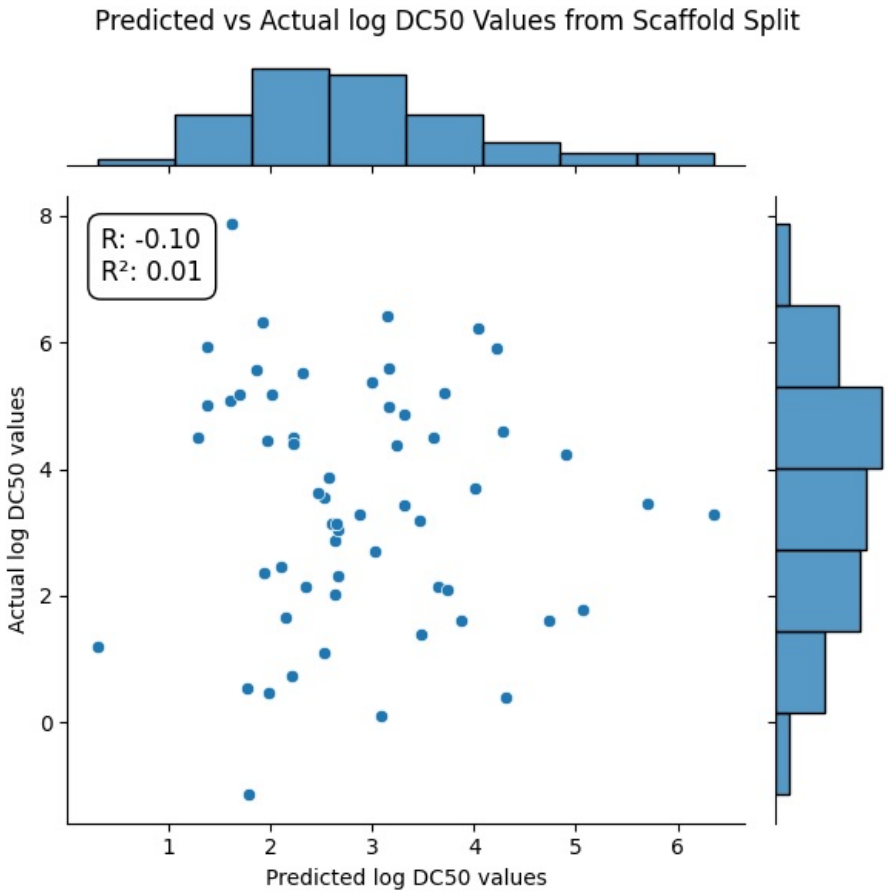
In [133…
```
metrics = [dc.metrics.Metric(dc.metrics.mean_squared_error), dc.metrics.Metric(dc.metrics.pearsonr), dc.metrics
eval_metrics = protac_model_scaffold.evaluate(test_scaffold, metrics)
preds = protac_model_scaffold.predict(test_scaffold)
```

In [134…
```
for k, v in eval_metrics.items():
  print('{}: {}'.format(k, v))
```

```
mean_squared_error: 5.991774828135091
pearsonr: -0.10286796793151554
pearson_r2_score: 0.010581818826359309
```

In [136…
```
import seaborn as sns
preds_and_labels = np.concatenate((test_scaffold.y.reshape([60, 1]), preds.reshape([60, 1])), axis=1)
pred_df = pd.DataFrame(preds_and_labels, columns=['Actual log DC50 values', 'Predicted log DC50 values'])
sns.jointplot(pred_df, x='Predicted log DC50 values', y='Actual log DC50 values')
plt.annotate(f"R: {eval_metrics['pearsonr']:.2f}\nR²: {eval_metrics['pearson_r2_score']:.2f}",
             xy=(0.05, 0.95),
             xycoords='axes fraction',
             ha='left',
             va='top',
             fontsize=12,
             bbox=dict(boxstyle='round,pad=0.5', edgecolor='black', facecolor='white'))
# Set the title
plt.suptitle('Predicted vs Actual log DC50 Values from Scaffold Split')

# Adjust the position of the title to avoid overlap with the plot
plt.tight_layout()
plt.show()
```



The model does significantly worse on the held out scaffold test set which was expected given the simplicity of the model. Developing far more complex models which can generalize out of distribution is a key area of focus in many areas of research from molecule property prediction to computer vision to natural language processing. In general, I hope this tutorial was a informative introduction into the world of PROTACs. Follow along as we explore how we can think about PROTAC design in the next tutorial!

5. References

[1] Kelm, J.M., Pandey, D.S., Malin, E. et al. PROTAC'ing oncoproteins: targeted protein degradation for cancer therapy. Mol Cancer. 2023, 22, 62. https://doi.org/10.1186/s12943-022-01707-5

[2] Tu, Y., Chen, C., Pan, J., Xu, J., Zhou, Z. G., & Wang, C. Y. The Ubiquitin Proteasome Pathway (UPP) in the regulation of cell cycle control and DNA damage repair and its implication in tumorigenesis. International journal of clinical and experimental pathology. 2012, 5, 8.

[3] Sun, X., Gao, H., Yang, Y. et al. PROTACs: great opportunities for academia and industry. Sig Transduct Target Ther. 2019, 4, 64. https://doi.org/10.1038/s41392-019-0101-6

[4] Che Y, Gilbert AM, Shanmugasundaram V, Noe MC. Inducing protein-protein interactions with molecular glues. Bioorg Med Chem Lett. 2018, 28, 15. https://doi.org/10.1016/j.bmcl.2018.04.046.

[5] Békés, M., Langley, D.R. & Crews, C.M. PROTAC targeted protein degraders: the past is prologue. Nat Rev Drug Discov. 2022, 21, 181–200. https://doi.org/10.1038/s41573-021-00371-6

[6] Liu, Z., Hu, M., Yang, Y. et al. An overview of PROTACs: a promising drug discovery paradigm. Mol Biomed. 2022, 3 (46). https://doi.org/10.1186/s43556-022-00112-0

[7] Gaoqi Weng, Xuanyan Cai, Dongsheng Cao, Hongyan Du, Chao Shen, Yafeng Deng, Qiaojun He, Bo Yang, Dan Li, Tingjun Hou, PROTAC-DB 2.0: an updated database of PROTACs, Nucleic Acids Research. 2023, 51 (D1), Pages D1367–D1372, https://doi.org/10.1093/nar/gkac946

[8] Cheng T, Zhao Y, Li X, Lin F, Xu Y, Zhang X, Li Y, Wang R, Lai L. Computation of octanol-water partition coefficients by guiding an additive model with knowledge. J Chem Inf Model. 2007, 47 (6), 2140-8. https://doi.org/10.1021/ci700257y.

[9] Glem RC, Bender A, Arnby CH, Carlsson L, Boyer S, Smith J. Circular fingerprints: flexible molecular descriptors with applications from physical chemistry to ADME. IDrugs. 2006, 9 (3).

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Druggability Assessment with Fpocket and Machine Learning

Author: Anamika Yadav

Table of Contents:

- Introduction
  - Understanding Druggability
  - Methods to assess drugabililty
  - Application of Machine Learning in Druggability Assessment
- Practical Application of Machine Learning for Druggability Prediction
  - Building a dataset
  - Fpocket to idneitfy binding pockets
  - ML model to classify the binding pockets

Introduction

In this tutorial, we will explore the concept of druggability and its crucial role in identifying successful drug targets. We will then apply a machine learning model to classify drug targets as highly druggable or less druggable, helping us assess the potential of a protein to be effectively targeted by a drug.

Understanding Druggability

Protein Pockets

Protein pockets, also known as binding pockets or active sites, are regions on the surface of a protein where small molecules, such as drugs, can bind. These pockets are formed by the three-dimensional folding of the protein. Protein pockets are characterized by specific amino acids lining the pocket that interact with ligands through various forces, such as hydrogen bonds, hydrophobic interactions, van der Waals forces, and ionic bonds. These pockets are crucial as they can be active sites where catalytic activity occurs, or allosteric sites where binding modulates the protein's function without directly involving the active site.

Identifying protein pockets or binding sites on disease-related proteins is essential for selecting targets for new drugs. Once a binding site is known, drugs can be designed to fit precisely into these sites, enhancing their efficacy and reducing side effects. Understanding the binding site helps in modifying drug molecules to increase their affinity and specificity.

Binding sites are central to the concept of druggability, as they are the points of interaction between a drug and its target protein. The characteristics of binding sites, such as their geometric and chemical properties, determine whether a protein can be effectively targeted by a drug. By understanding and analyzing these sites, we can identify druggable targets, design and optimize drugs, and predict the druggability of new proteins, ultimately facilitating the development of effective and safe therapeutic agents.

To learn more about binding sites, check out this additional DeepChem tutorial on the topic: Introduction to Binding Sites

Druggability

Druggability is the measure of whether a biological drug target, like a protein, can be effectively targeted and modulated by a drug to treat a disease. It basically refers to how suitable a protein is for being targeted by a drug. Not all proteins are good drug targets. A druggable protein has certain characteristics that make it possible to design a drug to interact with it effectively. For instance, a druggable protein must have accessible and well-defined binding sites or pockets that can interact with drug molecules.

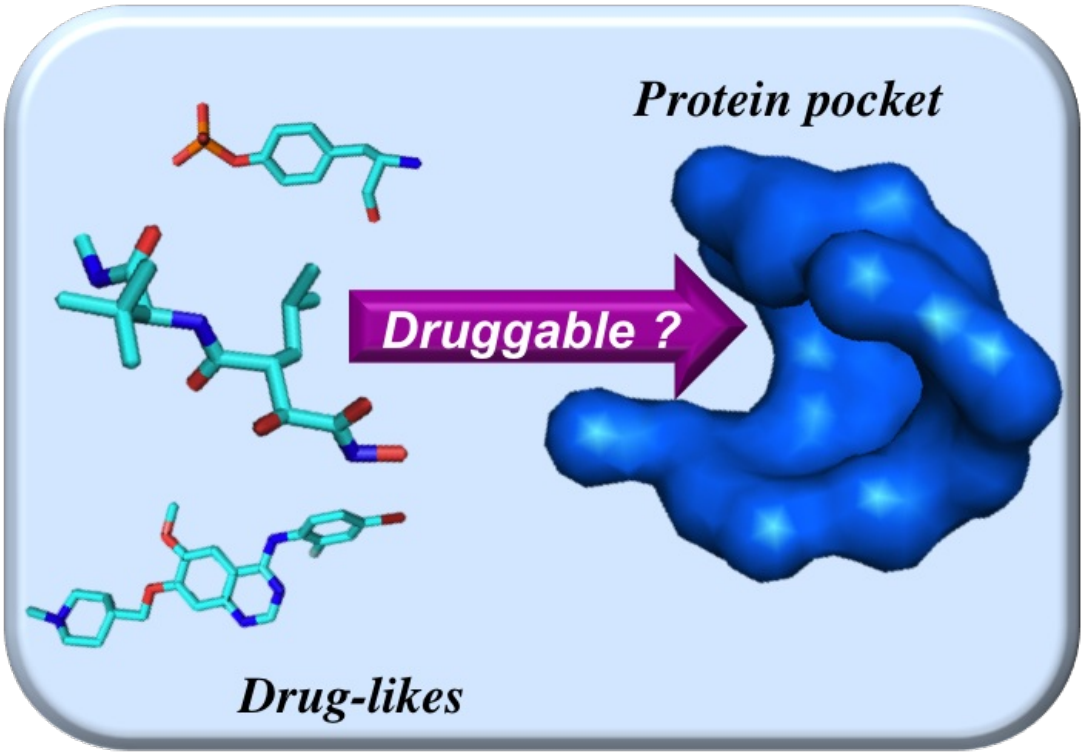


Fig. 1: Druggable pocket correspond to a protein region capable of binding a drug-like molecule.(source)

Properties of a Druggable Target

Structurally, a druggable target must have well-defined binding pockets where potential drugs can bind. These pockets, identified through techniques like X-ray crystallography or computational modeling, should be of suitable size, shape, and chemical composition to accommodate drug-like molecules.

The identification and characterization of binding pockets involve a detailed analysis of their key properties, including volume, hydrophobicity, and the presence of polar residues. The volume of a binding pocket dictates the size of the ligands that can be accommodated, with larger pockets able to bind larger or more complex molecules, offering more points of interaction. However, excessively large pockets can sometimes be less selective, leading to off-target effects. Hydrophobic regions within the binding pocket interact with non-polar parts of drug molecules through van der Waals forces and hydrophobic interactions, crucial for the binding stability of many drugs, particularly those targeting intracellular proteins where the environment is less aqueous. Polar residues within the pocket can form hydrogen bonds and ionic interactions with the drug, which are often key determinants of binding affinity and specificity. The distribution and accessibility of these polar residues are carefully analyzed to optimize drug design.

Another critical aspect of binding pockets is their dynamic behavior and flexibility. Binding pockets are not always static; they can undergo conformational changes upon ligand binding. This dynamic behavior, known as induced fit, allows the pocket to better accommodate different ligands, enhancing binding affinity and specificity. Molecular dynamics simulations are particularly useful in studying these conformational changes, providing insights into how flexible pockets can adapt to various drug molecules. Understanding this flexibility is essential for designing drugs that can bind effectively even as the protein changes shape.

The balance between hydrophobic and hydrophilic areas within the pocket also influences the type of ligands that can bind effectively. Hydrophobic pockets are better suited for non-polar ligands, while hydrophilic or polar pockets favor ligands that can form hydrogen bonds and ionic interactions. The density and distribution of alpha spheres, geometric constructs used to model the cavities within binding pockets, help in understanding the compactness and accessibility of the pocket. A high alpha sphere density typically indicates a well-defined pocket with the potential for strong ligand interactions. Additionally, the surface area of the pocket, particularly the solvent-accessible surface area (SASA), is crucial as it indicates how much of the pocket is exposed and available for binding, providing further insights into the druggability of the target.

Why is Druggability Important?

Before investing a lot of time and money into developing a new drug, scientists want to ensure that the target they are aiming at has a good chance of responding to a drug. If a target is druggable, it means there's a better chance that a drug can bind to it, affect its function, and ultimately help treat the disease. This helps prioritize targets that are more likely to lead to successful drug development.

Despite decades of experimental investigations in the drug discovery domain, about 96% overall failure rate has been recorded in drug development due to the “undruggability” of various identified disease targets and other challenges.

Druggability assessment of a target protein is crucial for several reasons:

1. **Prioritizing "Druggable" Pockets:** Not all regions of a promising target protein are suitable for drug binding. Druggability assessment tools, such as fpocket or SiteMap, are employed to identify pockets on the protein surface that are amenable to drug interaction. These pockets should be accessible, possess favorable physicochemical properties (such as hydrophobicity and the presence of hydrogen bond donors/acceptors), and ideally, should not be essential for the protein's normal function to avoid potential side effects if the pocket is targeted by a drug.
2. **Reducing Risks of Off-Target Effects:** Off-target effects occur when a drug interacts with unintended proteins, leading to adverse side effects. By conducting a thorough druggability assessment, researchers can identify target proteins with minimal risk of such interactions. This involves analyzing the protein’s structure and sequence to detect potential promiscuous binding sites that might interact with a broad range of molecules, thereby increasing the risk of off-target effects.
3. **Predicting Potential Safety Issues:** Druggability assessment can also highlight potential safety concerns related to targeting specific proteins. For instance, if the target protein shares significant structural or sequence similarity with proteins involved in critical biological processes, inhibiting it could lead to unintended consequences. This consideration is essential to avoid disrupting essential functions that could lead to toxicity or other adverse effects .

Methods for Druggability Assessment

Experimental druggability assessments often utilize high-throughput or NMR-based screening with libraries of molecules that exhibit lead-like characteristics. However, the high cost of equipment and the potential for ambiguous results, often due to insufficient compound libraries, limit the effectiveness of these experimental approaches. As a result, computational methods have gained prominence in recent decades, particularly those based on virtual screening and machine learning. These approaches typically involve four primary druggability assessment methods:

1. Sequence-Based Druggability Assessment:

Sequence-based methods analyze the amino acid sequence of a protein to predict its potential as a drug target. The sequence of amino acids determines the protein's function and can reveal conserved regions that may form binding pockets for drugs. This method also provides insights into essential physicochemical properties such as solvent accessibility, hydrophobicity, charge, and polarity.

Sequence-based assessments are often used in machine-learning algorithms to predict druggability. They can help identify functional domains within a protein. However, relying solely on sequence data can be limiting, as it does not provide a full picture of the protein's structure or how accessible these domains are to drug molecules.

Examples include CHEMBL, LncRNA2Target, and MiRBase, which are databases and tools that aid in predicting druggability based on protein sequences.

2. Structure-Based Druggability Assessment:

Structure-based methods examine the 3D structure of a protein to identify and evaluate potential drug-binding pockets. For a small drug-like molecule to effectively bind, the protein must have a pocket that is appropriately sized, with a deep hydrophobic cavity to encapsulate the drug. Large, exposed polar sites are generally less druggable compared to smaller, more hydrophobic pockets.

This method provides a more detailed and reliable prediction compared to sequence-based methods, as it considers the physical characteristics of the protein's binding sites. The identified pockets are then compared against a reference set of known biological targets to assess their druggability.

Notable tools include DOGSiteScorer, Metapocket, Fpocket, PockDrug Server, SiteMap, and Open Targets, which help identify and analyze potential drug-binding pockets.

3. Ligand-Based Druggability Assessment:

Ligand-based methods focus on the likelihood that a protein can bind to known drug-like molecules, called ligands. By examining endogenous compounds and their interactions with the target, researchers can predict how well a new drug might interact with the protein.

This approach leverages existing data on ligands and their binding capabilities, which can provide valuable insights when predicting the druggability of new targets.

Examples include BindingDB, PubChem, SwissTargetPrediction and TargetHunter, which are comprehensive databases of ligand-protein interactions.

4. Precedence-Based Druggability Assessment:

This method relies on historical data of proteins that have already been successfully targeted by drugs. If a similar

protein has proven to be druggable in the past, it is more likely that a new, related target will also be druggable.

Precedence-based methods offer the highest confidence in druggability predictions because they are based on proven, established targets. However, while this method provides a strong basis for predicting success, it does not guarantee that new drugs targeting similar proteins will succeed.

Databases such as DrugBank, ClinicalTrials.gov, and DrugCentral store detailed information on existing drug targets and compounds currently undergoing clinical trials.

Application of Machine Learning in Druggability Assessment

Machine-learning (ML) based methodologies can handle vast datasets, extract complex patterns, provide more accurate predictions, and reduce human bias, ultimately speeding up the druggability assessment. Various machine learning algorithms have been employed to develop in silico models for the druggability assessment which help to analyse biological data, predict binding site interactions, and identify potential drug target and drug candidates.

The ML methods are categorized into supervised and unsupervised learning techniques, each serving different purposes within the drug discovery pipeline.

- The supervised learning focuses on tasks where the outcome is known and involves models like decision trees, random forests, SVM, and Bayesian networks for tasks ranging from disease-druggability predictions to target identification.
- Unsupervised Learning involves clustering techniques like K-Means, hierarchical clustering, and HMM for tasks like molecular designing and feature selection.

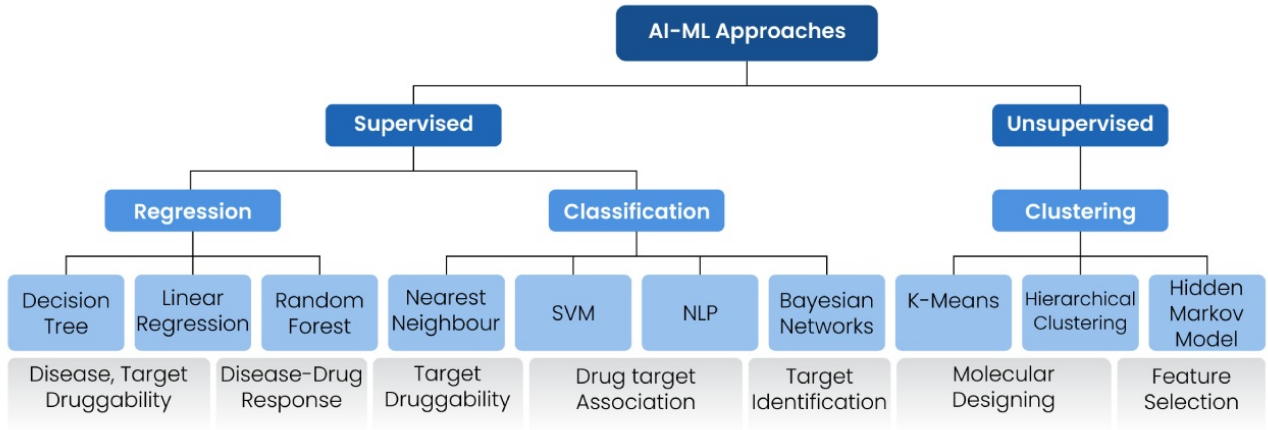


Fig 2 : Methods to predict the druggability based on machine learning (Source)

Note: For information on various ML models, please refer to the ML resources provided at the end of this tutorial. In this section, we will concentrate on the application of ML approaches specifically in druggability assessment.

Fig 2 illustrates how various AI and machine learning (ML) techniques are applied in the assessment of druggability.

In Supervised Learning, models like Decision Trees and Random Forest are used to predict the druggability of a target, such as a protein involved in a disease. These models can also predict the disease-drug response, which helps determine how well a drug might work in treating a specific disease. Classification methods, like Nearest Neighbour and SVM (Support Vector Machine), are used for drug target association, identifying which drugs are likely to interact with which targets. NLP (Natural Language Processing) helps analyze vast amounts of scientific literature to uncover potential drug targets, while Bayesian Networks assist in target identification, pinpointing which proteins or molecules are the best candidates for drug development.

Unsupervised Learning focuses on Clustering techniques, such as K-Means and Hierarchical Clustering, which group similar molecules or biological data together. This is crucial for molecular designing, where scientists design new drugs based on the properties of these clusters. Hidden Markov Models (HMM) are used for feature selection, identifying the most important characteristics of a protein that determine its druggability.

Application ML and AI based approaches in different Druggability Assessment methods:

1. Sequence-Based Druggability Assessment:

- Supervised Learning: Nearest Neighbour, SVM, and Random Forest can be used to predict the druggability of a target based on its sequence features (like amino acid composition and conserved regions). NLP techniques can process and extract relevant information from genetic databases or literature to predict potential binding sites based on sequence data.

- Unsupervised Learning : Hierarchical Clustering and K-Means can group protein sequences into clusters based on their similarities, which helps in identifying conserved regions or sequence motifs linked to druggability.

2. Structure-Based Druggability Assessment:

- Supervised Learning: Decision Trees and Bayesian Networks can predict whether specific structural features (like the size and hydrophobicity of binding pockets) make a protein druggable. Random Forest models can aggregate predictions about various structural features to give a more accurate overall assessment.
- Unsupervised Learning: Hidden Markov Models (HMMs) can model protein structural dynamics and predict how likely a given binding site is to interact with a drug. K-Means clustering can identify common structural features across different proteins that correlate with druggability.

3. Ligand-Based Druggability Assessment:

- Supervised Learning: SVM and NLP can be used to predict drug-target associations by analyzing known ligands and their binding affinities with various proteins. Random Forest models can improve the accuracy of these predictions by considering multiple ligand features simultaneously.
- Unsupervised Learning: K-Means and Hierarchical Clustering can group ligands based on their chemical properties, aiding in the identification of new ligand-based druggable targets.

4. Precedence-Based Druggability Assessment:

- Supervised Learning: Bayesian Networks and Decision Trees can help in predicting the success of new targets by analyzing previous data on established drug targets and their associated compounds. Random Forests can combine different data points from established targets to predict the druggability of new, similar targets.
- Unsupervised Learning: HMMs can be used to model the progression of drug development for targets with existing precedents, helping in feature selection and identifying key characteristics of successful targets.

Practical Application of Machine Learning for Druggability Prediction

With a solid understanding of druggability and the application of machine learning (ML) in druggability assessment, we are now ready to conduct an experiment to classify protein targets into 'highly druggable' and 'less druggable' categories using a Random Forest model.

Our dataset consists of proteins paired with their corresponding druggability labels - 'highly druggable' and 'less druggable'. To identify the druggable pockets within these proteins, we'll utilize Fpocket, a structure-based druggability assessment tool. Fpocket excels at identifying and characterizing pockets on the surface of proteins, which are potential binding sites for small molecules. These pockets are key regions where drug molecules might interact with the protein to exert a therapeutic effect.

The experiment involves two primary steps:

1. **Identifying Druggable Pockets**: Using Fpocket, we will analyze the protein structures to find pockets that might serve as effective binding sites for small molecules. Fpocket evaluates various characteristics of these pockets, such as size, shape, depth, and hydrophobicity, which are crucial indicators of druggability.

2. **Training the Random Forest Model**: Once we have characterized the pockets, the Random Forest model will be trained on these features along with the corresponding druggability labels. Random Forest is a machine learning algorithm that uses multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. It operates by constructing a multitude of decision trees during training and outputs either the mode of the classes (for classification) or the mean prediction (for regression) of the individual trees. The algorithm improves the accuracy and robustness of predictions by averaging the results from multiple decision trees, reducing the risk of overfitting and handling complex data with higher reliability. The model will learn to distinguish between 'highly druggable' and 'less druggable' pockets based on the patterns it identifies in the training data. After training, the model can classify new protein pockets as either highly druggable or less druggable with a high degree of accuracy.

While this model will provide valuable insights into whether a protein pocket is likely to be druggable, it's crucial to remember that druggability is just one aspect of a protein's potential as a drug target. Other important factors to consider include:

- Biological Relevance: The role of the protein in disease processes and whether modulating this protein will have a therapeutic effect.
- Feasibility: Practical considerations such as how easily a drug can reach the target protein in the body, and whether the protein is expressed in the right tissues at the right levels.
- Off-target effects: The potential for off-target effects and toxicity, which could arise if the protein is similar to other essential proteins in the body.

Building the Dataset

In this tutorial we'll use the NRLD dataset which has been widely used to study the druggability. It is a comprehensive, nonredundant data set containing crystal structures of 71 highly druggable and 44 less druggable proteins compiled by literature search and data mining published in the paper : DrugPred: A Structure-Based Approach To Predict Protein Druggability Developed Using an Extensive Nonredundant Data Set

The authors have only published the list of PDB code along with the labels. So, we'll first fetch the protein structure using Biopython. The labels are **'D'** for highly druggable and **'N'** for less druggable protein targets.

You can also use your own dataset if you have the labels. To obtain the structure of protein from your dataset you can refer the DeepChem tutorial: Protein Structure Prediction with ESMFold.

In [ ]:
```
!pip install biopython
```

```
Collecting biopython
  Downloading biopython-1.84-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from biopython) (1.25.2)
Downloading biopython-1.84-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 23.9 MB/s eta 0:00:00
Installing collected packages: biopython
Successfully installed biopython-1.84
```

In [ ]:
```
proteins_list = ['1pwm', '1lox', '3etr', '3f1q', '3ia4', '2cl5', '1uou', '1t46', '1unl', '1q41', '2i1m', '1pmn'
labels = ['D', 'D', 'D', '
```

Install the requirements

In [ ]:
```
!apt-get install -y build-essential git
# Clone the fpocket repository
!git clone https://github.com/Discngine/fpocket.git
%cd fpocket
!make
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
build-essential is already the newest version (12.9ubuntu3).
git is already the newest version (1:2.34.1-1ubuntu1.11).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Cloning into 'fpocket'...
remote: Enumerating objects: 11316, done.
remote: Counting objects: 100% (5152/5152), done.
remote: Compressing objects: 100% (1379/1379), done.
remote: Total 11316 (delta 3897), reused 4572 (delta 3764), pack-reused 6164
Receiving objects: 100% (11316/11316), 127.93 MiB | 21.88 MiB/s, done.
Resolving deltas: 100% (7346/7346), done.
Updating files: 100% (3895/3895), done.
/content/fpocket
cd src/qhull/ && make
make[1]: Entering directory '/content/fpocket/src/qhull'
mkdir -p bin lib
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/rbox/rbox.o src/rbox/rbox
.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/global.o s
rc/libqhull/global.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/stat.o src
/libqhull/stat.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/geom2.o sr
c/libqhull/geom2.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/poly2.o sr
c/libqhull/poly2.c
src/libqhull/poly2.c: In function ‘qh_delridge’:
src/libqhull/poly2.c:1127:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs
/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
 1127 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/merge.o sr
c/libqhull/merge.c
src/libqhull/merge.c: In function ‘qh_all_merges’:
src/libqhull/merge.c:219:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/
gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  219 |   void **freelistp;  /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~
src/libqhull/merge.c: In function ‘qh_appendmergeset’:
src/libqhull/merge.c:322:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/
gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  322 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~
src/libqhull/merge.c: In function ‘qh_mergecycle_ridges’:
src/libqhull/merge.c:2091:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs
/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
 2091 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/libqhull.o
src/libqhull/libqhull.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/geom.o src
/libqhull/geom.c
src/libqhull/geom.c: In function ‘qh_projectpoint’:
src/libqhull/geom.c:897:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  897 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~
src/libqhull/geom.c: In function ‘qh_setfacetplane’:
src/libqhull/geom.c:935:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  935 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/poly.o src
/libqhull/poly.c
src/libqhull/poly.c: In function ‘qh_delfacet’:
src/libqhull/poly.c:248:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  248 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~
src/libqhull/poly.c: In function ‘qh_makenew_nonsimplicial’:
src/libqhull/poly.c:564:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  564 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
```

```
      |          ^~~~~~~~
src/libqhull/poly.c: In function ‘qh_newfacet’:
src/libqhull/poly.c:987:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  987 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~
src/libqhull/poly.c: In function ‘qh_newridge’:
src/libqhull/poly.c:1020:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
 1020 |   void **freelistp;   /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/qset.o src/libqhull/qset.c
src/libqhull/qset.c: In function ‘qh_setfree’:
src/libqhull/qset.c:718:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  718 |   void **freelistp;  /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~
src/libqhull/qset.c: In function ‘qh_setnew’:
src/libqhull/qset.c:928:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  928 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/mem.o src/libqhull/mem.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/random.o src/libqhull/random.c
src/libqhull/random.c: In function ‘qh_argv_to_command’:
src/libqhull/random.c:84:6: warning: this ‘else’ clause does not guard... []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wmisleading-indentation-Wmisleading-indentation]8;;]
   84 |     }else
      |      ^~~~
src/libqhull/random.c:86:7: note: ...this statement, but the latter is misleadingly indented as if it were guarded by the ‘else’
   86 |       strcat(command, s);
      |       ^~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/usermem.o src/libqhull/usermem.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/userprintf.o src/libqhull/userprintf.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/io.o src/libqhull/io.c
In file included from src/libqhull/libqhull.h:38,
                 from src/libqhull/qhull_a.h:28,
                 from src/libqhull/io.c:21:
src/libqhull/io.c: In function ‘qh_printfacetridges’:
src/libqhull/qset.h:138:38: warning: this ‘for’ clause does not guard... []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wmisleading-indentation-Wmisleading-indentation]8;;]
  138 |         if (((variable= NULL), set)) for (\
      |                                      ^~~
src/libqhull/libqhull.h:936:34: note: in expansion of macro ‘FOREACHsetelement_’
  936 | #define FOREACHridge_(ridges)    FOREACHsetelement_(ridgeT, ridges, ridge)
      |                                  ^~~~~~~~~~~~~~~~~
src/libqhull/io.c:2622:7: note: in expansion of macro ‘FOREACHridge_’
 2622 |       FOREACHridge_(facet->ridges)
      |       ^~~~~~~~~~~~
src/libqhull/io.c:2624:9: note: ...this statement, but the latter is misleadingly indented as if it were guarded by the ‘for’
 2624 |         qh_fprintf(fp, 9185, "\n");
      |         ^~~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/user.o src/libqhull/user.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/rboxlib.o src/libqhull/rboxlib.c
In file included from /usr/include/string.h:535,
                 from src/libqhull/rboxlib.c:22:
In function ‘strncat’,
    inlined from ‘qh_rboxpoints’ at src/libqhull/rboxlib.c:385:5:
/usr/include/x86_64-linux-gnu/bits/string_fortified.h:138:10: warning: ‘__builtin___strncat_chk’ output may be truncated copying between 0 and 199 bytes from a string of length 199 []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wstringop-truncation-Wstringop-truncation]8;;]
  138 |   return __builtin___strncat_chk (__dest, __src, __len,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  139 |                                   __glibc_objsize (__dest));
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~
src/libqhull/rboxlib.c: In function ‘qh_rboxpoints’:
src/libqhull/rboxlib.c:93:7: warning: variable ‘coincidentcount’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   93 |   int coincidentcount=0, coincidenttotal=0, coincidentpoints=0;
      |       ^~~~~~~~~~~~~~
src/libqhull/rboxlib.c:93:45: warning: variable ‘coincidentpoints’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   93 |   int coincidentcount=0, coincidenttotal=0, coincidentpoints=0;
```

```
      |                                                ^~~~~~~~~~~~~~~
src/libqhull/rboxlib.c:94:30: warning: variable ‘seed’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   94 |   int cubesize, diamondsize, seed=0, count, apex;
      |                              ^~~~
src/libqhull/rboxlib.c:95:7: warning: variable ‘dim’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   95 |   int dim=3, numpoints=0, totpoints, addpoints=0;
      |       ^~~
src/libqhull/rboxlib.c:95:14: warning: variable ‘numpoints’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   95 |   int dim=3, numpoints=0, totpoints, addpoints=0;
      |              ^~~~~~~~~
src/libqhull/rboxlib.c:95:38: warning: variable ‘addpoints’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   95 |   int dim=3, numpoints=0, totpoints, addpoints=0;
      |                                      ^~~~~~~~~
src/libqhull/rboxlib.c:96:7: warning: variable ‘issphere’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |       ^~~~~~~~
src/libqhull/rboxlib.c:96:19: warning: variable ‘isaxis’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |                   ^~~~~~
src/libqhull/rboxlib.c:96:30: warning: variable ‘iscdd’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |                              ^~~~~
src/libqhull/rboxlib.c:96:39: warning: variable ‘islens’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |                                       ^~~~~~
src/libqhull/rboxlib.c:96:49: warning: variable ‘isregular’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |                                                 ^~~~~~~~~
src/libqhull/rboxlib.c:96:62: warning: variable ‘iswidth’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |                                                               ^~~~~~~
src/libqhull/rboxlib.c:96:73: warning: variable ‘addcube’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   96 |   int issphere=0, isaxis=0,  iscdd=0, islens=0, isregular=0, iswidth=0, addcube=0;
      |                                                                         ^~~~~~~
src/libqhull/rboxlib.c:97:16: warning: variable ‘isspiral’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   97 |   int isgap=0, isspiral=0, NOcommand=0, adddiamond=0;
      |                ^~~~~~~~
src/libqhull/rboxlib.c:97:28: warning: variable ‘NOcommand’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   97 |   int isgap=0, isspiral=0, NOcommand=0, adddiamond=0;
      |                            ^~~~~~~~~
src/libqhull/rboxlib.c:97:41: warning: variable ‘adddiamond’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   97 |   int isgap=0, isspiral=0, NOcommand=0, adddiamond=0;
      |                                         ^~~~~~~~~~
src/libqhull/rboxlib.c:98:7: warning: variable ‘israndom’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   98 |   int israndom=0, istime=0;
      |       ^~~~~~~~
src/libqhull/rboxlib.c:98:19: warning: variable ‘istime’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   98 |   int israndom=0, istime=0;
      |                   ^~~~~~
src/libqhull/rboxlib.c:99:7: warning: variable ‘isbox’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   99 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |       ^~~~~
src/libqhull/rboxlib.c:99:16: warning: variable ‘issimplex’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   99 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |                ^~~~~~~~~
src/libqhull/rboxlib.c:99:29: warning: variable ‘issimplex2’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   99 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |                             ^~~~~~~~~~
src/libqhull/rboxlib.c:99:43: warning: variable ‘ismesh’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   99 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |                                           ^~~~~~
src/libqhull/rboxlib.c:100:10: warning: variable ‘width’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
```

```
  100 |   double width=0.0, gap=0.0, radius=0.0, coincidentradius=0.0;
      |          ^~~~~
src/libqhull/rboxlib.c:100:30: warning: variable ‘radius’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  100 |   double width=0.0, gap=0.0, radius=0.0, coincidentradius=0.0;
      |                              ^~~~~~
src/libqhull/rboxlib.c:100:42: warning: variable ‘coincidentradius’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  100 |   double width=0.0, gap=0.0, radius=0.0, coincidentradius=0.0;
      |                                          ^~~~~~~~~~~~~~~~
src/libqhull/rboxlib.c:101:33: warning: variable ‘meshm’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  101 |   double coord[MAXdim], offset, meshm=3.0, meshn=4.0, meshr=5.0;
      |                                 ^~~~~
src/libqhull/rboxlib.c:101:44: warning: variable ‘meshn’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  101 |   double coord[MAXdim], offset, meshm=3.0, meshn=4.0, meshr=5.0;
      |                                            ^~~~~
src/libqhull/rboxlib.c:101:55: warning: variable ‘meshr’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  101 |   double coord[MAXdim], offset, meshm=3.0, meshn=4.0, meshr=5.0;
      |                                                       ^~~~~
src/libqhull/rboxlib.c:102:20: warning: variable ‘simplex’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  102 |   double *coordp, *simplex= NULL, *simplexp;
      |                    ^~~~~~~
src/libqhull/rboxlib.c:104:39: warning: variable ‘lensangle’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  104 |   double norm, factor, randr, rangap, lensangle=0, lensbase=1;
      |                                       ^~~~~~~~~
src/libqhull/rboxlib.c:104:52: warning: variable ‘lensbase’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  104 |   double norm, factor, randr, rangap, lensangle=0, lensbase=1;
      |                                                    ^~~~~~~~
src/libqhull/rboxlib.c:105:34: warning: variable ‘cube’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  105 |   double anglediff, angle, x, y, cube=0.0, diamond=0.0;
      |                                  ^~~~
src/libqhull/rboxlib.c:105:44: warning: variable ‘diamond’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  105 |   double anglediff, angle, x, y, cube=0.0, diamond=0.0;
      |                                            ^~~~~~~
src/libqhull/rboxlib.c:106:10: warning: variable ‘box’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  106 |   double box= qh_DEFAULTbox; /* scale all numbers before output */
      |          ^~~
src/libqhull/rboxlib.c:109:26: warning: variable ‘first_point’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
  109 |   char *s= command, *t, *first_point= NULL;
      |                          ^~~~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic/userprintf_rbox.o src/libqhull/userprintf_rbox.c
=========================================
==== If ar fails, try make qhullx ====
=========================================
ar -rs lib/libqhullstatic.a src/libqhullstatic/global.o src/libqhullstatic/stat.o src/libqhullstatic/geom2.o src/libqhullstatic/poly2.o src/libqhullstatic/merge.o src/libqhullstatic/libqhull.o src/libqhullstatic/geom.o src/libqhullstatic/poly.o src/libqhullstatic/qset.o src/libqhullstatic/mem.o src/libqhullstatic/random.o src/libqhullstatic/usermem.o src/libqhullstatic/userprintf.o src/libqhullstatic/io.o src/libqhullstatic/user.o src/libqhullstatic/rboxlib.o src/libqhullstatic/userprintf_rbox.o
ar: creating lib/libqhullstatic.a
#If 'ar -rs' fails try using 'ar -s' with 'ranlib'
#ranlib lib/libqhullstatic.a
gcc -o bin/rbox src/rbox/rbox.o -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -Llib -lqhullstatic -lm
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/qdelaunay/qdelaun.o src/qdelaunay/qdelaun.c
gcc -o bin/qdelaunay src/qdelaunay/qdelaun.o -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -Llib -lqhullstatic -lm
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/qhalf/qhalf.o src/qhalf/qhalf.c
gcc -o bin/qhalf src/qhalf/qhalf.o -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -Llib -lqhullstatic -lm
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/qhull/unix_r.o src/qhull/unix_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/global_r.o src/libqhull_r/global_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/stat_r.o src/libqhull_r/stat_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/geom2_r.o src/libqhull_r/geom2_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/poly2_r.o src/libqhull_r/poly2_r.c
```

```
src/libqhull_r/poly2_r.c: In function ‘qh_delridge’:
src/libqhull_r/poly2_r.c:1127:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
 1127 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/merge_r.o src/libqhull_r/merge_r.c
src/libqhull_r/merge_r.c: In function ‘qh_all_merges’:
src/libqhull_r/merge_r.c:219:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  219 |   void **freelistp;  /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~~
src/libqhull_r/merge_r.c: In function ‘qh_appendmergeset’:
src/libqhull_r/merge_r.c:322:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  322 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~~
src/libqhull_r/merge_r.c: In function ‘qh_mergecycle_ridges’:
src/libqhull_r/merge_r.c:2090:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
 2090 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/libqhull_r.o src/libqhull_r/libqhull_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/geom_r.o src/libqhull_r/geom_r.c
src/libqhull_r/geom_r.c: In function ‘qh_projectpoint’:
src/libqhull_r/geom_r.c:897:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  897 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~~
src/libqhull_r/geom_r.c: In function ‘qh_setfacetplane’:
src/libqhull_r/geom_r.c:935:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  935 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/poly_r.o src/libqhull_r/poly_r.c
src/libqhull_r/poly_r.c: In function ‘qh_delfacet’:
src/libqhull_r/poly_r.c:248:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  248 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~~
src/libqhull_r/poly_r.c: In function ‘qh_makenew_nonsimplicial’:
src/libqhull_r/poly_r.c:564:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  564 |   void **freelistp; /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~~
src/libqhull_r/poly_r.c: In function ‘qh_newfacet’:
src/libqhull_r/poly_r.c:987:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  987 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~~
src/libqhull_r/poly_r.c: In function ‘qh_newridge’:
src/libqhull_r/poly_r.c:1020:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
 1020 |   void **freelistp;   /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/qset_r.o src/libqhull_r/qset_r.c
src/libqhull_r/qset_r.c: In function ‘qh_setfree’:
src/libqhull_r/qset_r.c:718:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  718 |   void **freelistp;  /* used if !qh_NOmem by qh_memfree_() */
      |          ^~~~~~~~~
src/libqhull_r/qset_r.c: In function ‘qh_setnew’:
src/libqhull_r/qset_r.c:928:10: warning: variable ‘freelistp’ set but not used []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-but-set-variable-Wunused-but-set-variable]8;;]
  928 |   void **freelistp; /* used if !qh_NOmem by qh_memalloc_() */
      |          ^~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/mem_r.o src/libqhull_r/mem_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/random_r.o src/libqhull_r/random_r.c
src/libqhull_r/random_r.c: In function ‘qh_argv_to_command’:
src/libqhull_r/random_r.c:84:6: warning: this ‘else’ clause does not guard... []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wmisleading-indentation-Wmisleading-indentation]8;;]
   84 |     }else
      |      ^~~~
src/libqhull_r/random_r.c:86:7: note: ...this statement, but the latter is misleadingly indented as if it were guarded by the ‘else’
   86 |       strcat(command, s);
      |       ^~~~~~
```

```
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/usermem_
r.o src/libqhull_r/usermem_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/userprin
tf_r.o src/libqhull_r/userprintf_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/io_r.o s
rc/libqhull_r/io_r.c
In file included from src/libqhull_r/libqhull_r.h:32,
                 from src/libqhull_r/qhull_ra.h:28,
                 from src/libqhull_r/io_r.c:21:
src/libqhull_r/io_r.c: In function ‘qh_printfacetridges’:
src/libqhull_r/qset_r.h:143:38: warning: this ‘for’ clause does not guard... []8;;https://gcc.gnu.org/onlinedocs
/gcc/Warning-Options.html#index-Wmisleading-indentation-Wmisleading-indentation]8;;]
  143 |         if (((variable= NULL), set)) for (\
      |                                      ^~~
src/libqhull_r/libqhull_r.h:926:34: note: in expansion of macro ‘FOREACHsetelement_’
  926 | #define FOREACHridge_(ridges)    FOREACHsetelement_(ridgeT, ridges, ridge)
      |                                  ^~~~~~~~~~~~~~~~~
src/libqhull_r/io_r.c:2622:7: note: in expansion of macro ‘FOREACHridge_’
 2622 |       FOREACHridge_(facet->ridges)
      |       ^~~~~~~~~~~~
src/libqhull_r/io_r.c:2624:9: note: ...this statement, but the latter is misleadingly indented as if it were gua
rded by the ‘for’
 2624 |         qh_fprintf(qh, fp, 9185, "\n");
      |         ^~~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/user_r.o
src/libqhull_r/user_r.c
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/rboxlib_
r.o src/libqhull_r/rboxlib_r.c
In file included from /usr/include/string.h:535,
                 from src/libqhull_r/rboxlib_r.c:22:
In function ‘strncat’,
    inlined from ‘qh_rboxpoints’ at src/libqhull_r/rboxlib_r.c:358:5:
/usr/include/x86_64-linux-gnu/bits/string_fortified.h:138:10: warning: ‘__builtin___strncat_chk’ output may be t
runcated copying between 0 and 199 bytes from a string of length 199 []8;;https://gcc.gnu.org/onlinedocs/gcc/War
ning-Options.html#index-Wstringop-truncation-Wstringop-truncation]8;;]
  138 |   return __builtin___strncat_chk (__dest, __src, __len,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  139 |                                   __glibc_objsize (__dest));
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~
src/libqhull_r/rboxlib_r.c: In function ‘qh_rboxpoints’:
src/libqhull_r/rboxlib_r.c:75:7: warning: variable ‘coincidentcount’ might be clobbered by ‘longjmp’ or ‘vfork’
[]8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   75 |   int coincidentcount=0, coincidenttotal=0, coincidentpoints=0;
      |       ^~~~~~~~~~~~~~
src/libqhull_r/rboxlib_r.c:75:45: warning: variable ‘coincidentpoints’ might be clobbered by ‘longjmp’ or ‘vfork
’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   75 |   int coincidentcount=0, coincidenttotal=0, coincidentpoints=0;
      |                                             ^~~~~~~~~~~~~~~~
src/libqhull_r/rboxlib_r.c:76:30: warning: variable ‘seed’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https
://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   76 |   int cubesize, diamondsize, seed=0, count, apex;
      |                              ^~~~
src/libqhull_r/rboxlib_r.c:77:7: warning: variable ‘dim’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https:/
/gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   77 |   int dim=3 , numpoints= 0, totpoints, addpoints=0;
      |       ^~~
src/libqhull_r/rboxlib_r.c:77:15: warning: variable ‘numpoints’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   77 |   int dim=3 , numpoints= 0, totpoints, addpoints=0;
      |               ^~~~~~~~~
src/libqhull_r/rboxlib_r.c:77:40: warning: variable ‘addpoints’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   77 |   int dim=3 , numpoints= 0, totpoints, addpoints=0;
      |                                        ^~~~~~~~~
src/libqhull_r/rboxlib_r.c:78:7: warning: variable ‘issphere’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;ht
tps://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
      |       ^~~~~~~~
src/libqhull_r/rboxlib_r.c:78:19: warning: variable ‘isaxis’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;htt
ps://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
      |                   ^~~~~~
src/libqhull_r/rboxlib_r.c:78:30: warning: variable ‘iscdd’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;http
s://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
      |                              ^~~~~
src/libqhull_r/rboxlib_r.c:78:40: warning: variable ‘islens’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;htt
ps://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
      |                                        ^~~~~~
src/libqhull_r/rboxlib_r.c:78:51: warning: variable ‘isregular’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;
https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
```

```
      |                                                 ^~~~~~~~
src/libqhull_r/rboxlib_r.c:78:64: warning: variable ‘iswidth’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
      |                                                                ^~~~~~~
src/libqhull_r/rboxlib_r.c:78:75: warning: variable ‘addcube’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   78 |   int issphere=0, isaxis=0,  iscdd= 0, islens= 0, isregular=0, iswidth=0, addcube=0;
      |                                                                           ^~~~~~~
src/libqhull_r/rboxlib_r.c:79:16: warning: variable ‘isspiral’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   79 |   int isgap=0, isspiral=0, NOcommand= 0, adddiamond=0;
      |                ^~~~~~~~
src/libqhull_r/rboxlib_r.c:79:28: warning: variable ‘NOcommand’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   79 |   int isgap=0, isspiral=0, NOcommand= 0, adddiamond=0;
      |                            ^~~~~~~~~
src/libqhull_r/rboxlib_r.c:79:42: warning: variable ‘adddiamond’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   79 |   int isgap=0, isspiral=0, NOcommand= 0, adddiamond=0;
      |                                          ^~~~~~~~~~
src/libqhull_r/rboxlib_r.c:80:7: warning: variable ‘israndom’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   80 |   int israndom=0, istime=0;
      |       ^~~~~~~~
src/libqhull_r/rboxlib_r.c:80:19: warning: variable ‘istime’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   80 |   int israndom=0, istime=0;
      |                   ^~~~~~
src/libqhull_r/rboxlib_r.c:81:7: warning: variable ‘isbox’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   81 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |       ^~~~~
src/libqhull_r/rboxlib_r.c:81:16: warning: variable ‘issimplex’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   81 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |                ^~~~~~~~~
src/libqhull_r/rboxlib_r.c:81:29: warning: variable ‘issimplex2’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   81 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |                             ^~~~~~~~~~
src/libqhull_r/rboxlib_r.c:81:43: warning: variable ‘ismesh’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   81 |   int isbox=0, issimplex=0, issimplex2=0, ismesh=0;
      |                                           ^~~~~~
src/libqhull_r/rboxlib_r.c:82:10: warning: variable ‘width’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   82 |   double width=0.0, gap=0.0, radius=0.0, coincidentradius=0.0;
      |          ^~~~~
src/libqhull_r/rboxlib_r.c:82:30: warning: variable ‘radius’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   82 |   double width=0.0, gap=0.0, radius=0.0, coincidentradius=0.0;
      |                              ^~~~~~
src/libqhull_r/rboxlib_r.c:82:42: warning: variable ‘coincidentradius’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   82 |   double width=0.0, gap=0.0, radius=0.0, coincidentradius=0.0;
      |                                          ^~~~~~~~~~~~~~~~
src/libqhull_r/rboxlib_r.c:83:33: warning: variable ‘meshm’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   83 |   double coord[MAXdim], offset, meshm=3.0, meshn=4.0, meshr=5.0;
      |                                 ^~~~~
src/libqhull_r/rboxlib_r.c:83:44: warning: variable ‘meshn’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   83 |   double coord[MAXdim], offset, meshm=3.0, meshn=4.0, meshr=5.0;
      |                                            ^~~~~
src/libqhull_r/rboxlib_r.c:83:55: warning: variable ‘meshr’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   83 |   double coord[MAXdim], offset, meshm=3.0, meshn=4.0, meshr=5.0;
      |                                                       ^~~~~
src/libqhull_r/rboxlib_r.c:84:20: warning: variable ‘simplex’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   84 |   double *coordp, *simplex= NULL, *simplexp;
      |                    ^~~~~~~
src/libqhull_r/rboxlib_r.c:86:39: warning: variable ‘lensangle’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   86 |   double norm, factor, randr, rangap, lensangle= 0, lensbase= 1;
      |                                       ^~~~~~~~~
src/libqhull_r/rboxlib_r.c:86:53: warning: variable ‘lensbase’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   86 |   double norm, factor, randr, rangap, lensangle= 0, lensbase= 1;
      |                                                     ^~~~~~~~
src/libqhull_r/rboxlib_r.c:87:34: warning: variable ‘cube’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
```

```
   87 |   double anglediff, angle, x, y, cube= 0.0, diamond= 0.0;
      |                                  ^~~~
src/libqhull_r/rboxlib_r.c:87:45: warning: variable ‘diamond’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   87 |   double anglediff, angle, x, y, cube= 0.0, diamond= 0.0;
      |                                             ^~~~~~~
src/libqhull_r/rboxlib_r.c:88:10: warning: variable ‘box’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   88 |   double box= qh_DEFAULTbox; /* scale all numbers before output */
      |          ^~~
src/libqhull_r/rboxlib_r.c:91:26: warning: variable ‘first_point’ might be clobbered by ‘longjmp’ or ‘vfork’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wclobbered-Wclobbered]8;;]
   91 |   char *s= command, *t, *first_point= NULL;
      |                          ^~~~~~~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/libqhullstatic_r/userprintf_rbox_r.o src/libqhull_r/userprintf_rbox_r.c
ar -rs lib/libqhullstatic_r.a src/libqhullstatic_r/global_r.o src/libqhullstatic_r/stat_r.o src/libqhullstatic_r/geom2_r.o src/libqhullstatic_r/poly2_r.o src/libqhullstatic_r/merge_r.o src/libqhullstatic_r/libqhull_r.o src/libqhullstatic_r/geom_r.o src/libqhullstatic_r/poly_r.o src/libqhullstatic_r/qset_r.o src/libqhullstatic_r/mem_r.o src/libqhullstatic_r/random_r.o src/libqhullstatic_r/usermem_r.o src/libqhullstatic_r/userprintf_r.o src/libqhullstatic_r/io_r.o src/libqhullstatic_r/user_r.o src/libqhullstatic_r/rboxlib_r.o src/libqhullstatic_r/userprintf_rbox_r.o
ar: creating lib/libqhullstatic_r.a
#ranlib lib/libqhullstatic_r.a
gcc -o bin/qhull src/qhull/unix_r.o -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -Llib -lqhullstatic_r -lm
chmod +x eg/q_test eg/q_eg eg/q_egtest
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/qconvex/qconvex.o src/qconvex/qconvex.c
src/qconvex/qconvex.c: In function ‘run_qconvex’:
src/qconvex/qconvex.c:72:38: warning: passing argument 5 of ‘qh_init_A’ from incompatible pointer type []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wincompatible-pointer-types-Wincompatible-pointer-types]8;;]
   72 |   qh_init_A(fin, fout, stderr, argc, argv);  /* sets qh qhull_command */
      |                                      ^~~~
      |                                      |
      |                                      const char **
In file included from src/qconvex/qconvex.c:12:
src/libqhull/libqhull.h:1083:79: note: expected ‘char **’ but argument is of type ‘const char **’
 1083 | void    qh_init_A(FILE *infile, FILE *outfile, FILE *errfile, int argc, char *argv[]);
      |                                                                         ~~~~~~^~~~~
gcc -c -O3 -ansi -Isrc -fpic -Wall -Wcast-qual -Wextra -Wwrite-strings -Wshadow -o src/qvoronoi/qvoronoi.o src/qvoronoi/qvoronoi.c
src/qvoronoi/qvoronoi.c: In function ‘run_qvoronoi’:
src/qvoronoi/qvoronoi.c:285:39: warning: passing argument 5 of ‘qh_init_A’ from incompatible pointer type []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wincompatible-pointer-types-Wincompatible-pointer-types]8;;]
  285 |   qh_init_A (fin, fout, stderr, argc, argv);  /* sets qh qhull_command */
      |                                       ^~~~
      |                                       |
      |                                       const char **
In file included from src/qvoronoi/qvoronoi.c:20:
src/libqhull/libqhull.h:1083:79: note: expected ‘char **’ but argument is of type ‘const char **’
 1083 | void    qh_init_A(FILE *infile, FILE *outfile, FILE *errfile, int argc, char *argv[]);
      |                                                                         ~~~~~~^~~~~
src/qvoronoi/qvoronoi.c:237:16: warning: unused variable ‘i’ []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wunused-variable-Wunused-variable]8;;]
  237 |   int curlong, i,totlong; /* used !qh_NOmem */
      |                ^
make[1]: Leaving directory '/content/fpocket/src/qhull'
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/fpmain.c -o obj/fpmain.o
src/fpmain.c: In function ‘open_file_format’:
src/fpmain.c:239:16: warning: ‘pdb’ may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
  239 |         return pdb;
      |                ^~~
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/psorting.c -o obj/psorting.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/pscoring.c -o obj/pscoring.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/utils.c -o obj/utils.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/pertable.c -o obj/pertable.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/memhandler.c -o obj/memhandler.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/voronoi.c -o obj/voronoi.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/sort.c -o obj/sort.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/calc.c -o obj/calc.o
```

```
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/writepdb.c -o obj/writepdb.o
src/writepdb.c: In function ‘write_pdb_atom_line’:
src/writepdb.c:94:18: warning: comparison between pointer and integer
   94 |         if(insert!=NULL) {
      |                  ^~
src/writepdb.c:101:34: warning: ‘%05x’ directive writing between 5 and 8 bytes into a region of size 6 []8;;http
s://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  101 |                 sprintf(id_buf, "%05x", id);
      |                                  ^~~~
src/writepdb.c:101:33: note: directive argument in the range [100000, 2147483647]
  101 |                 sprintf(id_buf, "%05x", id);
      |                                 ^~~~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/writepdb.h:14,
                 from src/writepdb.c:2:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:38:10: note: ‘__builtin___sprintf_chk’ output between 6 and 9 bytes
into a destination of size 6
   38 |   return __builtin___sprintf_chk (__s, __USE_FORTIFY_LEVEL - 1,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   39 |                                   __glibc_objsize (__s), __fmt,
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   40 |                                   __va_arg_pack ());
      |                                   ~~~~~~~~~~~~~~~~~
src/writepdb.c:99:34: warning: ‘%5d’ directive writing between 5 and 11 bytes into a region of size 6 []8;;https
://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
   99 |                 sprintf(id_buf, "%5d", id);
      |                                  ^~~
src/writepdb.c:99:33: note: directive argument in the range [-2147483648, 99999]
   99 |                 sprintf(id_buf, "%5d", id);
      |                                 ^~~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/writepdb.h:14,
                 from src/writepdb.c:2:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:38:10: note: ‘__builtin___sprintf_chk’ output between 6 and 12 bytes
into a destination of size 6
   38 |   return __builtin___sprintf_chk (__s, __USE_FORTIFY_LEVEL - 1,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   39 |                                   __glibc_objsize (__s), __fmt,
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   40 |                                   __va_arg_pack ());
      |                                   ~~~~~~~~~~~~~~~~~
src/writepdb.c:104:38: warning: ‘%4d’ directive writing between 4 and 11 bytes into a region of size 5 []8;;http
s://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  104 |                 sprintf(res_id_buf, "%4d", res_id);
      |                                      ^~~
src/writepdb.c:104:37: note: directive argument in the range [-2147483648, 9999]
  104 |                 sprintf(res_id_buf, "%4d", res_id);
      |                                     ^~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/writepdb.h:14,
                 from src/writepdb.c:2:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:38:10: note: ‘__builtin___sprintf_chk’ output between 5 and 12 bytes
into a destination of size 5
   38 |   return __builtin___sprintf_chk (__s, __USE_FORTIFY_LEVEL - 1,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   39 |                                   __glibc_objsize (__s), __fmt,
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   40 |                                   __va_arg_pack ());
      |                                   ~~~~~~~~~~~~~~~~~
src/writepdb.c: In function ‘write_pqr_atom_line’:
src/writepdb.c:185:34: warning: ‘%05x’ directive writing between 5 and 8 bytes into a region of size 7 []8;;http
s://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  185 |                 sprintf(id_buf, "%05x", id);
      |                                  ^~~~
src/writepdb.c:185:33: note: directive argument in the range [100000, 2147483647]
  185 |                 sprintf(id_buf, "%05x", id);
      |                                 ^~~~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/writepdb.h:14,
                 from src/writepdb.c:2:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:38:10: note: ‘__builtin___sprintf_chk’ output between 6 and 9 bytes
into a destination of size 7
   38 |   return __builtin___sprintf_chk (__s, __USE_FORTIFY_LEVEL - 1,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   39 |                                   __glibc_objsize (__s), __fmt,
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   40 |                                   __va_arg_pack ());
      |                                   ~~~~~~~~~~~~~~~~~
src/writepdb.c:183:34: warning: ‘%5d’ directive writing between 5 and 11 bytes into a region of size 7 []8;;http
s://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  183 |                 sprintf(id_buf, "%5d", id);
      |                                  ^~~
```

```
src/writepdb.c:183:33: note: directive argument in the range [-2147483648, 99999]
  183 |                 sprintf(id_buf, "%5d", id);
      |                                 ^~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/writepdb.h:14,
                 from src/writepdb.c:2:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:38:10: note: '__builtin___sprintf_chk' output between 6 and 12 bytes
into a destination of size 7
   38 |   return __builtin___sprintf_chk (__s, __USE_FORTIFY_LEVEL - 1,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   39 |                                   __glibc_objsize (__s), __fmt,
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   40 |                                   __va_arg_pack ());
      |                                   ~~~~~~~~~~~~~~~~~
src/writepdb.c:188:38: warning: '%4d' directive writing between 4 and 11 bytes into a region of size 6 []8;;http
s://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  188 |                 sprintf(res_id_buf, "%4d", res_id);
      |                                      ^~~
src/writepdb.c:188:37: note: directive argument in the range [-2147483648, 9999]
  188 |                 sprintf(res_id_buf, "%4d", res_id);
      |                                     ^~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/writepdb.h:14,
                 from src/writepdb.c:2:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:38:10: note: '__builtin___sprintf_chk' output between 5 and 12 bytes
into a destination of size 6
   38 |   return __builtin___sprintf_chk (__s, __USE_FORTIFY_LEVEL - 1,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   39 |                                   __glibc_objsize (__s), __fmt,
      |                                   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   40 |                                   __va_arg_pack ());
      |                                   ~~~~~~~~~~~~~~~~~
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/rpdb.c -o obj/rpdb.o
src/rpdb.c: In function 'rpdb_open':
src/rpdb.c:902:42: warning: 'z' may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
  902 |                 if (x < 9990 && y < 9990 && z < 9990)
      |                     ~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~
src/rpdb.c:902:30: warning: 'y' may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
  902 |                 if (x < 9990 && y < 9990 && z < 9990)
      |                     ~~~~~~~~~^~~~~~~~~~
src/rpdb.c:902:20: warning: 'x' may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedocs/g
cc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
  902 |                 if (x < 9990 && y < 9990 && z < 9990)
      |                    ^
src/rpdb.c: In function 'rpdb_read':
src/rpdb.c:1234:48: warning: 'tmpz' may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedo
cs/gcc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
 1234 |                 if (tmpx < 9990 && tmpy < 9990 && tmpz < 9990)
      |                     ~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~
src/rpdb.c:1234:33: warning: 'tmpy' may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedo
cs/gcc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
 1234 |                 if (tmpx < 9990 && tmpy < 9990 && tmpz < 9990)
      |                     ~~~~~~~~~~~~^~~~~~~~~~~~~~
src/rpdb.c:1234:20: warning: 'tmpx' may be used uninitialized in this function []8;;https://gcc.gnu.org/onlinedo
cs/gcc/Warning-Options.html#index-Wmaybe-uninitialized-Wmaybe-uninitialized]8;;]
 1234 |                 if (tmpx < 9990 && tmpy < 9990 && tmpz < 9990)
      |                    ^
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/tparams.c -o obj/tparams.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/fparams.c -o obj/fparams.o
src/fparams.c: In function 'get_fpocket_args':
src/fparams.c:296:24: warning: passing argument 1 of 'strcpy' from incompatible pointer type []8;;https://gcc.gn
u.org/onlinedocs/gcc/Warning-Options.html#index-Wincompatible-pointer-types-Wincompatible-pointer-types]8;;]
  296 |                 strcpy(&residue_string, pt);
      |                        ^~~~~~~~~~~~~~~
      |                        |
      |                        char * (*)[8000]
In file included from /usr/include/features.h:486,
                 from /usr/include/x86_64-linux-gnu/bits/libc-header-start.h:33,
                 from /usr/include/stdio.h:27,
                 from src/../headers/fparams.h:16,
                 from src/fparams.c:2:
/usr/include/x86_64-linux-gnu/bits/string_fortified.h:77:1: note: expected 'char * restrict' but argument is of
type 'char * (*)[8000]'
   77 | __NTH (strcpy (char *__restrict __dest, const char *__restrict __src))
      | ^~~~~
src/fparams.c:297:23: warning: assignment to 'char *' from incompatible pointer type 'char **' []8;;https://gcc.
gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wincompatible-pointer-types-Wincompatible-pointer-types]8;;]
  297 |                 rest2 = residue_string;
```

```
      |                       ^
src/fparams.c:304:58: warning: this statement may fall through []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wimplicit-fallthrough=-Wimplicit-fallthrough=]8;;]
  304 |                       par->xpocket_residue_number[pti] = (unsigned short)atoi(apt); // fprintf(stdout, "residuenumber: %d\n", atoi(apt));
      |                       ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^~~~~~~~~~~~~~~~~~~~~~~~~
src/fparams.c:305:21: note: here
  305 |                     case 1:
      |                     ^~~~
src/fparams.c:306:25: warning: this statement may fall through []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wimplicit-fallthrough=-Wimplicit-fallthrough=]8;;]
  306 |                       strncpy(&(par->xpocket_insertion_code[pti]), apt, 1);
      |                       ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
src/fparams.c:307:21: note: here
  307 |                     case 2:
      |                     ^~~~
In file included from /usr/include/stdio.h:894,
                 from src/../headers/fparams.h:16,
                 from src/fparams.c:2:
In function ‘fprintf’,
    inlined from ‘parse_clustering_method’ at src/fparams.c:631:9,
    inlined from ‘parse_clustering_method’ at src/fparams.c:623:5:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:105:10: warning: ‘%s’ directive argument is null []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  105 |   return __fprintf_chk (__stream, __USE_FORTIFY_LEVEL - 1, __fmt,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  106 |                         __va_arg_pack ());
      |                         ~~~~~~~~~~~~~~~~
In function ‘fprintf’,
    inlined from ‘parse_distance_measure’ at src/fparams.c:646:9,
    inlined from ‘parse_distance_measure’ at src/fparams.c:638:5:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:105:10: warning: ‘%s’ directive argument is null []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  105 |   return __fprintf_chk (__stream, __USE_FORTIFY_LEVEL - 1, __fmt,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  106 |                         __va_arg_pack ());
      |                         ~~~~~~~~~~~~~~~~
In function ‘fprintf’,
    inlined from ‘parse_distance_measure’ at src/fparams.c:646:9,
    inlined from ‘parse_distance_measure’ at src/fparams.c:638:5,
    inlined from ‘DEPR_get_fpocket_args’ at src/fparams.c:474:27:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:105:10: warning: ‘%s’ directive argument is null []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  105 |   return __fprintf_chk (__stream, __USE_FORTIFY_LEVEL - 1, __fmt,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  106 |                         __va_arg_pack ());
      |                         ~~~~~~~~~~~~~~~~
In function ‘fprintf’,
    inlined from ‘parse_clustering_method’ at src/fparams.c:631:9,
    inlined from ‘parse_clustering_method’ at src/fparams.c:623:5,
    inlined from ‘DEPR_get_fpocket_args’ at src/fparams.c:471:27:
/usr/include/x86_64-linux-gnu/bits/stdio2.h:105:10: warning: ‘%s’ directive argument is null []8;;https://gcc.gnu.org/onlinedocs/gcc/Warning-Options.html#index-Wformat-overflow=-Wformat-overflow=]8;;]
  105 |   return __fprintf_chk (__stream, __USE_FORTIFY_LEVEL - 1, __fmt,
      |          ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  106 |                         __va_arg_pack ());
      |                         ~~~~~~~~~~~~~~~~
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/pocket.c -o obj/pocket.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/refine.c -o obj/refine.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/descriptors.c -o obj/descriptors.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/aa.c -o obj/aa.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/fpocket.c -o obj/fpocket.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/write_visu.c -o obj/write_visu.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/fpout.c -o obj/fpout.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/atom.c -o obj/atom.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/writepocket.c -o obj/writepocket.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/voronoi_lst.c -o obj/voronoi_lst.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/asa.c -o obj/asa.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/clusterlib.c -o obj/clusterlib.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/include -Iplugins/LINUXAMD64/molfile -c src/energy.c -o obj/energy.o
```

```
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/topology.c -o obj/topology.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/read_mmcif.c -o obj/read_mmcif.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/geom2.c -o src/qhull/src/libqhull/geom2.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/geom.c -o src/qhull/src/libqhull/geom.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/global.c -o src/qhull/src/libqhull/global.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/io.c -o src/qhull/src/libqhull/io.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/libqhull.c -o src/qhull/src/libqhull/libqhull.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/mem.c -o src/qhull/src/libqhull/mem.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/merge.c -o src/qhull/src/libqhull/merge.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/poly2.c -o src/qhull/src/libqhull/poly2.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/poly.c -o src/qhull/src/libqhull/poly.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/qset.c -o src/qhull/src/libqhull/qset.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/random.c -o src/qhull/src/libqhull/random.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/rboxlib.c -o src/qhull/src/libqhull/rboxlib.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/stat.c -o src/qhull/src/libqhull/stat.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/user.c -o src/qhull/src/libqhull/user.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/usermem.c -o src/qhull/src/libqhull/usermem.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/userprintf.c -o src/qhull/src/libqhull/userprintf.o
gcc -O -g -pg -ansi -c src/qhull/src/libqhull/userprintf_rbox.c -o src/qhull/src/libqhull/userprintf_rbox.o
LD_LIBRARY_PATH=plugins/LINUXAMD64/molfile gcc obj/fpmain.o obj/psorting.o obj/pscoring.o obj/utils.o obj/pertab
le.o obj/memhandler.o obj/voronoi.o obj/sort.o obj/calc.o obj/writepdb.o obj/rpdb.o obj/tparams.o obj/fparams.o
obj/pocket.o obj/refine.o obj/descriptors.o obj/aa.o obj/fpocket.o obj/write_visu.o obj/fpout.o obj/atom.o obj/w
ritepocket.o obj/voronoi_lst.o obj/asa.o obj/clusterlib.o obj/energy.o obj/topology.o obj/read_mmcif.o src/qhull
/src/libqhull/geom2.o src/qhull/src/libqhull/geom.o src/qhull/src/libqhull/global.o src/qhull/src/libqhull/io.o
src/qhull/src/libqhull/libqhull.o src/qhull/src/libqhull/mem.o src/qhull/src/libqhull/merge.o src/qhull/src/libq
hull/poly2.o src/qhull/src/libqhull/poly.o src/qhull/src/libqhull/qset.o src/qhull/src/libqhull/random.o src/qhu
ll/src/libqhull/rboxlib.o src/qhull/src/libqhull/stat.o src/qhull/src/libqhull/user.o src/qhull/src/libqhull/use
rmem.o src/qhull/src/libqhull/userprintf.o src/qhull/src/libqhull/userprintf_rbox.o src/qhull/src/qvoronoi/qvoro
noi.o src/qhull/src/qconvex/qconvex.o -o bin/fpocket -lm -Lplugins/LINUXAMD64/molfile plugins/LINUXAMD64/molfile
/libmolfile_plugin.a -lstdc++
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/tpmain.c -o obj/tpmain.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/tpocket.c -o obj/tpocket.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/neighbor.c -o obj/neighbor.o
LD_LIBRARY_PATH=plugins/LINUXAMD64/molfile gcc obj/tpmain.o obj/psorting.o obj/pscoring.o obj/utils.o obj/pertab
le.o obj/memhandler.o obj/voronoi.o obj/sort.o obj/calc.o obj/writepdb.o obj/rpdb.o obj/tparams.o obj/fparams.o
obj/pocket.o obj/refine.o obj/tpocket.o obj/descriptors.o obj/aa.o obj/fpocket.o obj/write_visu.o obj/fpout.o ob
j/atom.o obj/writepocket.o obj/voronoi_lst.o obj/neighbor.o obj/asa.o obj/clusterlib.o obj/energy.o obj/topology
.o obj/read_mmcif.o src/qhull/src/qvoronoi/qvoronoi.o src/qhull/src/qconvex/qconvex.o src/qhull/src/libqhull/geo
m2.o src/qhull/src/libqhull/geom.o src/qhull/src/libqhull/global.o src/qhull/src/libqhull/io.o src/qhull/src/lib
qhull/libqhull.o src/qhull/src/libqhull/mem.o src/qhull/src/libqhull/merge.o src/qhull/src/libqhull/poly2.o src/
qhull/src/libqhull/poly.o src/qhull/src/libqhull/qset.o src/qhull/src/libqhull/random.o src/qhull/src/libqhull/r
boxlib.o src/qhull/src/libqhull/stat.o src/qhull/src/libqhull/user.o src/qhull/src/libqhull/usermem.o src/qhull/
src/libqhull/userprintf.o src/qhull/src/libqhull/userprintf_rbox.o -o bin/tpocket -lm -Lplugins/LINUXAMD64/molfi
le plugins/LINUXAMD64/molfile/libmolfile_plugin.a -lstdc++
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/dpmain.c -o obj/dpmain.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/dpocket.c -o obj/dpocket.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/dparams.c -o obj/dparams.o
LD_LIBRARY_PATH=plugins/LINUXAMD64/molfile gcc obj/dpmain.o obj/psorting.o obj/pscoring.o obj/dpocket.o obj/dpar
ams.o obj/voronoi.o obj/sort.o obj/rpdb.o obj/descriptors.o obj/neighbor.o obj/atom.o obj/aa.o obj/pertable.o ob
j/calc.o obj/utils.o obj/writepdb.o obj/memhandler.o obj/pocket.o obj/refine.o obj/fparams.o obj/fpocket.o obj/f
pout.o obj/writepocket.o obj/write_visu.o obj/asa.o obj/read_mmcif.o obj/voronoi_lst.o obj/clusterlib.o src/qhul
l/src/libqhull/geom2.o src/qhull/src/libqhull/geom.o src/qhull/src/libqhull/global.o src/qhull/src/libqhull/io.o
src/qhull/src/libqhull/libqhull.o src/qhull/src/libqhull/mem.o src/qhull/src/libqhull/merge.o src/qhull/src/libq
hull/poly2.o src/qhull/src/libqhull/poly.o src/qhull/src/libqhull/qset.o src/qhull/src/libqhull/random.o src/qhu
ll/src/libqhull/rboxlib.o src/qhull/src/libqhull/stat.o src/qhull/src/libqhull/user.o src/qhull/src/libqhull/use
rmem.o src/qhull/src/libqhull/userprintf.o src/qhull/src/libqhull/userprintf_rbox.o src/qhull/src/qvoronoi/qvoro
noi.o src/qhull/src/qconvex/qconvex.o obj/energy.o obj/topology.o -o bin/dpocket -lm -Lplugins/LINUXAMD64/molfil
e plugins/LINUXAMD64/molfile/libmolfile_plugin.a -lstdc++
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/mdpmain.c -o obj/mdpmain.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/mdpocket.c -o obj/mdpocket.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/mdpbase.c -o obj/mdpbase.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/mdpout.c -o obj/mdpout.o
gcc -W -Wextra -Wwrite-strings -Wstrict-prototypes -DM_OS_LINUX -DMNO_MEM_DEBUG -O2 -g -pg -std=gnu99 -Iplugins/
include -Iplugins/LINUXAMD64/molfile -c src/mdparams.c -o obj/mdparams.o
LD_LIBRARY_PATH=plugins/LINUXAMD64/molfile gcc obj/mdpmain.o obj/mdpocket.o obj/mdpbase.o obj/mdpout.o obj/psort
ing.o obj/pscoring.o obj/mdparams.o obj/voronoi.o obj/sort.o obj/rpdb.o obj/descriptors.o obj/neighbor.o obj/ato
m.o obj/aa.o obj/pertable.o obj/calc.o obj/utils.o obj/writepdb.o obj/memhandler.o obj/pocket.o obj/refine.o obj
/fparams.o obj/fpocket.o obj/fpout.o obj/writepocket.o obj/write_visu.o obj/asa.o obj/read_mmcif.o obj/voronoi_l
st.o obj/clusterlib.o src/qhull/src/libqhull/geom2.o src/qhull/src/libqhull/geom.o src/qhull/src/libqhull/global
.o src/qhull/src/libqhull/io.o src/qhull/src/libqhull/libqhull.o src/qhull/src/libqhull/mem.o src/qhull/src/libq
hull/merge.o src/qhull/src/libqhull/poly2.o src/qhull/src/libqhull/poly.o src/qhull/src/libqhull/qset.o src/qhul
```

```
l/src/libqhull/random.o src/qhull/src/libqhull/rboxlib.o src/qhull/src/libqhull/stat.o src/qhull/src/libqhull/us
er.o src/qhull/src/libqhull/usermem.o src/qhull/src/libqhull/userprintf.o src/qhull/src/libqhull/userprintf_rbox
.o src/qhull/src/qvoronoi/qvoronoi.o src/qhull/src/qconvex/qconvex.o obj/energy.o obj/topology.o -o bin/mdpocket
-lm -Lplugins/LINUXAMD64/molfile plugins/LINUXAMD64/molfile/libmolfile_plugin.a -lstdc++
```

Run Fpocket for each of the protein and save the output in output directory

In [ ]:
```python
import subprocess
def run_fpocket(pdb_file_path):
    """
    Runs fpocket on the given PDB file to find binding pockets.

    Parameters:
    pdb_file_path (str): The path to the PDB file.

    Returns:
    str: The path to the fpocket output directory.
    """
    try:
        # Run fpocket
        command = ["bin/fpocket", "-f", pdb_file_path]
        subprocess.run(command, check=True)

        # Extract the output directory path
        pdb_base = os.path.basename(pdb_file_path).split('.')[0]
        output_dir = f"{pdb_base}_out"
    except subprocess.CalledProcessError as e:
        print(f'Error running fpocket on {pdb_file_path}: {e}')
        return None
```

Specify the base directory where the pdb files are stored

In [ ]:
```python
base_pdb_files = '/content/pdb_files/'
```

Run Fpocket

In [ ]:
```python
for protein in proteins_list:
  protein_structure = os.path.join(base_pdb_files, f'{protein}.pdb')
  run_fpocket(protein_structure)
```

Let's have a look at the fpocket output

In [ ]:
```python
# We'll have a look at the protein /content/pdb_files/1ajs_out

fpocket_info_dir = '/content/pdb_files/1ajs_out/1ajs_info.txt'
with open(fpocket_info_dir, 'r') as file:
  for line in file:
    print(line.strip())
```

```
Pocket 1 :
Score :         0.924
Druggability Score :    0.535
Number of Alpha Spheres :       95
Total SASA :    9.960
Polar SASA :    7.544
Apolar SASA :   2.415
Volume :        493.527
Mean local hydrophobic density :        21.440
Mean alpha sphere radius :      3.774
Mean alp. sph. solvent access :         0.449
Apolar alpha sphere proportion :        0.263
Hydrophobicity score:   26.957
Volume score:    4.348
Polarity score:  14
Charge score :   4
Proportion of polar atoms:      49.020
Alpha sphere density :  5.127
Cent. of mass - Alpha Sphere max dist:  11.895
Flexibility :   0.031

Pocket 2 :
Score :         0.323
Druggability Score :    0.615
Number of Alpha Spheres :       83
Total SASA :    157.049
Polar SASA :    99.083
Apolar SASA :   57.966
Volume :        479.790
Mean local hydrophobic density :        15.053
Mean alpha sphere radius :      3.795
Mean alp. sph. solvent access :         0.423
```

```
Apolar alpha sphere proportion :        0.229
Hydrophobicity score:   21.000
Volume score:    4.100
Polarity score:  6
Charge score :   1
Proportion of polar atoms:      54.167
Alpha sphere density :  5.868
Cent. of mass - Alpha Sphere max dist:  17.292
Flexibility :   0.090

Pocket 3 :
Score :         0.308
Druggability Score :    0.848
Number of Alpha Spheres :       67
Total SASA :    172.894
Polar SASA :    69.038
Apolar SASA :   103.856
Volume :        695.777
Mean local hydrophobic density :        23.312
Mean alpha sphere radius :      3.936
Mean alp. sph. solvent access :         0.537
Apolar alpha sphere proportion :        0.478
Hydrophobicity score:   29.350
Volume score:    4.450
Polarity score:  9
Charge score :   0
Proportion of polar atoms:      37.736
Alpha sphere density :  6.774
Cent. of mass - Alpha Sphere max dist:  19.126
Flexibility :   0.087

Pocket 4 :
Score :         0.170
Druggability Score :    0.001
Number of Alpha Spheres :       37
Total SASA :    77.069
Polar SASA :    32.387
Apolar SASA :   44.682
Volume :        291.443
Mean local hydrophobic density :        5.000
Mean alpha sphere radius :      3.776
Mean alp. sph. solvent access :         0.544
Apolar alpha sphere proportion :        0.162
Hydrophobicity score:   29.100
Volume score:    3.700
Polarity score:  4
Charge score :   2
Proportion of polar atoms:      44.444
Alpha sphere density :  3.532
Cent. of mass - Alpha Sphere max dist:  9.721
Flexibility :   0.067

Pocket 5 :
Score :         0.124
Druggability Score :    0.016
Number of Alpha Spheres :       39
Total SASA :    77.117
Polar SASA :    31.227
Apolar SASA :   45.890
Volume :        303.785
Mean local hydrophobic density :        16.000
Mean alpha sphere radius :      4.001
Mean alp. sph. solvent access :         0.587
Apolar alpha sphere proportion :        0.436
Hydrophobicity score:   45.154
Volume score:    5.000
Polarity score:  4
Charge score :   1
Proportion of polar atoms:      42.857
Alpha sphere density :  2.914
Cent. of mass - Alpha Sphere max dist:  6.045
Flexibility :   0.084

Pocket 6 :
Score :         0.116
Druggability Score :    0.002
Number of Alpha Spheres :       16
Total SASA :    66.472
Polar SASA :    19.375
Apolar SASA :   47.097
Volume :        273.670
Mean local hydrophobic density :        9.000
Mean alpha sphere radius :      3.896
```

```
Mean alp. sph. solvent access :         0.607
Apolar alpha sphere proportion :        0.625
Hydrophobicity score:   32.429
Volume score:    3.714
Polarity score:  3
Charge score :   -1
Proportion of polar atoms:      30.000
Alpha sphere density :  3.763
Cent. of mass - Alpha Sphere max dist:  6.233
Flexibility :   0.132

Pocket 7 :
Score :         0.100
Druggability Score :    0.006
Number of Alpha Spheres :       23
Total SASA :    55.512
Polar SASA :    20.490
Apolar SASA :   35.021
Volume :        180.286
Mean local hydrophobic density :        9.000
Mean alpha sphere radius :      3.700
Mean alp. sph. solvent access :         0.507
Apolar alpha sphere proportion :        0.435
Hydrophobicity score:   18.714
Volume score:    4.857
Polarity score:  5
Charge score :   1
Proportion of polar atoms:      38.889
Alpha sphere density :  2.712
Cent. of mass - Alpha Sphere max dist:  7.423
Flexibility :   0.082

Pocket 8 :
Score :         0.096
Druggability Score :    0.040
Number of Alpha Spheres :       46
Total SASA :    120.073
Polar SASA :    45.200
Apolar SASA :   74.873
Volume :        356.338
Mean local hydrophobic density :        15.889
Mean alpha sphere radius :      3.856
Mean alp. sph. solvent access :         0.467
Apolar alpha sphere proportion :        0.391
Hydrophobicity score:   40.909
Volume score:    4.545
Polarity score:  3
Charge score :   1
Proportion of polar atoms:      35.714
Alpha sphere density :  4.130
Cent. of mass - Alpha Sphere max dist:  9.926
Flexibility :   0.114

Pocket 9 :
Score :         0.094
Druggability Score :    0.626
Number of Alpha Spheres :       59
Total SASA :    149.738
Polar SASA :    50.713
Apolar SASA :   99.026
Volume :        515.888
Mean local hydrophobic density :        30.188
Mean alpha sphere radius :      3.879
Mean alp. sph. solvent access :         0.490
Apolar alpha sphere proportion :        0.542
Hydrophobicity score:   9.812
Volume score:    4.375
Polarity score:  8
Charge score :   2
Proportion of polar atoms:      41.667
Alpha sphere density :  4.421
Cent. of mass - Alpha Sphere max dist:  10.792
Flexibility :   0.146

Pocket 10 :
Score :         0.082
Druggability Score :    0.001
Number of Alpha Spheres :       23
Total SASA :    53.189
Polar SASA :    30.244
Apolar SASA :   22.945
Volume :        215.752
Mean local hydrophobic density :        2.000
```

Mean alpha sphere radius : 3.912
Mean alp. sph. solvent access : 0.482
Apolar alpha sphere proportion : 0.130
Hydrophobicity score: 18.091
Volume score: 4.182
Polarity score: 5
Charge score : -1
Proportion of polar atoms: 50.000
Alpha sphere density : 2.367
Cent. of mass - Alpha Sphere max dist: 5.442
Flexibility : 0.070

Pocket 11 :
Score : 0.067
Druggability Score : 0.005
Number of Alpha Spheres : 33
Total SASA : 98.192
Polar SASA : 49.907
Apolar SASA : 48.285
Volume : 230.851
Mean local hydrophobic density : 9.000
Mean alpha sphere radius : 3.640
Mean alp. sph. solvent access : 0.400
Apolar alpha sphere proportion : 0.303
Hydrophobicity score: 20.091
Volume score: 3.818
Polarity score: 5
Charge score : 2
Proportion of polar atoms: 50.000
Alpha sphere density : 3.626
Cent. of mass - Alpha Sphere max dist: 8.639
Flexibility : 0.149

Pocket 12 :
Score : 0.066
Druggability Score : 0.000
Number of Alpha Spheres : 18
Total SASA : 70.906
Polar SASA : 40.715
Apolar SASA : 30.191
Volume : 276.108
Mean local hydrophobic density : 3.000
Mean alpha sphere radius : 3.912
Mean alp. sph. solvent access : 0.764
Apolar alpha sphere proportion : 0.222
Hydrophobicity score: -5.625
Volume score: 4.000
Polarity score: 6
Charge score : -2
Proportion of polar atoms: 42.857
Alpha sphere density : 3.259
Cent. of mass - Alpha Sphere max dist: 6.493
Flexibility : 0.248

Pocket 13 :
Score : 0.061
Druggability Score : 0.000
Number of Alpha Spheres : 16
Total SASA : 53.881
Polar SASA : 32.144
Apolar SASA : 21.737
Volume : 207.140
Mean local hydrophobic density : 0.000
Mean alpha sphere radius : 3.859
Mean alp. sph. solvent access : 0.483
Apolar alpha sphere proportion : 0.000
Hydrophobicity score: 6.857
Volume score: 3.714
Polarity score: 4
Charge score : 0
Proportion of polar atoms: 46.667
Alpha sphere density : 2.604
Cent. of mass - Alpha Sphere max dist: 5.709
Flexibility : 0.040

Pocket 14 :
Score : 0.052
Druggability Score : 0.056
Number of Alpha Spheres : 34
Total SASA : 106.933
Polar SASA : 52.590
Apolar SASA : 54.343
Volume : 337.196

Mean local hydrophobic density : 19.000
Mean alpha sphere radius : 3.914
Mean alp. sph. solvent access : 0.594
Apolar alpha sphere proportion : 0.588
Hydrophobicity score: 16.182
Volume score: 4.182
Polarity score: 5
Charge score : 0
Proportion of polar atoms: 40.000
Alpha sphere density : 3.739
Cent. of mass - Alpha Sphere max dist: 9.922
Flexibility : 0.107

Pocket 15 :
Score : 0.049
Druggability Score : 0.001
Number of Alpha Spheres : 30
Total SASA : 105.156
Polar SASA : 68.927
Apolar SASA : 36.229
Volume : 305.590
Mean local hydrophobic density : 8.000
Mean alpha sphere radius : 3.764
Mean alp. sph. solvent access : 0.478
Apolar alpha sphere proportion : 0.300
Hydrophobicity score: 32.600
Volume score: 4.000
Polarity score: 6
Charge score : -1
Proportion of polar atoms: 44.444
Alpha sphere density : 3.793
Cent. of mass - Alpha Sphere max dist: 8.473
Flexibility : 0.126

Pocket 16 :
Score : 0.038
Druggability Score : 0.002
Number of Alpha Spheres : 26
Total SASA : 87.662
Polar SASA : 35.734
Apolar SASA : 51.928
Volume : 386.685
Mean local hydrophobic density : 6.000
Mean alpha sphere radius : 3.878
Mean alp. sph. solvent access : 0.578
Apolar alpha sphere proportion : 0.269
Hydrophobicity score: 34.875
Volume score: 4.875
Polarity score: 5
Charge score : 2
Proportion of polar atoms: 40.909
Alpha sphere density : 3.155
Cent. of mass - Alpha Sphere max dist: 8.944
Flexibility : 0.084

Pocket 17 :
Score : 0.027
Druggability Score : 0.005
Number of Alpha Spheres : 22
Total SASA : 93.222
Polar SASA : 40.087
Apolar SASA : 53.136
Volume : 316.036
Mean local hydrophobic density : 12.000
Mean alpha sphere radius : 3.913
Mean alp. sph. solvent access : 0.541
Apolar alpha sphere proportion : 0.591
Hydrophobicity score: 20.750
Volume score: 4.500
Polarity score: 6
Charge score : 0
Proportion of polar atoms: 47.619
Alpha sphere density : 3.575
Cent. of mass - Alpha Sphere max dist: 8.418
Flexibility : 0.199

Pocket 18 :
Score : 0.023
Druggability Score : 0.005
Number of Alpha Spheres : 24
Total SASA : 64.654
Polar SASA : 23.594
Apolar SASA : 41.059

Volume : 215.579
Mean local hydrophobic density : 10.000
Mean alpha sphere radius : 3.658
Mean alp. sph. solvent access : 0.416
Apolar alpha sphere proportion : 0.458
Hydrophobicity score: 50.889
Volume score: 3.889
Polarity score: 2
Charge score : 1
Proportion of polar atoms: 43.750
Alpha sphere density : 2.216
Cent. of mass - Alpha Sphere max dist: 6.211
Flexibility : 0.081

Pocket 19 :
Score : 0.004
Druggability Score : 0.000
Number of Alpha Spheres : 34
Total SASA : 97.832
Polar SASA : 50.735
Apolar SASA : 47.098
Volume : 258.882
Mean local hydrophobic density : 4.000
Mean alpha sphere radius : 3.867
Mean alp. sph. solvent access : 0.559
Apolar alpha sphere proportion : 0.147
Hydrophobicity score: 33.889
Volume score: 3.333
Polarity score: 3
Charge score : 1
Proportion of polar atoms: 45.455
Alpha sphere density : 2.835
Cent. of mass - Alpha Sphere max dist: 7.099
Flexibility : 0.045

Pocket 20 :
Score : -0.001
Druggability Score : 0.001
Number of Alpha Spheres : 21
Total SASA : 90.847
Polar SASA : 30.466
Apolar SASA : 60.381
Volume : 299.652
Mean local hydrophobic density : 9.000
Mean alpha sphere radius : 4.087
Mean alp. sph. solvent access : 0.603
Apolar alpha sphere proportion : 0.476
Hydrophobicity score: 25.750
Volume score: 4.125
Polarity score: 4
Charge score : 0
Proportion of polar atoms: 40.000
Alpha sphere density : 3.251
Cent. of mass - Alpha Sphere max dist: 7.450
Flexibility : 0.080

Pocket 21 :
Score : -0.002
Druggability Score : 0.000
Number of Alpha Spheres : 23
Total SASA : 114.955
Polar SASA : 89.595
Apolar SASA : 25.360
Volume : 358.544
Mean local hydrophobic density : 0.000
Mean alpha sphere radius : 3.920
Mean alp. sph. solvent access : 0.533
Apolar alpha sphere proportion : 0.000
Hydrophobicity score: -5.375
Volume score: 4.375
Polarity score: 7
Charge score : 2
Proportion of polar atoms: 72.727
Alpha sphere density : 4.083
Cent. of mass - Alpha Sphere max dist: 8.764
Flexibility : 0.137

Pocket 22 :
Score : -0.013
Druggability Score : 0.000
Number of Alpha Spheres : 18
Total SASA : 89.372
Polar SASA : 53.143

```
Apolar SASA :   36.229
Volume :        317.468
Mean local hydrophobic density :        3.000
Mean alpha sphere radius :      4.007
Mean alp. sph. solvent access :         0.567
Apolar alpha sphere proportion :        0.222
Hydrophobicity score:   26.444
Volume score:    4.000
Polarity score:  4
Charge score :   1
Proportion of polar atoms:      42.105
Alpha sphere density :  3.200
Cent. of mass - Alpha Sphere max dist:  8.017
Flexibility :   0.158

Pocket 23 :
Score :         -0.015
Druggability Score :    0.000
Number of Alpha Spheres :       19
Total SASA :    98.373
Polar SASA :    62.145
Apolar SASA :   36.229
Volume :        333.904
Mean local hydrophobic density :        0.000
Mean alpha sphere radius :      4.018
Mean alp. sph. solvent access :         0.736
Apolar alpha sphere proportion :        0.053
Hydrophobicity score:   0.143
Volume score:    4.143
Polarity score:  6
Charge score :   -2
Proportion of polar atoms:      50.000
Alpha sphere density :  3.504
Cent. of mass - Alpha Sphere max dist:  6.972
Flexibility :   0.329

Pocket 24 :
Score :         -0.015
Druggability Score :    0.035
Number of Alpha Spheres :       36
Total SASA :    105.121
Polar SASA :    50.956
Apolar SASA :   54.165
Volume :        314.308
Mean local hydrophobic density :        20.000
Mean alpha sphere radius :      3.935
Mean alp. sph. solvent access :         0.440
Apolar alpha sphere proportion :        0.583
Hydrophobicity score:   25.700
Volume score:    4.000
Polarity score:  4
Charge score :   2
Proportion of polar atoms:      41.667
Alpha sphere density :  2.846
Cent. of mass - Alpha Sphere max dist:  8.157
Flexibility :   0.076

Pocket 25 :
Score :         -0.023
Druggability Score :    0.005
Number of Alpha Spheres :       26
Total SASA :    93.656
Polar SASA :    35.690
Apolar SASA :   57.966
Volume :        305.760
Mean local hydrophobic density :        11.000
Mean alpha sphere radius :      3.949
Mean alp. sph. solvent access :         0.631
Apolar alpha sphere proportion :        0.462
Hydrophobicity score:   24.429
Volume score:    4.571
Polarity score:  4
Charge score :   1
Proportion of polar atoms:      34.783
Alpha sphere density :  2.826
Cent. of mass - Alpha Sphere max dist:  7.508
Flexibility :   0.129

Pocket 26 :
Score :         -0.040
Druggability Score :    0.020
Number of Alpha Spheres :       55
Total SASA :    197.532
```

```
Polar SASA :    47.786
Apolar SASA :   149.746
Volume :        676.425
Mean local hydrophobic density :        13.375
Mean alpha sphere radius :      4.011
Mean alp. sph. solvent access :         0.557
Apolar alpha sphere proportion :        0.291
Hydrophobicity score:   14.467
Volume score:    3.800
Polarity score:  8
Charge score :   -2
Proportion of polar atoms:      34.146
Alpha sphere density :  4.940
Cent. of mass - Alpha Sphere max dist:  11.982
Flexibility :   0.043

Pocket 27 :
Score :         -0.089
Druggability Score :    0.000
Number of Alpha Spheres :       22
Total SASA :    121.997
Polar SASA :    68.861
Apolar SASA :   53.136
Volume :        404.828
Mean local hydrophobic density :        4.000
Mean alpha sphere radius :      4.034
Mean alp. sph. solvent access :         0.488
Apolar alpha sphere proportion :        0.227
Hydrophobicity score:   -1.300
Volume score:    3.700
Polarity score:  8
Charge score :   0
Proportion of polar atoms:      52.381
Alpha sphere density :  3.501
Cent. of mass - Alpha Sphere max dist:  7.081
Flexibility :   0.056

Pocket 28 :
Score :         -0.116
Druggability Score :    0.001
Number of Alpha Spheres :       16
Total SASA :    81.657
Polar SASA :    32.144
Apolar SASA :   49.513
Volume :        263.256
Mean local hydrophobic density :        7.000
Mean alpha sphere radius :      4.122
Mean alp. sph. solvent access :         0.557
Apolar alpha sphere proportion :        0.500
Hydrophobicity score:   -11.833
Volume score:    5.333
Polarity score:  5
Charge score :   3
Proportion of polar atoms:      28.571
Alpha sphere density :  1.896
Cent. of mass - Alpha Sphere max dist:  5.439
Flexibility :   0.071
```

Now, let's find the pocket with the highest druggability score in each of the target proteins and store its features in a DataFrame for training a model in the next step.

In [ ]:

```python
import re

def identify_most_druggable_pocket(pocket_df):
  # Find the pocket with the highest druggability score
  pocket_df['Druggability Score'] = pocket_df['Druggability Score'].astype(float)
  best_pocket_df = pocket_df.loc[pocket_df['Druggability Score'].idxmax()]
  best_pocket_df = pd.DataFrame(best_pocket_df).T
  return best_pocket_df

def extract_features(pocket_info):
  pocket_data = []
  # Read the file content line by line
  with open(pocket_info, 'r') as file:
    current_pocket_info = {}
    for line in file:
        if "Pocket" in line:
            if current_pocket_info:
                pocket_data.append(current_pocket_info)
            current_pocket_info = {'Pocket': line.strip()}
        else:
            if ':' in line:
```

```
                key, value = line.split(':')
                current_pocket_info[key.strip()] = value.strip()

    # Append the last pocket information
    if current_pocket_info:
        pocket_data.append(current_pocket_info)

  # Convert the list of dictionaries to a DataFrame
  pocket_df = pd.DataFrame(pocket_data)
  pocket_df.drop(columns=['Pocket'], inplace=True)
  return pocket_df
```

In [ ]:
```
pocket_dataset = {}
for protein in proteins_list:
  pocket_info = f'{base_pdb_files}{protein}_out/{protein}_info.txt'
  pocket_df = extract_features(pocket_info)
  best_pocket_df = identify_most_druggable_pocket(pocket_df)
  pocket_dataset[protein] = best_pocket_df

# Combine all dataframes into one with a new column for the keys
dataset_fpocket = pd.concat(pocket_dataset.values(), keys=pocket_dataset.keys()).reset_index(level=0).rename(co
dataset_fpocket.rename(columns={'Key': 'pdb code'}, inplace=True)
```

Add the label's to the dataset

In [ ]:
```
# # Set the key column as the index our dataframe from intial csv file with pdb code and labels and the datafra
dataset_fpocket.set_index('pdb code', inplace=True)
# dataset.set_index('pdb code', inplace=True)

# # Merge the dataframes on the index to create a datset
# dataset_fpocket = dataset_fpocket.join(dataset, how='outer')
```

ML model to classify the binding pockets

Here, we are using random forest algorithm to train our model. However, you are free to experiment with supervised model for classification of your choice.

In [ ]:
```
import pandas as pd
from sklearn.metrics import classification_report, accuracy_score, roc_curve, roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier

# Separate features and labels
X = dataset_fpocket
y = labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Initialize the Random Forest model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

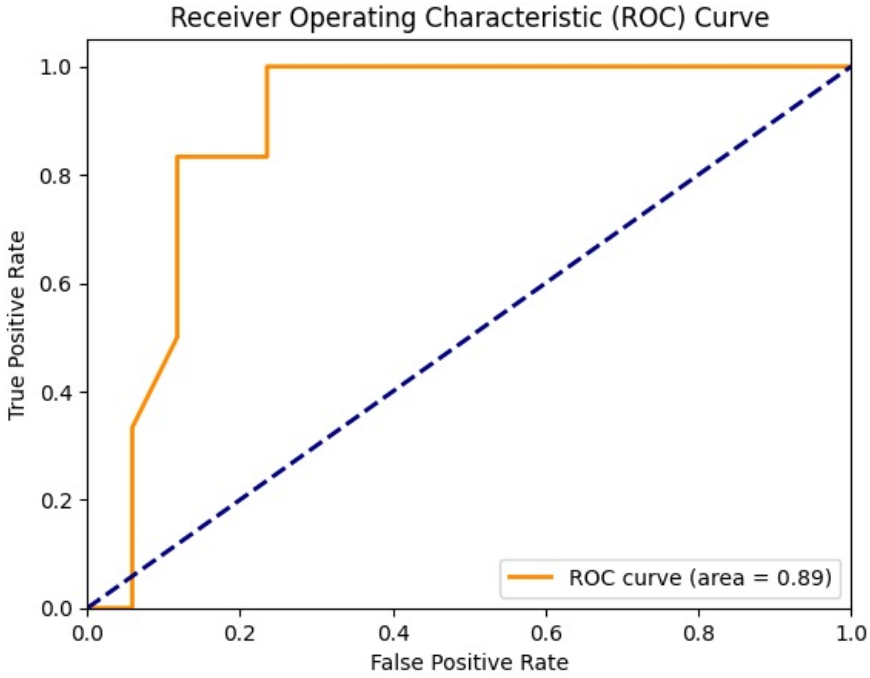
```
Accuracy: 0.8695652173913043
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.88      0.91        17
           1       0.71      0.83      0.77         6

    accuracy                           0.87        23
   macro avg       0.83      0.86      0.84        23
weighted avg       0.88      0.87      0.87        23
```

In [ ]:
```
# Calculate the ROC curve and AUC
y_prob = model.predict_proba(X_test)[:, 1]
fpr, tpr, thresholds = roc_curve(y_test, y_prob)
roc_auc = roc_auc_score(y_test, y_prob)
```

```
import matplotlib.pyplot as plt
# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



The model achieved an accuracy of 86.96%. The precision, recall, and F1-score were 0.94, 0.88, and 0.91 for class 1(highly druggable), and 0.71, 0.83, and 0.77 for class 1(less druggable), indicating strong performance overall but slightly lower precision for class 1.

Classifying Your Protein's Druggability

Let's find out if your protein is highly druggable or less druggable by following these steps. Assuming you already have the 3D structure of the protein in .pdb file format, we'll use the ML model to classify the protein pockets effectively.

1. Run Fpocket to Identify the Pockets
2. Idenitfy the pocket with highest druggable score
3. Extract the features of most druggable feature in the target
4. Use the above trained model to make a prediction

In case you do not have the 3D structure of your protein, don't worry! You have two options:

- Fetch the Structure Using PDB Code: You can use the `fetch_protein_structure` method defined above to fetch the structure using the PDB code of the protein.
- Predict the Structure Using ESMfold: If you have the protein sequence, you can use ESMfold to obtain the structure of the protein. For more information, you can refer to the DeepChem tutorial: Protein Structure Prediction with ESMFold.

By following these steps, you can systematically determine the druggability of a protein pocket, combining advanced computational tools like Fpocket with the predictive power of machine learning. This process provides a robust method for identifying promising drug targets.

In [ ]:
```
fetch_protein_structure('1mbn', '/content/1mbn/')
run_fpocket('/content/1mbn/1mbn.pdb')
output_dir = '/content/1mbn'
pdb_code = '1mbn'

target_pocket_info = f'{output_dir}{pdb_code}_out/{protein}_info.txt'
target_pocket_df = extract_features(pocket_info)
best_pocket_df = identify_most_druggable_pocket(pocket_df)
prediction = label_encoder.inverse_transform(model.predict(best_pocket_df))
print(prediction)
```

```
Downloading PDB structure '1mbn'...
['N']
```

The predicted label is 'N' which means protein '1mbn' isn't highly druggable. Hence, it shouldn't be used as a drug target.

ML Resources for Druggability Assessment

Here are some useful resources to help you understand the different ML methods mentioned in this tutorial:

Supervised Learning

Regression

- *Understanding* Decision Trees in Machine Learning
- Decision Tree Regression - An In-depth Guide
- Linear Regression: Introduction to Linear Regression
- Linear Regression Explained
- Random Forest: Random Forest in Machine Learning
- Random Forest: A Complete Guide for Machine Learning

Classification

- K-Nearest Neighbours Algorithm
- Support Vector Machine (SVM): Introduction to Support Vector Machines
- Natural Language Processing (NLP): NLP Overview
- Getting Started with NLP
- Bayesian Networks: Introduction to Bayesian Networks

Unsupervised Learning

Clustering

- K-Means Clustering: K-Means Clustering Algorithm Explained
- K-Means Clustering with Python
- Hierarchical Clustering: Understanding Hierarchical Clustering
- Hidden Markov Model (HMM): Introduction to Hidden Markov Models

Feel free to explore these resources as you progress through the notebook to deepen your understanding of the ML methods used in druggability assessment.

References

1. Hopkins, A. L., & Groom, C. R. (2002). The druggable genome. *Nature Reviews Drug Discovery*, 1(9), 727-730. DOI: 10.1038/nrd892
2. Yu, L., Xue, L., Liu, F., Li, Y., Jing, R., & Luo, J. (2022). The applications of deep learning algorithms on in silico druggable proteins identification. Journal of Advanced Research, 41, 219-231. DOI: 10.1016/j.jare.2022.01.009
3. Hajduk, P. J., Huth, J. R., & Tse, C. (2005). Predicting protein druggability. *Drug Discovery Today*, 10(23-24), 1675-1682. DOI: 10.1016/S1359-6446(05)03624-203624-2)
4. Halgren, T. A. (2009). Identifying and characterizing binding sites and assessing druggability. *Journal of Chemical Information and Modeling*, 49(2), 377-389. DOI: 10.1021/ci800324m
5. Peters, J. U. (2013). Polypharmacology–foe or friend? *Journal of Medicinal Chemistry*, 56(22), 8955-8971. DOI: 10.1021/jm400856t
6. Ashley, E. A. (2016). Towards precision medicine. *Nature Reviews Genetics*, 17(9), 507-522. DOI: 10.1038/nrg.2016.86
7. Agoni, C., Olotu, F.A., Ramharack, P. et al. Druggability and drug-likeness concepts in drug design: are biomodelling and predictive tools having their say?. J Mol Model 26, 120 (2020). DOI: 10.1007/s00894-020-04385-6
8. Abi Hussein H, Geneix C, Petitjean M, Borrel A, Flatters D, Camproux AC. Global vision of druggability issues: applications and perspectives. Drug Discov Today. 22, 404–415 (2017)
9. Arrowsmith, J. Phase III and submission failures: 2007-2010. Nature Reviews Drug Discovery. 10(2), 1-2 (2011)
10. Excelra. (2024). Identifying Druggable Therapeutic Targets: Unveiling Promising Avenues in Drug Discovery. *Excelra White Paper*. Retrieved from https://www.excelra.com/whitepaper/identifying-druggable-therapeutic-targets-unveiling-promising-avenues-in-drug-discovery/

11. Aguti R, Gardini E, Bertazzo M, Decherchi S and Cavalli A (2022) Probabilistic Pocket Druggability Prediction via One-Class Learning. Front. Pharmacol. 13:870479. DOI: 10.3389/fphar.2022.870479)

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing this Tutorial

If you found this tutorial useful, please consider citing it as:

```
@manual{Bioinformatics,
 title={Druggability Assessment with Fpocket and Machine Learning},
 organization={DeepChem},
 author={Yadav, Anamika },
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Druggablity_Assessment_wit

 year={2024},
}
```

Protein Deep Learning

by David Ricardo Figueroa Blanco

In this tutorial we will compare protein sequences featurization such as one hot encoders and aminoacids composition. We will use some tools of DeepChem and additional packages to create a model to predict melting temperature of proteins (a good measurement of protein stability)

The melting temperature (MT) of a protein is a measurement of protein stability. This measure could vary from a big variety of experimental conditions, however, curated databases cand be found in literature https://aip.scitation.org/doi/10.1063/1.4947493. In this paper we can find a lot of thermodynamic information of proteins and therefore a big resource for the study of protein stability. Other information related with protein stability could be the change in Gibbs Free Energy
$\Delta\Delta G^\circ$
due to a mutation.

The study of protein stability is important in areas such as protein engineering and biocatalysis because catalytic efficiency could be directly related to the tertiary structure of the protein in study.

Setup

To run DeepChem within Colab, you'll need to run the following installation commands. This will take about 5 minutes to run to completion and install your environment. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install Anaconda on your local machine.

In [ ]:
```
!curl -Lo conda_installer.py https://raw.githubusercontent.com/deepchem/deepchem/master/scripts/colab_install.py
import conda_installer
conda_installer.install()
!/root/miniconda/bin/conda info -e
```

In [ ]:
```
!pip install --pre deepchem
!pip install propy3
```

Data extraction

In this cell, we download the dataset published in the paper https://aip.scitation.org/doi/10.1063/1.4947493 from the DeepChem dataset repository

In [1]:
```
import deepchem as dc
import os
from deepchem.utils import download_url
data_dir = dc.utils.get_data_dir()
download_url("https://deepchemdata.s3-us-west-1.amazonaws.com/datasets/pucci-proteins-appendixtable1.csv",dest_
print('Dataset Dowloaded at {}'.format(data_dir))
dataset_file = os.path.join(data_dir, "pucci-proteins-appendixtable1.csv")
```

```
Dataset Dowloaded at /tmp
```

A closer look of the dataset: Contains the PDBid and the respective mutation and change in thermodynamical properties in each studied protein

In [2]:
```
import pandas as pd
data = pd.read_csv(dataset_file)
data
```

Out[2]:

	Unnamed: 0	N	PDBid	Chain	RESN	RESwt	RESmut	ΔTmexp	Tmexp [wt]	ΔΔHmexp	...	ΔΔGexp(T)	T	N
0	NaN	1	1aky	A	8	VAL	ILE	-1.5	47.6	70	...	5.0	25	
1	NaN	2	1aky	A	48	GLN	GLU	-1.3	47.6	60	...	4.0	25	
2	NaN	3	1aky	A	77	THR	HIS	-1.1	47.6	130	...	9.0	25	
3	NaN	4	1aky	A	110	THR	HIS	-4.8	47.6	165	...	11.0	25	
4	NaN	5	1aky	A	169	ASN	ASP	-0.6	47.6	140	...	9.0	25	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1621	NaN	1622	5pti_m52l	A	15	LYS	SER	-1.3	91.7	-5	...	1.2	25	
1622	NaN	1623	5pti_m52l	A	15	LYS	THR	-1.1	91.7	-9	...	-3.6	25	
1623	NaN	1624	5pti_m52l	A	15	LYS	VAL	-6.3	91.7	4	...	4.7	25	
1624	NaN	1625	5pti_m52l	A	15	LYS	TRP	-7.5	91.7	17	...	8.5	25	
1625	NaN	1626	5pti_m52l	A	15	LYS	TYR	-6.6	91.7	4	...	4.6	25	

1626 rows × 23 columns

Here we extract a small DataFrame that only contains a unique PDBid code and its respective melting temperature

In [3]:
```
WT_Tm = data[['PDBid','Tmexp [wt]']]
WT_Tm.set_index('PDBid',inplace=True)
```

In [4]:
```
WT_Tm
```

Out[4]:

PDBid	Tmexp [wt]
1aky	47.6
1aky	47.6
1aky	47.6
1aky	47.6
1aky	47.6
...	...
5pti_m52l	91.7
5pti_m52l	91.7
5pti_m52l	91.7
5pti_m52l	91.7
5pti_m52l	91.7

1626 rows × 1 columns

Here we create a dictionary that contains as keys, the pdbid of each protein and as values, the wild type melting temperature

In [5]:
```
dict_WT_TM = {}
for k,v in WT_Tm.itertuples():
    if(k not in dict_WT_TM):
        dict_WT_TM[k]=float(v)
```

Here we extract proteins with mutations reported only in chain A

In [6]:
```
pdbs = data[data['PDBid'].str.len()<5]
pdbs = pdbs[pdbs['Chain'] == "A"]
```

In [7]:
```
pdbs[['RESN','RESwt','RESmut']]
```

Out[7]:

	RESN	RESwt	RESmut
0	8	VAL	ILE
1	48	GLN	GLU
2	77	THR	HIS
3	110	THR	HIS
4	169	ASN	ASP
...	...	...	...
1604	36	GLY	ALA
1605	36	GLY	SER
1606	37	GLY	ALA
1607	39	ARG	ALA
1608	46	LYS	ALA

1509 rows × 3 columns

This cell extracts the total number of mutations and changes in MT. In addition, we use a dicctionary to convert the residue mutation in a one letter code.

In [8]:
```
alls=[]
for resnum,wt in pdbs[['RESN','RESwt','RESmut','PDBid','ΔTmexp']].iteritems():
    alls.append(wt.values)
d = {'CYS': 'C', 'ASP': 'D', 'SER': 'S', 'GLN': 'Q', 'LYS': 'K',
     'ILE': 'I', 'PRO': 'P', 'THR': 'T', 'PHE': 'F', 'ASN': 'N',
     'GLY': 'G', 'HIS': 'H', 'LEU': 'L', 'ARG': 'R', 'TRP': 'W',
     'ALA': 'A', 'VAL':'V', 'GLU': 'E', 'TYR': 'Y', 'MET': 'M'}
resnum=alls[0]
wt=[d[x.strip()] for x in alls[1]] # extract the Wildtype aminoacid with one letter code
mut=[d[x.strip()] for x in alls[2]] # extract the Mutation aminoacid with one letter code
codes=alls[3] # PDB code
tms=alls[4] # Melting temperature
```

In [9]:
```
print("pdbid {}, WT-AA {}, Resnum {}, MUT-AA {},DeltaTm {}".format(codes[0],wt[0],resnum[0],mut[0],tms[0]))
```

```
pdbid 1aky, WT-AA V, Resnum 8, MUT-AA I,DeltaTm -1.5
```

PDB Download

Here we download all the pdbs by PDBID using the pdbfixer tool

In [10]:
```
from pdbfixer import PDBFixer
from simtk.openmm.app import PDBFile
```

In [ ]:
```
!mkdir PDBs
```

Using the fixer from pdbfixer we download each protein from its PDB code and fix some common problems present in the Protein Data Bank Files. This process will take around 15 minutes and 100 Mb. The use of the PDBFixer can be found in https://htmlpreview.github.io/?https://github.com/openmm/pdbfixer/blob/master/Manual.html . In our case, we download the pdb file from the pdb code and perform some curation such as find Nonstandar or missing residues, fix missing atoms

In [ ]:
```
import os
import time
t0 = time.time()

downloaded = os.listdir("PDBs")
PDBs_ids= set(pdbs['PDBid'])
pdb_list = []
print("Start Download ")
for pdbid in PDBs_ids:
    name=pdbid+".pdb"
    if(name in downloaded):
        continue
    try:
        fixer = PDBFixer(pdbid=pdbid)
        fixer.findMissingResidues()
        fixer.findNonstandardResidues()
        fixer.replaceNonstandardResidues()
        fixer.removeHeterogens(True)
```

```
        fixer.findMissingAtoms()
        fixer.addMissingAtoms()
        PDBFile.writeFile(fixer.topology, fixer.positions, open('./PDBs/%s.pdb' % (pdbid), 'w'),keepIds=True)
    except:
        print("Problem with {}".format(pdbid))
print("Total Time {}".format(time.time()-t0))
```

The following function help us to mutate a sequence denoted as A###B where A is the wildtype aminoacid, ### the position and, B the new aminoacid

In [11]:
```
import re
def MutateSeq(seq,Mutant):
    '''
    Mutate a sequence based on a string (Mutant) that has the notation :
    A###B where A is the wildtype aminoacid ### the position and B the mutation
    '''
    aalist = re.findall('([A-Z])([0-9]+)([A-Z])', Mutant)

    #(len(aalist)==1):
    newseq=seq
    listseq=list(newseq)
    for aas in aalist:
        wildAA = aas[0]
        pos = int(aas[1]) -1
        if(pos >= len(listseq)):
            print("Mutation not in the range of the protein")
            return None
        MutAA = aas[-1]

        if(listseq[pos]==wildAA):

            listseq[pos]=MutAA

        else:
            #print("WildType AA does not match")
            return None
    return("".join(listseq))
```

The following function help us to identify a sequence of aminoacids base on PDB structures

In [12]:
```
from Bio.PDB.PDBParser import PDBParser
from Bio.PDB.Polypeptide import PPBuilder
ppb=PPBuilder()
def GetSeqFromPDB(pdbid):
    structure = PDBParser().get_structure(pdbid.split(".")[0], 'PDBs/{}'.format(pdbid))
    seqs=[]
    return ppb.build_peptides(structure)
```

Some examples of the described functions : GetSeqFromPDB. Take one pdb that we previously downloaded and extract the sequence in one letter code

In [13]:
```
import warnings; warnings.simplefilter('ignore')
test="1ezm"
print(test)
seq = GetSeqFromPDB("{}.pdb".format(test))[0].get_sequence()
print("Original Sequence")
print(seq)
```

```
1ezm
Original Sequence
AEAGGPGGNQKIGKYTYGSDYGPLIVNDRCEMDDGNVITVDMNSSTDDSKTTPFRFACPTNTYKQVNGAYSPLNDAHFFGGVVFKLYRDWFGTSPLTHKLYMKVHYGRSVEN
AYWDGTAMLFGDGATMFYPLVSLDVAAHEVSHGFTEQNSGLIYRGQSGGMNEAFSDMAGEAAEFYMRGKNDFLIGYDIKKGSGALRYMDQPSRDGRSIDNASQYYNGIDVHH
SSGVYNRAFYLLANSPGWDTRKAFEVFVDANRYYWTATSNYNSGACGVIRSAQNRNYSAADVTRAFSTVGVTCPSAL
```

Information about the mutation

In [14]:
```
informSeq=GetSeqFromPDB(test+".pdb")[0].__repr__()
print("Seq information",informSeq)
start = re.findall('[0-9]+',informSeq)[0]
print("Reported Mutation {}{}{}".format("R",179,"A"))
numf =179 - int(start) + 1  # fix some cases of negative aminoacid numbers
```

```
Seq information <Polypeptide start=1 end=301>
Reported Mutation R179A
```

Mutation in the sequence.

In [15]:
```
mutfinal = "R{}A".format(numf)
print("Real Mutation = ",mutfinal)
mutseq = MutateSeq(seq,mutfinal)
```

```
print(mutseq)
```

```
Real Mutation =  R179A
AEAGGPGGNQKIGKYTYGSDYGPLIVNDRCEMDDGNVITVDMNSSTDDSKTTPFRFACPTNTYKQVNGAYSPLNDAHFFGGVVFKLYRDWFGTSPLTHKLYMKVHYGRSVEN
AYWDGTAMLFGDGATMFYPLVSLDVAAHEVSHGFTEQNSGLIYRGQSGGMNEAFSDMAGEAAEFYMAGKNDFLIGYDIKKGSGALRYMDQPSRDGRSIDNASQYYNGIDVHH
SSGVYNRAFYLLANSPGWDTRKAFEVFVDANRYYWTATSNYNSGACGVIRSAQNRNYSAADVTRAFSTVGVTCPSAL
```

In this for loop we extract the sequences of all proteins in the dataset. In addition we created the mutated sequences and append the change in MT. In some cases, gaps in pdbs will cause that mutateSeq function fails, therefore this entries will be avoided. This is an important step in the whole process because creates a final tabulated data that contains the sequence and the Melting temperature (our label)

In [16]:
```
information = {}
count = 1
failures=[]
for code,tm,numr,wt_val,mut_val in zip(codes,tms,resnum,wt,mut):
    count += 1
    seq = GetSeqFromPDB("{}.pdb".format(code))[0].get_sequence()
    mutfinal="WT"
    if("{}-{}".format(code,mutfinal) not in information):
        informSeq=GetSeqFromPDB(code+".pdb")[0].__repr__()
        start = re.findall('[-0-9]+',informSeq)[0]
    if(int(start)<0):
        numf =numr - int(start) # if start is negative 0 is not used as resnumber
    else:
        numf =numr - int(start) + 1
    mutfinal = "{}{}{}".format(wt_val,numf,mut_val)
    mutseq = MutateSeq(seq,mutfinal)
    if(mutseq==None):
        failures.append((code,mutfinal))
        continue
    information["{}-{}".format(code,mutfinal)]=[mutseq,dict_WT_TM[code]-float(tm)]
```

Deep Learning and Machine Learning Models using proteins sequences

In [17]:
```
import deepchem as dc
import tensorflow as tf
```

Here we extract two list, sequences (data) and melting temperature (label)

In [18]:
```
seq_list=[]
deltaTm=[]
for i in information.values():
    seq_list.append(i[0])
    deltaTm.append(i[1])
```

In [19]:
```
max_seq= 0
for i in seq_list:
    if(len(i)>max_seq):
        max_seq=len(i)
```

Here we use a OneHotFeaturizer in order to convert protein sequences in numeric arrays

In [20]:
```
codes = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L',
         'M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
OneHotFeaturizer = dc.feat.OneHotFeaturizer(codes,max_length=max_seq)
features = OneHotFeaturizer.featurize(seq_list)
```

Note that the OneHotFeaturizer produces a matrix that contains the OneHot Vector for each sequence.

In [21]:
```
features_vector  = []
for i in range(len(features)):
    features_vector.append(features[i].flatten())
```

In [22]:
```
dc_dataset = dc.data.NumpyDataset(X=features_vector,y=deltaTm)
```

In [23]:
```
dc_dataset
```

Out[23]:
```
<NumpyDataset X.shape: (1497, 13188), y.shape: (1497,), w.shape: (1497,), task_names: [0]>
```

Here we create a spliiter to perform a tran_test split of the dataset

In [41]:
```
from deepchem import splits
splitter = splits.RandomSplitter()
train, test  = splitter.train_test_split(dc_dataset,seed=42)
```

Here we create a neuronal network using tensorflow-keras and using a loss function of "MAE" to evaluate the regression result.

In [42]:
```
import tensorflow.keras as keras
#from  keras import layers
model = tf.keras.Sequential([
    keras.layers.Dense(units=32, activation='relu', input_shape=dc_dataset.X.shape[1:]),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(units=32, activation='relu'),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(units=1),
])
```

In [43]:
```
model.compile(loss='mae', optimizer='adam')

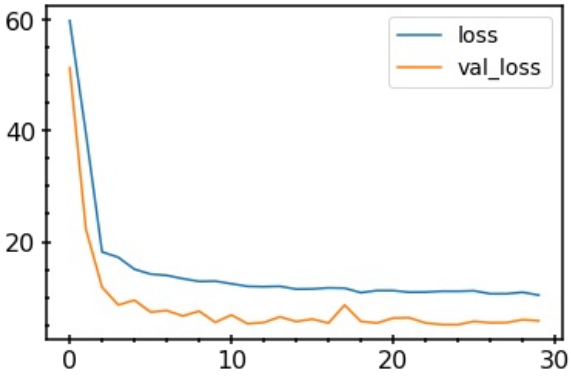
print(model.summary())

history = model.fit(
    train.X, train.y,
    validation_data=(test.X,test.y),
    batch_size=100,
    epochs=30,
)

## perform a plot of loss vs epochs
import matplotlib.pyplot as plt
history_df = pd.DataFrame(history.history)
history_df[['loss', 'val_loss']].plot()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_6 (Dense)              (None, 32)                422048
_________________________________________________________________
dropout_4 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_7 (Dense)              (None, 32)                1056
_________________________________________________________________
dropout_5 (Dropout)          (None, 32)                0
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 33
=================================================================
Total params: 423,137
Trainable params: 423,137
Non-trainable params: 0
_________________________________________________________________
None
Epoch 1/30
12/12 [==============================] - 1s 20ms/step - loss: 62.4284 - val_loss: 51.3894
Epoch 2/30
12/12 [==============================] - 0s 10ms/step - loss: 45.6483 - val_loss: 22.2757
Epoch 3/30
12/12 [==============================] - 0s 12ms/step - loss: 19.4803 - val_loss: 11.7996
Epoch 4/30
12/12 [==============================] - 0s 11ms/step - loss: 18.0858 - val_loss: 8.6031
Epoch 5/30
12/12 [==============================] - 0s 11ms/step - loss: 14.5904 - val_loss: 9.4577
Epoch 6/30
12/12 [==============================] - 0s 10ms/step - loss: 14.3444 - val_loss: 7.3325
Epoch 7/30
12/12 [==============================] - 0s 10ms/step - loss: 13.6787 - val_loss: 7.5799
Epoch 8/30
12/12 [==============================] - 0s 10ms/step - loss: 14.0674 - val_loss: 6.6186
Epoch 9/30
12/12 [==============================] - 0s 11ms/step - loss: 12.8215 - val_loss: 7.4920
Epoch 10/30
12/12 [==============================] - 0s 11ms/step - loss: 13.0748 - val_loss: 5.4614
Epoch 11/30
12/12 [==============================] - 0s 11ms/step - loss: 12.3646 - val_loss: 6.7943
Epoch 12/30
12/12 [==============================] - 0s 10ms/step - loss: 11.5250 - val_loss: 5.2098
Epoch 13/30
12/12 [==============================] - 0s 11ms/step - loss: 11.5254 - val_loss: 5.4262
Epoch 14/30
12/12 [==============================] - 0s 11ms/step - loss: 12.1312 - val_loss: 6.4607
Epoch 15/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4129 - val_loss: 5.6157
Epoch 16/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4406 - val_loss: 6.0425
Epoch 17/30
12/12 [==============================] - 0s 12ms/step - loss: 11.4762 - val_loss: 5.3330
Epoch 18/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4820 - val_loss: 8.5933
Epoch 19/30
12/12 [==============================] - 0s 11ms/step - loss: 11.1607 - val_loss: 5.6460
Epoch 20/30
12/12 [==============================] - 0s 12ms/step - loss: 11.2637 - val_loss: 5.3362
Epoch 21/30
12/12 [==============================] - 0s 11ms/step - loss: 11.4390 - val_loss: 6.2368
Epoch 22/30
12/12 [==============================] - 0s 10ms/step - loss: 10.8070 - val_loss: 6.2875
Epoch 23/30
12/12 [==============================] - 0s 11ms/step - loss: 11.1277 - val_loss: 5.3496
Epoch 24/30
12/12 [==============================] - 0s 10ms/step - loss: 11.1626 - val_loss: 5.0670
Epoch 25/30
12/12 [==============================] - 0s 10ms/step - loss: 10.9386 - val_loss: 5.0440
Epoch 26/30
12/12 [==============================] - 0s 14ms/step - loss: 11.0402 - val_loss: 5.6448
Epoch 27/30
12/12 [==============================] - 0s 14ms/step - loss: 10.7278 - val_loss: 5.3868
Epoch 28/30
12/12 [==============================] - 0s 13ms/step - loss: 10.6073 - val_loss: 5.4149
Epoch 29/30
12/12 [==============================] - 0s 12ms/step - loss: 11.0812 - val_loss: 5.9349
Epoch 30/30
12/12 [==============================] - 0s 14ms/step - loss: 10.2336 - val_loss: 5.7489
```

Out[43]: <AxesSubplot:>



DeepChem Keras Model

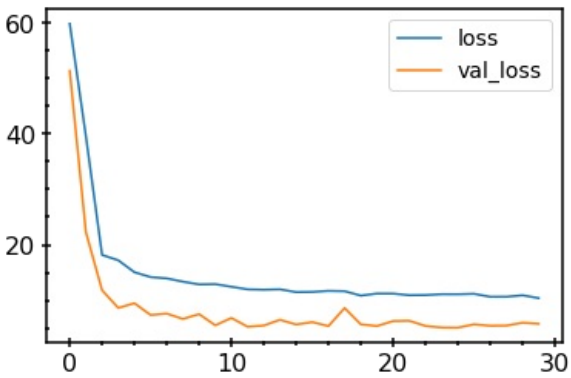
Note that DeepChem Keras model allows to create a deepchem model based on the previously built model of keras. All the information in the training can be access with model.model.history

In [44]:
```
model_dc = dc.models.KerasModel(model, dc.models.losses.L1Loss())
model_dc.fit(train)
```

Out[44]: 10.399123382568359

In [45]:
```
History_df = pd.DataFrame(model_dc.model.history.history)
History_df[['loss', 'val_loss']].plot()
```

Out[45]: <AxesSubplot:>



In [48]:
```
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)
print('test dataset R2:', model_dc.evaluate(test, [metric]))
```

```
test dataset R2: {'pearson_r2_score': 0.7125054529591869}
```

Examples of Classic ML models

Finally, we will compare others descriptros such as AAcomposition and Composition,transition and distribution of AA (https://www.pnas.org/content/92/19/8700)

In [49]:
```
from propy import PyPro
```

In the following cell, we are creating and pyPro Object based on the protein sequence. Pypro allows us the calculation of amino acid composition vectors
Here we create a list with the aminoacido composition vector for each sequence used in the previous model.

In [50]:
```
import numpy as np
aaComplist = []
CTDList =[]
for seq in seq_list:
    Obj = PyPro.GetProDes(seq)
    aaComplist.append(np.array(list(Obj.GetAAComp().values())))
    CTDList.append(np.array(list(Obj.GetCTD().values())))
```

In [51]:
```
dc_dataset_aacomp = dc.data.NumpyDataset(X=aaComplist,y=deltaTm)
dc_dataset_ctd = dc.data.NumpyDataset(X=CTDList,y=deltaTm)
```

Evaluation of classical machine learning models

In the following cell we create a randomForest Regressor and the deepchem SklearnModel. As it was used in the DL models, here we use "MAE" score to evaluate the results of the regression

In [52]:
```
from deepchem import splits
splitter = splits.RandomSplitter()
train, test  = splitter.train_test_split(dc_dataset_aacomp,seed=42)
from sklearn.ensemble import RandomForestRegressor
from deepchem.utils.evaluate import Evaluator
import pandas as pd
print("RandomForestRegressor")
seed = 42 # Set a random seed to get stable results
sklearn_model = RandomForestRegressor(n_estimators=100, max_features='sqrt')
sklearn_model.random_state = seed
model = dc.models.SklearnModel(sklearn_model)
model.fit(train)
metric = dc.metrics.Metric(dc.metrics.mae_score)
train_score = model.evaluate(train, [metric])
test_score = model.evaluate(test, [metric])
print("Train score is : {}".format(train_score))
print("Test score is : {}".format(test_score))
```

```
RandomForestRegressor
Train score is : {'mae_score': 1.7916551501995608}
Test score is : {'mae_score': 3.8967191996673947}
```

In the following cell we create a Suport Vector Regressor and the deepchem SklearnModel. As it was used in the DL models, here we use "MAE" score to evaluate the results of the regression

In [53]:
```
print("SupportVectorMachineRegressor")
from sklearn.svm import SVR
svr_sklearn = SVR(kernel="poly",degree=4)
svr_sklearn.random_state = seed
model = dc.models.SklearnModel(svr_sklearn)
model.fit(train)
metric = dc.metrics.Metric(dc.metrics.mae_score)
train_score = model.evaluate(train, [metric])
test_score = model.evaluate(test, [metric])
print("Train score is : {}".format(train_score))
print("Test score is : {}".format(test_score))
```

```
SupportVectorMachineRegressor
Train score is : {'mae_score': 3.275727325767219}
Test score is : {'mae_score': 4.058136267284038}
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

An Introduction to Antibody Design Using Protein Language Models

By Dhuvi Karthikeyan and Aaron Menezes

This DeepChem tutorial is designed to serve as a brief primer for antibody design via protein language models. Antibodies are immune proteins also known as immunoglobulins that are naturally produced in the body and bind/inactivate viruses and other pathogens. They are a valuable therapeutic and save lives in immune checkpoint inhibitors for cancer and as neutralizing antibodies for acute viral infection. In addition, they are well known outside the hospital walls for their ability to bind and stick to arbitrary molecular targets, a useful feature in the basic sciences and industrial biochemical facilities.

This tutorial aims to provide a quick overview of key immunology concepts needed to understand antibody structure and function in the broader context of the immune system. We make some assumptions with familiarity with large language models. Take a look at our other tuorial. For the sake of brevity we provide links on non-essential topics that point to external sources wherever possible. Follow along to learn more about the immune system, and protein language models for guided Ab design.

Note: This tutorial is loosely based on the 2023 Nature Biotechnology paper titled "Efficient evolution of human antibodies from general protein language models" [1] by Hie et al. We thank the authors for making their methods and data available and accessible.

1. Immunology 101

1.1 What is the immune system?

Like other body systems such as the digestive system or the cardiovascular system, the immune system is fundamentally a collection of specialized cells that operate together to accomplish a specific homeostatic function. The immune system is responsible for the protection of our body's vast resources (lipids, carbodhydrates, enzymes, proteins), especially against opportunistic threats of the outside world, such as viruses and bacteria. A very simple yet powerful component of the immune system is the skin, which keeps what's out, out and what's in, in. However, what happens when we get a cut/scrape or when we walk past someone who's coughing and accidentally breathe in the droplets? Thankfully, it turns out the immune system has a whole army of highly specialized white blood cells patrolling our blood and lymph nodes, ready to launch a multi-layered response against these and the countless other cases where foreign objects may enter our bodies. Through these white bloods cells, the immune system is able to accomplish its primary function, **self-nonself discrimination**, the accurate identification of molecules that are sufficiently dissimilar than those derived from a healthy person.

If you would like to learn more about the complex problem of self non-self discrimination, and appreciate the theory behind the immune system's organization and function, we recommend checking out the following works:

- A Theory of Self-Nonself Discrimination [2]
- The common sense of the self-nonself discrimination, 2005 [3]
- Conceptual aspects of self and nonself discrimination, 2011 [4]
- Self-Nonself Discrimination due to Immunological Nonlinearities: the Analysis of a Series of Models by Numerical Methods, 1987 [5]
- A biological context for the self-nonself discrimination and the regulation of effector class by the immune system, 2005 [6]

1.2 Innate vs. Adaptive Immune System

Remarkably, these white blood cells operate independently, adding and subtracting from the local context (tissue microenvironment), orchestrating cohesive responses in a completely decentralized manner. Individual white blood cells interact with their environments through general cytokine and chemokine surface receptors that facilitate gradient sensing of the local environment. This is the primary axis by which immune cells are able to home to sites of infection/wound healing and send pro-inflammatory or anti-inflammatory signals to neighboring cells. In addition, nearly all immune cells require some level of activation by means of receptor:ligand stimulation by non-self signatures. How specific these receptors are helps highlight the delineation between the two arms of the immune system: the innate and adaptive immune systems.

1.2.1 The Innate Immune System

The innate immune system is the body's first line of defense, and encompasses different types of cells (and proteins) that recognize broadly non-self signals known as pathogen-associated or damage-associated molecular patterns (PAMPs

or DAMPs). The effector cells of the innate immune system are decorated with ~20 different kinds of pattern recognition receptors (PRRs) that recognize broad signals (single stranded DNA, LPS, other signals of pathogen activity). Being the first responders, the innate immune system handles containment, engulfing the offending signals and otherwise blocking it off, and tries to destroy the threat. Lastly, these cells simultaneously sound the alarm, putting the local area into a heightened state of threat detection and infection prevention, such as spiking a fever and initiating swelling to draw in more immune cells.

1.2.2 The Adaptive Immune System

The adaptive immune system on the other hand, is slow to respond and is often brought in by innate cell activation. Instead of broad PRRs, it relies on tens of millions of somatically rearranged and stochastically generated highly specific adaptive immune receptors (AIRs) whose shape complementarity allows them to identify their complementary molecular patterns called epitopes. There are many more differences between the innate and adaptive immune systems, such as the latter's ability to develop a durable memory response, and those are introduced in greater detail here [7]. It is important to note however, that it is through the independent and asynchronous operation of both the innate and adaptive immune system that we see the dynamics [8] of threat detection, message passing, calling for reinforcements, homing of adaptive immune cells, and activation/expansion of these cells, resulting in complete clearance of the pathogen.Another key distinction between the innate and adaptive immune system is the formation of a robust memory response. Amazingly, evolution has steered this system to imprint past pathogen exposures so that upon re-exposure to the same signal, a small pool of memory cells is activated and antigen-specifc adaptive immune cells rapidly proliferatre and clear the antigen-source [9].

```
algorithmic
Adaptive Immune Algorithm

1. Nascent progenitor cells undergoes somatic recombination to stochastically generate an
Adaptive Immune Receptor (AIR).
2. Self-selecting methods of ensuring self-tolerance remove cells that react too strongly to
the self before being released into the blood.
3. In periphery, naive (antigen-inexperienced) cells interact with antigens via AIRs in search
of their cognate epitope.
4. Upon recognition of a sufficiently strong epitope reaction, they divide rapidly and
overwhelm the offending antigen with sheer numbers and specialized effector function.
5. After the threat has cleared, this expanded population contracts as cells die without
activation signalling, and a lasting pool of memory cells remains.
6. Memory pool persists and is reactivated in an effector state upon antigen reintroduction.
```

1.2.3 Effector Cells of the Adaptive Immune System

There are two major types of effector cells in the adaptive immune response: T-cells and B-cells. Both the T-cell and B-cell populations in the body are contextualized through their adaptive immune receptors, T-cell Receptors (TCRs) and B-cell Receptors (BCRs), respectively. We can think of both populations as repertoires of receptors, conferring protection against the threats recognized by the repertoire. T-cells help maitnain cellular immunity by interrogating the intracellular component of our bodies' by using their TCRs to scan recycled protein fragments presented at the cell surface (Read more about T-cell mediated immunity here [10]). B-cells, on the other hand, use their BCRs to survey the extracellular compartment and are tasked with upkeeping humoral immunity: neutralizing threats floating around in the blood and plasma. As effector cells of the adaptive immune response, both T-cell and B-cells are similar in their development and their operation as a unit, though the specifics of per cell function are quite different.

Note: A helpful distinction between antigens and epitopes is that an antigen is something that broadly generates an immune response and can have multiple epitopes. Epitopes are specific molecular patterns that have a matching paratope (binding surface of an adaptive immune receptor).

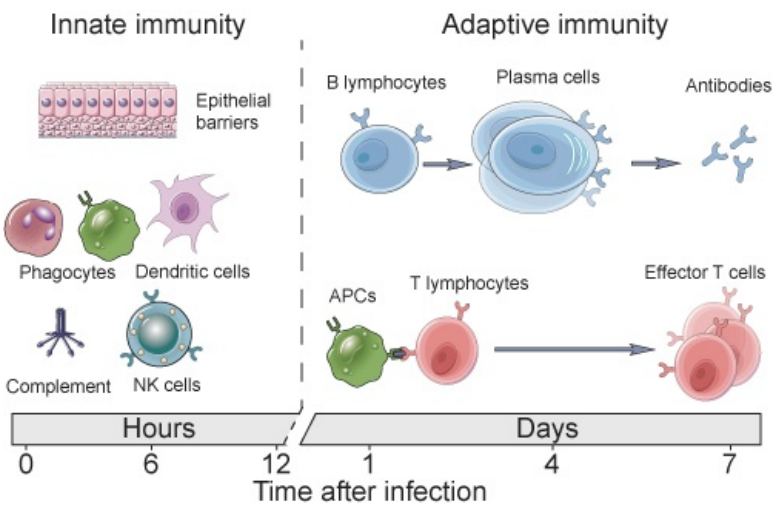


Image Source: Creative Diagnostics

1.3 B-Cells and Antibodies

B-cells, or B-lymphocytes, get their name not from their origin in the bone marrow, but from their discovery in a particular organ of the chicken[11]. These cells circulate in the bloodstream, equipped with unique B-cell receptors (BCRs) that allow them to recognize specific antigens, leading to their activation. This process often requires additional stimulation from helper T-cells which provide essential co-stimulatory signals, an additional layer of verification for non-selfness.

Over the course of the COVID pandemic, whether we wanted to or not were exposed to the concept of antibodies and learned of their association with some sort of protective capacity against SARS-COV-2. But what are they, and where do they originate from?

Antibodies (Abs) are typically represented as Y-shaped proteins that bind to their cognate epitope surfaces with high specificity and affinity, similar to how TCRs and BCRs bind to their epitopes. This is because antibodies are the soluble form of the B-cell receptor that is secreted into the blood upon B-cell activation in the presence of its cognate antigen. The secretion of large amounts of antibodies is the primary effector function of B-cells. Upon activation, a B-cell will divide, with the daughter cells inheriting the same BCR, and some of these cells will differentiate into plasma cells, which are the Ab factories capable of secreting thousands of Abs/min. This is especially useful upon antigen re-encounter where a large amount of antibodies are released by memory cells which neutralize the pathogen even before we develop the symptoms of infection (this is what most common vaccines are designed to do).

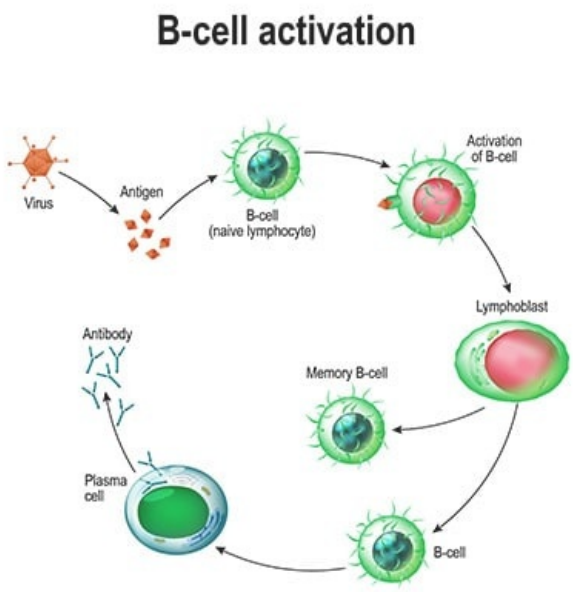


Image Source: Beckman

Neutralizing mechanisms of pathogenesis is only one way that antibody tagging is useful to immune defense. Antibody tagging plays a key role in a number of humoral immunity processes:

1. Neutralization: De-activation of pathogenic function by near complete coating of the functional component of pathogens or toxins by antibodies to inhibit interaction with host cells (i.e. and antibody that binds to the surface glycoproteins on SARS-COV2 now inhibit that virus particle's ability to enter cells expressing ACE2).
2. Opsonization: Partial coating of pathogens enhances rates phagocytosis and removal from the blood by cells of the innate immune system.
3. Agglutination/Precipitation): Since antibodies have 2 arms (each arm of the Y), they can cross-link and form anitbody-antigens chains which can precipitate out of the plasma and increase their chances of being recognized as aberrant and cleared by phagocytes.
4. Complement Activation: The complement system is a collection of inactive proteins and protein precursors are self-amplifying on activation and help with multiple aspects of humoral immunity. Yet another function of antibodies is their role in initiating the complement cascade that ends in the lysis or phagocytosis of pathogens.

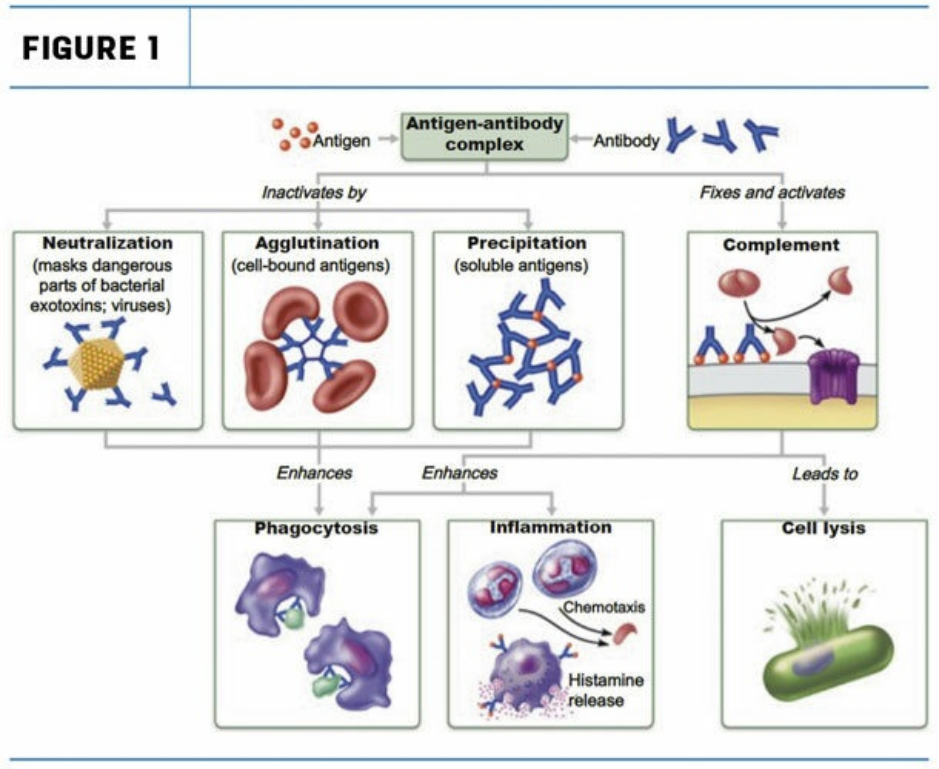


Image Source: The Immune System: Innate and Adaptive Body Defenses Figure 21.15 pulled from [Source]

Given the importance of B-cell mediated immunity, as operationalized by the body's antibody repertoire, it's clear that the diversity of BCR clones plays a critical role in our ability to mount an effective response against a pathogen. The maintenance of a robust BCR repertoire highlights not only the complexity of the immune response but also underscores the potential for leveraging the modularity of this mechanism to introduce new clones for their extraordinary precision in therapeutics such as vaccine development.

1.4 Antibody Sequence, Structure, and Function

The remarkable diversity of antibodies is achieved through somatic recombination, or gene rearrangement at the DNA level that occurs outside of meiosis. The AIR-specific somatic recombination is known as V(D)J recombination and generates both TCR and BCR diversity. During V(D)J recombination, a single gene per set is sampled from the set of variable (V), diversity (D), and joining (J) gene segments and randomly joined together with some baked in error (insertions) to create stable BCRs with unique antigen-binding sites. Additionally, B-cells have an additional process that further amplifies the diversity as well as the functional capacity of antigen-specific antibodies. This process is known as somatic hypermutation. When an activated B-cell divides, somatic hypermutation (SHM) introduces point mutations in the variable regions of BCR generating minor variants of each BCR. These daughter cells compete for survival signals mediated through antigen binding such that only the stronger binders survive.

Structurally, antibodies are composed of two identical light chains and two identical heavy chains, linked by disulfide bonds. Each chain contributes to the formation of the antigen-binding site, located in the variable regions. Within these regions, hypervariable loops known as complementarity determining regions (CDRs) dictate the specificity and affinity of the antibody-antigen interaction. This specificity is measured in terms of affinity using the dissociation constant (Kd), and the avidity (affinity over multi-valent binding sites, see IgM, IgA). The antibody molecule is divided into two main functional regions:

1. Fab Region (Fragment, antigen-binding): Contains the variable regions of the light and heavy chains, responsible for antigen recognition and binding.
2. Fc Region (Fragment, crystallizable): Composed of the constant regions of the heavy chains, mediates interactions with innate immune cells and the complement system.

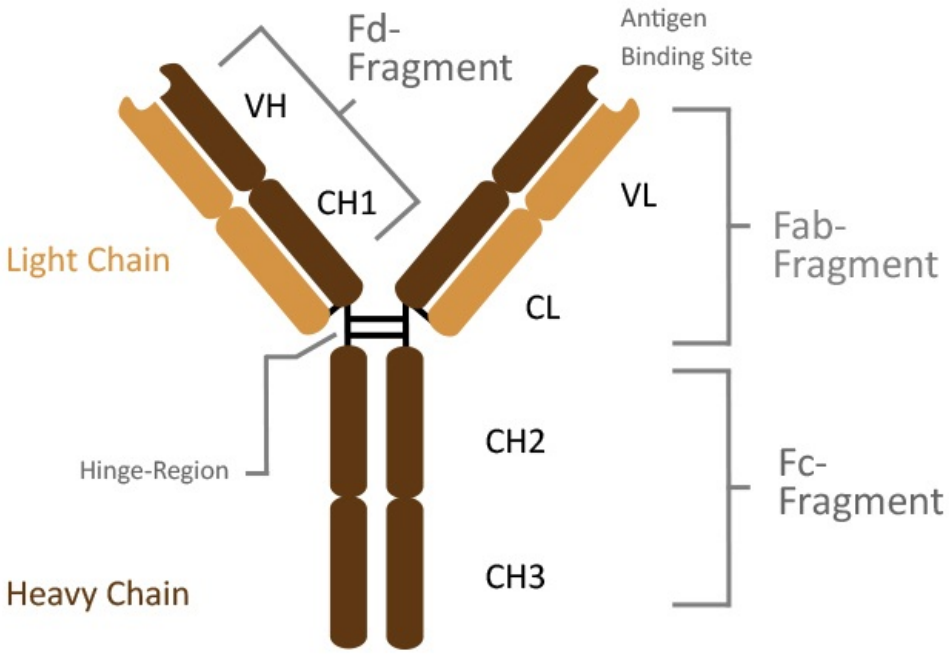


Image Source: Dianova: Antibody Structure \

By harnessing the selection of evolutionary pressures during somatic hypermutation, the B-cell compartment uses a powerful method of further tuning the antibody specificity to have some of the highest affinity interactions in the known protein universe [12]. Their high precision and binding affinities have caused their broad adoption in not only therapeutics but commercial and research applications as well as to tag proteins in solution in flow cytometry, CyTOF, immuno-precipitation, and other target identification assays.

1.5 Current Paradigms for Antibodies as a Therapeutic Modality

Given their unparalleled ability to precisely and durably bind arbitrary targets, there has been a significant interest in possessing antibodies for desired targets. There are a number of therapeutic use cases for these antibodies, for diseases ranging from transplant rejection, non-Hodgkin's lymphoma, immune checkpoint inhibitors for cancer immunotherapy, psoriasis, multiple sclerosis, Crohn's disease, and many more. While antibodies against common pathogens can be isolated from the serum of convalescent individuals and screened for specificity, the process of procuring novel antibodies is substantially more challenging and involves inoculating an animal with an antigen and isolating the antibodies after. For example, anti-venom is a solution of antibodies derived from animals (typically horses) against cytotoxic proteins found in venom. This procedure is both resource and labor intensive. This is because after inoculation and isolation of antibodies from the animal, there is an additional step of screening them for reactivity against the target using reaction chemistry methods such as surface plamon resonance or bio-layer interferometry. As such there has been a great deal of interest in methods of *in-silico* antibody design. A number of approaches have shown reasonable degrees of success in this task from guided evolution based approaches [1] to newer diffusion based [13] approaches. In this tutorial we will pay homage to the former.

2. Let's Code! Designing Antibodies via Directed Evolution

2.1 Overview

Now that we have the minimal backgrounded needed to understand the antibody design proble and the necessary language model background, we can jump right into antibody design via directed evolution, as shown in the figure below:

\

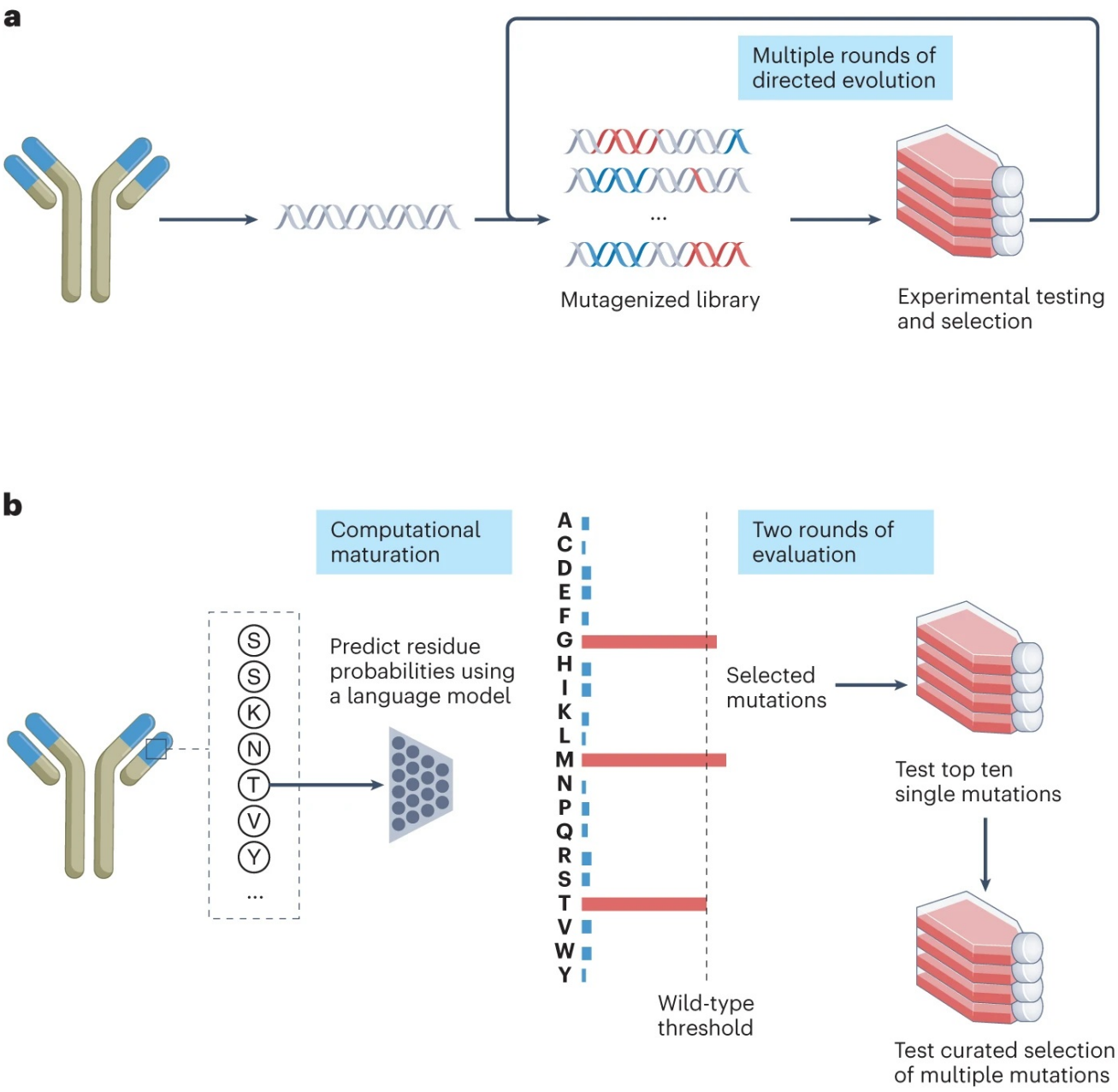


Image Source: Figure 1. Outeiral et. al

2.2 Setup/Methodology

In Hie et al. the authors decide to use a general protein language model instead of one trained specifically on antibody sequences. They use the ESM-1b and ESM-1v models which were trained on UniRef50 and UniRef90 [14], respectively. For their directed evolution studies they select seven therapeutic antibodies associated with viral infection spanning Influenza, Ebolavirus, and SARS-COv2. The authors use a straightforward and exhaustive mutation scheduler in mutating every residue in the antigen binding region to every other residue and computing the likelihood of the sequence. Sequences with likelihoods greater than or equal to the WT sequence were kept for experimental validation. For our purposes, we need not be as thorough and can use a slightly expedited method by taking the top-k mutations at a specific point.

Inspired by the work of Hie et al., we first define the pLM driven directed evolution task as simply passing in a masked antibody sequence to a pLM that was previously trained on the masked language modeling objective and examining the token probabilities for the masked amino acids. It really is that easy!

For reference we break the task down into the following steps:

```
# Antibody Design via pLM Directed Evolution
1. Select a pre-trained model language model (can be pre-trained on all domains or exclusively
antibodies)
2. Choose an antibody to mutate.
3. Determine the amino acid(s) to mask out*.
4. Pass the tokenized sequences into the pLM
5. Sample tokens according to a heuristic for increased fitness
```

*Modification of antibodies needs to focus only on the variable regions as the amino acids at the interface are the ones

responsible for driving affinity. Making edits to the constant region would actually be detrimental to antibodies' effector function in the complement system as well as potentially disrupt binding to innate immune receptors. \

2.2.1 Loading the Model + Tokenizer

For this exploration, we pay homage to an early Antibody Language Model AbLang [15]. AbLang is a masked language model based on the RoBERTa [16] model, and pre-trained on antibody sequences from the [observed antibody space (OAS) [17]. AbLang consists of two models, one trained on the heavy chain sequences and one trained on the light chain sequences and the authors demonstrate its usefulness over broader protein language models such as ESM-1b, contradicting the findings put forth in Hie et al. Both the heavy and light chain models are identical in architecture with a
d_{model}
of 768, max position embedding of 160, and 12 transformer block layers, totaling ~86M parameters.

In [ ]:
```
from transformers import AutoModel, AutoTokenizer, AutoModelForMaskedLM

# Get the tokenizer
tokenizer = AutoTokenizer.from_pretrained('qilowoq/AbLang_light')

# Get the light chain model
mlm_light_chain_model = AutoModelForMaskedLM.from_pretrained('qilowoq/AbLang_light')
# Get the heavy chain model
mlm_heavy_chain_model = AutoModelForMaskedLM.from_pretrained('qilowoq/AbLang_heavy')
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/
tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:   0%|          | 0.00/367 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/71.0 [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/3.02k [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/125 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/848 [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/343M [00:00<?, ?B/s]
config.json:   0%|          | 0.00/848 [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/343M [00:00<?, ?B/s]
```

In [ ]:
```
# Lets take a look at the model parameter count and architecture
n_params = sum(p.numel() for p in mlm_heavy_chain_model.parameters())
print(f'The Ablang model has {n_params} trainable parameters. \n')
mlm_heavy_chain_model
```

```
The Ablang model has 85809432 trainable parameters.
```

Out[ ]:
```
RobertaForMaskedLM(
  (roberta): RobertaModel(
    (embeddings): RobertaEmbeddings(
      (word_embeddings): Embedding(24, 768, padding_idx=21)
      (position_embeddings): Embedding(160, 768, padding_idx=21)
      (token_type_embeddings): Embedding(2, 768)
      (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.1, inplace=False)
    )
    (encoder): RobertaEncoder(
      (layer): ModuleList(
        (0-11): 12 x RobertaLayer(
          (attention): RobertaAttention(
            (self): RobertaSelfAttention(
              (query): Linear(in_features=768, out_features=768, bias=True)
              (key): Linear(in_features=768, out_features=768, bias=True)
              (value): Linear(in_features=768, out_features=768, bias=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
            (output): RobertaSelfOutput(
              (dense): Linear(in_features=768, out_features=768, bias=True)
              (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.1, inplace=False)
            )
          )
          (intermediate): RobertaIntermediate(
            (dense): Linear(in_features=768, out_features=3072, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): RobertaOutput(
            (dense): Linear(in_features=3072, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)

          )
        )
      )
    )
  )
  (lm_head): RobertaLMHead(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (layer_norm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (decoder): Linear(in_features=768, out_features=24, bias=True)
  )
)
```

2.2.2 Example Antibody Sequence

Next let's choose a heavy chain and light chain for designing. These were chosen from the Ablang examples page on HuggingFace,

In [ ]:
```
# Lets take the variable regions of the heavy and light chains

heavy_chain_example =  'EVQLQESGPGLVKPSETLSLTCTVSGGPINNAYWTWIRQPPGKGLEYLGYVYHTGVTNYNPSLKSRLTITIDTSRKQLSLSLKFVTA
light_chain_example = 'GSELTQDPAVSVALGQTVRITCQGDSLRNYYASWYQQKPRQAPVLVFYGKNNRPSGIPDRFSGSSSGNTASLTISGAQAEDEADYYCN
```

2.2.3 Masking the Sequence

One of the crucial parameters with this approach is in the determination of which residues to mask and re-design. Let's start off by first setting up some reproducible code so that we can apply the masking procedure to any number of sequences at arbitrary points.

In [ ]:
```
# Sequnece masking convenience function
def mask_seq_pos(sequence: str,
                      idx: int,
                      mask='[MASK]'):
    '''Given an arbitrary antibody sequence with and a seqeunce index,
    convert the residue at that index into the mask token.

    '''
    cleaned_sequence = sequence.replace(' ', '')  # Get ride of extraneous spaces if any
    assert abs(idx) < len(sequence), "Zero-indexed value needs to be less than sequence length minus one."
    cleaned_sequence = list(cleaned_sequence)  # Turn the sequence into a list
    cleaned_sequence[idx] = '*'                 # Mask the sequence at idx
    masked_sequence = ' '.join(cleaned_sequence) # Convert list -> seq
    masked_sequence = masked_sequence.replace('*', mask)
    return masked_sequence

# Test
assert mask_seq_pos('CAT', 1)=='C [MASK] T'
```

```
#TODO: Add unit tests with pytest where you can check that the assert has been hit
```

2.2.3 Model Inference

In [ ]:
```
### Step 1. Mask the light_chain sequence
mask_idx = 9
masked_light_chain = mask_seq_pos(light_chain_example, idx=mask_idx)
### Step 2. Tokenize
tokenized_input = tokenizer(masked_light_chain, return_tensors='pt')
### Step 3. Light Chain Model
mlm_output = mlm_light_chain_model(**tokenized_input)
### Step 4. Decode the outputs to see what the model has placed
decoded_outs = tokenizer.decode(mlm_output.logits.squeeze().argmax(dim=1), skip_special_tokens=True)
print(f'Model predicted: {decoded_outs.replace(" ", "")[9]} at index {mask_idx}')
print(f'Predicted Sequence: {decoded_outs.replace(" ", "")}')
print(f'Starting  Sequence: {light_chain_example}')
```

```
Model predicted: S at index 9
Predicted Sequence: SADSSSCGVSSTVAHGQTLKINSQGQRHSLYYVRWYQQKPGLAPLLLIYGKNSRPSGIPDRFSGSKSGTTASLTITGLQAEDEADYYCQQSG
GSGGHLTVGGGALLATLTQ
Starting  Sequence: GSELTQDPAVSVALGQTVRITCQGDSLRNYYASWYQQKPRQAPVLVFYGKNNRPSGIPDRFSGSSSGNTASLTISGAQAEDEADYYCNSRDS
SSNHLVFGGGTKLTVLSQ
```

2.2.4 HuggingFace Pipeline Object

Hold on, given the tokenized input with only one masked token, we would expect to see only one change to the the sequence. However, what we get back is something a lot more different that what we put in. Luckily, there's something in the HuggingFace software suite that we can use to address this: Pipelines

HuggingFace Pipelines:

1. Pipeline object is a wrapper for inference and can be treated like an object for API calls
2. There is a fill-mask pipeline that we can use which accepts a single mask token in out input and outputs a dictionary of the score of that sequence, the imputed token, and the reconstructed full sequence.

Lets see it in action:

In [ ]:
```
from transformers import pipeline
filler = pipeline(task='fill-mask', model=mlm_light_chain_model, tokenizer=tokenizer)
filler(masked_light_chain) # fill in the mask
```

Out[ ]:
```
[{'score': 0.13761496543884277,
  'token': 7,
  'token_str': 'S',
  'sequence': 'G S E L T Q D P A S S V A L G Q T V R I T C Q G D S L R N Y Y A S W Y Q Q K P R Q A P V L V F Y
G K N N R P S G I P D R F S G S S S G N T A S L T I S G A Q A E D E A D Y Y C N S R D S S S N H L V F G G G T K
L T V L S Q'},
 {'score': 0.1152879148721695,
  'token': 6,
  'token_str': 'E',
  'sequence': 'G S E L T Q D P A E S V A L G Q T V R I T C Q G D S L R N Y Y A S W Y Q Q K P R Q A P V L V F Y
G K N N R P S G I P D R F S G S S S G N T A S L T I S G A Q A E D E A D Y Y C N S R D S S S N H L V F G G G T K
L T V L S Q'},
 {'score': 0.0989701896905899,
  'token': 9,
  'token_str': 'N',
  'sequence': 'G S E L T Q D P A N S V A L G Q T V R I T C Q G D S L R N Y Y A S W Y Q Q K P R Q A P V L V F Y
G K N N R P S G I P D R F S G S S S G N T A S L T I S G A Q A E D E A D Y Y C N S R D S S S N H L V F G G G T K
L T V L S Q'},
 {'score': 0.08586061000823975,
  'token': 14,
  'token_str': 'A',
  'sequence': 'G S E L T Q D P A A S V A L G Q T V R I T C Q G D S L R N Y Y A S W Y Q Q K P R Q A P V L V F Y
G K N N R P S G I P D R F S G S S S G N T A S L T I S G A Q A E D E A D Y Y C N S R D S S S N H L V F G G G T K
L T V L S Q'},
 {'score': 0.07652082294225693,
  'token': 8,
  'token_str': 'T',
  'sequence': 'G S E L T Q D P A T S V A L G Q T V R I T C Q G D S L R N Y Y A S W Y Q Q K P R Q A P V L V F Y
G K N N R P S G I P D R F S G S S S G N T A S L T I S G A Q A E D E A D Y Y C N S R D S S S N H L V F G G G T K
L T V L S Q'}]
```

Congratulations you have now designed 5 new antibodies!

Disclaimer: For a more thorough antibody (re)design, we will typically want to follow an approach like what was done in Hie et al. where every point along the sequence will be mutated and the total number of sequences will be collated and scored with the top-100 or so antibodies being expressed for validation. If you would like to explore this feel free to try it out yourself as a challenge!

You can also refer to the real data in Hie et al. to see if any of the predicted ones were found to work well and increase fitness.

2.3 Limitations

While promising, this approach is obviously not without its shortcomings. Key limitations include:

- Fixed length antibody design since masked tokens are applied in a 1:1 fashion.
- Lack of target information included during conditional sampling step which can influence choice of amino acid given the sequence context.
- Approach is sensitive to choice of protein language model

This letter [18] provides a great synopsis of Hie et al.'s work, which by extension apply to the methods presented in this tutorial as well.

Citing this Tutorial

If you found this tutorial useful, please consider citing it as:

```
@manual{Bioinformatics,
 title={An Introduction to Antibody Design Using Protein Language Models},
 organization={DeepChem},
 author={Karthikeyan, Dhuvarakesh and Menezes, Aaron},
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/DeepChem_AntibodyTutorial_

 year={2024},
}
```

Works Cited

[1] Hie, B.L., Shanker, V.R., Xu, D. et al. Efficient evolution of human antibodies from general protein language models. Nat Biotechnol 42, 275–283 (2024). https://doi.org/10.1038/s41587-023-01763-2

[2] Bretscher, P., & Cohn, M. (1970). A Theory of Self-Nonself Discrimination. Science, 169(3950), 1042–1049. doi:10.1126/science.169.3950.1042

[3] Cohn, M. The common sense of the self-nonself discrimination. Springer Semin Immun 27, 3–17 (2005). https://doi.org/10.1007/s00281-005-0199-1

[4] Gonzalez S, González-Rodríguez AP, Suárez-Álvarez B, López-Soto A, Huergo-Zapico L, Lopez-Larrea C. Conceptual aspects of self and nonself discrimination. Self Nonself. 2011 Jan;2(1):19-25. doi: 10.4161/self.2.1.15094. Epub 2011 Jan 1. PMID: 21776331; PMCID: PMC3136900.

[5] ROB J. DE BOER, PAULINE HOGEWEG, Self-Nonself Discrimination due to Immunological Nonlinearities: the Analysis of a Series of Models by Numerical Methods, Mathematical Medicine and Biology: A Journal of the IMA, Volume 4, Issue 1, 1987, Pages 1–32, https://doi.org/10.1093/imammb/4.1.1

[6] Cohn, M. A biological context for the self-nonself discrimination and the regulation of effector class by the immune system. Immunol Res 31, 133–150 (2005). https://doi.org/10.1385/IR:31:2:133

[7] Janeway CA Jr, Travers P, Walport M, et al. Immunobiology: The Immune System in Health and Disease. 5th edition. New York: Garland Science; 2001. Principles of innate and adaptive immunity. Available from: https://www.ncbi.nlm.nih.gov/books/NBK27090/

[8] Perelson, A. Modelling viral and immune system dynamics. Nat Rev Immunol. 2. , 28–36 (2002). https://doi.org/10.1038/nri700

[9] Shittu, A. (n.d.). Understanding immunological memory. ASM.org. https://asm.org/articles/2023/may/understanding-immunological-memory

[10] Janeway CA Jr, Travers P, Walport M, et al. Immunobiology: The Immune System in Health and Disease. 5th edition. New York: Garland Science; 2001. Chapter 8, T Cell-Mediated Immunity. Available from: https://www.ncbi.nlm.nih.gov/books/NBK10762/

[11] Glick, B., Chang, T. S., & Jaap, R. G. (1956). The Bursa of Fabricius and Antibody Production. Poultry Science, 35(1), 224–225. doi:10.3382/ps.0350224

[12] Nooren, I. M. (2003). NEW EMBO MEMBER'S REVIEW: Diversity of protein-protein interactions. EMBO Journal, 22(14),

3486–3492. https://doi.org/10.1093/emboj/cdg359

[13] Karolis Martinkus, Jan Ludwiczak, Kyunghyun Cho, Wei-Ching Liang, Julien Lafrance-Vanasse, Isidro Hotzel, Arvind Rajpal, Yan Wu, Richard Bonneau, Vladimir Gligorijevic, & Andreas Loukas. (2024). AbDiffuser: Full-Atom Generation of in vitro Functioning Antibodies.

[14] Baris E. Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, Cathy H. Wu, UniRef: comprehensive and non-redundant UniProt reference clusters, Bioinformatics, Volume 23, Issue 10, May 2007, Pages 1282–1288, https://doi.org/10.1093/bioinformatics/btm098

[15] Tobias H Olsen, Iain H Moal, Charlotte M Deane, AbLang: an antibody language model for completing antibody sequences, Bioinformatics Advances, Volume 2, Issue 1, 2022, vbac046, https://doi.org/10.1093/bioadv/vbac046

[16] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, & Veselin Stoyanov. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[17] Olsen TH, Boyles F, Deane CM. Observed Antibody Space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences. Protein Sci. 2022 Jan;31(1):141-146. doi: 10.1002/pro.4205. Epub 2021 Oct 29. PMID: 34655133; PMCID: PMC8740823.

[18] Outeiral, C., Deane, C.M. Perfecting antibodies with language models. Nat Biotechnol 42, 185–186 (2024). https://doi.org/10.1038/s41587-023-01991-6

Protein Language Models (Intuition): A First Look at Modeling Syntax and Semantics of the Known Protein Universe

By Dhuvi Karthikeyan, Aaron Menezes, Elisa Gómez de Lope, and Rakshit Singh

Inspired by the success of Large Language Models (LLMs) such as BERT, T5, and GPT, that have demonstrated state of the art performance on sentiment analysis, summarization, question answering, and classification tasks, protein language models (pLMs) have demonstrated similar sweeping success across a broad array of protein specific tasks. These tasks include contact prediction, mutational landscape fitness prediction, binding site prediction, property prediction, and much more! In this tutorial, we explore the fundamental intuition driving the success of protein language models, by developing a strong intuition of what is actually happening under the hood and the resulting pros and cons of their usage. If you'd like to learn more about language models in various other domains, check out DeepChem's very own ChemBERTa, a large language model trained on the chemical domain here.



Table of Contents:

1. Introduction
2. What is a language model?
3. Methods for learning language
4. How do Protein Language Models (pLMs) work?
5. MSA-aware vs non-MSA-aware protein language models
6. Evolutionary statistics of Hemoglobin β and its ProtBERT learned representation
7. Concluding thoughts

1. Introduction

This DeepChem tutorial is designed to serve as an introductory primer on protein language models, a powerful and versatile method of processing protein sequence information inspired by methods from the natural language space. Over the past decade, natural language processing has shown the strength of using learned representations to encapsulate the semantic meaning of text data. Notable models like word2vec [1] and GloVe [2] proved that self-supervised pre-training on large, unlabeled corpora effectively creates robust feature embeddings that maintain similarity and analogy in language. However, these models were limited in utility by their context-free embeddings. The advent of context-aware models, starting with BERT [3], led to numerous sequence models applicable beyond language domains. In biology, self-supervised pre-training on protein language models has achieved state-of-the-art performance in various tasks by deriving context-aware amino acid embeddings that can be finteuned to capture information on structure [4] and function [5] of proteins.

This tutorial aims to provide an overview of the concepts and intuition of protein language models that are needed to work with them and understand their input/outputs, strengths, failure modes. We skip over the detailed breakdown of their architecture, but invite the community to add content as they see fit in the form of a pull request to build upon this.

Disclaimer: For brevity sake, we make some assumptions with familiarity to the multi-layered perceptron, neural networks, and learning by gradient descent. Additionally we assume some fluency with probability theory on matters such as discrete vs. continuous distributions, likelihood, and conditional distributions. We provide links on non-obvious topics and concepts to external sources wherever necessary to bring the audience a vetted and beginner friendly source to start learning on the more complicated topics. Follow along for a high-level overview into the reason that protein language models have been so successful across a broad range of tasks.

2. What is a language model?

Under the hood, all language models are nothing more than probability distributions over tokens, or discrete sub-sequences. In natural language, a very intuitive set of tokens are the words of a language, or perhaps even the characters. Both have their own pros and cons. For simplicity, let's work with words as tokens here, though this changes for proteins. Since the learned distribution is over discrete units (words), this distribution is a categorical distribution, not a continuous one. To make this more concrete, take for example a common language model that you have likely interacted with many times in your life: text auto-complete. Text auto-complete is a conditional language model that takes the previous words you have written and then computes the conditional probability over all the words in its vocabulary and returns the highest probability words based on the context. If you'd like a very intuitive and fine-grained

explanation for both the form and function of language models, the 3Blue1Brown walk-through is a great resource that breaks down the basics of the architecture, the flow of information, and the process of training a specific LLM (GPT). In this section we skip over the architecture of the language models and instead leave them as a black-box, focusing more on the how language models learn from sequences to better motivate their use in the protein domain.

A simple way to visualize what a language model is doing in the background is to think of the language model as updating and indexing a huge square matrix of transition probabilities of size
DxD
, where
D
is the vocabulary size of the model. Here vocabulary size refers to the number of unique words or sub-words that make up the state space of the categorical distribution. So a model that only knows the words ['a', 'boy', 'cute', 'is', 'student', 'the' 'walking'] has a vocabulary size of 7. If we start off with an untrained model that is randomly initialized, we can use a uniform initialization we would get a transition matrix that looks something like this where we introduce a special word to designate the end of sequence (EOS):

	a	boy	cute	is	student	the	walking	EOS
a	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
boy	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
cute	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
is	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
student	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
the	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125
walking	0.125	0.125	0.125	0.125	0.125	0.125	0.125	0.125

However, if we look at some of the transition probabilities, we can immediately see that the model is not very good. For example, the probability of the word 'a' coming after 'a' should be close to 0. Same goes for the word 'the' coming after 'a'. It's pretty clear that we need some way of training this model so that we can get some realistic transition probabilities.

3. Methods for learning language

3.1 Causal Modeling

The first language models were trained on the principle of causal language modeling, where the model is tasked with next word prediction during each training step.

$$\text{The quick brown fox jumped over the lazy -----.}$$

$P(x_t|$

After enough rounds of this training protocol the model learns a much more plausible distribution over the words - something that looks like the following:

	a	boy	cute	is	student	the	walking	EOS
a	0.0	0.5	0.1	0.05	0.25	0.05	0.05	0.0
boy	0.15	0.0	0.1	0.4	0.05	0.15	.05	0.1
cute	0.05	0.2	0.0	0.1	0.25	0.1	0.0	0.3
is	0.2	0.0	0.3	0.0	0.0	0.2	0.3	0.0
student	0.15	0.05	0.1	0.5	0.0	0.05	.05	0.1
the	0.0	0.5	.2	0.0	0.25	0.0	0.05	0.0
walking	0.1	.0	.05	.2	.05	0.35	0.0	0.3
EOS	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Here we can see that the model has learned that the words above are not typically repeated twice in a row. It assigns subject words ['boy', 'student'] after the word 'the' with higher probability than the verbs ['is', 'walking']. If we start at 'the' and sample the most likely words at each transition we can generate the following sentence as a path through the model: 'the' -> 'boy' -> 'is' -> 'walking' -> 'EOS'. This mode of sampling a word at every time step and then conditioning on the previously sampled words is known as auto-regressive generation.

3.2 The Power of Neural Networks

A key point here that motivates the use of increasingly complicated neural networks for language modeling tasks is that with our illustrative example, we have the transition probability matrix and each step we sample from it like a markov chain. However, there are longer range dependencies in language that are not captured simply by conditioning on the previous word. So why not construct matrices that map transition probabilities between pairs of words or triples, or even more? Beyond issues of computational feasibility, this model would require that all possible n-grams would have been seen at least once during training, which greatly limits the models learning and usability. Neural networks have emerged as a great way of using large contexts and generating a neural representation of the sequence before indexing a loose transition matrix that maps between the neural representation and all the words in the vocabulary. For this reason, we often see in the language model space the distribution over the vocabulary from the last layer of a neural network as a one-dimensional probability distribution rather than a probability matrix. Keeping this in mind, we can see how these language models accommodate another method of learning language that draws from the context before **AND** after a word.

3.3 Masked Language Modeling

Causal language modeling has a key drawback in that sometimes the necessary context to make sense of a word in a sentence comes after the word and not before. Masked language modeling is like causal modeling, but makes use of the fact that context may come before and after the word of interest.

$$\text{The quick brown [MASK] jumped over the lazy dog.}$$

$P(x_t|$

This approach is what underlies the powerful BERT [3] language model, where they used a masking rate of about 15% of the words. Amazingly, this approach has been tried on sequences other than language and has been shown to be a robust model for learning the syntax and semantics of sequential data of various modalities including time series data, videos, and yes even proteins!

4. How do Protein Language Models (pLMs) work?

Inspired by the success of large language models (LLMs) in a broad variety of natural language tasks, protein language models represent a powerful new approach in understanding the syntax and semantics of protein sequences [6]. These models are trained on using the masked language modeling objective to mask out portions of the sequence and infer what amino acids belong across billions of protein sequences, learning to identify patterns and relationships within the sequences that are crucial for their structure and function. This step is called pre-training and it imbues the language model with a general understanding of the structural dependencies within the language, in this case proteins.

An optional second training step known as fine-tuning can be applied on a pre-trained protein language model, to further train it on a specific task with protein sequence examples annotated with labels. In practice, starting from the pretrained weights has shown to have better performance than starting from randomly initialized weights as the model simply learns how to use strong representations of the inputs (learned during pretraining) instead of jointly learning the representation AND how to use it. PLMs finetuned on the mappings between specific protein families or functional classes can significantly enhance predictive power compared to non-pretrained models, and can be applied in a number of different use cases, such as predicting binding sites or the effects of mutations.

One of the most compelling benefits of PLMs is their ability to capture coevolutionary relationships within and across protein sequences [7]. In the same way that words in a sentence co-occur to convey coherent meaning, amino acid residues in a protein sequence co-evolve to maintain the protein's structural integrity and functionality. PLMs capture these coevolutionary patterns, allowing for the prediction of how changes in one part of a protein may affect other parts. Thus, from a design perspective, the directed evolution task is an area where PLMs offer substantial advantages. In a directed evolution experiment, a naturally occurring protein can be mutated according to any arbitrary heuristic and is then checked if a desired function has improved. Since PLMs capture intra-sequence conditional distributions, this process can be vastly streamlined by masking portions of the protein we wish to 'mutate' and sampling from the distribution of what amino acids are strong candidates to occur given the rest of the sequence. PLMs thus have the potential to significantly reduce experimental burden by identify promising candidates a higher hit rate.

4.1 Reconciling Sequence and Structure

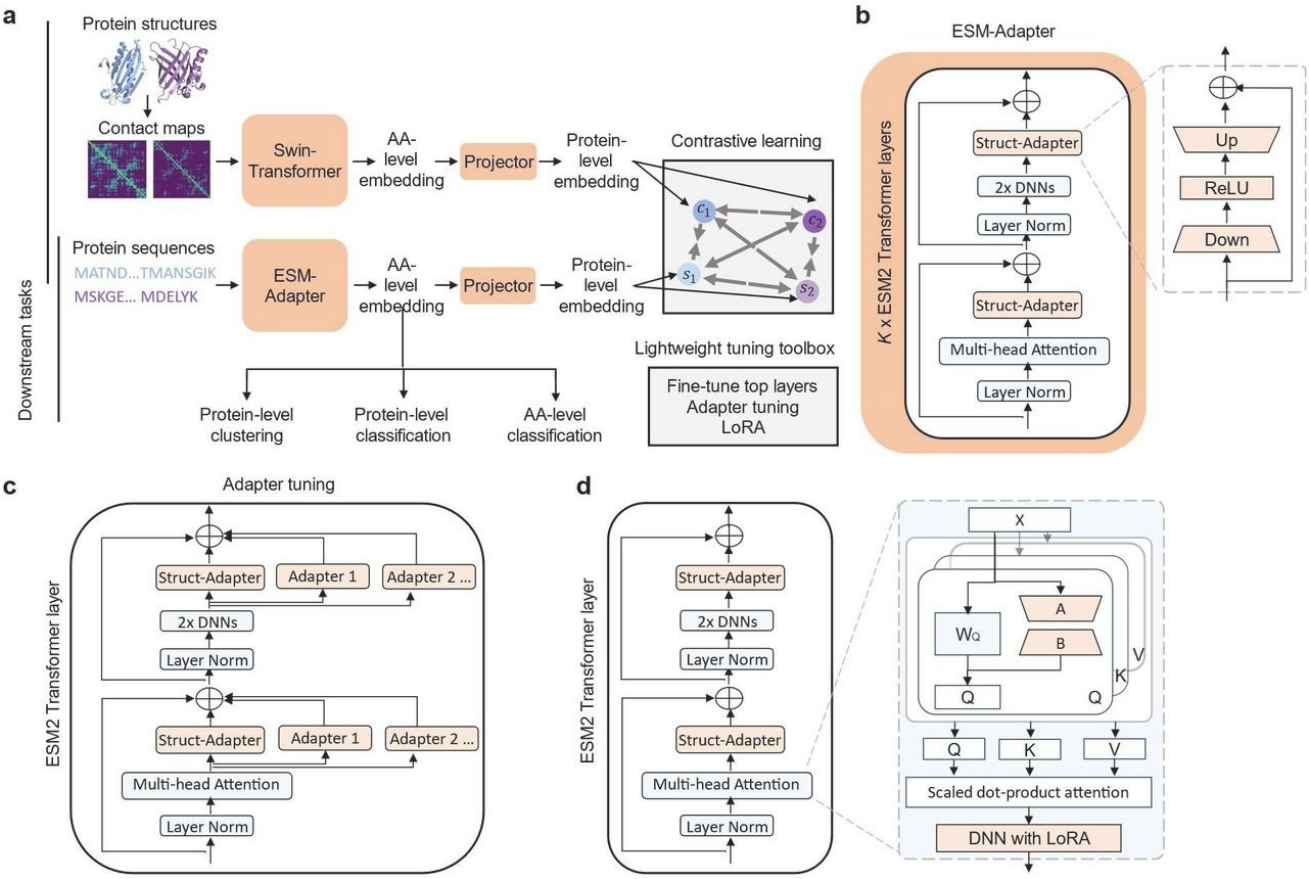
Some protein language models combine in their training input amino acid sequence data with structural data, such as 3D coordinates of atoms in the protein. The goal is to explicitly incorporate structural information information, aiming to enhance the representation and ultimately prediction of unseen protein structure and functions. This is in contrast to sequence-only models that implicitly model structure which is more closely conserved across proteins via homology.

Models like ESM-1b [4] and ESM-2 [8] are examples of sequence-only pLMs that do not explicitly incorporate 3D structural information. These sequence-based pLMs have demonstrated impressive performance on a variety of protein function prediction tasks by learning patterns from large protein sequence datasets. However, the lack of structural

information can limit the generalization capabilities of sequence-only PLMs. This is true especially for applications heavily dependent on protein structure such as contact prediction. Moreoever, the inclusion of structural information helps overcome the distributional biases that exist in the training datasets of sequences.

Structure-aware pLMs like S-PLM[9] and ESM-Fold [8] are trained on both sequence and structural information, and in turn generate protein representations that encode both sequence and structural information. These models use various methods such as multi-view contrastive learning to align the sequence and structure representations in a shared latent space (S-PLM). The structural awareness enables them to achieve comparable or superior performance to specialized structure-based methods or sequence-based pLMs, particularly for applications that heavily rely on protein structure.

Interestingly, the recently released ESM-3 [10] pLM reasons over sequence, structure, and function, meaning that for each protein, its sequence, structure, and function are extracted, tokenized, and partially masked during pre-training.



The framework of S-PLM and lightweight tuning strategies for downstream supervised learning. a, The framework of S-PLM: During pretraining, the model inputs both the amino acid sequences and contact maps derived from protein structures simultaneously. After pretraining, the ESM-Adapter that generates the AA-level embeddings before the projector layer is used for downstream tasks. The entire ESM-Adapter model can be fully frozen or learnable through lightweight tuning. b, Architecture of the ESM-Adapter. c, Adapter tunning for supervised downstream tasks. d, LoRA tuning for supervised downstream tasks is implemented. Adapted from [9].

5. MSA-aware vs non-MSA-aware protein language models

Multiple Sequence Alignment (MSA) is a method used to align three or more biological sequences (protein or nucleic acid) to identify regions of similarity that may indicate functional, structural, or evolutionary relationships. MSAs are a cornerstone in bioinformatics for tasks such as phylogenetic analysis, structure prediction, and function annotation.

In the context of pLMs, MSA provides evolutionary context to the representations of protein sequences. PLMs can be MSA-aware and non-MSA-aware:

MSA-aware models:

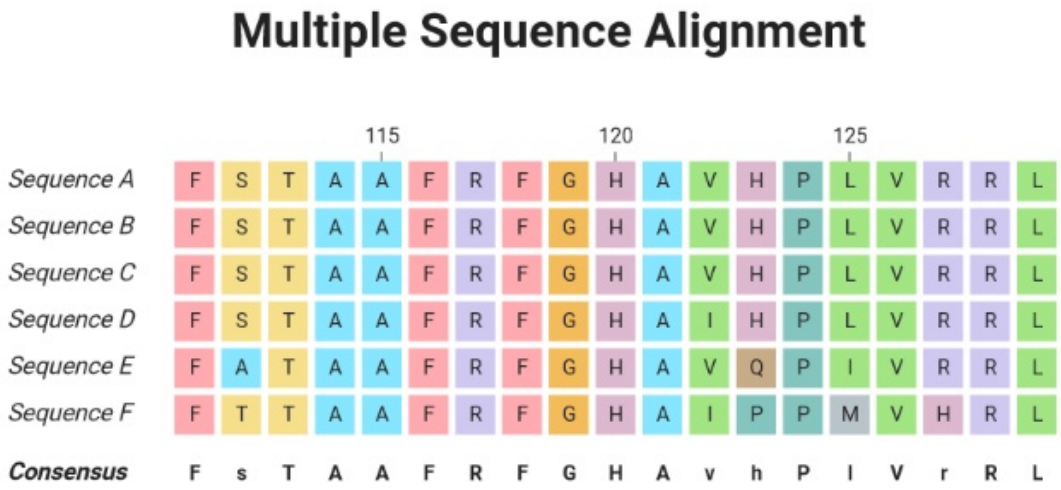
MSA-aware models, such as the MSA Transformer [11], Evoformer (used in AlphaFold) [12] and ESM-MSA [11], are trained on datasets that include MSAs as input to incorporate evolutionary information and relationships between sequences to learn richer representations. They align multiple homologous sequences to capture conserved and variable regions. The rationale is that conserved regions often indicate functionally or structurally important parts of the protein, while variable regions can provide insights into evolutionary divergence and adaptation.

MSA-aware models can provide deeper insights into protein function and structure due to the evolutionary context. However, they are computationally intensive and require high-quality MSAs, which may not be available for all protein families.

Non-MSA-aware models:

Non-MSA-aware models, such as ESMFold (ESM-2)[8], ProtBERT [6] and TAPE, treat each protein sequence independently and do not explicitly incorporate evolutionary information from MSAs. They are trained on large datasets of individual protein sequences, learning patterns and representations directly from the sequence data.

While they can generalize well to diverse sequences and are computationally efficient, they may miss out on the evolutionary context that can be crucial for certain tasks.



"*Multiple Sequence Alignment*". *BioRender.*

Benefits and challenges of MSA-aware models:

- Evolutionary insight: MSAs provide evolutionary information, highlighting conserved residues that are often critical for protein function and structure.
- Improved predictions: By incorporating evolutionary context, MSA-aware models can improve performance on tasks such as secondary structure prediction, contact prediction, and function annotation.
- Functional and structural understanding: MSAs help in identifying functionally important regions and understanding the structural constraints of proteins.
- Computational complexity: Generating and processing MSAs is computationally expensive and time-consuming.
- Data availability: High-quality MSAs are not available for all protein families, especially those with few known homologs.
- Model complexity: MSA-aware models are more complex and require sophisticated architectures to effectively utilize the evolutionary information.

Other considerations:

- The performance benchmark of both MSA-aware and not MSA-aware for predicting the 3D structure of proteins, as well as their function and other properties is currently an active topic of research.
- Interestingly, MSA-free models have reported ability to efficiently generate sufficiently accurate MSAs that can be used as input for the MSA-aware models.

Without further ado, lets explore some of the properties of protein language models in the wild!

6. Evolutionary statistics of Hemoglobin β and its ProtBERT learned representation



Image Source: *Adapted from "Représentation simplifiée de l'hémoglobine et de l'hème". Wikimedia Commons.*

Hemoglobin is the protein responsible for transporting oxygen from the lungs to all the cells of our body via red blood cells. Hemoglobin is a great protein to interrogate the behaviors of protein language models as it is highly conserved in certain regions across species, and also slightly variable in other places. What would we expect the distribution over amino acids to look like if we mask out a highly conserved region? What about a highly diverse region? Let's find out.

Hemoglobin Sequence Homology across closely related mammals (from [13]):

In [ ]:
```
hemoglobin_beta = {
'human':
"MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVL
'chimpanzee':
"MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTORFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVL
'camel':
"MVHLSGDEKNAVHGLWSKVKVDEVGGEALGRLLVVYPWTRRFFESFGDLSTADAVMNNPKVKAHGSKVLNSFGDGLNHLDNLKGTYAKLSELHCDKLHVDPENFRLLGNVL
'rabbit':
"MVHLSSEEKSAVTALWGKVNVEEVGGEALGRLLVVYPWTQRFFESFGDLSSANAVMNNPKVKAHGKKVLAAFSEGLSHLDNLKGTFAKLSELHCDKLHVDPENFRLLGNVL
'pig':
"MVHLSAEEKEAVLGLWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSNADAVMGNPKVKAHGKKVLQSFSDGLKHLDNLKGTFAKLSELHCDQLHVDPENFRLLGNVI
'horse':
"*VQLSGEEKAAVLALWDKVNEEEVGGEALGRLLVVYPWTQRFFDSFGDLSNPGAVMGNPKVKAHGKKVLHSFGEGVHHLDNLKGTFAALSELHCDKLHVDPENFRLLGNVL
'bovine':
"M**LTAEEKAAVTAFWGKVKVDEVGGEALGRLLVVYPWTQRFFESFGDLSTADAVMNNPKVKAHGKKVLDSFSNGMKHLDDLKGTFAALSELHCDKLHVDPENFKLLGNVL
'sheep':
"M**LTAEEKAAVTGFWGKVKVDEVGAEALGRLLVVYPWTQRFFEHFGDLSNADAVMNNPKVKAHGKKVLDSFSNGMKHLDDLKGTFAQLSELHCDKLHVDPENFRLLGNVL
}
```

As we can see there is a great degree of overlap between the hemoglobin β subunits across the animal kingdom. The part of the hemoglobin sequence that is essential to the function of carrying oxygen is the part that binds to the heme group. This is handled by a single amino acid, namely the Histidine (H) near position 92 on the beta chain, in the middle of the underlined subsequences above. Unsurprsingly, given its functional importance, the amino acid (H) at position is unchanged across all species. Can a language model recapitulate this?

ProtBERT learned representation of Hemoglobin β

ProtBERT [6] is a BERT style protein language model that was trained via masked amino acid modeling on Uniref100 [14], a dataset consisting of 217 million protein sequences, and 88B amino acids. The Uniref database contains deduplicated protein sequences from UniProt where they are clustered together, and thus deduplicated, given the threhold of sequence identity between species. Uniref100 takes 100% sequence identity, while Uniref90 does 90% and Uniref50% has a cutoff of 50%. As such, ProtBERT was trained on the largest of these databases. Lets load up ProtBERT and see what it looks like.

In [ ]:
```
### Step 1. Download ProtBERT Weights from the HF repo
from transformers import BertForMaskedLM, BertTokenizer, pipeline
tokenizer = BertTokenizer.from_pretrained("Rostlab/prot_bert", do_lower_case=False )
model = BertForMaskedLM.from_pretrained("Rostlab/prot_bert")
model
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/
tokens), set it as secret in your Google Colab and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json:   0%|          | 0.00/86.0 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/81.0 [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download`
is deprecated and will be removed in version 1.0.0. Downloads always resume when possible. If you want to force
a new download, use `force_download=True`.
  warnings.warn(
config.json:   0%|          | 0.00/361 [00:00<?, ?B/s]
pytorch_model.bin:   0%|          | 0.00/1.68G [00:00<?, ?B/s]
Some weights of the model checkpoint at Rostlab/prot_bert were not used when initializing BertForMaskedLM: ['ber
t.pooler.dense.bias', 'bert.pooler.dense.weight', 'cls.seq_relationship.bias', 'cls.seq_relationship.weight']
- This IS expected if you are initializing BertForMaskedLM from the checkpoint of a model trained on another tas
k or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTrainin
g model).
- This IS NOT expected if you are initializing BertForMaskedLM from the checkpoint of a model that you expect to
be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification mo
del).
```

Out[ ]:
```
BertForMaskedLM(
  (bert): BertModel(
    (embeddings): BertEmbeddings(
      (word_embeddings): Embedding(30, 1024, padding_idx=0)
      (position_embeddings): Embedding(40000, 1024)
      (token_type_embeddings): Embedding(2, 1024)
      (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
      (dropout): Dropout(p=0.0, inplace=False)
    )
    (encoder): BertEncoder(

      (layer): ModuleList(
        (0-29): 30 x BertLayer(
          (attention): BertAttention(
            (self): BertSdpaSelfAttention(
              (query): Linear(in_features=1024, out_features=1024, bias=True)
              (key): Linear(in_features=1024, out_features=1024, bias=True)
              (value): Linear(in_features=1024, out_features=1024, bias=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
            (output): BertSelfOutput(
              (dense): Linear(in_features=1024, out_features=1024, bias=True)
              (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
              (dropout): Dropout(p=0.0, inplace=False)
            )
          )
          (intermediate): BertIntermediate(
            (dense): Linear(in_features=1024, out_features=4096, bias=True)
            (intermediate_act_fn): GELUActivation()
          )
          (output): BertOutput(
            (dense): Linear(in_features=4096, out_features=1024, bias=True)
            (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.0, inplace=False)
          )
        )
      )
    )
    (cls): BertOnlyMLMHead(
      (predictions): BertLMPredictionHead(
        (transform): BertPredictionHeadTransform(
          (dense): Linear(in_features=1024, out_features=1024, bias=True)
          (transform_act_fn): GELUActivation()
          (LayerNorm): LayerNorm((1024,), eps=1e-12, elementwise_affine=True)
        )
        (decoder): Linear(in_features=1024, out_features=30, bias=True)
      )
    )
  )
```

In [ ]:
```
### Step 2. Mask the F8 Histidine of Hemoglobin B Subunit
human_heme = list(hemoglobin_beta['human'])
human_heme[92] = "[MASK]"
masked_heme = ' '.join(human_heme)
print(masked_heme)
```

```
M V H L T P E E K S A V T A L W G K V N V D E V G G E A L G R L L V V Y P W T Q R F F E S F G D L S T P D A V M
G N P K V K A H G K K V L G A F S D G L A H L D N L K G T F A T L S E L [MASK] C D K L H V D P E N F R L L G N V
L V C V L A H H F G K E F T P P V Q A A Y Q K V V A G V A N A L A H K Y H
```

In [ ]:
```
### Step 3. Tokenize the Sequence
tokenized_sequence = tokenizer(masked_heme, return_tensors='pt')
tokenized_sequence
```

Out[ ]:
```
{'input_ids': tensor([[ 2, 21,  8, 22,  5, 15, 16,  9,  9, 12, 10,  6,  8, 15,  6,  5, 24,  7,
         12,  8, 17,  8, 14,  9,  8,  7,  7,  9,  6,  5,  7, 13,  5,  5,  8,  8,
         20, 16, 24, 15, 18, 13, 19, 19,  9, 10, 19,  7, 14,  5, 10, 15, 16, 14,
          6,  8, 21,  7, 17, 16, 12,  8, 12,  6, 22,  7, 12, 12,  8,  5,  7,  6,
         19, 10, 14,  7,  5,  6, 22,  5, 14, 17,  5, 12,  7, 15, 19,  6, 15,  5,
         10,  9,  5,  4, 23, 14, 12,  5, 22,  8, 14, 16,  9, 17, 19, 13,  5,  5,
          7, 17,  8,  5,  8, 23,  8,  5,  6, 22, 22, 19,  7, 12,  9, 19, 15, 16,
         16,  8, 18,  6,  6, 20, 18, 12,  8,  8,  6,  7,  8,  6, 17,  6,  5,  6,
         22, 12, 20, 22,  3]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
         1, 1, 1, 1, 1]])}
```

In [ ]:
```
### Step 4. Pass the tokenized sequence through ProtBERT
model_outs = model(**tokenized_sequence)
model_outs
```

Out[ ]:
```
MaskedLMOutput(loss=None, logits=tensor([[[-18.5725, -18.4833, -18.4627,  ..., -18.6075, -18.6886, -18.5834],
         [-19.9242, -21.0844, -20.1253,  ..., -18.1157, -19.4804, -19.9197],
         [-17.1929, -18.2174, -17.2419,  ..., -18.0640, -17.8393, -17.1908],
         ...,
         [-19.3707, -19.7466, -19.2460,  ..., -21.4760, -20.1202, -22.1149],
         [-19.6403, -20.1926, -18.5551,  ..., -17.4572, -18.9988, -21.0471],
         [-18.8277, -19.8305, -18.5255,  ..., -16.8518, -18.2768, -18.0429]]],
       grad_fn=<ViewBackward0>), hidden_states=None, attentions=None)
```

In [ ]:
```
### Step 5. Transform the Logits
import torch.nn.functional as F
logits = model_outs.logits.squeeze()[1:-1] # Ignore SOS and EOS special tokens
print(logits.shape)
softmaxed = F.softmax(logits, dim=1).detach().numpy() # Softmax to normalize the logits to sum to 1
```

```
torch.Size([147, 30])
```

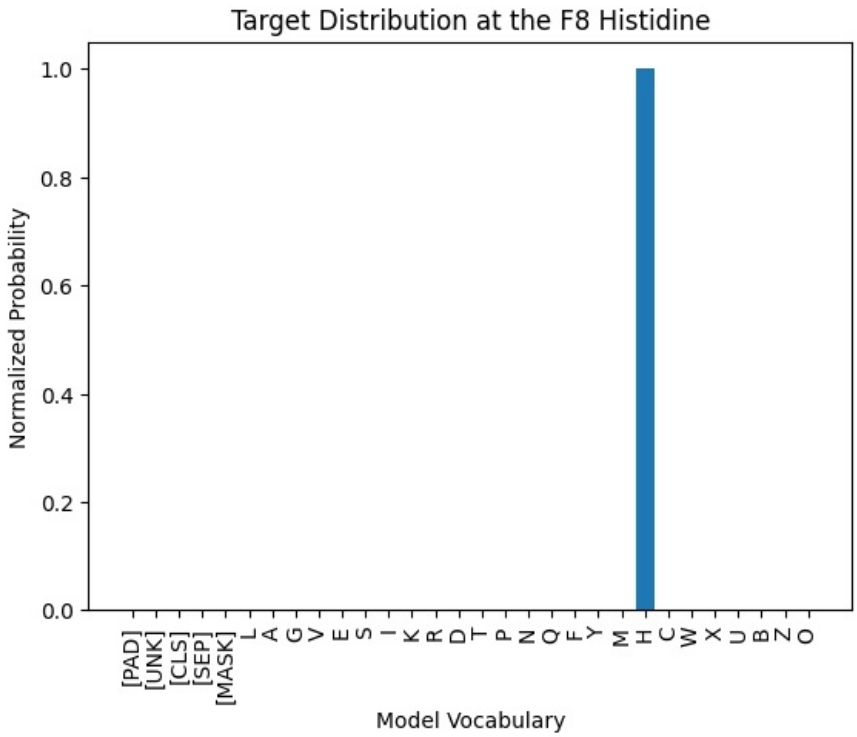
In [ ]:
```
### Step 6. Decode the Logits Using Greedy Decoding (Max Probability at Each Timestep)
decoded_outputs = tokenizer.batch_decode(softmaxed.argmax(axis=1))
decoded_sequence = ''.join(decoded_outputs)
print(decoded_sequence)
print(f'The filled-in masked sequence is: {decoded_sequence[92]}')
```

```
MVHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLKHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLV
CVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH
The filled-in masked sequence is: H
```

Sanity Check: Whew, looks like the pLM ProtBERT was able to recapitulate the correct amino acid at that position. But how confident was the model? Let's visualize the distribution at that position and see what other amino acids the model was choosing between.

In [ ]:
```
### Step 7. Visualize the Token Distribution at the F8 Histidine
import matplotlib.pyplot as plt

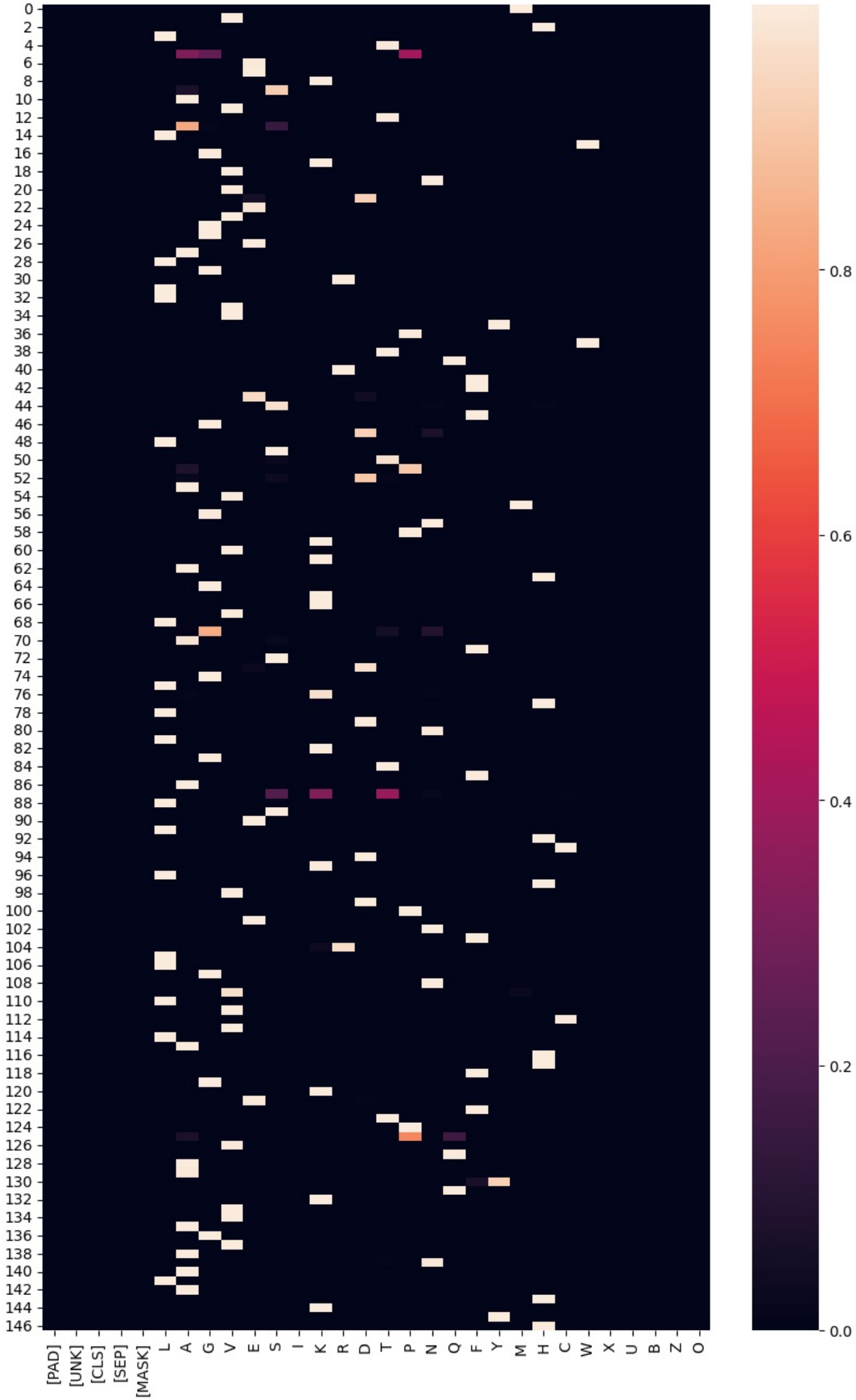
plt.bar(tokenizer.get_vocab().keys(), softmaxed[92])
plt.ylabel('Normalized Probability')
plt.xlabel('Model Vocabulary')
plt.title('Target Distribution at the F8 Histidine')
plt.xticks(rotation='vertical')
plt.show()
```



In [ ]:

```
### [EXTRA] Step 8. Visualize the Logits Map Across All Positions
import seaborn as sns

plt.figure(figsize=(10,16))
sns.heatmap(softmaxed, xticklabels=tokenizer.get_vocab())
plt.show()
```

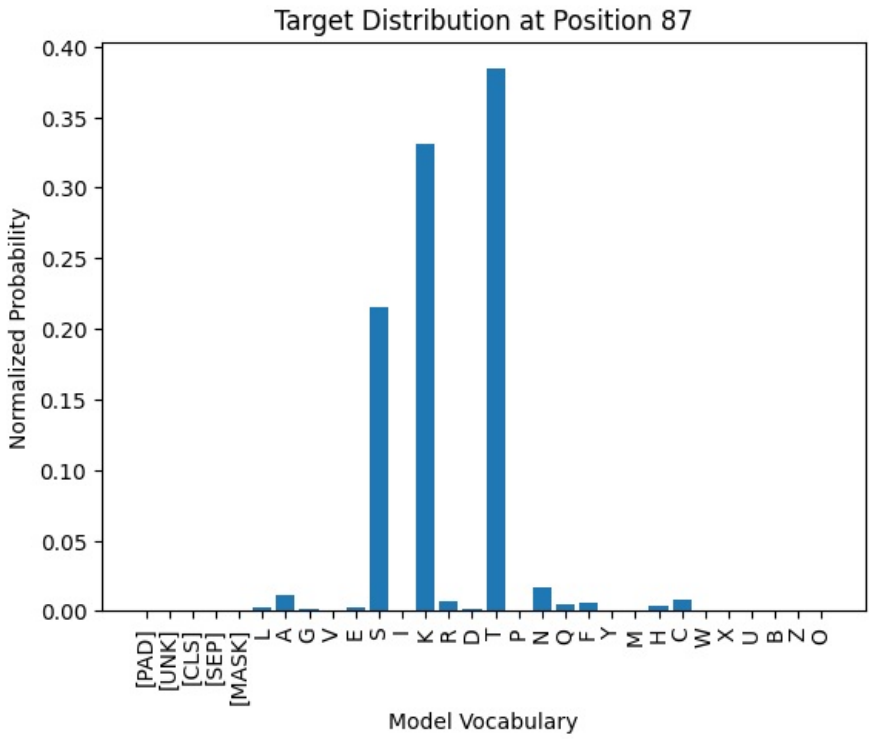


In [ ]:

```
### [EXTRA] Step 9. Look at a Low Confidence Region

plt.bar(tokenizer.get_vocab().keys(), softmaxed[87])
plt.ylabel('Normalized Probability')
plt.xlabel('Model Vocabulary')
```

```
plt.title('Target Distribution at Position 87')
plt.xticks(rotation='vertical')
plt.show()
```



In [ ]:
```
for animal in hemoglobin_beta:
    print(f'{animal} has residue {hemoglobin_beta[animal][87]} at position 87')
```

```
human has residue T at position 87
chimpanzee has residue T at position 87
camel has residue K at position 87
rabbit has residue K at position 87
pig has residue K at position 87
horse has residue A at position 87
bovine has residue A at position 87
sheep has residue Q at position 87
```

As we can see from the above, at the positions where is lower confidence, there tends to be an increase in diversity among the different species. This aligns well with out understanding of what the categorical distribution would look like if we took calculated the probabilities of each of the amino acids using all the homologous proteins in the protein universe.

7. Concluding Thougts

We hope you liked this Tutorial 0 on protein language models. While subsequent tutorials will cover more of the architecture of the protein language models, the learned representations, and the applications of this remarkable class of methods, we hope that this work helps ground you in when going through all the details. Analyzing the input/outputs of pLMs using this lens has helped me understand why performance disparities for certain examples and understand the failure modes that these models can encounter. For a quick reference, some of their strengths and limitations as they fall within the scope of the tutorial are summarized below:

7.1 Strengths

- pLMs learn co-evolutionary statistics of residues across diverse protein families [7].
- They capture information on structure and function from protein sequence alone (most available and accurate modality by far).

7.2 Limitations

- pLMs demonstrate poorer performance on learning the sequence distributions of highly mutated/variable protein sequences. They bias towards germline sequences [15]
- Current pLMs are biased towards the sequences derived from canonically studied model organisms [16]

Citing this Tutorial

If you found this tutorial useful, please consider citing it as:

```
@manual{Bioinformatics,
title={Protein Language Models (Intuition): A First Look at Modeling Syntax and Semantics of
the Known  Protein Universe},
organization={DeepChem},
author={Karthikeyan, Dhuvarakesh and Menezes, Aaron and de Lope, Elisa Gomez},
howpublished =
{\\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/ProteinLM_Tutorial0.ipynb

year={2024},
},
```

References

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv [Cs.CL]. Retrieved from http://arxiv.org/abs/1301.3781

[2] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[4] Rao, R., Meier, J., Sercu, T., Ovchinnikov, S., & Rives, A. (2020). Transformer protein language models are unsupervised structure learners. bioRxiv. doi:10.1101/2020.12.15.422761

[5] Ibtehaz, N., Kagaya, Y., & Kihara, D. (2023). Domain-PFP: Protein Function Prediction Using Function-Aware Domain Embedding Representations. bioRxiv. doi:10.1101/2023.08.23.554486

[6] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, & Burkhard Rost. (2021). ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing.

[7] Zhidian Zhang, Hannah K. Wayment-Steele, Garyk Brixi, Haobo Wang, Matteo Dal Peraro, Dorothee Kern, Sergey Ovchinnikov bioRxiv 2024.01.30.577970; doi: https://doi.org/10.1101/2024.01.30.577970

[8] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, Alexander Rives bioRxiv 2022.07.20.500902; doi: https://doi.org/10.1101/2022.07.20.500902

[9] Wang D, Pourmirzaei M, Abbas UL, Zeng S, Manshour N, Esmaili F, Poudel B, Jiang Y, Shao Q, Chen J, Xu D. S-PLM: Structure-aware Protein Language Model via Contrastive Learning between Sequence and Structure. bioRxiv [Preprint]. 2024 May 13:2023.08.06.552203. doi: 10.1101/2023.08.06.552203. PMID: 37609352; PMCID: PMC10441326.

[10] Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, Alexander Rives bioRxiv 2024.07.01.600583; doi: https://doi.org/10.1101/2024.07.01.600583

[11] Roshan M Rao, Jason Liu, Robert Verkuil, Joshua Meier, John Canny, Pieter Abbeel, Tom Sercu, Alexander Rives Proceedings of the 38th International Conference on Machine Learning, PMLR 139:8844-8856, 2021.

[12] Jumper, J., Evans, R., Pritzel, A. et al. Highly accurate protein structure prediction with AlphaFold. Nature 596, 583–589 (2021). https://doi.org/10.1038/s41586-021-03819-2

[13] Ali, A., Baby, B., Soman, S.S. et al. Molecular insights into the interaction of hemorphin and its targets. Sci Rep 9, 14747 (2019). https://doi.org/10.1038/s41598-019-50619-w

[14] Baris E. Suzek, Hongzhan Huang, Peter McGarvey, Raja Mazumder, Cathy H. Wu, UniRef: comprehensive and non-redundant UniProt reference clusters, Bioinformatics, Volume 23, Issue 10, May 2007, Pages 1282–1288, https://doi.org/10.1093/bioinformatics/btm098

[15] Shaw, A., Spinner, H., Shin, J., Gurev, S., Rollins, N., & Marks, D. (2023). Removing bias in sequence models of protein fitness. bioRxiv. doi:10.1101/2023.09.28.560044

[16] Ding, F., & Steinhardt, J. (2024). Protein language models are biased by unequal sequence sampling across the tree of life. doi:10.1101/2024.03.07.584001

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

How to contribute to this/other tutorials

If you have any feedback on the given tutorial, please feel free to reach out by opening a github issue on this tutorial. If you'd like to build off this tutorial series, there are a couple of places that would be natural and impactful extensions to this tutorial:

1. Attention Mechanisms and their Significance to Protein Language Models
2. Zhang et al's Categorical Jacobian analysis of pLMs

Feel free to contribute your own ideas on the matter as well.

Introduction to binding sites

By Elisa Gómez de Lope | Twitter

In this tutorial, you will explore the fundamental concepts and computational methods for studying binding sites to enhance your understanding and analysis of molecular interactions.

Table of Contents:

- Introduction
- Basic concepts
- Types of binding sites
- Computational methods to study binding sites
- DeepChem tools
- How does a binding pocket look like?
- Further Reading

This tutorial is made to run without any GPU support, and can be used in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Introduction

Binding sites are specific locations on a molecule where ligands, such as substrates, inhibitors, or other molecules, can attach through various types of molecular interactions.

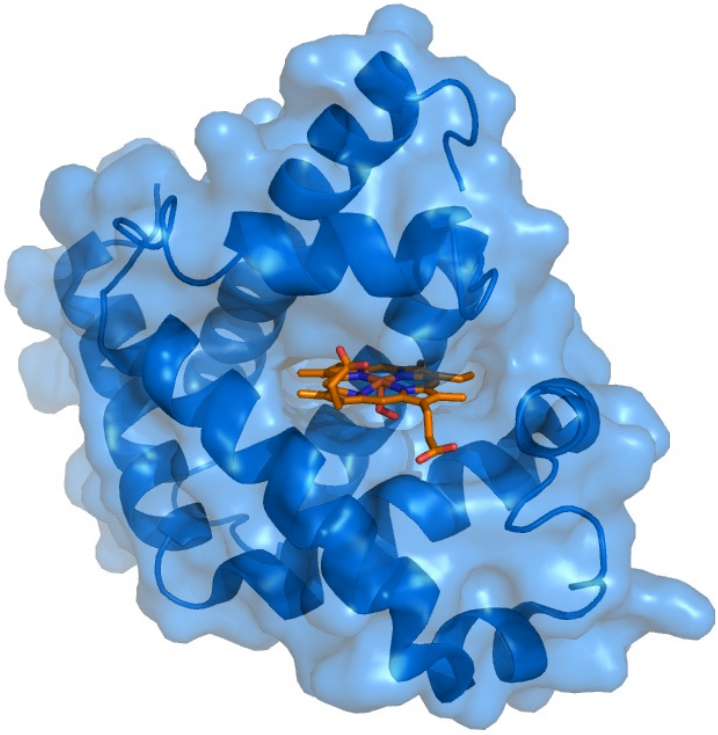
Binding sites are crucial for the function of many biological molecules. They are typically located on the surface of proteins or within their three-dimensional structure. When a ligand binds to a binding site, it can induce a conformational change in the protein, which can either activate or inhibit the protein's function. This binding process is essential for numerous biological processes, including enzyme catalysis, signal transduction, and molecular recognition.

For example, in enzymes, the binding site where the substrate binds is often referred to as the active site. In receptors, the binding site for signaling molecules (such as hormones or neurotransmitters) is critical for transmitting signals inside the cell.

Understanding binding sites is particularly relevant for the development of new drugs, as it can lead to the development of more effective and selective drugs from multiple angles:

- Target identification: Identifying binding sites on target proteins allows researchers to design molecules that can specifically interact with these sites, leading to the development of drugs that can modulate the protein's activity.
- Drug design: Knowledge of the structure and properties of binding sites enables the design of drugs with high specificity and affinity, reducing off-target effects and increasing efficacy.
- Optimization: Detailed understanding of binding interactions helps improving the binding characteristics of drug candidates, such as increasing binding affinity and selectivity.

Additionally, knowledge of the structure and properties of binding sites enables the design of drugs with high specificity and affinity, reducing off-target effects and increasing efficacy.



Myoglobin (blue) with its ligand heme (orange) bound. Based on PDB: 1MBO.

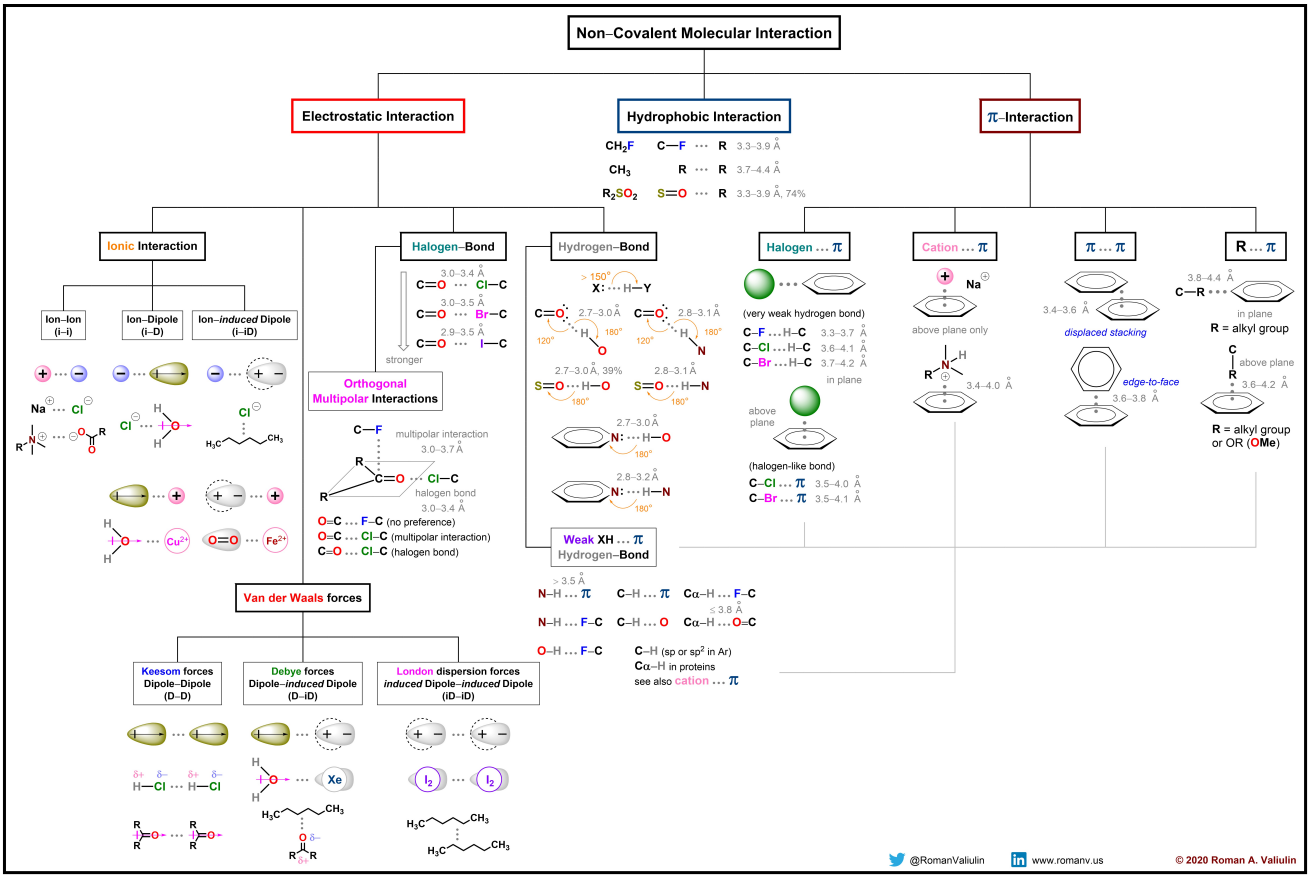
Basic concepts

Here we cover some basic notions to understand the science of binding site identification.

Molecular Interactions

The specific interactions that occur at the binding site can be of various types, including (non-exhaustive list):

- Hydrogen Bonding: Weak electrostatic interactions between hydrogen atoms bonded to highly electronegative atoms (like oxygen, nitrogen, or fluorine) and other electronegative atoms or functional groups. Hydrogen bonding is important for stabilizing protein-ligand complexes and can be enhanced by halogen bonding.
- Halogen Bonding: A type of intermolecular interaction where a halogen atom (like iodine or fluorine) acts as an acceptor, forming a bond with a hydrogen atom or a multiple bond. Halogen bonding can significantly enhance the affinity of ligands for binding sites.
- Orthogonal Multipolar Interactions: Interactions between backbone carbonyls, amide-containing side chains, guanidinium groups, and sulphur atoms, which can also enhance binding affinity.
- Van der Waals Forces: Weak, non-specific interactions arising from induced electrical interactions between closely approaching atoms or molecules. They usually provide additional stabilization and contribute to the overall binding affinity, especially in close-contact regions.
- Metal coordination: Interactions between metal ions (e.g., zinc, magnesium) and ligands that have lone pairs of electrons (e.g., histidine, cysteine, water). These interactions are typically coordinate covalent bonds, where both electrons in the bond come from the ligand, and are crucial in metalloenzymes and metalloproteins, where metal ions often play a key role in catalytic activity and structural stability.
- Polar Interactions: Interactions between polar functional groups, such as hydrogen bond donors (e.g., backbone NH, polarized Cα–H, polar side chains, and protein-bound water) and acceptors (e.g., backbone carbonyls, amide-containing side chains, and guanidinium groups).
- Hydrophobic Interactions: Non-polar interactions between lipophilic side chains, which can contribute to the binding affinity of ligands.
- Pi Interactions: Interactions between aromatic rings (Pi-Pi), and aromatic rings with other types of molecules (Halogen-Pi, Cation-Pi,...). They occur in binding sites with aromatic residues such as phenylalanine, tyrosine, and tryptophan, and stabilize the binding complex through stacking interactions.



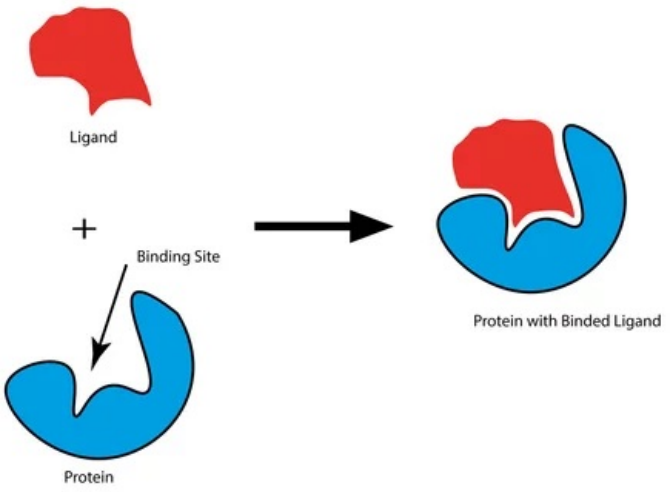
Valiulin, Roman A. "Non-Covalent Molecular Interactions". Cheminfographic, 13 Apr. 2020, https://cheminfographic.wordpress.com/2020/04/13/non-covalent-molecular-interactions

Ligands

Ligands are molecules that bind to specific sites on proteins or other molecules, facilitating various biological processes.

Ligands can be classified into different types based on their function:

- Substrates: Molecules that bind to an enzyme's active site and undergo a chemical reaction.
- Inhibitors: Molecules that bind to an enzyme or receptor and block its activity.
- Activators: Molecules that bind to an enzyme or receptor and increase its activity.
- Cofactors: Non-protein molecules (metal ions, vitamins, or other small molecules) that bind to enzymes to modify their activity. They can act as activators or inhibitors depending on the specific enzyme and the binding site.
- Signaling Lipids: Lipid-based molecules that act as signaling molecules, such as steroid hormones.
- Neurotransmitters: Chemical messengers that transmit signals between neurons and their target cells.



Protein Ligand Docking. Adobe Stock #216974924.

The binding of ligands to their target sites is influenced by various physicochemical properties:

- Size and Shape: Ligands must be the appropriate size and shape to fit into the binding site.
- Charge: Electrostatic interactions, such as ionic bonds, can contribute to ligand binding.

- Hydrophobicity: Hydrophobic interactions between non-polar regions of the ligand and the binding site can stabilize the complex.
- Hydrogen Bonding: Hydrogen bonds between the ligand and the binding site can also play a crucial role in binding affinity.

The specific interactions between a ligand and its binding site, as well as the physicochemical properties of the ligand, are essential for understanding and predicting ligand-receptor binding events

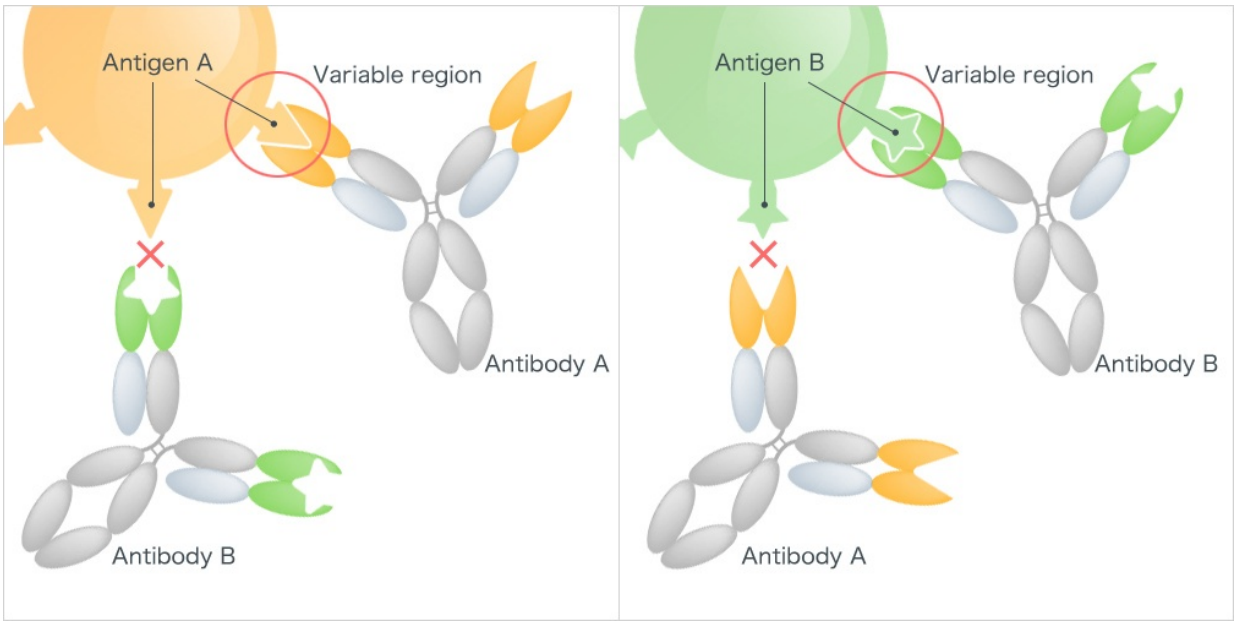
Binding Affinity and Specificity

Binding affinity is the strength of the binding interaction between a biomolecule (e.g., a protein or DNA) and its ligand or binding partner (e.g., a drug or inhibitor). It is typically measured and reported by the equilibrium dissociation constant (K_d), which is used to evaluate and rank the strengths of bimolecular interactions. The smaller the K_d value, the greater the binding affinity of the ligand for its target. Conversely, the larger the K_d value, the more weakly the target molecule and ligand are attracted to and bind to one another.

Binding affinity is influenced by non-covalent intermolecular interactions, such as hydrogen bonding, electrostatic interactions, hydrophobic interactions, and van der Waals forces between the two molecules. The presence of other molecules can also affect the binding affinity between a ligand and its target.

Binding specificity refers to the selectivity of a ligand for binding to a particular site or target. Highly specific ligands will bind tightly and selectively to their intended target, while less specific ligands may bind to multiple targets with varying affinities. It is determined by the complementarity between the ligand and the binding site, including factors such as size, shape, charge, and hydrophobicity (see section above on ligands). Specific interactions, like hydrogen bonding and ionic interactions, contribute to the selectivity of the binding.

Binding specificity is crucial in various biological processes, such as enzymatic reactions or drug-target interactions, as it allows for specific and regulated interactions, which is essential for the proper functioning of biological systems.



Antigen-antibody interactions are particularly highly specific binding sites (often also characterized by high affinity). The specificity of these interactions is fundamental to ensure precise immune recognition and response. Kyowa Kirin. "Specificity of Antibodies".

In summary, binding affinity measures the strength of the interaction between a ligand and its target, while binding specificity determines the selectivity of the ligand for a particular binding site or target.

Thermodynamics of Binding

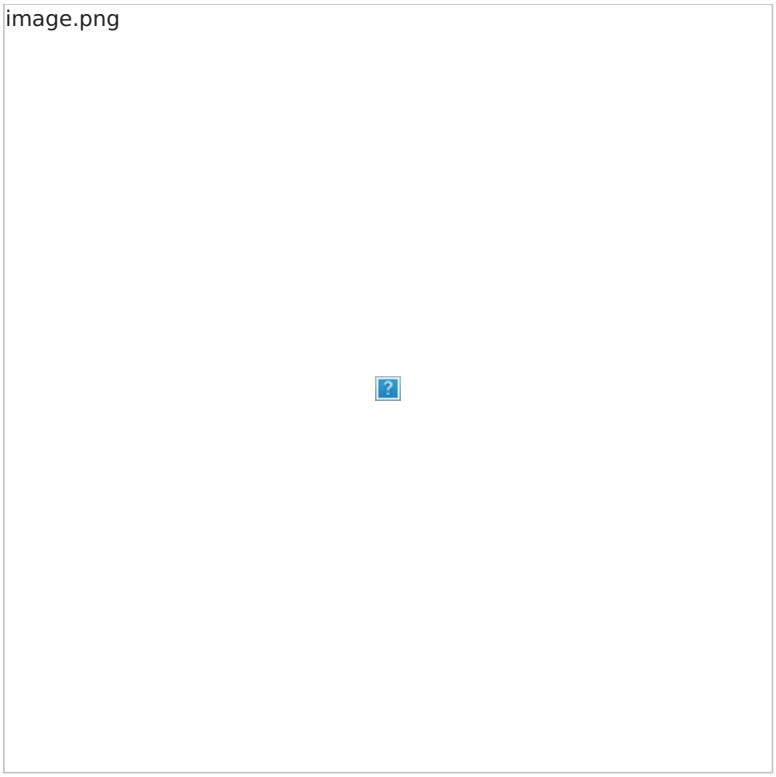
The thermodynamics of binding involves the interplay of enthalpy (H), entropy (S), and Gibbs free energy (G) to describe the binding of ligands to binding sites. These thermodynamic parameters are crucial in understanding the binding process and the forces involved.

- **Enthalpy (**
  H
) is a measure of the total energy change during a process. In the context of binding, enthalpy represents the energy change associated with the formation of the ligand-binding site complex. A negative enthalpy change indicates that the binding process is exothermic, meaning that heat is released during binding. Conversely, a positive enthalpy change indicates an endothermic process, where heat is absorbed during binding.

- **Entropy (**
  S
) measures the disorder or randomness of a system. In binding, entropy represents the change in disorder associated with the formation of the ligand-binding site complex. A negative entropy change indicates a decrease in disorder, which is often associated with the formation of a more ordered complex. Conversely, a positive entropy change indicates an increase in disorder, which can be seen in the disruption of the binding site or the ligand.

- **Gibbs free energy (**
  G
) is a measure of the energy change during a process that takes into account both enthalpy and entropy. It is defined as
  $G = H - TS$
  , where
  T
  is the temperature in Kelvin. Gibbs free energy represents the energy available for work during a process. In binding, a negative Gibbs free energy change indicates that the binding process is spontaneous and favorable, while a positive Gibbs free energy change indicates that the process is non-spontaneous and less favorable.

image.png



Calculated variation of the Gibbs free energy (G), entropy (S), enthalpy (H), and heat capacity (Cp) of a given reaction plotted as a function of temperature (T). The solid curves correspond to a pressure of 1 bar; The dashed curve shows variation in ΔG at 8 GPa. Ghiorso, Mark S., Yang, Hexiong and Hazen, Robert M. "Thermodynamics of cation ordering in karrooite (MgTi2O5)" American Mineralogist, vol. 84, no. 9, 1999, pp. 1370-1374. https://doi.org/10.2138/am-1999-0914

- **Binding isotherms** are models that describe the relationship between the concentration of ligand and the occupancy of binding sites. These isotherms are crucial in understanding the binding process and the forces involved. One example is the Langmuir isotherm, which assumes that the binding site is homogeneous and the ligand binds to the site with a single binding constant.

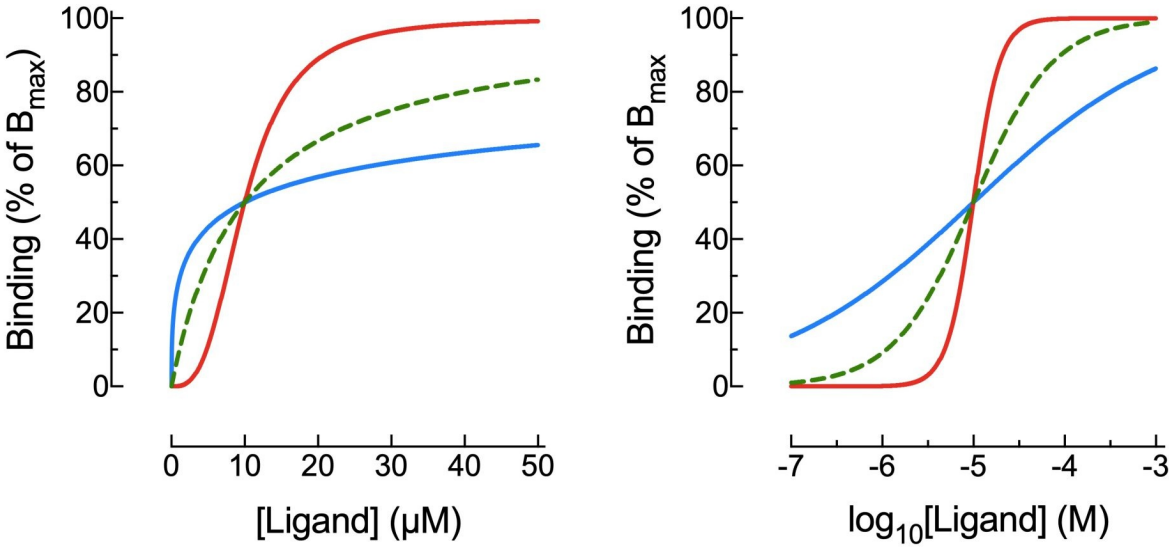
- **Cooperative binding** occurs when the binding of one ligand molecule affects the binding of subsequent ligand molecules. This can lead to non-linear binding isotherms, where the binding of ligands is enhanced or inhibited by the presence of other ligands. Cooperative binding is often seen in systems where multiple binding sites are involved or where the binding site is heterogeneous.

Kinetics of Binding

The kinetics of binding involves the study of the rates at which ligands bind to and dissociate from binding sites.

- **Rate constants** for association and dissociation are essential in describing the kinetics of binding. They represent the rate at which the ligand binds or dissociates to the site, respectively.

- **Kinetic models** are used to describe the binding process. One commonly used kinetic model to describe enzyme kinetics is the Michaelis-Menten model, which assumes that the enzyme has a single binding site and that the binding of the substrate is reversible. There are other kinetic models, including the Langmuir adsorption model.



Binding curves for three ligands following the Hill-Langmuir model, each with a K_d *(equilibrium dissociation constant) of 10 μM for its target protein. The blue ligand shows negative cooperativity of binding, meaning that binding of the first ligand reduces the binding affinity of the remaining site(s) for binding of a second ligand. The red ligand shows positive cooperativity of binding, meaning that binding of the first ligand increases the binding affinity of the remaining site(s) for binding of a second ligand. "Hill-Langmuir equation". Open Educational Alberta. https://openeducationalberta.ca/abcofpkpd/chapter/hill-langmuir/.*

Types of binding sites

Binding sites can be classified based on various criteria depending on the context. This is a non-exhaustive list of binding sites categories based on the type of molecule they bind to:

- Protein Binding Sites: These are regions on a protein where other molecules can bind. They can be further divided into:
  - Active Sites: Regions where enzymes bind substrates and catalyze chemical reactions. Example: The active site of the enzyme hexokinase binds to glucose and ATP, catalyzing the phosphorylation of glucose.
  - Allosteric Sites: Regions where ligands bind and alter the protein's activity without being part of the active site. Example: The binding of 2,3-bisphosphoglycerate (2,3-BPG) to hemoglobin enhances the ability of hemoglobin to release oxygen where it is most needed.
  - Regulatory Sites: Regions where ligands bind and regulate protein activity or localization. Example: Binding of a regulatory protein to a specific site on a receptor can modulate the receptor's activity.
- Nucleic Acid Binding Sites: These are regions on DNA or RNA where other molecules can bind. They can be further divided into:
  - Transcription Factor Binding Sites: Regions where transcription factors bind to regulate gene expression. Example: The TATA box is a DNA sequence that transcription factors bind to initiate transcription.
  - Restriction Sites: Regions where restriction enzymes bind to cleave DNA. Example: The EcoRI restriction enzyme recognizes and cuts the DNA sequence GAATTC.
  - Recombination Sites: Regions where site-specific recombinases bind to facilitate genetic recombination. Example: The loxP sites are recognized by the Cre recombinase enzyme to mediate recombination.
- Small Molecule Binding Sites: These are regions on proteins or nucleic acids where small molecules like drugs or substrates bind. They can be further divided into:
  - Compound Binding Sites: Regions typically located at the active site of the enzyme where the substrate binds and undergoes a chemical reaction, usually reversible. Example: The binding site for the drug aspirin on the

enzyme cyclooxygenase (COX) inhibits its activity.
    - Cofactor Binding Sites: Regions where cofactors can bind, sometimes permanently and covalently attached to the protein, and can be located at various sites. Example: The binding site for the heme cofactor in hemoglobin, which is essential for oxygen transport.
- Ion and Water Binding Sites: These are regions on proteins or nucleic acids where ions or water molecules bind. The calcium-binding sites in calmodulin, which are crucial for its role in signal transduction.

Computational methods to study binding sites

Multiple computational methods have been developed in the last decades to study and identify binding sites, offering insights that complement experimental approaches. These methods enable the prediction of binding affinities, the identification of potential drug candidates, and the understanding of the dynamic nature of binding interactions. Here are some key computational approaches used to study binding sites:

Method	Description	Applications	Tools	Output
Molecular Docking	Predicts the preferred orientation of a ligand when bound to a protein or nucleic acid binding site.	Used extensively in drug discovery to screen large libraries of compounds and identify potential drug candidates.	AutoDock, Glide, DOCK	Binding poses, binding affinities, interaction maps
Molecular Dynamics (MD) Simulations	Provides a dynamic view of the binding process by simulating the physical movements of atoms and molecules over time.	Used to study the stability of ligand binding, conformational changes, and the effect of mutations on binding affinity.	GROMACS, AMBER, CHARMM	Trajectories of atomic positions, binding free energies, insights into the flexibility and dynamics of binding sites
Quantum Mechanics/Molecular Mechanics (QM/MM) Methods	Combines quantum mechanical calculations for the active site with molecular mechanical calculations for the rest of the system.	Used to study reaction mechanisms, electronic properties, and the role of metal ions in binding sites.	Gaussian, ORCA, Q-Chem	Detailed electronic structure information, reaction pathways, energy profiles
Homology Modeling	Predicts the 3D structure of a protein based on the known structure of a homologous protein.	Useful for studying binding sites in proteins for which no experimental structure is available.	MODELLER, SWISS-MODEL, Phyre2	Predicted 3D structures that can be used for docking and MD simulations
Virtual Screening	Computationally screens large libraries of compounds to identify potential ligands that bind to a target binding site.	Widely used in drug discovery to prioritize compounds for experimental testing.	Schrodinger's Glide, AutoDock Vina, GOLD	Ranked lists of compounds with predicted binding affinities and poses
Machine Learning and AI	Uses machine learning and AI techniques to predict binding affinities, identify binding sites, and generate new ligand structures.	Enhancing the accuracy of docking predictions, predicting drug-target interactions, and designing novel compounds.	DeepChem, TensorFlow, PyTorch, protein language models (ESM2)	Predictive models, binding affinity predictions, novel ligand designs
Protein-Ligand Interaction Fingerprints (PLIF)	Computational representations of the interactions between a protein and a ligand.	Used to compare binding modes, identify key interactions, and cluster similar binding sites.	MOE, Schrödinger's Maestro	Interaction maps, similarity scores

Recently, large language models, particularly protein language models (PLMs), have emerged as powerful tools for predicting protein properties. These models typically use a transformer-based architecture to process protein sequences, learning relationships between amino acids and protein properties. PLMs can then be fine-tuned for specific tasks, such as binding site prediction, reducing the need for large, specific training datasets and offering high scalability.

DeepChem tools for studying binding sites

DeepChem in particular has a few tools and capabilities for identifying binding pockets on proteins:

- BindingPocketFinder: This is an abstract superclass in DeepChem that provides a template for child classes to algorithmically locate potential binding pockets on proteins. The idea is to help identify regions of the protein that may be good interaction sites for ligands or other molecules.
- ConvexHullPocketFinder: This is a specific implementation of the BindingPocketFinder class that uses the convex hull of the protein structure to find potential binding pockets. It takes in a protein structure and returns a list of binding pockets represented as CoordinateBoxes.
- Pose generators: Pose generation is the task of finding a “pose”, that is a geometric configuration of a small molecule interacting with a protein. A key step in computing the binding free energy of two complexes is to find low energy “poses”, that is energetically favorable conformations of molecules with respect to each other. This can be

useful for identifying favorable binding modes and orientations (low energy poses) of ligands within a protein's binding site. Current implementations allow for Autodock Vina and GNINA.
- Docking: There is a generic docking implementation that depends on provide pose generation and pose scoring utilities to perform docking.

There is a tutorial on using machine learning and molecular docking methods to predict the binding energy of a protein-ligand complex, and another tutorial on using atomic convolutions in particular to model such interactions.

How does a binding site look like?

Let's visualize a binding pocket in greater detail. For this purpose, we need to download the structure of a protein that contains a binding site. As an illustrative example, we show here a well-known protein-ligand pair: the binding of the drug Imatinib (Gleevec) to the Abl kinase domain of the BCR-Abl fusion protein, which is commonly associated with chronic myeloid leukemia (CML). The PDB ID for this complex is '1IEP'.

In [ ]:
```
# setup
!pip install py3Dmol biopython requests
import requests
import py3Dmol
from Bio.PDB import PDBParser, NeighborSearch
from io import StringIO
```

```
Requirement already satisfied: py3Dmol in /usr/local/lib/python3.10/dist-packages (2.1.0)
Requirement already satisfied: biopython in /usr/local/lib/python3.10/dist-packages (1.83)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.31.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from biopython) (1.25.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
s) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.
0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (20
24.6.2)
```

In [ ]:
```
# Download the protein-ligand complex (PDB file)
pdb_id = "1IEP"
url = f"https://files.rcsb.org/download/{pdb_id}.pdb"
response = requests.get(url)
pdb_content = response.text
```

Let's see what's inside the pdb file:

In [ ]:
```
# Parse the PDB content to identify chains and ligands
chains = {}
for line in pdb_content.splitlines():
    if line.startswith("ATOM") or line.startswith("HETATM"):
        chain_id = line[21]
        resn = line[17:20].strip()
        if chain_id not in chains:
            chains[chain_id] = {'residues': set(), 'ligands': set()}
        chains[chain_id]['residues'].add(resn)
        if line.startswith("HETATM"):
            chains[chain_id]['ligands'].add(resn)
chains_info = chains
print("Chains and Ligands Information:")
for chain_id, info in chains_info.items():
    print(f"Chain {chain_id}:")
    print(f"  Residues: {sorted(info['residues'])}")
    print(f"  Ligands: {sorted(info['ligands'])}")
```

```
Chains and Ligands Information:
Chain A:
  Residues: ['ALA', 'ARG', 'ASN', 'ASP', 'CL', 'CYS', 'GLN', 'GLU', 'GLY', 'HIS', 'HOH', 'ILE', 'LEU', 'LYS', 'M
ET', 'PHE', 'PRO', 'SER', 'STI', 'THR', 'TRP', 'TYR', 'VAL']
  Ligands: ['CL', 'HOH', 'STI']
Chain B:
  Residues: ['ALA', 'ARG', 'ASN', 'ASP', 'CL', 'CYS', 'GLN', 'GLU', 'GLY', 'HIS', 'HOH', 'ILE', 'LEU', 'LYS', 'M
ET', 'PHE', 'PRO', 'SER', 'STI', 'THR', 'TRP', 'TYR', 'VAL']
  Ligands: ['CL', 'HOH', 'STI']
```

We have a homodimer of two identical chains and several ligands: CL, HOH, and STI. CL and HOH are common solvents and ions binding to the protein structure, while STI is the ligand of interest (Imatinib).

For visualization, let's extract the information of chain A and its ligands:

In [ ]:
```
chain_id = "A"
chain_lines = []
for line in pdb_content.splitlines():
```

```
    if line.startswith("HETATM") or line.startswith("ATOM"):
        if line[21] == chain_id:
            chain_lines.append(line)
    elif line.startswith("TER"):
        if chain_lines and chain_lines[-1][21] == chain_id:
            chain_lines.append(line)
chain_A = "\n".join(chain_lines)
```

In [ ]:
```
view = py3Dmol.view()

# Add the extracted chain model
view.addModel(chain_A, "pdb")

# Set the styles
view.setStyle({'cartoon': {'color': 'darkgreen'}})
for ligand in chains_info[chain_id]['ligands']:
    view.addStyle({'chain': 'A', 'resn': ligand}, {'stick': {'colorscheme': 'yellowCarbon'}})
view.zoomTo()

view.show()
```

Now let's highlight the binding pocket in the protein ribbon. For this purpose we need to parse the PDB content again to identify residues belonging to chain A and STI ligand:

In [ ]:
```
parser = PDBParser()
structure = parser.get_structure('protein', StringIO(pdb_content))

# Extract residues of chain A and the ligand STI
str_chain_A = structure[0][chain_id]
ligand = None
ligand_resname = "STI"
for residue in structure.get_residues():
    if residue.id[0] != ' ' and residue.resname == ligand_resname:
        ligand = residue
        break
if ligand is None:
    raise ValueError("Ligand STi not found in the PDB content")

chain_A_atoms = [atom for atom in str_chain_A.get_atoms()]
ligand_atoms = [atom for atom in ligand.get_atoms()]
```

```
/usr/local/lib/python3.10/dist-packages/Bio/PDB/StructureBuilder.py:89: PDBConstructionWarning: WARNING: Chain A is discontinuous at line 5038.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/Bio/PDB/StructureBuilder.py:89: PDBConstructionWarning: WARNING: Chain B is discontinuous at line 5079.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/Bio/PDB/StructureBuilder.py:89: PDBConstructionWarning: WARNING: Chain A is discontinuous at line 5118.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/Bio/PDB/StructureBuilder.py:89: PDBConstructionWarning: WARNING: Chain B is discontinuous at line 5217.
  warnings.warn(
```

Now we can find the residues in chain A within a certain distance from the ligand. Here we set this distance to 5Å.

In [ ]:
```
binding_residues = set()
distance_threshold = 5.0
ns = NeighborSearch(chain_A_atoms)

for ligand_atom in ligand_atoms:
    close_atoms = ns.search(ligand_atom.coord, distance_threshold)
    for atom in close_atoms:
        binding_residues.add(atom.get_parent().id[1])
```

And let's see the visualization, now with the binding pocket:

In [ ]:
```
# Create a py3Dmol view
view = py3Dmol.view()

# Add the chain model
view.addModel(chain_A, "pdb")

# Set the style for the protein
view.setStyle({'cartoon': {'color': 'darkgreen'}})

# Color the binding pocket residues
for resi in binding_residues:
    view.addStyle({'chain': chain_id, 'resi': str(resi)}, {'cartoon': {'colorscheme': 'greenCarbon'}})

# Color the ligand
view.addStyle({'chain': chain_id, 'resn': ligand_resname}, {'stick': {'colorscheme': 'yellowCarbon'}})

# Zoom to the view
view.zoomTo()

# Show the view
view.show()
```

Further Reading

For further reading on computational methods for binding sites and protein language models here are a couple of great resources:

- Exploring the computational methods for protein-ligand binding site prediction -Getting started with protein language models

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing this tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

```
@manual{Bioinformatics,
 title={Introduction to Binding Sites},
 organization={DeepChem},
 author={Gómez de Lope, Elisa},
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Introduction_to_Binding_Si

 year={2024},
}
```

Tutorial Part 13: Modeling Protein-Ligand Interactions

By Nathan C. Frey | Twitter and Bharath Ramsundar | Twitter

In this tutorial, we'll walk you through the use of machine learning and molecular docking methods to predict the binding energy of a protein-ligand complex. Recall that a ligand is some small molecule which interacts (usually non-covalently) with a protein. Molecular docking performs geometric calculations to find a “binding pose” with a small molecule interacting with a protein in a suitable binding pocket (that is, a region on the protein which has a groove in which the small molecule can rest).

The structure of proteins can be determined experimentally with techniques like Cryo-EM or X-ray crystallography. This can be a powerful tool for structure-based drug discovery. For more info on docking, read the AutoDock Vina paper and the `deepchem.dock` documentation. There are many graphical user and command line interfaces (like AutoDock) for performing molecular docking. Here, we show how docking can be performed programmatically with DeepChem, which enables automation and easy integration with machine learning pipelines.

As you work through the tutorial, you'll trace an arc including

1. Loading a protein-ligand complex dataset (PDBbind)
2. Performing programmatic molecular docking
3. Featurizing protein-ligand complexes with interaction fingerprints
4. Fitting a random forest model and predicting binding affinities

To start the tutorial, we'll use a simple pre-processed dataset file that comes in the form of a gzipped file. Each row is a molecular system, and each column represents a different piece of information about that system. For instance, in this example, every row reflects a protein-ligand complex, and the following columns are present: a unique complex identifier; the SMILES string of the ligand; the binding affinity (Ki) of the ligand to the protein in the complex; a Python `list` of all lines in a PDB file for the protein alone; and a Python `list` of all lines in a ligand file for the ligand alone.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5 minutes to run to completion and install your environment.

In [1]:
```
!pip install -q condacolab
import condacolab
condacolab.install()
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the syst
em package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
     Everything looks OK!
```

In [2]:
```
!conda install -c conda-forge openmm
```

```
Collecting package metadata (current_repodata.json): - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / done
Solving environment: \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / done

# All requested packages already installed.
```

In [3]:
```
!pip install deepchem
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: deepchem in /usr/local/lib/python3.10/site-packages (2.7.1)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/site-packages (from deepchem) (1.24.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/site-packages (from deepchem) (1.2.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/site-packages (from deepchem) (1.2.0)
Requirement already satisfied: scipy<1.9 in /usr/local/lib/python3.10/site-packages (from deepchem) (1.8.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/site-packages (from deepchem) (2.0.1)
Requirement already satisfied: rdkit in /usr/local/lib/python3.10/site-packages (from deepchem) (2023.3.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/site-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/site-packages (from pandas->deepchem) (
2023.3)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/site-packages (from pandas->deepchem)
(2023.3)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/site-packages (from rdkit->deepchem) (9.5.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/site-packages (from scikit-lear
n->deepchem) (3.1.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/site-packages (from python-dateutil>=2.8.2-
>pandas->deepchem) (1.16.0)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the syst
em package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [4]:
```
!conda install -c conda-forge pdbfixer
```

```
Collecting package metadata (current_repodata.json): - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - done
Solving environment: | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ done

# All requested packages already installed.
```

In [5]:
```
!conda install -c conda-forge vina
```

```
Collecting package metadata (current_repodata.json): - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \
| / - \ | / - \ | / - \ | / - \ | / - \ done
Solving environment: / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / - \ | / done

# All requested packages already installed.
```

Protein-ligand complex data

It is really helpful to visualize proteins and ligands when doing docking. Unfortunately, Google Colab doesn't currently support the Jupyter widgets we need to do that visualization. Install `MDTraj` and `nglview` on your local machine to view the protein-ligand complexes we're working with.

In [6]:
```
!pip install -q mdtraj nglview
# !jupyter-nbextension enable nglview --py --sys-prefix  # for jupyter notebook
# !jupyter labextension install  nglview-js-widgets  # for jupyter lab
```

```
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the syst
em package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [8]:
```
import os
import numpy as np
import pandas as pd

import tempfile

from rdkit import Chem
from rdkit.Chem import AllChem
import deepchem as dc

from deepchem.utils import download_url, load_from_disk
```

```
Skipped loading modules with pytorch-geometric dependency, missing a dependency. No module named 'torch_geometri
c'
Skipped loading modules with pytorch-geometric dependency, missing a dependency. cannot import name 'DMPNN' from
'deepchem.models.torch_models' (/usr/local/lib/python3.10/site-packages/deepchem/models/torch_models/__init__.py
)
Skipped loading modules with pytorch-lightning dependency, missing a dependency. No module named 'pytorch_lightn
ing'
Skipped loading some Jax models, missing a dependency. No module named 'haiku'
```

To illustrate the docking procedure, here we'll use a csv that contains SMILES strings of ligands as well as PDB files for the ligand and protein targets from PDBbind. Later, we'll use the labels to train a model to predict binding affinities. We'll also show how to download and featurize PDBbind to train a model from scratch.

In [9]:
```
data_dir = dc.utils.get_data_dir()
dataset_file = os.path.join(data_dir, "pdbbind_core_df.csv.gz")

if not os.path.exists(dataset_file):
    print('File does not exist. Downloading file...')
    download_url("https://s3-us-west-1.amazonaws.com/deepchem.io/datasets/pdbbind_core_df.csv.gz")
    print('File downloaded...')

raw_dataset = load_from_disk(dataset_file)
raw_dataset = raw_dataset[['pdb_id', 'smiles', 'label']]
```

Let's see what `raw_dataset` looks like:

In [10]:
```
raw_dataset.head(2)
```

Out[10]:

	pdb_id	smiles	label
0	2d3u	CC1CCCCC1S(O)(O)NC1CC(C2CCC(CN)CC2)SC1C(O)O	6.92
1	3cyx	CC(C)(C)NC(O)C1CC2CCCCC2C[NH+]1CC(O)C(CC1CCCC...	8.00

Fixing PDB files

Next, let's get some PDB protein files for visualization and docking. We'll use the PDB IDs from our `raw_dataset` and download the pdb files directly from the Protein Data Bank using `pdbfixer`. We'll also sanitize the structures with RDKit. This ensures that any problems with the protein and ligand files (non-standard residues, chemical validity, etc.) are corrected. Feel free to modify these cells and pdbids to consider new protein-ligand complexes. We note here that PDB files are complex and human judgement is required to prepare protein structures for docking. DeepChem includes a number of docking utilites to assist you with preparing protein files, but results should be inspected before docking is attempted.

In [11]:
```
from openmm.app import PDBFile
from pdbfixer import PDBFixer

from deepchem.utils.vina_utils import prepare_inputs
```

In [12]:
```
# consider one protein-ligand complex for visualization
pdbid = raw_dataset['pdb_id'].iloc[1]
ligand = raw_dataset['smiles'].iloc[1]
```

In [13]:
```
%%time
fixer = PDBFixer(pdbid=pdbid)
PDBFile.writeFile(fixer.topology, fixer.positions, open('%s.pdb' % (pdbid), 'w'))

p, m = None, None
# fix protein, optimize ligand geometry, and sanitize molecules
try:
    p, m = prepare_inputs('%s.pdb' % (pdbid), ligand)
except:
    print('%s failed PDB fixing' % (pdbid))

if p and m:  # protein and molecule are readable by RDKit
    print(pdbid, p.GetNumAtoms())
    Chem.rdmolfiles.MolToPDBFile(p, '%s.pdb' % (pdbid))
    Chem.rdmolfiles.MolToPDBFile(m, 'ligand_%s.pdb' % (pdbid))
```

```
<timed exec>:7: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
Warning: importing 'simtk.openmm' is deprecated.  Import 'openmm' instead.
3cyx 1510
CPU times: user 2.04 s, sys: 157 ms, total: 2.2 s
Wall time: 4.32 s
```

Visualization

If you're outside of Colab, you can expand these cells and use `MDTraj` and `nglview` to visualize proteins and ligands.

In [14]:
```
import mdtraj as md
import nglview

from IPython.display import display, Image
```

Let's take a look at the first protein ligand pair in our dataset:

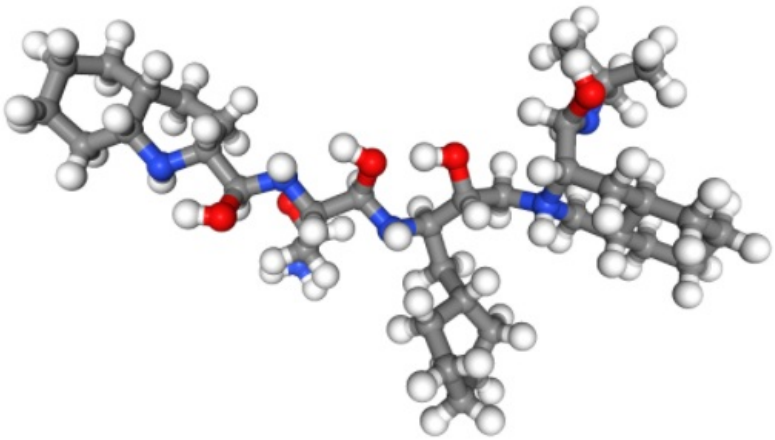
In [15]:
```
protein_mdtraj = md.load_pdb('3cyx.pdb')
ligand_mdtraj = md.load_pdb('ligand_3cyx.pdb')
```

We'll use the convenience function `nglview.show_mdtraj` in order to view our proteins and ligands. Note that this will only work if you uncommented the above cell, installed nglview, and enabled the necessary notebook extensions.

In [16]:
```
v = nglview.show_mdtraj(ligand_mdtraj)
```

In [17]:
```
display(v)  # interactive view outside Colab
```

```
NGLWidget()
```



Now that we have an idea of what the ligand looks like, let's take a look at our protein:

In [18]:
```
view = nglview.show_mdtraj(protein_mdtraj)
display(view)  # interactive view outside Colab
```

```
NGLWidget()
```



Molecular Docking

Ok, now that we've got our data and basic visualization tools up and running, let's see if we can use molecular docking to estimate the binding affinities between our protein ligand systems.

There are three steps to setting up a docking job, and you should experiment with different settings. The three things we need to specify are 1) how to identify binding pockets in the target protein; 2) how to generate poses (geometric configurations) of a ligand in a binding pocket; and 3) how to "score" a pose. Remember, our goal is to identify candidate ligands that strongly interact with a target protein, which is reflected by the score.

DeepChem has a simple built-in method for identifying binding pockets in proteins. It is based on the convex hull method. The method works by creating a 3D polyhedron (convex hull) around a protein structure and identifying the surface atoms of the protein as the ones closest to the convex hull. Some biochemical properties are considered, so the method is not purely geometrical. It has the advantage of having a low computational cost and is good enough for our purposes.

In [19]:
```
finder = dc.dock.binding_pocket.ConvexHullPocketFinder()
pockets = finder.find_pockets('3cyx.pdb')
len(pockets)  # number of identified pockets
```

Out[19]:
```
36
```

Pose generation is quite complex. Luckily, using DeepChem's pose generator will install the AutoDock Vina engine under

the hood, allowing us to get up and running generating poses quickly.

In [20]:
```
vpg = dc.dock.pose_generation.VinaPoseGenerator()
```

We could specify a pose scoring function from `deepchem.dock.pose_scoring`, which includes things like repulsive and hydrophobic interactions and hydrogen bonding. Vina will take care of this, so instead we'll allow Vina to compute scores for poses.

In [7]:
```
!mkdir -p vina_test
```

In [21]:
```
%%time
complexes, scores = vpg.generate_poses(molecular_complex=('3cyx.pdb', 'ligand_3cyx.pdb'),  # protein-ligand fil
                                       out_dir='vina_test',
                                       generate_scores=True
                                      )
```

```
CPU times: user 41min 4s, sys: 21.9 s, total: 41min 26s
Wall time: 28min 32s
```

```
/usr/local/lib/python3.10/site-packages/vina/vina.py:260: DeprecationWarning: `np.int` is a deprecated alias for
the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is
safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If yo
u wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#dep
recations
  self._voxels = np.ceil(np.array(box_size) / self._spacing).astype(np.int)
```

We used the default value for `num_modes` when generating poses, so Vina will return the 9 lowest energy poses it found in units of `kcal/mol`.

In [22]:
```
scores
```

Out[22]:
```
[-9.484, -9.405, -9.195, -9.151, -8.9, -8.696, -8.687, -8.633, -8.557]
```

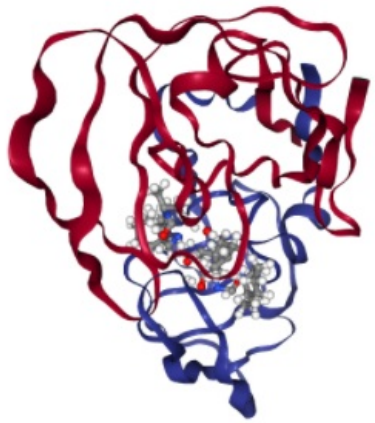
Can we view the complex with both protein and ligand? Yes, but we'll need to combine the molecules into a single RDkit molecule.

In [23]:
```
complex_mol = Chem.CombineMols(complexes[0][0], complexes[0][1])
```

Let's now visualize our complex. We can see that the ligand slots into a pocket of the protein.

In [24]:
```
v = nglview.show_rdkit(complex_mol)
display(v)
```

```
NGLWidget()
```



Now that we understand each piece of the process, we can put it all together using DeepChem's `Docker` class. Docker creates a generator that yields tuples of posed complexes and docking scores.

In [25]:
```
docker = dc.dock.docking.Docker(pose_generator=vpg)
posed_complex, score = next(docker.dock(molecular_complex=('3cyx.pdb', 'ligand_3cyx.pdb'),
                                        use_pose_generator_scores=True))
```

Modeling Binding Affinity

Docking is a useful, albeit coarse-grained tool for predicting protein-ligand binding affinities. However, it takes some time, especially for large-scale virtual screenings where we might be considering different protein targets and thousands of potential ligands. We might naturally ask then, can we train a machine learning model to predict docking scores? Let's try and find out!

We'll show how to download the PDBbind dataset. We can use the loader in MoleculeNet to get the 4852 protein-ligand complexes from the "refined" set or the entire "general" set in PDBbind. For simplicity, we'll stick with the ~100 complexes we've already processed to train our models.

Next, we'll need a way to transform our protein-ligand complexes into representations which can be used by learning algorithms. Ideally, we'd have neural protein-ligand complex fingerprints, but DeepChem doesn't yet have a good learned fingerprint of this sort. We do however have well-tuned manual featurizers that can help us with our challenge here.

We'll make use of two types of fingerprints in the rest of the tutorial, the `CircularFingerprint` and `ContactCircularFingerprint`. DeepChem also has voxelizers and grid descriptors that convert a 3D volume containing an arragment of atoms into a fingerprint. These featurizers are really useful for understanding protein-ligand complexes since they allow us to translate complexes into vectors that can be passed into a simple machine learning algorithm. First, we'll create circular fingerprints. These convert small molecules into a vector of fragments.

In [26]:
```
pdbids = raw_dataset['pdb_id'].values
ligand_smiles = raw_dataset['smiles'].values
```

In [29]:
```
%%time
for (pdbid, ligand) in zip(pdbids, ligand_smiles):
  fixer = PDBFixer(url='https://files.rcsb.org/download/%s.pdb' % (pdbid))
  PDBFile.writeFile(fixer.topology, fixer.positions, open('%s.pdb' % (pdbid), 'w'))

  p, m = None, None
  # skip pdb fixing for speed
  try:
    p, m = prepare_inputs('%s.pdb' % (pdbid), ligand, replace_nonstandard_residues=False,
                          remove_heterogens=False, remove_water=False,
                          add_hydrogens=False)
  except:
    print('%s failed sanitization' % (pdbid))

  if p and m:  # protein and molecule are readable by RDKit
    Chem.rdmolfiles.MolToPDBFile(p, '%s.pdb' % (pdbid))
    Chem.rdmolfiles.MolToPDBFile(m, 'ligand_%s.pdb' % (pdbid))
```

```
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:11:45] UFFTYPER: Unrecognized atom type: S_5+4 (7)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
3cyx failed sanitization
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:02] UFFTYPER: Warning: hybridization set to SP3 for atom 17
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:04] UFFTYPER: Warning: hybridization set to SP3 for atom 6
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:06] UFFTYPER: Warning: hybridization set to SP3 for atom 1
[15:12:06] UFFTYPER: Unrecognized atom type: S_5+4 (21)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:23] UFFTYPER: Warning: hybridization set to SP3 for atom 20
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
```

```
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:31] UFFTYPER: Warning: hybridization set to SP3 for atom 19
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:12:35] UFFTYPER: Warning: hybridization set to SP3 for atom 29
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:13:03] UFFTYPER: Unrecognized atom type: S_5+4 (39)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:13:37] UFFTYPER: Warning: hybridization set to SP3 for atom 33
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:01] UFFTYPER: Unrecognized atom type: S_5+4 (11)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:02] UFFTYPER: Unrecognized atom type: S_5+4 (47)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:14] UFFTYPER: Unrecognized atom type: S_5+4 (1)
```

```
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:27] UFFTYPER: Warning: hybridization set to SP3 for atom 6
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:33] UFFTYPER: Unrecognized atom type: S_5+4 (47)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:43] UFFTYPER: Unrecognized atom type: S_5+4 (28)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:55] UFFTYPER: Warning: hybridization set to SP3 for atom 17
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:14:57] UFFTYPER: Warning: hybridization set to SP3 for atom 6
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:08] Explicit valence for atom # 388 O, 3, is greater than permitted
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:15] UFFTYPER: Warning: hybridization set to SP3 for atom 9
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:19] UFFTYPER: Unrecognized atom type: S_5+4 (6)
3utu failed sanitization
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:29] UFFTYPER: Unrecognized atom type: S_5+4 (1)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
```

```
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:39] UFFTYPER: Unrecognized atom type: S_5+4 (19)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:43] UFFTYPER: Unrecognized atom type: S_5+4 (21)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:15:57] UFFTYPER: Unrecognized atom type: S_5+4 (9)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:16:01] UFFTYPER: Warning: hybridization set to SP3 for atom 18
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:16:21] UFFTYPER: Warning: hybridization set to SP3 for atom 17
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:16:42] UFFTYPER: Warning: hybridization set to SP3 for atom 10
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:19] UFFTYPER: Unrecognized atom type: S_5+4 (13)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
```

```
ction in deepchem.utils.docking_utils.
[15:17:25] UFFTYPER: Unrecognized atom type: S_5+4 (10)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:27] UFFTYPER: Unrecognized atom type: S_5+4 (6)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:28] UFFTYPER: Warning: hybridization set to SP3 for atom 11
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:46] UFFTYPER: Unrecognized atom type: S_5+4 (8)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:17:58] UFFTYPER: Unrecognized atom type: S_5+4 (4)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:02] UFFTYPER: Unrecognized atom type: S_5+4 (9)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:15] UFFTYPER: Unrecognized atom type: S_5+4 (1)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:32] UFFTYPER: Unrecognized atom type: S_5+4 (23)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:35] UFFTYPER: Unrecognized atom type: S_5+4 (22)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:18:42] UFFTYPER: Warning: hybridization set to SP3 for atom 8
[15:18:42] UFFTYPER: Warning: hybridization set to SP3 for atom 24
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
```

```
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:01] UFFTYPER: Warning: hybridization set to SP3 for atom 16
[15:19:01] UFFTYPER: Unrecognized atom type: S_5+4 (20)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:02] UFFTYPER: Unrecognized atom type: S_5+4 (6)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:05] UFFTYPER: Unrecognized atom type: S_5+4 (6)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
1hfs failed sanitization
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:22] UFFTYPER: Warning: hybridization set to SP3 for atom 20
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:41] Explicit valence for atom # 1800 C, 5, is greater than permitted
[15:19:41] UFFTYPER: Unrecognized atom type: S_5+4 (11)
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:42] UFFTYPER: Warning: hybridization set to SP3 for atom 11
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:57] UFFTYPER: Warning: hybridization set to SP3 for atom 9
[15:19:57] UFFTYPER: Warning: hybridization set to SP3 for atom 23
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 8
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 12
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 34
[15:19:59] UFFTYPER: Warning: hybridization set to SP3 for atom 41
<timed exec>:8: DeprecationWarning: Call to deprecated function prepare_inputs. Please use the corresponding fun
ction in deepchem.utils.docking_utils.
CPU times: user 4min 9s, sys: 3.31 s, total: 4min 12s
Wall time: 8min 19s
```

In [28]:
```
proteins = [f for f in os.listdir('.') if len(f) == 8 and f.endswith('.pdb')]
ligands = [f for f in os.listdir('.') if f.startswith('ligand') and f.endswith('.pdb')]
```

We'll do some clean up to make sure we have a valid ligand file for every valid protein. The lines here will compare the PDB IDs between the ligand and protein files and remove any proteins that don't have corresponding ligands.

In [30]:
```
# Handle failed sanitizations
failures = set([f[:-4] for f in proteins]) - set([f[7:-4] for f in ligands])
for pdbid in failures:
  proteins.remove(pdbid + '.pdb')
```

In [31]:
```
len(proteins), len(ligands)
```

Out[31]:
```
(190, 190)
```

In [32]:
```
pdbids = [f[:-4] for f in proteins]
small_dataset = raw_dataset[raw_dataset['pdb_id'].isin(pdbids)]
labels = small_dataset.label
```

In [33]:
```
fp_featurizer = dc.feat.CircularFingerprint(size=2048)
```

In [34]:
```
features = fp_featurizer.featurize([Chem.MolFromPDBFile(l) for l in ligands])
```

In [35]:
```
dataset = dc.data.NumpyDataset(X=features, y=labels, ids=pdbids)
train_dataset, test_dataset = dc.splits.RandomSplitter().train_test_split(dataset, seed=42)
```

The convenience loader `dc.molnet.load_pdbbind` will take care of downloading and featurizing the pdbbind dataset under the hood for us. This will take quite a bit of time and compute, so the code to do it is commented out. Uncomment it and grab a cup of coffee if you'd like to featurize all of PDBbind's refined set. Otherwise, you can continue with the small dataset we constructed above.

In [36]:
```
# # Uncomment to featurize all of PDBBind's "refined" set
# pdbbind_tasks, (train_dataset, valid_dataset, test_dataset), transformers = dc.molnet.load_pdbbind(
#     featurizer=fp_featurizer, set_name="refined", reload=True,
#     data_dir='pdbbind_data', save_dir='pdbbind_data')
```

Now, we're ready to do some learning!

To fit a deepchem model, first we instantiate one of the provided (or user-written) model classes. In this case, we have a created a convenience class to wrap around any ML model available in Sci-Kit Learn that can in turn be used to interoperate with deepchem. To instantiate an `SklearnModel`, you will need (a) task_types, (b) model_params, another `dict` as illustrated below, and (c) a `model_instance` defining the type of model you would like to fit, in this case a `RandomForestRegressor`.

In [37]:
```
from sklearn.ensemble import RandomForestRegressor

from deepchem.utils.evaluate import Evaluator
import pandas as pd
```

In [38]:
```
seed = 42 # Set a random seed to get stable results
sklearn_model = RandomForestRegressor(n_estimators=100, max_features='sqrt')
sklearn_model.random_state = seed
model = dc.models.SklearnModel(sklearn_model)
model.fit(train_dataset)
```

Note that the
R^2
value for the test set indicates that the model isn't producing meaningful outputs. It turns out that predicting binding affinities is *hard*. This tutorial isn't meant to show how to create a state-of-the-art model for predicting binding affinities, but it gives you the tools to generate your own datasets with molecular docking, featurize complexes, and train models.

In [39]:
```
# use Pearson correlation so metrics are > 0
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)

evaluator = Evaluator(model, train_dataset, [])
train_r2score = evaluator.compute_model_performance([metric])
print("RF Train set R^2 %f" % (train_r2score["pearson_r2_score"]))

evaluator = Evaluator(model, test_dataset, [])
test_r2score = evaluator.compute_model_performance([metric])
print("RF Test set R^2 %f" % (test_r2score["pearson_r2_score"]))
```

```
RF Train set R^2 0.888697
RF Test set R^2 0.007797
```

We're using a very small dataset and an overly simplistic representation, so it's no surprise that the test set performance is quite bad.

In [40]:
```
# Compare predicted and true values
list(zip(model.predict(train_dataset), train_dataset.y))[:5]
```

Out[40]:
```
[(6.862549999999994, 7.4),
 (6.616400000000008, 6.85),
 (4.852004999999995, 3.4),
 (6.43060000000001, 6.72),
 (8.66322999999999, 11.06)]
```

In [41]:
```
list(zip(model.predict(test_dataset), test_dataset.y))[:5]
```

Out[41]:
```
[(5.960549999999999, 4.21),
 (6.051305714285715, 8.7),
 (5.799900000000003, 6.39),
 (6.433881666666665, 4.94),
 (6.7465399999999995, 9.21)]
```

The protein-ligand complex view.

In the previous section, we featurized only the ligand. This time, let's see if we can do something sensible with our protein-ligand fingerprints that make use of our structural information. To start with, we need to re-featurize the dataset but using the contact fingerprint this time.

In [42]:
```
fp_featurizer = dc.feat.ContactCircularFingerprint(size=2048)
```

In [43]:
```
features = fp_featurizer.featurize(zip(ligands, proteins))
dataset = dc.data.NumpyDataset(X=features, y=labels, ids=pdbids)
train_dataset, test_dataset = dc.splits.RandomSplitter().train_test_split(dataset, seed=42)
```

```
[15:21:40] Explicit valence for atom # 3 C, 5, is greater than permitted
Mol [H]OC([H])([H])[C@]1([H])O[C@@]2([H])SC([H])([H])(N([H])C([H])([H])C([H])([H])[H])N([H])[C@@]2([H])[C@@]([H]
)(O[H])[C@]1([H])O[H] failed sanitization
[15:21:47] Explicit valence for atom # 214 O, 3, is greater than permitted
Mol CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CN
C3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H
]2CCCN2C(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2
CCCN2C(=O)CNC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=
O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)
NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@@H]2CSSC[C@H](NC(=O
)[C@@H](NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]3CCCN
3C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[
C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(N)=O)NC(=O)CN)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)C(
=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@
H]([C@@H](C)CC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@@H](CC(=O)O)C(=
O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N2)CSSC[C@@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@
H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)NCC(=O)N[C@@H](CC
2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C
2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CNC3=C2C=CC
=C3)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@
H](CCC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N2C3C(=O)O32)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)CNC
(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@H](C(C)C)NC1=O.CC[C@H](C)[C@H](
NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@]12C(CC(=O)O)C13C(C)[C@H](
N)C(=O)N32)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H]
(CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(
=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C
)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=C(O)C=C2
)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[
C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=
O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](
CCCCN)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=
O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C
@H](C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC3=CNC=N3)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCCNC(=N)N)C(=
O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@
@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](
CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N3CCC[C@H]3C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)
N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC3=CNC=N3)C(=O)N3CCC[C@H]
3C(=O)N[C@H](C(=O)N[C@@H](CSSC[C@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)N)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](
CCCNC(=N)N)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCC
N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H]
(CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)
N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@]34C(=O)O3C4C3=CC=C(O)C=C3)[C@@H](C)O)C(=O)N[C@@H
](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C
@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H]
(C)C(=O)NCC(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H]
(C(=O)NCC(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)
N[C@H](C=O)CCC(=O)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@
H](C)CC)[C@@H](C)CC)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC2=O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H](CC2=CN
C=N2)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC2=CNC
3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)CNC1=O)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O
)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](N)CC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@
H](CO)CC1CCC(OS(O)(O)O)CC1.CO[C@H]1CC[C@H](S(O)(O)N[C@@H](C[C@@H](O)NC[C@H]2CC[C@H](CN)CC2)C(O)N2CCC[C@H]2[C@H](
O)NCC2CCC(C(N)N)CC2)CC1Cl.O.O.O
```

[15:22:27] Explicit valence for atom # 5358 O, 3, is greater than permitted
Mol CC(O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H]1CC(O)=OO=C([C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CNC=N2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)

(CC(=O)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]1CCCN1
C(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)N
C(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@
H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C
@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](C)NC
(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](
CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](N)C(C)C)C(C)
C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H]
(C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C
(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[
C@@H](C)CC)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@
H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=
O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)
O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O
)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(
=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N1CCC[C@H]1C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H
](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)
C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)
C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C
=C1)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(
=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O
)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C
=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@
@H](C)C=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C.CC[C@H](C)[C@H](NC(=O)CN
C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=
C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=
O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO
)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)N)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CC1=CC=C(O
)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O
)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H]
(CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C
@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(
=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H
](CCCNC(=N)N)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N
[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C
@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N
[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)
C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)
N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H]
(C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C
@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC
(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC
NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)N)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(=O)N[C@@
H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](
CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C
(=O)N1CCC[C@H]1C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@
H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)
O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1
=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=
O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC
C(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@
H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CC1

0.0.0.

NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC

O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]
](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCC
N)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(
=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](
C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(
=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@
H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)N
C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCC
N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=
C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)
CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@
H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](
CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(
=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(
=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@
H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C
@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H
](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](CCCCN)NC(=O)[C@@H](NC(=
O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C
CC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)C
NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](NC(=O)
)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1
=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O
)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[
C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H
](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)CNC(=O)[C
@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)
NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[
C@@H](NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(
=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C
@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C
C(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](
CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)
[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H
]1CCCN1C(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN
1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](N)CCC(N)=O)C(C)C)C(C)C)[C@@H](C
)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@
H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)
[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C
@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C
@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@
H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](
C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@
H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)O.Cl.Cl.Cl.Cl.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0
.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0

[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C

(=O)O)NC(=O)[C@H](CCCCN)NC1=O)C(C)C.N=C(N)NCCCCC(=O)O.O.O.

1C=CC=C2)C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1CC1N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1OC1=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H]1CCCNC2=[N+](N2)O=C(N)CC[C@@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)NC1=O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C1CCC1N)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CC(C)

)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(
=O)CNC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C
)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CC
(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](
CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=
CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H
](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O
)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCC
NC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=
O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H
](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H]
(CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@
@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[
C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](C
H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC
(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CN)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](
C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@
@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CCC
NC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=C
C=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N
)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C
(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C
)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=
N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@
H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C
CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C
@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC
(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCC
N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C
(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C
@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C
=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N1CCC[C
@H]1C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C
(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)
C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N
[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)
C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)
C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)
C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CCSC)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H]
(CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]
(CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N
[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)
O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CO)C(=O)N[C@@
H](CO)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)
N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)
C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=
CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C
(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=CC=
C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O
)O)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)
N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(N)=
O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H]
(CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@
H](C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)

=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H]1CCC(=O)NC(N)CCC[C@@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC2N3C(N)=[N+]23)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC2=OO2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CS)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCSC)NC1=O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC

NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](

@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC

(NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C
C1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C
C1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N
1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC
(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1C
CCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O
)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](
CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](
NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(
=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)N
C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C
(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](
C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C
=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@
@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C
@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C
@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO
)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=
O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC
(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C
@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)N
C(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C
@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O

C(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O

O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)

[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C

CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)
CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C
(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C
C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O
)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O
)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC
C(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)
NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC
(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=
O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=
O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=
O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)N
C(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(
=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC
NC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)
[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C
@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](N)CO)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=
O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O
)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N
[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]
(NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C
C1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C
C1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N
1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC
(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1C
CCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O
)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](
CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](
CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](
NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(
=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O
)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](N)CO)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H
](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)
C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H
](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N
[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.0.0.0

0.0.0.

C2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC2=OO2)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H]2CC(O)=ON2C(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](N)CCCCN)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)CCC(=O)ONCCCC[C@@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N2CCC[C@H]2C(=O)N[C@H]2CSSC[C@@H](C=O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H]3CCCN3C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(=O)O)NC2=O)[C@@H](C)CC)[C@@H](C)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C(O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)C1=O.CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CO)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O

C(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC1=O.CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC2=NN2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H]2CC(O)=O3N2C3(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]3CCCN3C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C4)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)CNC(=O)[C@H](CS)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CS)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]3CCCN3C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CC(N)=O)NC(=O)C(CCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@]3(N)CC3CCN)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)CC2C(N)=O)C2CC(=O)O2)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)CSSC[C@@H](C=O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC1=O.C[C@@H]1ONC(O)[C@@H]1C[C@H](N)C(O)O.C[C@@H]1ONC(O)[C@@H]1C[C@H](N)C(O)O.C[C@H]1ONC(O)[C@@H]1C[C@H](N)C(O)O.0.0.0

CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CO)NC(=O)[C@@H]2CCCN2C(=O)[C@@H](NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@H]3O[C@H](CO)[C@@H](O)[C@H](O)[C@H]3NC(C)O)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC3=ON3)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H]3CCCN3C(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H]3CCCN3C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)C(C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC2=OO2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N1)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)C(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CSSC[C@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCN1NC1=N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](N)CO)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NC1=O)C(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(=O)N[C@@H]1O[C@H](CO)[C@@H](O)[C@H](O)[C@H]1NC(C)O)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(=O)N[C@@H]1O[C@H](CO)[C@@H](O)[C@H](O)[C@H]1NC(C)O)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC

)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](N)CCSC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C0)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)NCC(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CS)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CS)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CS)C(=O)N[C@@H](C)C(=O)N[C@H](

@H](CCCNC(=N)N)C(=O)O)[C@@H](C)O)C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC3=CNC=N3)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H]4CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H]4CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H]4CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@H]3CSSC[C@@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC4=CC=C(O)C=C4)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)

@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N3CCC[C@H]3C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](CC(N)=O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C3CN3C)C(=O)N[C@@H](C3C4C(=O)O43)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N2)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)C(=O)N[C@H]2CC(N)=ONC(=O)C[C@H](NC(=O)[C@H](CC(C)C)NC2=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2OC2=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N1)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1CC=CC=C1)C(=O)N[C@@H](CCS)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@]1(C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=C(O)C=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H]2CSSC[C@@H](C(=O)N[C@H](C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC3=CNC4=C3C=CC=C4)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C3=OC4567=O89%10N(C(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)N3)[C@@]8([C@@H](C)O)C9(=O)[N+]%104[C@H]5C607)C(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCC(=O)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H]([C@@H](C)O)NC(=O)CNC(=O)[C@H](CC(=O)N[C@@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)NCC(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O



(C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N
[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](
CC2=CNC=N2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2
C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](C
C2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(
=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[
C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=
O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC
(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)NCC(=O)N[C@@H](C)
C(=O)N[C@@H](CS)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N2CCC[C@H]2C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(
=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CNC=N2)C(=O)NCC(=O)
N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O
)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C
@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C
O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O
)N[C@H](C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC2
=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@
@H](CCCNC(=N)N)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N
[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)NCC(=O)N[C@@H](CCCNC(=N)N)C(
=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CS)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C
(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](C
)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@
H](CC(C)C)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)
C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)NCC(=O)N
[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=
C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=
CNC=N2)C(=O)N[C@@]23C(=O)O24OC34)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)
C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@
@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[
C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)O)NC(=O)[C@H
](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCCN
C(=N)N)NC1=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O.CC[C@H](C)[C@H]
(NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C
@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
](CCCCN)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O
)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H]1CCCN1C(=O)[C@@H]1CC
CN1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H
](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(
=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H
](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=
O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC
(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H]
(CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(
N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)
[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O
)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O
)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C
C(C)C)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)
NC(=O)[C@H](CO)NC(=O)[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O
)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)C
(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@H](C=O)CCC(N)=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C
@@H](C)O)C(C)C)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C
@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=
N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CN
C(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C
@H](CC1=CNC=N1)NC(=O)[C@@H]1CCCN1C(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
S)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@
H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)
NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)
[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](
(C)C)C(=O)N[C@@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC2=C1C=
CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)
N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H]
(C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C(

CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@H](C=O)CCC(N)=O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C.O.O.O



)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)
[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@
@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)
O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)
[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)O.CC[C@H](C)[C@H](NC(=O)[C@H](CC(C)C)
NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)N
C(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=
O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](N
C(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](
C)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(N)=O
)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(
=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
(CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC=ON)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=
O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
C(=O)O)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CO)NC(=O)[C@H](CS)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](N
C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=
O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)
O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](N
C(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]
(CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=C
NC2=C1C=CC=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(
=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[
C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)N
C(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
(CCSC)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](
C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=
O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)N
C(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CNC2=
C1C=CC=C2)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[
C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC
(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](
C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)N
C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCC(=O)O)
NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[
C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](
CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC
1=CNC=N1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H
](CCCCN)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC
=C2)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C
CCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@
H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)
)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)
NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)CNC(
=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)CNC(
=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)

NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(
=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C
@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C
@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C
@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[
C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)
=O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC
(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)
[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)
NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@@
H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)
[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[
C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](
NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](N)CCC(N)=O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)
O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C
)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C
(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@
H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@
H](C)CC)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@
H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)
[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@
@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)
C)[C@@H](C)CC)[C@@H](C)O)C(=O)O.Cl.Cl.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.
O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O.O

1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC
(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H
](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C
)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[
C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC
C(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@
H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H
](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O
)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)
[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H
](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C
@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(
=O)[C@@H](N)CC(N)=O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](
C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C
@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C
@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@H](
C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C
(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O
)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)
C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C
(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(N
)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O
)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(
=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C
@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C
@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H]
(C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[
C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](CC1=CNC2=C1C
=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@
@H](C)C(=O)N[C@@H](CS)C(=O)NCC(=O)N1O=C(O)C[C@H]1C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)
C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O
)N[C@H](C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC
N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@
H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H
](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N
[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)
N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(
=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC
CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C
(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CC

)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(
=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@
H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](
C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(
=O)NCC(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C
CC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N
)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)
NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(
N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=
C2)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)
N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=
O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C
@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H
](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC
(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(
=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C
@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)
N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(
N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCC
N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H
](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC
C(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H]
(C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H]
(CS)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@]1(C(=O)N[C@@H](CC(
C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N2CCC[C@H]2
C(=O)N[C@@H](CC2O=C2O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H]2CCCNC(=N)NNC(=O)C
C[C@@H](C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N3CCC[C@H]3C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H]
(CO)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC3C4CN34)C(=O)N[C@@H](CCC
(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@
@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H
](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC3=C
C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)
C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)
C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)
C)C(=O)N[C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCN3C4=NN43)C(=O)N[C@@H](CCCCN)
C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N3CCC[C@H]3C(=O)N[C@@H](
CCCNC3=NN3)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[
C@@H](CC(C)C)C(=O)N3CCC[C@H]3C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CN
C=N3)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C
@@H](CCCCN)C(=O)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C3CCCN3)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)C
C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]
(CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC
=C3)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)
[C@H](C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC
(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC
CCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CO)NC2=O)[C@@H](C)O)CC1CCN)C(C)C)[C@@H](C)O)C(C)C)C(
C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@
@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)C)
C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H]
(NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[
C@H](CC(C)C)NC(=O)[C@@H](N)CCC(=O)O)C(C)C)[C@@H](C)CC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](
C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=C
NC=N1)C(=O)O)[C@@H](C)O.CC[C@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCN1N=C1N)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@@H
]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=CC=C
1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCCNC(=N)N)NC(=
O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=
O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](C
C(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC
(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CS)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N
C(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C
@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H]
(CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC
CNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CO)NC
(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)CNC(=O)[C@H](CO)
NC(=O)[C@@H]1CCCC2NNC2(=O)CC[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H
](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](
CC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]
(NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)[C@
H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=C
NC=N2)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC(N)=O)NC(=
O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O
)[C@H](CO)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]2CCCN2C(=O)[C@H](C
C2=CNC3=C2C=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(N)=O
)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](C
CSC)NC(=O)[C@H](CS)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O
)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C
C(C)C)NC(=O)CNC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CCC(N)=O)NC(=O)CN

C(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN[C@@](CO)(NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]3CCCN3C(=O)[C@@H](NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC3CN3)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)CNC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CS)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CCSC)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)CNC(=O)CNC(=O)CNC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)CNC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC3=CC=C(O)C=C3)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](

CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)N
C(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(
=O)[C@@H](NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC3=CC=C(O)C=C
3)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)
=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H
](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC3=CC=C(
O)C=C3)NC(=O)[C@H](CC3=CC=CC=C3)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CC(C)C)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@
H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC3=CNC=N3)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H]3CCCN3C(=O)[
C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]3CCCN3C(=O)[C@H](CCSC)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)[C@H](CC
CCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC3=
CC=CC=C3)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC3=CNC4=C3C=CC=C4)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]
(CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)
[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(N)=O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C
@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C

=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)

```
](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC
(=O)CNC(=O)CNC(=O)[C@@H]4CCCN4C(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](C
CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](CC4=CC=C(O)C=C4)
NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC4=CC=C(O)C=C4)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)
C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CC4=CNC5=C4C=CC=C5)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC4=CC=CC=C4)
NC(=O)[C@@H](NC(=O)[C@@H]4CSSC[C@H](NC(=O)[C@@H](NC(=O)[C@H](CC5=CNC=N5)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC5=C
C=C(O)C=C5)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=
O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC5=CC=C(O)C=C5)NC(=O)[C@H](CC5=CC=CC=C5)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)
=O)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC5=CC=CC=C5)NC(=O)[C@H](CC5=CNC=N5)NC
(=O)[C@@H]5CCCN5C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H]5CCCN5C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC5=CC=CC
=C5)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC5=CNC6=C5C=CC=C6)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](N
C(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC5=CNC=N5)NC(=O)[C@H](CCC(N)=O)NC(
=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]5CSSC[C@H](NC(=O)[C@@H](NC(=O)[C@H
](CC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@
H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CNC7=C
6C=CC=C7)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](
CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H
](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CS)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC6=CNC=N6)NC(=O
)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@@H]
6CCCN6C(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O
)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C
@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=CC=C6)N
C(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC(N)=O)NC(=O)[
C@@H](NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C
@@H]6CCCN6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)
[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC
(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)
C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC
(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H]6CCCN
6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC6
=CC=C(O)C=C6)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(
=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=
O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7
)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=
C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[
C@H](CCC(N)=O)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](
(CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CNC=N6)NC(=
O)CNC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O
)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=
O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@@H]6O[C@H](CO)[C@@H](O[C@@H]7O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)N6O=C6[C@H](CC(N)=O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CNC=N6)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)N[C@@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)[C@@H]6CCCN6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)C(=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC6=CC=C(O)C=C6)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC6=CNC7=C6C=CC=C7)C(=O)N[C@@H](CC(N)=O)C(=O)N5)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(=O)N[C@@H](CC5=CC=C(O)C=C5)C(=O)N[C@@H](CC5=CC=CC=C5)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N4)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N2)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)CSSC[C@@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)[N+]23CCC[C@H]2C(=O)O3)
NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CNC=N2)N
```

C(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC1=O.CC[C@H](C)[C@@H]1NC(=O)CNC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@H](CCSC)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)CNC(=O)CN

(CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CO)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CNC=N6)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)N[C@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)[C@@H]6CCCN6C(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]6CCCN6C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)CNC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)N[C@@H]6O[C@H](CO)[C@@H](O)[C@H](O)[C@H]6NC(C)O)NC(=O)CNC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC6=CC=CC=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC6=CNC=N6)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC6=CNC7=C6C=CC=C7)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC6=CC=C(O)C=C6)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C

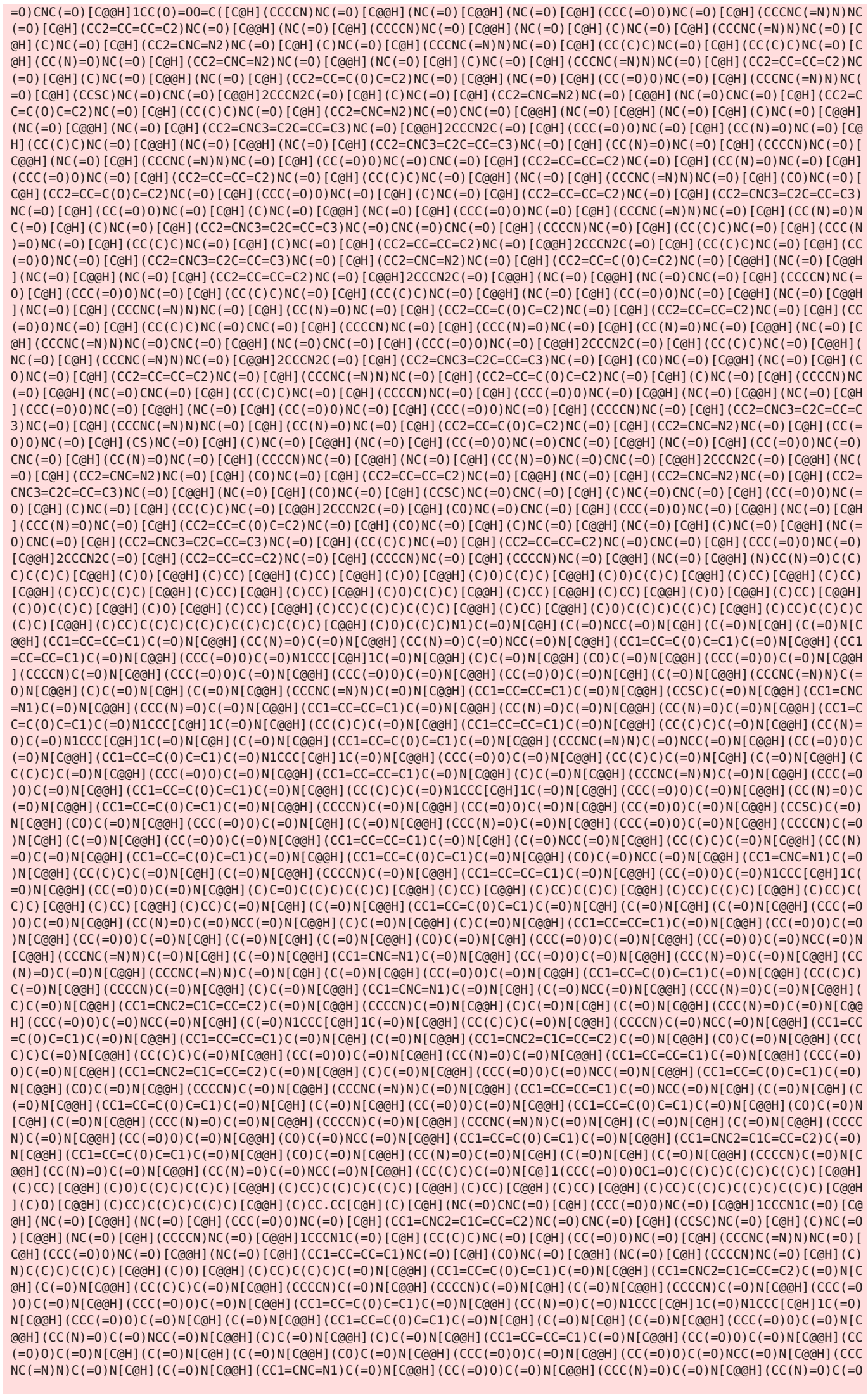
C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)NCC(=O)N[C@H](

CNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](
CS)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C
1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=
O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CNC2=C1C=C
C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](
CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(
=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O
)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N1CCC[C@H]1C
(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=
O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@
H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=
O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](
CCC(=O)O)C(=O)N[C@H]1CCC(=O)N2NC2CCC[C@@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)
C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H
](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC
CN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC
[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC2=ON2
)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H]
(CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC
2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC
CN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O
)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H]
(CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H
](CC2=CC=CC=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](
CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)
N[C@H](C(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H]2C3(C4C5CN54)O=C23N[C@@H]
(CCC2CN2)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O
)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O
)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H]
(CC2=CNC=N2)C(=O)N[C@@H](CC2=CNC=N2)C2=OO2)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H
](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(
C)C)C(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CC(C)C)NC
(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC
CNC(=N)N)NC(=O)[C@H](CS)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H]([C@@H](C)O)NC(=O)[
C@H](CC(C)C)NC(=O)[C@H](CC2=CNC3=C2C=CC=C3)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O
)[C@H](C)NC(=O)[C@H](CCCN2N=C2N)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)C)C)NC(=O)[C@H](CS)NC(=O)[C@H](CCCCN)N
C(=O)[C@H](CO)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H]2CCCN2C(=O)[C@H](C(C)C)NC(=O)
[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC2=CNC=N2)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(N)=O)N
C(=O)[C@H]([C@@H](C)CC)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=
O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)[C
@H](C(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=
O)[C@H](CCSC)NC1=O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O
)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@
H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@
H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O
)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O
)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)
CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C).CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC1=CNC2=C1
C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(
=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CO)
NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN
)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=C
C=CC=C1)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(N)=O
)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O
)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@
@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O
)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC
CCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@@H
](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC
CCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)
O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CC=C(O)C=C1
)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)
[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC
(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](
CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=
O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC
(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[
C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)
[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)
=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)
[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC
(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCC(N)=O)NC(=
O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCSC)NC(=O)CNC
(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[
C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@
H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)
[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CNC2=C1C=C
C=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](
CO)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](
CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)
NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](N)CC(N)=O)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)
[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C
(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](
CCSC)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC
NC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=
CC=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)

N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@H]1CCC(=O)ONCCCC[C@@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCSC)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)NCC(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)

O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC2=OO2)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CS)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C=O)CCCCN)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C23C45CN426NC536=N)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](C(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C(C)C)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H]1CCC(=O)OO=C(O)CC[C@H](N)C(=O)N1)C(C)C)[C@@H](C)CC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCCN)C(=O)O)[C@@H](C)O.Cl.Cl.Cl.Cl.Cl.0.0.0

)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N1CCC[C@H]1C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCSC)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)



)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N[C@@]12C(=O)O1C2CC(=O)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)C(=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](CC(=O)O)C(=O)N1CCC[C@H]1C=O)C(C)C)C(C)C)[C@@H](C)CC.O.O.O

[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O.CC[C@H](C)[C@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)CNC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)CNC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CS)NC(=O)[C@H](CCC(=O)O)NC(=O)CNC(=O)[C@H](CCCCN)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CO)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CO)NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](

@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCSC)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](N)CCCCN)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C1=O[C@@](CO)(C2=OO2)N1)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CS)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CCSC)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CS)NC(=O)[C@@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCSC)NC(=O)CNC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCCCN)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)CC)[C@@H](C)O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CS)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@]12O=C13O1C2013)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C.CO[C@H](C1CCCCC1)C(O)N1CC2NNC(NC(O)C3CCC(N4CCN(C)CC4)CC3)C2C1.CO[C@H](C1CCCCC1)C(O)N1CC2NNC(NC(O)C3CCC(N4CCN(C)CC4)CC3)C2C1.0.0.0

=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H]1CSSC[C@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)CNC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC2=CC=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC2=CC=C(O)C=C2)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)CNC(=O)[C@@H]2CCCN2C(=O)[C@H](C)NC(=O)[C@@H](N)CO)C(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](C(C)C)C(=O)NCC(=O)N2CCC[C@H]2C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H]([C@@H](C)O)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@@H]([C@@H](C)O)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N2CCC[C@H]2C(=O)NCC(=O)N[C@@H](C)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](C)C(=O)N[C@@H](CS)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]([C@@H](C)CC)C(=O)N[C@@H](CCC(=O)O)C(=O)NCC(=O)N2CCC[C@H]2C(=O)N[C@@H](CCCCN)C(=O)NCC(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)NCC(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CO)C(=O)N2CCC[C@H]2C(=O)N[C@@H](C)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N2CCC[C@H]2C(=O)N[C@H](C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CCC(N)=O)C(=O)N2CCC[C@H]2C(=O)N[C@@H](CC(C)

2CNC3C(N)NCNC32)[C@H](O)[C@@H]1O.C[S@@H](CC[C@H](N)C(O)O)C[C@H]1O[C@@H](N2CNC3C(N)NCNC32)[C@H](O)[C@@H]1O.O.O.O.
O.O.O.O.

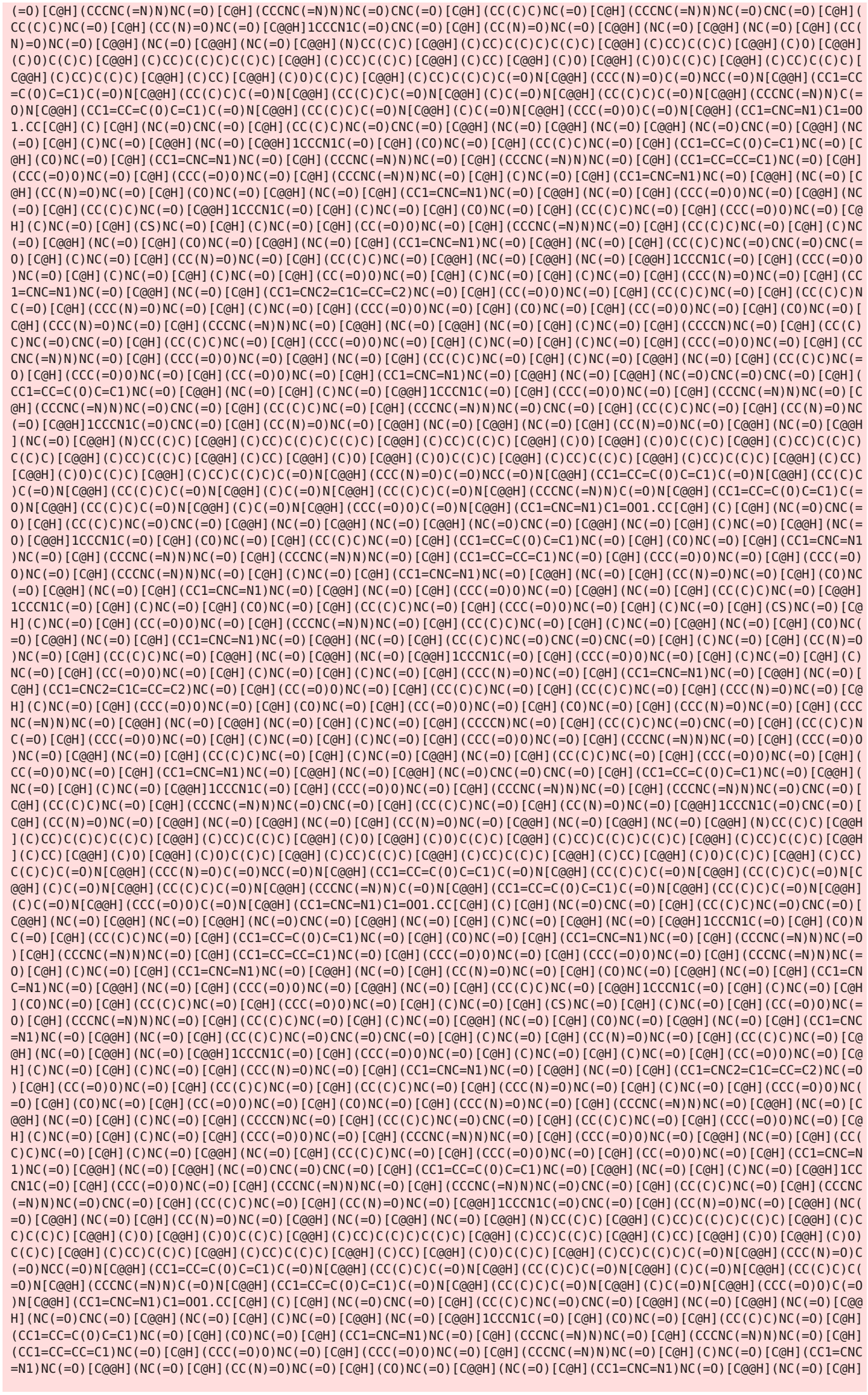
0.0.0.

N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CC
CN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H
](N)CC(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](
C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(
C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H]
(CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(
C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)
C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN
1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](
CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]
(CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC
(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C
@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C
@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(
=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](
CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C
C(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2
=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C
@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O
)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC
C(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H]
(NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O
)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C
)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(N)=O)
NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)
C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H]
(C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[
C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N
[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@
H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[
C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CO)NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=
CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC
(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO
)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)
NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)
N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)
O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC
(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](
CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=
O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO
)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)
O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)
NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)



NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CC=CC=C1)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](C)NC(=O)[C@@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](CC(N)=O)NC(=O)[C@@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)C(=O)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCC(N)=O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](C)NC(=O)[C@@H](CCCCN)NC(=O)[C@@H](CC(C)C)NC(=O)CNC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H](CC(N)=O)NC(=O)[C@@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@@H](CC(N)=O)NC(=O)[C@@H](CO)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](C)NC(=O)[C@@H](CO)NC(=O)[C@@H](CC(C)C)NC(=O)[C@@H](CCC(=O)O)NC(=O)[C@@H](C)NC(=O)[C@@H](CS)NC(=O)[C@@H](C)NC(=O)[C@@H](CC(=O)O)NC(=O)[C@@H](CCCNC(=N)N)NC(=O)[C@@H](CC1=CNC=N1)C(=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](CC1=CC=C(O)C=C1

```
](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O
)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](
C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)C
NC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(
=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H
](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(
=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](C
C(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H
](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)
NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C
@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(
C)C)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC
(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)O.CC[C@H](C)[C@H](NC(=O)C
)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)O.CC[C@H](C)[C@H](NC(=O)C
NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](
NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CN
C=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC
(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C
O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C
@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O
)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO
)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(
N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H
](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(
=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O
)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H]
(CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C
)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(
=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C
@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@
@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(
=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(
=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[
C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(=O)N[C@@H](CCC(N)=O)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CNC=N1)C1=OO1.CC[C@H](C)[C@H](NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[
C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[
C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](C)NC(=O)[C@H](CC1
=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[
C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)N
C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)
[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C
@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C
@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)
[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H
](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](C
C(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)
[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)
[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[
C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@
@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC
```



(CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CS)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)CNC(=O)[C@H](C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)CNC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](N)CC(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)CC

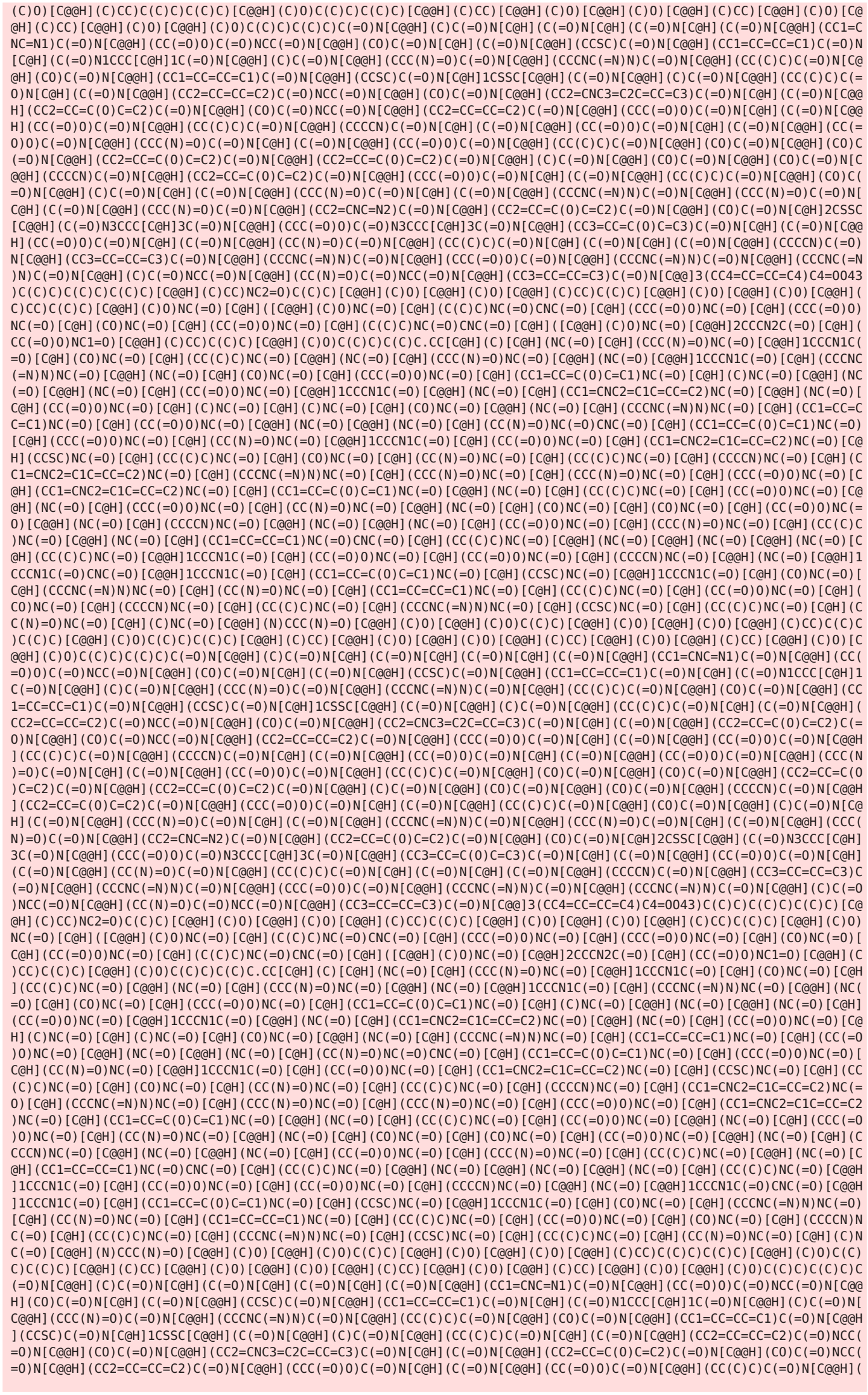
.0.0.0

CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@]34CO3C4=O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](C)O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2

](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](N)CC(=O)O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO

C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN

C(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(=O)N[C@@H](C)C(=O)N[C@H]([C@@H](C)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](C)C1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)CNC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H]




CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N
[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2
=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O
)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)
=O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=
CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O
)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)
C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N
)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC(N)=
O)C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[
C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O
)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@
H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C
(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](
NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H
](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1
CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(
=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](NC(=O)
[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@
@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC
(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(
=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(
O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O
)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=
O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O
)CNC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=
O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=
CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C
@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=
O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O
)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C
@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C
@H](C(=O)N[C@H](C(=O)N[C@@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@H](C(=O
)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[
C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSS
C[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C
@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C
2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O
)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[
C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@
@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O
)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[
C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC
2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)
N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(
=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O
)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](
CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)NC2=O)C(C)C)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(
=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)
[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[
C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)N
C(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H]
(CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O
)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@
H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(
N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC
(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=
O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=
O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC
(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H](NC(=O)[C
@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@
@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)
NC(=O)[C@@H](NC(=O)[C@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@H]1CS
SC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C
@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C
2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O
)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[
C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@
@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O
)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[
C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C
@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C
3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)NC(=O)N[C@H](C(=O)N[C@H](

C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)NCC(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@]3(CC4=CC=CC=C4)C4=O043)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.CC[C@H](C)[C@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@H](CCC(N)=O)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCCNC(=N)N)NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H]1CCCN1C(=O)[C@@H](NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C)NC(=O)[C@H](C)NC(=O)[C@H](CO)NC(=O)[C@@H](NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)CNC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC(N)=O)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CO)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CO)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCC(N)=O)NC(=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC(=O)O)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H]1CCCN1C(=O)CNC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CC=C(O)C=C1)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CO)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CCCNC(=N)N)NC(=O)[C@H](CCSC)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](C)NC(=O)[C@@H](N)CCC(N)=O)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)C(C)C)C(C)C)[C@@H](C)O)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC(=O)O)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N1CCC[C@H]1C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CCSC)C(=O)N[C@H]1CSSC[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=CC=C2)C(=O)NCC(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CNC3=C2C=CC=C3)C(=O)N[C@H](C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)NCC(=O)N[C@@H](CC2=CC=CC=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@H](C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC2=CNC=N2)C(=O)N[C@@H](CC2=CC=C(O)C=C2)C(=O)N[C@@H](CO)C(=O)N[C@H]2CSSC[C@@H](C(=O)N3CCC[C@H]3C(=O)N[C@@H](CCC(=O)O)C(=O)N3CCC[C@H]3C(=O)N[C@@H](CC3=CC=C(O)C=C3)C(=O)N[C@H](C(=O)N[C@@H](CC(=O)O)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC3=CC=CC=C3)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](C)CC)NC2=O)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)C(C)C)[C@@H](C)O)NC(=O)[C@H]([C@@H](C)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CO)NC(=O)[C@H](CC(=O)O)NC(=O)[C@H](C(C)C)NC(=O)CNC(=O)[C@H]([C@@H](C)O)NC(=O)[C@@H]2CCCN2C(=O)[C@H](CC(=O)O)NC1=O)[C@@H](C)CC)C(C)C)[C@@H](C)O)C(C)C)C(C)C.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.NCCC1CNC2CCC(O)CC12.0.0.0

.0.0.0

```
CCC(N)=O)NC(=O)[C@H](CC1=CNC=N1)NC(=O)[C@@H](NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCSC)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CCC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC(N)=O)NC(=O)[C@H](CC(C)C)NC(=O)[C@H](CC(=O)O)NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CCC(N)=O)NC(=O)[C@H](CCC(=O)O)NC(=O)[C@H](CCCCN)NC(=O)[C@@H](NC(=O)[C@@H](NC(=O)[C@H](CC1=CC=CC=C1)NC(=O)[C@H](CC(C)C)NC(=O)[C@@H]1CCCN1C(=O)[C@H](CC1=CNC2=C1C=CC=C2)NC(=O)[C@H](CCCCN)NC(=O)[C@H](CO)NC(=O)[C@@H](N)CCC(=O)O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(N)=O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](C)C(=O)N[C@]1([C@@H](C)O)OC1=O)C(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)NCC(=O)N[C@@H](CC(=O)N[C@@H]1O[C@H](CO)[C@@H](O[C@H]2O[C@H](CO)[C@@H](O)[C@H](O)[C@H]2NC(C)O)[C@H](O)[C@H]1NC(C)O)C(=O)NCC(=O)N[C@H](C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC1C2=OO21)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](CCSC)C(=O)NCC(=O)N[C@H](C(=O)N[C@H](C(=O)N[C@@H](CC1=CNC=N1)C(=O)NCC(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)NCC(=O)N[C@@H](CC(C)C)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(C)C)C(=O)N[C@H](C(=O)N[C@@H](CCCCN)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](C(=O)N[C@@H](C)C(=O)N[C@@H](C)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](CCC(=O)O)C(=O)N[C@@H](C)C(=O)N[C@@H](CC(C)C)C(=O)N[C@@H](CO)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@@H](CCCNC(=N)N)C(=O)N[C@H](C(=O)N[C@@H](CCSC)C(=O)N[C@@H](CC1=CNC=N1)C(=O)N[C@@H](CC1=CC=C(O)C=C1)C(=O)N[C@@H](CC1=CNC2=C1C=CC=C2)C(=O)N[C@@H](C)C(=O)N[C@H](C(=O)N[C@@H](CC1=CC=CC=C1)C(=O)N[C@@H](C)C(=O)N[C@@H](CCCCN)C(=O)N[C@H](C(=O)NCC(=O)N[C@@H](CC(N)=O)C(=O)N1CCC[C@H]1C(=O)N[C@@H](CC(N)=O)C(=O)N[C@@H](CCC(=O)O)C(=O)N1CCC[C@H]1C=O)[C@@H](C)O)[C@@H](C)O)[C@@H](C)CC)[C@@H](C)O)C(C)C)C(C)C)C(C)C)[C@@H](C)CC)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)O)[C@@H](C)O)C(C)C)[C@@H](C)CC)C(C)C)C(C)C)[C@@H](C)CC.0.0.0
```

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Modeling Protein-Ligand Interactions with Atomic Convolutions

By Nathan C. Frey | Twitter and Bharath Ramsundar | Twitter

This DeepChem tutorial introduces the Atomic Convolutional Neural Network. We'll see the structure of the `AtomicConvModel` and write a simple program to run Atomic Convolutions.

ACNN Architecture

ACNN's directly exploit the local three-dimensional structure of molecules to hierarchically learn more complex chemical features by optimizing both the model and featurization simultaneously in an end-to-end fashion.

The atom type convolution makes use of a neighbor-listed distance matrix to extract features encoding local chemical environments from an input representation (Cartesian atomic coordinates) that does not necessarily contain spatial locality. The following methods are used to build the ACNN architecture:

- **Distance Matrix**
  The distance matrix R is constructed from the Cartesian atomic coordinates X. It calculates distances from the distance tensor D. The distance matrix construction accepts as input a $(N, 3)$ coordinate matrix C. This matrix is "neighbor listed" into a (N, M) matrix R.

```
R = tf.reduce_sum(tf.multiply(D, D), 3)     # D: Distance Tensor
    R = tf.sqrt(R)                              # R: Distance Matrix
    return R
```

- **Atom type convolution**
  The output of the atom type convolution is constructed from the distance matrix R and atomic number matrix Z. The matrix R is fed into a (1x1) filter with stride 1 and depth of N_{at}, where N_{at} is the number of unique atomic numbers (atom types) present in the molecular system. The atom type convolution kernel is a step function that operates on the neighbor distance matrix R.

- **Radial Pooling layer**
  Radial Pooling is basically a dimensionality reduction process that down-samples the output of the atom type convolutions. The reduction process prevents overfitting by providing an abstracted form of representation through feature binning, as well as reducing the number of parameters learned. Mathematically, radial pooling layers pool over tensor slices (receptive fields) of size (1x M x1) with stride 1 and a depth of N_r, where N_r is the number of desired radial filters and M is the maximum number of neighbors.

- **Atomistic fully connected network**
  Atomic Convolution layers are stacked by feeding the flattened (

N
,
N_{at}
,
N_r
) output of the radial pooling layer into the atom type convolution operation. Finally, we feed the tensor row-wise (per-atom) into a fully-connected network. The same fully connected weights and biases are used for each atom in a given molecule.

Now that we have seen the structural overview of ACNNs, we'll try to get deeper into the model and see how we can train it and what we expect as the output.

For the training, we will use the publicly available PDBbind dataset. In this example, every row reflects a protein-ligand complex and the target is the binding affinity (K_i) of the ligand to the protein in the complex.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5 minutes to run to completion and install your environment.

In [ ]:
```
!pip install -q condacolab
import condacolab
condacolab.install()
!/usr/local/bin/conda info -e
```

In [ ]:
```
!/usr/local/bin/conda install -c conda-forge pycosat mdtraj pdbfixer openmm -y -q  # needed for AtomicConvs
```

In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

In [ ]:
```
import deepchem as dc
import os

import numpy as np
import tensorflow as tf

import matplotlib.pyplot as plt

from rdkit import Chem

from deepchem.molnet import load_pdbbind
from deepchem.models import AtomicConvModel
from deepchem.feat import AtomicConvFeaturizer
```

Getting protein-ligand data

If you worked through Tutorial 13 on modeling protein-ligand interactions, you'll already be familiar with how to obtain a set of data from PDBbind for training our model. Since we explored molecular complexes in detail in the previous tutorial), this time we'll simply initialize an `AtomicConvFeaturizer` and load the PDBbind dataset directly using MolNet.

In [ ]:
```
f1_num_atoms = 100  # maximum number of atoms to consider in the ligand
f2_num_atoms = 1000  # maximum number of atoms to consider in the protein
max_num_neighbors = 12  # maximum number of spatial neighbors for an atom

acf = AtomicConvFeaturizer(frag1_num_atoms=f1_num_atoms,
                      frag2_num_atoms=f2_num_atoms,
                      complex_num_atoms=f1_num_atoms+f2_num_atoms,
                      max_num_neighbors=max_num_neighbors,
                      neighbor_cutoff=4)
```

`load_pdbbind` allows us to specify if we want to use the entire protein or only the binding pocket (`pocket=True`) for featurization. Using only the pocket saves memory and speeds up the featurization. We can also use the "core" dataset

of ~200 high-quality complexes for rapidly testing our model, or the larger "refined" set of nearly 5000 complexes for more datapoints and more robust training/validation. On Colab, it takes only a minute to featurize the core PDBbind set! This is pretty incredible, and it means you can quickly experiment with different featurizations and model architectures.

In [ ]:
```
%%time
tasks, datasets, transformers = load_pdbbind(featurizer=acf,
                                             save_dir='.',
                                             data_dir='.',
                                             pocket=True,
                                             reload=False,
                                             set_name='core')
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
CPU times: user 43.2 s, sys: 18.6 s, total: 1min 1s
Wall time: 1min 10s
```

Unfortunately, if you try to use the "refined" dataset, there are some complexes that cannot be featurized. To resolve this issue, rather than increasing `complex_num_atoms`, simply omit the lines of the dataset that have an `x` value of `None`

In [ ]:
```
class MyTransformer(dc.trans.Transformer):
  def transform_array(x, y, w, ids):
    kept_rows = x != None
    return x[kept_rows], y[kept_rows], w[kept_rows], ids[kept_rows],

datasets = [d.transform(MyTransformer) for d in datasets]
```

In [ ]:
```
datasets
```

Out[ ]:
```
(<DiskDataset X.shape: (154, 9), y.shape: (154,), w.shape: (154,), ids: ['1mq6' '3pe2' '2wtv' ... '3f3c' '4gqq'
'2x00'], task_names: [0]>,
 <DiskDataset X.shape: (19, 9), y.shape: (19,), w.shape: (19,), ids: ['3ivg' '4de1' '4tmn' ... '2vw5' '1w3l' '2
zjw'], task_names: [0]>,
 <DiskDataset X.shape: (20, 9), y.shape: (20,), w.shape: (20,), ids: ['1kel' '2w66' '2xnb' ... '2qbp' '3lka' '1
qi0'], task_names: [0]>)
```

In [ ]:
```
train, val, test = datasets
```

Training the model

Now that we've got our dataset, let's go ahead and initialize an `AtomicConvModel` to train. Keep the input parameters the same as those used in `AtomicConvFeaturizer`, or else we'll get errors. `layer_sizes` controls the number of layers and the size of each dense layer in the network. We choose these hyperparameters to be the same as those used in the original paper.

In [ ]:
```
acm = AtomicConvModel(n_tasks=1,
                      frag1_num_atoms=f1_num_atoms,
                      frag2_num_atoms=f2_num_atoms,
                      complex_num_atoms=f1_num_atoms+f2_num_atoms,
                      max_num_neighbors=max_num_neighbors,
                      batch_size=12,
                      layer_sizes=[32, 32, 16],
                      learning_rate=0.003,
                      )
```

In [ ]:
```
losses, val_losses = [], []
```

In [ ]:
```
%%time
max_epochs = 50

metric = dc.metrics.Metric(dc.metrics.score_function.rms_score)
step_cutoff = len(train)//12
def val_cb(model, step):
  if step%step_cutoff!=0:
      return
  val_losses.append(model.evaluate(val, metrics=[metric])['rms_score']**2)  # L2 Loss
  losses.append(model.evaluate(train, metrics=[metric])['rms_score']**2)  # L2 Loss

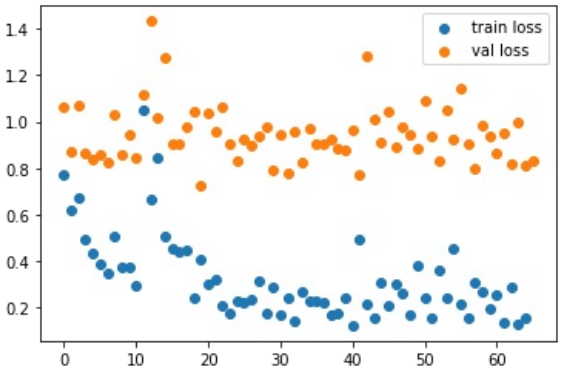
acm.fit(train, nb_epoch=max_epochs, max_checkpoints_to_keep=1,
                callbacks=[val_cb])
```

```
CPU times: user 2min 41s, sys: 11.4 s, total: 2min 53s
Wall time: 2min 47s
```

The loss curves are not exactly smooth, which is unsurprising because we are using 154 training and 19 validation

datapoints. Increasing the dataset size may help with this, but will also require greater computational resources.

In [ ]:
```
f, ax = plt.subplots()
ax.scatter(range(len(losses)), losses, label='train loss')
ax.scatter(range(len(val_losses)), val_losses, label='val loss')
plt.legend(loc='upper right');
```



The ACNN paper showed a Pearson
R^2
score of 0.912 and 0.448 for a random 80/20 split of the PDBbind core train/test sets. Here, we've used an 80/10/10 training/validation/test split and achieved similar performance for the training set (0.943). We can see from the performance on the training, validation, and test sets (and from the results in the paper) that the ACNN can learn chemical interactions from small training datasets, but struggles to generalize. Still, it is pretty amazing that we can train an `AtomicConvModel` with only a few lines of code and start predicting binding affinities!
From here, you can experiment with different hyperparameters, more challenging splits, and the "refined" set of PDBbind to see if you can reduce overfitting and come up with a more robust model.

In [ ]:
```
score = dc.metrics.Metric(dc.metrics.score_function.pearson_r2_score)
for tvt, ds in zip(['train', 'val', 'test'], datasets):
  print(tvt, acm.evaluate(ds, metrics=[score]))
```

```
train {'pearson_r2_score': 0.9311347622675604
val {'pearson_r2_score': 0.5162870575992874}
test {'pearson_r2_score': 0.4756633065901693}
```

Further reading

We have explored the ACNN architecture and used the PDBbind dataset to train an ACNN to predict protein-ligand binding energies. For more information, read the original paper that introduced ACNNs: Gomes, Joseph, et al. "Atomic convolutional networks for predicting protein-ligand binding affinity." arXiv preprint arXiv:1703.10603 (2017). There are many other methods and papers on predicting binding affinities. Here are a few interesting ones to check out: predictions using only ligands or proteins, molecular docking with deep learning, and AtomNet.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Applications of DeepChem with Alphafold: Docking and protein-ligand interaction from protein sequence

Author: Sriphani Vardhan Bellamkonda

In this tutorial, we explore a potential use case where we combine the capabilities of AlphaFold and DeepChem.

Alphafold2 has made immense strides in predicting protein structure folding without the use of costly lab equipment and DeepChem comprises a repertoire of easy to use modules which can then be applied on these protein structures for further analysis. In the first part of our tutorial we will predict the protein structure when given a protein sequence. Then, in the second part of our tutorial, we sample a few ligands from the protein-complex dataset(PDBbind) and perform programmatic docking to estimate binding affinities between our protein and a number of ligands.

This tutorial is meant to be run in google colab. You can follow the below link to open this notebook in colab.



Setup

We start off with all the installations and configurations. If you would like to directly skip reading this part, you can head over to the Input Query Section below.

We will first be installing deepchem as a runtime restart might be required. Along with that lets also install condacolab, vina and pdbfixer which will be used in the later parts of this tutorial.

In [ ]:
```
!pip install -q condacolab
import condacolab
condacolab.install()
!conda install -c conda-forge openmm -y
!mkdir vina_test

!conda install -c conda-forge pdbfixer
!conda install -c conda-forge vina
!pip install deepchem
!pip install -q mdtraj nglview
```

Restart may be prompted

Part 1: Predict Protein Structure in pdb format from a given input sequence

Note: The cells until part 2 of this tutorial are directly from the Colabfold's google colab implementation and have been further annotated in this tutorial.

Alphafold Colabfold Openfold .....so many folds!

In 2020, a major breakthrough in protein structure prediction occurred when AlphaFold2, a newer version of AlphaFold, vastly outperformed other models in the CASP (Critical Assessment of Protein Structure Prediction) competition. After this breakthrough event, several organizations worked ahead of AlphaFold2 and provided slightly different solutions for the protein folding problem. For example, ColabFold provides faster protein structure modeling because it uses a separate server for the sequence alignment step of AlphaFold2, and OpenFold provides a few enhancements, such as faster inference and custom CUDA attention kernels, while being implemented in PyTorch. Apart from these, there are other protein folding models, such as AlphaFold-multimer, OmegaFold, and RosettaFold, designed for different scenarios. Due to its simplicity, we will be using Colabfold in our tutorial.



ColabFold v1.5.3-patch: AlphaFold2 using MMseqs2

ColabFold is an easy to use protien structure and complex prediction model which uses Alphafold2 and Alphafold2-multimer for sequence to structure prediction of single and multiple protein chains. It also abstracts out the Sequence Alignment/Template Generation step or MSA, Multiple Sequence Alignemnt, through MMSeqs2 and HHsearch. MSA step is explained in more detail in the cells below.

For more details, checkout the ColabFold GitHub and read the below manuscript.

Mirdita M, Schütze K, Moriwaki Y, Heo L, Ovchinnikov S, Steinegger M. ColabFold: Making protein folding accessible to all. *Nature Methods*, 2022

We then create a job-name folder and store all the sequences as queries in a text file which we will reference later to call model inference on. The way that Alphafold is structured is that it interacts with a job directory to process the inputs, templates, and save the outpus. Hence, it is important to create a unique job folder.

In [ ]:
```
#@title Input protein sequence(s), then hit `Runtime` -> `Run all`
from google.colab import files
import os
import re
import hashlib
import random

from sys import version_info
python_version = f"{version_info.major}.{version_info.minor}"

# query_sequence = 'PIAQIHILEGRSDEQKETLIREVSEAISRSLDAPLTSVRVIITEMAKGHFGIGGELASK' #@param {type:"string"}
query_sequence='#'
#@markdown  - Use `:` to specify inter-protein chainbreaks for **modeling complexes** (supports homo- and hetro
jobname = 'test' #@param {type:"string"}
# number of models to use
num_relax = 0 #@param [0, 1, 5] {type:"raw"}
#@markdown - specify how many of the top ranked structures to relax using amber

use_amber = num_relax > 0
```

Use the below parameter to set template configs.

AlphaFold uses templates from a vast protein structure database to guide its predictions. It aligns the target protein's sequence with similar sequences from the database to generate structural constraints, aiding in the accurate prediction of the protein's 3D structure.

`none` = no template information is used. In this case the sequence alignment step is skipped and prediction is made without it.

`pdb100` = templates are detected from the pdb100 database. pdb100 is a subset of the Protein Data Bank (PDB) containing 100 diverse protein structures used for training and validation in structural biology and bioinformatics research.

`custom` = the user has the option to upload their own templates database to search on (PDB or mmCIF format)

In [ ]:
```
template_mode = "none" #@param ["none", "pdb100","custom"]
```

Lets specify a hashing function which we will use to create a job name. Hashing is mainly used to reduce the length of the folder name in order to have reasonable and unique jobname everytime we run the notebook.

In [ ]:
```
def add_hash(x,y):
  return x+"_"+hashlib.sha1(y.encode()).hexdigest()[:5]

# remove whitespaces
query_sequence = "".join(query_sequence.split())

basejobname = "".join(jobname.split())
basejobname = re.sub(r'\W+', '', basejobname)
jobname = add_hash(basejobname, query_sequence)
```

We then create the directory where we will store our protein seqeunce (query) and respective files which will be generated in the later parts of this tutorial. We also define a check function below which prevents us from creating duplicate directories.

In [ ]:
```
# check if directory with jobname exists
def check(folder):
  if os.path.exists(folder):
    return False
  else:
    return True
if not check(jobname):
  n = 0
  while not check(f"{jobname}_{n}"): n += 1
  jobname = f"{jobname}_{n}"

# make directory to save results
os.makedirs(jobname, exist_ok=True)
```

Based on the template_mode specified earlier, we automatically adjust a few parameters such as use_templates and custom_template_path. If the template_mode is custom, we will create an additional directory for it.

In [ ]:
```
if template_mode == "pdb100":
```

```
  use_templates = True
  custom_template_path = None
elif template_mode == "custom":
  custom_template_path = os.path.join(jobname,f"template")
  os.makedirs(custom_template_path, exist_ok=True)
  uploaded = files.upload()
  use_templates = True
  for fn in uploaded.keys():
    os.rename(fn,os.path.join(custom_template_path,fn))
else:
  custom_template_path = None
  use_templates = False
```

Install dependencies

Based on the parameters mentioned above we will respectively install the dependencies with the code below.

1. First we have to install the latest version of ColabFold from their github repo. After a successful installation, a file called COLABFOLD_READY will be created to mark its completion.

In [ ]:
```
%%time
import os
USE_AMBER = use_amber
USE_TEMPLATES = use_templates
PYTHON_VERSION = python_version

if not os.path.isfile("COLABFOLD_READY"):
  print("installing colabfold...")
  os.system("pip install -q --no-warn-conflicts 'colabfold[alphafold-minus-jax] @ git+https://github.com/sokryp
  os.system("pip install --upgrade dm-haiku")
  os.system("ln -s /usr/local/lib/python3.*/dist-packages/colabfold colabfold")
  os.system("ln -s /usr/local/lib/python3.*/dist-packages/alphafold alphafold")
  # patch for jax > 0.3.25
  os.system("sed -i 's/weights = jax.nn.softmax(logits)/logits=jnp.clip(logits,-1e8,1e8);weights=jax.nn.softmax
  os.system("touch COLABFOLD_READY")
```

```
CPU times: user 0 ns, sys: 35 µs, total: 35 µs
Wall time: 40.1 µs
```

2. Next, if we need amber relaxation or protein templates, we will install mamba which will help us in installing further packages

In [ ]:
```
if USE_AMBER or USE_TEMPLATES:
  if not os.path.isfile("CONDA_READY"):
    print("installing conda...")
    os.system("wget -qnc https://github.com/conda-forge/miniforge/releases/latest/download/Mambaforge-Linux-x86_
    os.system("bash Mambaforge-Linux-x86_64.sh -bfp /usr/local")
    os.system("mamba config --set auto_update_conda false")
    os.system("touch CONDA_READY")
```

3. Then, we will install hhsuite for db-search/tempate-retireval and openmm for amber relaxation.

In [ ]:
```
if USE_TEMPLATES and not os.path.isfile("HH_READY") and USE_AMBER and not os.path.isfile("AMBER_READY"):
  print("installing hhsuite and amber...")
  os.system(f"mamba install -y -c conda-forge -c bioconda kalign2=2.04 hhsuite=3.3.0 openmm=7.7.0 python='{PYTH
  os.system("touch HH_READY")
  os.system("touch AMBER_READY")
else:
  if USE_TEMPLATES and not os.path.isfile("HH_READY"):
    print("installing hhsuite...")
    os.system(f"mamba install -y -c conda-forge -c bioconda kalign2=2.04 hhsuite=3.3.0 python='{PYTHON_VERSION}
    os.system("touch HH_READY")
  if USE_AMBER and not os.path.isfile("AMBER_READY"):
    print("installing amber...")
    os.system(f"mamba install -y -c conda-forge openmm=7.7.0 python='{PYTHON_VERSION}' pdbfixer")
    os.system("touch AMBER_READY")
```

A little about the softwares we are installing:

HHsuite is a widely used open source software suite for sensitive sequence similarity searches and protein fold recognition.

Openmm is a high-performance toolkit for molecular simulation. In our tutorial this tooklkit helps us simulate amber force field which is used to "relax" the position of atoms with respect to each other in order to remove any definite obstructions between them. This helps us to better model protein folding in edge cases and offers better refinement of the protein structures.

Now lets specifiy the different msa options available.

MSA

Multi Sequence Alignment is a process in Alphafold which aligns multiple sequences of amino acids from different sources which have a similar sequence. In this step, Alphafold2 forms a grid aligning identical amino acids in the same columns and leaving gaps where there are differences. We have the option of how we would like to pair the respective MSA with the options: "unpaired_paired" to pair sequences from same species and an unpaired MSA, "unpaired" to seperate MSA for each chain, and "paired" to only use paired sequences. Additionally, we have multiple options in the way Alphafold searches for the respective sequences and few of them are mentioned below.

MMSEQ2: This is a sequence searching tool which finds similar sequences to our input sequence from a large database.

Single_sequence: This option restricts Alphafold from searching for any similar amino acid sequences and restricts it utilize the only given one.

Custom: This options lets Alphafold do the sequence search from user defined sequence search space.

In [ ]:
```
#@markdown ### MSA options (custom MSA upload, single sequence, pairing mode)
msa_mode = "mmseqs2_uniref_env" #@param ["mmseqs2_uniref_env", "mmseqs2_uniref","single_sequence","custom"]
pair_mode = "unpaired_paired" #@param ["unpaired_paired","paired","unpaired"] {type:"string"}
#@markdown - "unpaired_paired" = pair sequences from same species + unpaired MSA, "unpaired" = seperate MSA for
```

So based on the above msa parameters we will set the path to the A3M file. An A3M file (Alignment to Multiple Models) is a type of input file used in the protein structure prediction process which contains multiple sequence alignments (MSAs) of related protein sequences that are used as input data for the AlphaFold model. A3M file format is an extentions to FASTA file format and can be read about over at FASTA format extension.

Additionally, for the purpose of this tutorial we don't need to get into the details of custom MSA(where user inputs their own template databse for search).

In [ ]:
```
# decide which a3m to use
if "mmseqs2" in msa_mode:
  a3m_file = os.path.join(jobname,f"{jobname}.a3m")
```

Advanced settings

Below we can specify more advanced settings with Alphafold. We can choose which model parameters to use from the options given below i.e alphafold2, alphafold2_multimer_v1 etc.., recycle early stop tolerence, saving to google drive, and image resolution options. Also note that there is no need to fully understand the parameters and just stick to the defaults. But here are a few details about the parameters which can be changed.

model_type: If auto selected, will use alphafold2_ptm for monomer prediction and alphafold2_multimer_v3 for complex prediction. Any of the mode_types can be used (regardless if input is monomer or complex).

num_recycles: "auto" with other options available as ["auto", "0", "1", "3", "6", "12", "24", "48"]

recycle_early_stop_tolerance: "auto" with other options available as ["auto", "0.0", "0.5", "1.0"]

By "recycling" Alphafold refers to the process of iterative refinement of protein structure prediction to imporve accuracy.

Pairing strategy: "greedy" or "complete"

1. `greedy` = pair any taxonomically matching subsets
2. `complete` = all sequences have to match in one line.

In [ ]:
```
#@markdown ### Advanced settings
model_type = "auto" #@param ["auto", "alphafold2_ptm", "alphafold2_multimer_v1", "alphafold2_multimer_v2", "alp
#@markdown - if `auto` selected, will use `alphafold2_ptm` for monomer prediction and `alphafold2_multimer_v3`
#@markdown Any of the mode_types can be used (regardless if input is monomer or complex).
num_recycles = "3" #@param ["auto", "0", "1", "3", "6", "12", "24", "48"]
#@markdown - if `auto` selected, will use `num_recycles=20` if `model_type=alphafold2_multimer_v3`, else `num_r
recycle_early_stop_tolerance = "auto" #@param ["auto", "0.0", "0.5", "1.0"]
#@markdown - if `auto` selected, will use `tol=0.5` if `model_type=alphafold2_multimer_v3` else `tol=0.0`.
relax_max_iterations = 200 #@param [0, 200, 2000] {type:"raw"}
#@markdown - max amber relax iterations, `0` = unlimited (AlphaFold2 default, can take very long)
pairing_strategy = "greedy" #@param ["greedy", "complete"] {type:"string"}
#@markdown - `greedy` = pair any taxonomically matching subsets, `complete` = all sequences have to match in on
```

We can also set the maximum length of Multiple Sequence Alignment, Number of Seeds, and whether to use dropout or not.

max_msa = "auto" with other optinos as ["auto", "512:1024", "256:512", "64:128", "32:64", "16:32"]. Here left is the minimum and right is the maximum msa length.

num_seeds = 1 with other options as [1,2,4,8,16] {type:"raw"}

use_dropout = False other option True

In [ ]:
```
#@markdown #### Sample settings
#@markdown -  enable dropouts and increase number of seeds to sample predictions from uncertainty of the model.
#@markdown -  decrease `max_msa` to increase uncertainity
max_msa = "auto" #@param ["auto", "512:1024", "256:512", "64:128", "32:64", "16:32"]
num_seeds = 1 #@param [1,2,4,8,16] {type:"raw"}
use_dropout = False #@param {type:"boolean"}

num_recycles = None if num_recycles == "auto" else int(num_recycles)
recycle_early_stop_tolerance = None if recycle_early_stop_tolerance == "auto" else float(recycle_early_stop_tol
if max_msa == "auto": max_msa = None
```

Save to Google Drive

We define a simple save_to_google_drive function which authenticates into our respective google drive and allows to save our files.

In [ ]:
```
#@markdown #### Save settings
save_all = False #@param {type:"boolean"}
save_recycles = False #@param {type:"boolean"}
save_to_google_drive = False #@param {type:"boolean"}
#@markdown -  if the save_to_google_drive option was selected, the result zip will be uploaded to your Google D
dpi = 200 #@param {type:"integer"}
#@markdown - set dpi for image resolution

if save_to_google_drive:
  from pydrive.drive import GoogleDrive
  from pydrive.auth import GoogleAuth
  from google.colab import auth
  from oauth2client.client import GoogleCredentials
  auth.authenticate_user()
  gauth = GoogleAuth()
  gauth.credentials = GoogleCredentials.get_application_default()
  drive = GoogleDrive(gauth)
  print("You are logged into Google Drive and are good to go!")
```

Now we will run the prediction model using all the inputs and specifications from above.

We import the respective files from Colabfold's package for inference and plotting, and we then check if we have a specific GPU. We also define 2 helper functions: input_features_callback, and prediction_callback, which help us to visualize the respective input featues and prediction results.

Lets import utility, batch and plotting functions from colabfold!

In [ ]:
```
#@title Run Prediction
display_images = True #@param {type:"boolean"}

import sys
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from Bio import BiopythonDeprecationWarning
warnings.simplefilter(action='ignore', category=BiopythonDeprecationWarning)
from pathlib import Path
from colabfold.download import download_alphafold_params, default_data_dir
from colabfold.utils import setup_logging
from colabfold.batch import get_queries, run, set_model_type
from colabfold.plot import plot_msa_v2

import os
import numpy as np
```

Next we check if K80 GPU is available.

In [ ]:
```
try:
  K80_chk = os.popen('nvidia-smi | grep "Tesla K80" | wc -l').read()
except:
  K80_chk = "0"
  pass
if "1" in K80_chk:
  print("WARNING: found GPU Tesla K80: limited to total length < 1000")
  if "TF_FORCE_UNIFIED_MEMORY" in os.environ:
    del os.environ["TF_FORCE_UNIFIED_MEMORY"]
```

```
  if "XLA_PYTHON_CLIENT_MEM_FRACTION" in os.environ:
    del os.environ["XLA_PYTHON_CLIENT_MEM_FRACTION"]
```

Now, let's define some helper functions to visualize input features and output preduction.

In [ ]:
```
from colabfold.colabfold import plot_protein
from pathlib import Path
import matplotlib.pyplot as plt

# For some reason we need that to get pdbfixer to import
if use_amber and f"/usr/local/lib/python{python_version}/site-packages/" not in sys.path:
    sys.path.insert(0, f"/usr/local/lib/python{python_version}/site-packages/")

def input_features_callback(input_features):
  if display_images:
    plot_msa_v2(input_features)
    plt.show()
    plt.close()

def prediction_callback(protein_obj, length,
                        prediction_result, input_features, mode):
  model_name, relaxed = mode
  if not relaxed:
    if display_images:
      fig = plot_protein(protein_obj, Ls=length, dpi=150)
      plt.show()
      plt.close()
```

Let's define our logging environment and retrieve the input queries from our job folder.

In [ ]:
```
result_dir = jobname
log_filename = os.path.join(jobname,"log.txt")
setup_logging(Path(log_filename))
```

Input Query Sequence

Lets specify our input for Sequence to Structure prediction. We should specify the input sequence (seperated by ':' for specifying inter-protein chain breaks to model complexes) and whether we would like to use amber relaxation or not.

In [ ]:
```
query_sequence = 'PIAQIHILEGRSDEQKETLIREVSEAISRSLDAPLTSVRVIITEMAKGHFGIGGELASK' #@param {type:"string"}
```

We then store our query_sequence in a csv inside the jobname director. This facilates multiple query execution and provides input in the format expected by ColabFold.

In [ ]:
```
# save queries
queries_path = os.path.join(jobname, f"{jobname}.csv")
with open(queries_path, "w") as text_file:
  text_file.write(f"id,sequence\n{jobname},{query_sequence}")
```

We utilize the get_queries function, which is a colabfold utility function, to fetch the queries from the directory.

One downside to having a high number of sequence alignemts(~128) detected in the MSA step is that it is quadrateically expensive to continue the later steps in inference. In order, to reduce this the MSA's are clustered smartly, based on sequence similarity, in order to reduce the computational cost while ensuring that each MSA has some influence on the final prediction. This clustering step is captured by use_cluster_profile which is also configured below.

In [ ]:
```
queries, is_complex = get_queries(queries_path)
model_type = set_model_type(is_complex, model_type)

if "multimer" in model_type and max_msa is not None:
  use_cluster_profile = False
else:
  use_cluster_profile = True
```

Inference

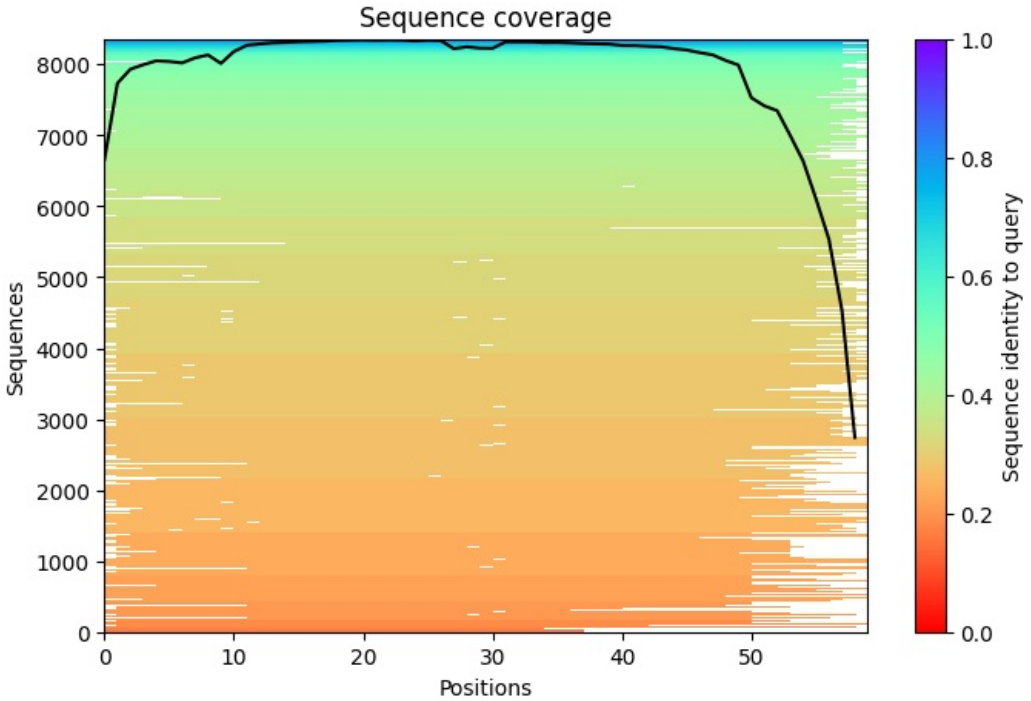
Our final step is to download the alphafold parameters for the model and run inference using all the previous given specifications and inputs.

We will then save the results in a zip file and download it.

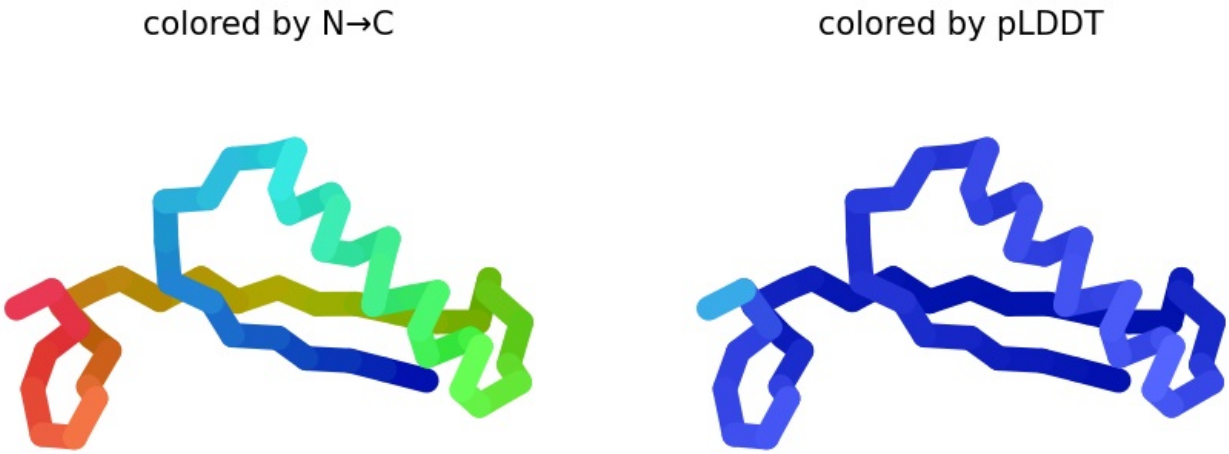
In [ ]:
```
download_alphafold_params(model_type, Path("."))
```

```
results = run(
    queries=queries,
    result_dir=result_dir,
    use_templates=use_templates,
    custom_template_path=custom_template_path,
    num_relax=num_relax,
    msa_mode=msa_mode,
    model_type=model_type,
    num_models=5,
    num_recycles=num_recycles,
    relax_max_iterations=relax_max_iterations,
    recycle_early_stop_tolerance=recycle_early_stop_tolerance,
    num_seeds=num_seeds,
    use_dropout=use_dropout,
    model_order=[1,2,3,4,5],
    is_complex=is_complex,
    data_dir=Path("."),
    keep_existing_results=False,
    rank_by="auto",
    pair_mode=pair_mode,
    pairing_strategy=pairing_strategy,
    stop_at_score=float(100),
    prediction_callback=prediction_callback,
    dpi=dpi,
    zip_results=False,
    save_all=save_all,
    max_msa=max_msa,
    use_cluster_profile=use_cluster_profile,
    input_features_callback=input_features_callback,
    save_recycles=save_recycles,
    user_agent="colabfold/google-colab-main",
)
results_zip = f"{jobname}.result.zip"
os.system(f"zip -r {results_zip} {jobname}")
```

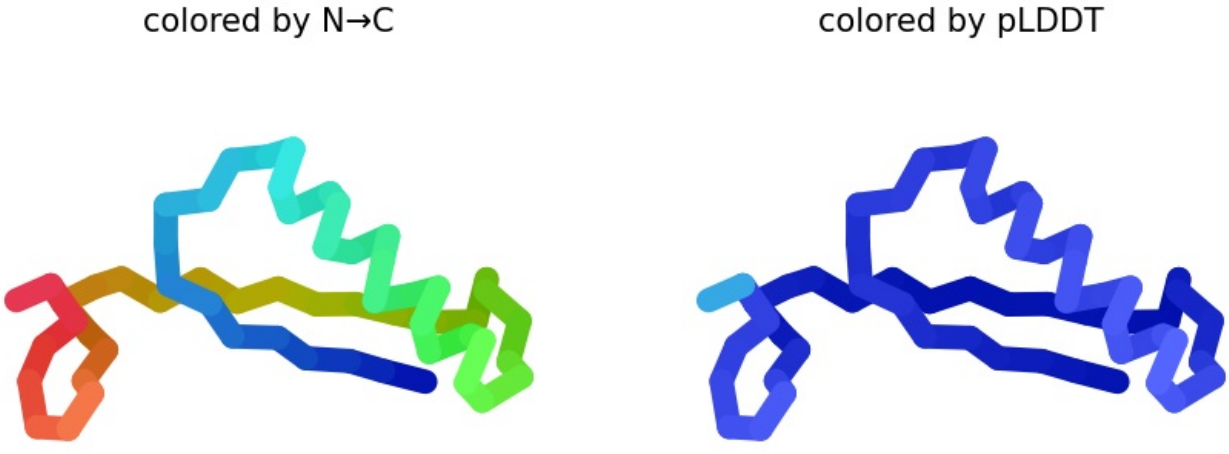
```
2023-11-14 00:35:33,396 Running on GPU
2023-11-14 00:35:33,401 Found 4 citations for tools or databases
2023-11-14 00:35:33,401 Query 1/1: test_d08f8 (length 59)
COMPLETE: 100%|██████████| 150/150 [elapsed: 00:02 remaining: 00:00]
```




```
2023-11-14 00:35:37,107 Setting max_seq=512, max_extra_seq=5120
2023-11-14 00:35:58,595 alphafold2_ptm_model_1_seed_000 recycle=0 pLDDT=96.6 pTM=0.755
2023-11-14 00:36:01,632 alphafold2_ptm_model_1_seed_000 recycle=1 pLDDT=96.6 pTM=0.758 tol=0.311
2023-11-14 00:36:04,668 alphafold2_ptm_model_1_seed_000 recycle=2 pLDDT=96.4 pTM=0.757 tol=0.043
2023-11-14 00:36:07,706 alphafold2_ptm_model_1_seed_000 recycle=3 pLDDT=96.2 pTM=0.757 tol=0.0399
2023-11-14 00:36:07,708 alphafold2_ptm_model_1_seed_000 took 25.1s (3 recycles)
```




```
2023-11-14 00:36:10,843 alphafold2_ptm_model_2_seed_000 recycle=0 pLDDT=96.9 pTM=0.76
2023-11-14 00:36:13,881 alphafold2_ptm_model_2_seed_000 recycle=1 pLDDT=96.9 pTM=0.765 tol=0.375
2023-11-14 00:36:16,920 alphafold2_ptm_model_2_seed_000 recycle=2 pLDDT=96.9 pTM=0.766 tol=0.12
2023-11-14 00:36:19,959 alphafold2_ptm_model_2_seed_000 recycle=3 pLDDT=96.8 pTM=0.767 tol=0.0779
2023-11-14 00:36:19,960 alphafold2_ptm_model_2_seed_000 took 12.2s (3 recycles)
```

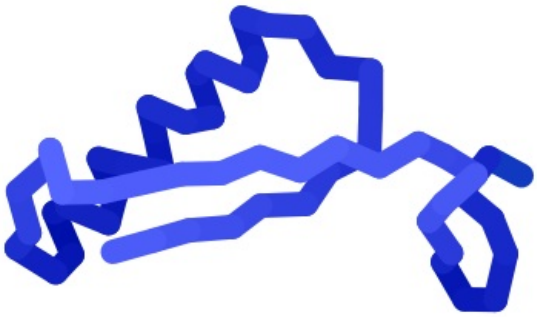



```
2023-11-14 00:36:23,114 alphafold2_ptm_model_3_seed_000 recycle=0 pLDDT=97.2 pTM=0.775
2023-11-14 00:36:26,153 alphafold2_ptm_model_3_seed_000 recycle=1 pLDDT=97.4 pTM=0.782 tol=0.297
2023-11-14 00:36:29,190 alphafold2_ptm_model_3_seed_000 recycle=2 pLDDT=97.4 pTM=0.782 tol=0.105
2023-11-14 00:36:32,230 alphafold2_ptm_model_3_seed_000 recycle=3 pLDDT=97.4 pTM=0.784 tol=0.0659
2023-11-14 00:36:32,231 alphafold2_ptm_model_3_seed_000 took 12.2s (3 recycles)
```

colored by N→C



colored by pLDDT

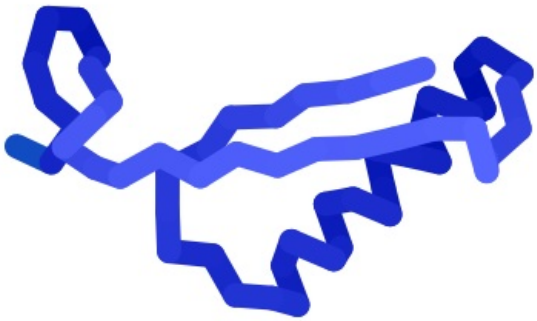


```
2023-11-14 00:36:35,372 alphafold2_ptm_model_4_seed_000 recycle=0 pLDDT=97.4 pTM=0.775
2023-11-14 00:36:38,411 alphafold2_ptm_model_4_seed_000 recycle=1 pLDDT=97.4 pTM=0.781 tol=0.282
2023-11-14 00:36:41,451 alphafold2_ptm_model_4_seed_000 recycle=2 pLDDT=97.2 pTM=0.778 tol=0.0839
2023-11-14 00:36:44,490 alphafold2_ptm_model_4_seed_000 recycle=3 pLDDT=97.1 pTM=0.779 tol=0.0421
2023-11-14 00:36:44,491 alphafold2_ptm_model_4_seed_000 took 12.2s (3 recycles)
```

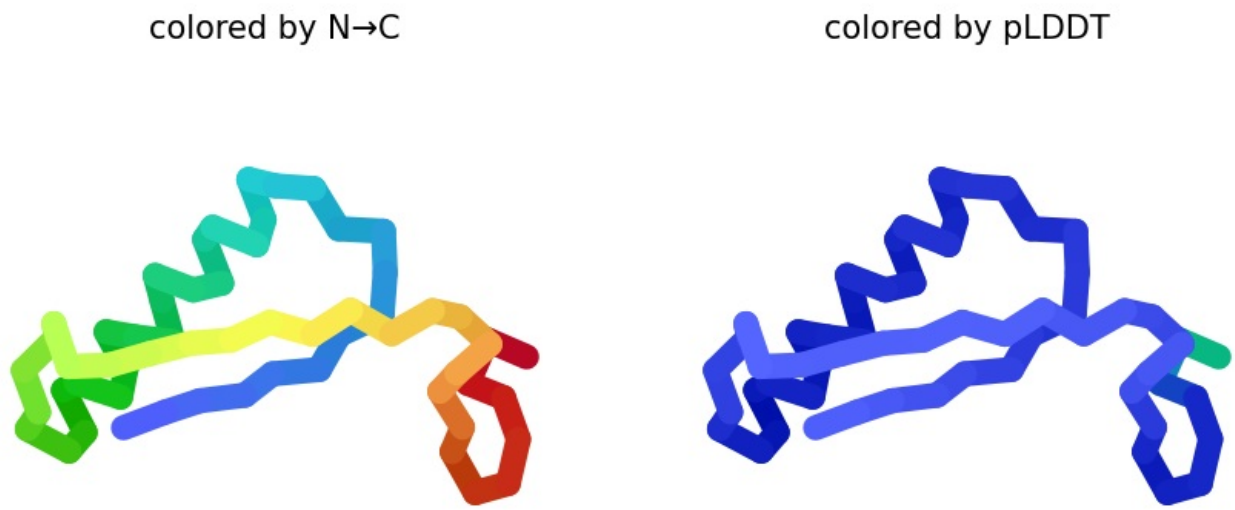
colored by N→C



colored by pLDDT



```
2023-11-14 00:36:47,670 alphafold2_ptm_model_5_seed_000 recycle=0 pLDDT=97.4 pTM=0.784
2023-11-14 00:36:50,712 alphafold2_ptm_model_5_seed_000 recycle=1 pLDDT=96.9 pTM=0.784 tol=0.237
2023-11-14 00:36:53,750 alphafold2_ptm_model_5_seed_000 recycle=2 pLDDT=96.3 pTM=0.777 tol=0.18
2023-11-14 00:36:56,788 alphafold2_ptm_model_5_seed_000 recycle=3 pLDDT=96.2 pTM=0.777 tol=0.121
2023-11-14 00:36:56,789 alphafold2_ptm_model_5_seed_000 took 12.2s (3 recycles)
```




```
2023-11-14 00:36:56,927 reranking models by 'plddt' metric
2023-11-14 00:36:56,927 rank_001_alphafold2_ptm_model_3_seed_000 pLDDT=97.4 pTM=0.784
2023-11-14 00:36:56,929 rank_002_alphafold2_ptm_model_4_seed_000 pLDDT=97.1 pTM=0.779
2023-11-14 00:36:56,930 rank_003_alphafold2_ptm_model_2_seed_000 pLDDT=96.8 pTM=0.767
2023-11-14 00:36:56,930 rank_004_alphafold2_ptm_model_1_seed_000 pLDDT=96.2 pTM=0.757
2023-11-14 00:36:56,931 rank_005_alphafold2_ptm_model_5_seed_000 pLDDT=96.2 pTM=0.777
2023-11-14 00:36:58,495 Done
```

Out[ ]: 0

Display 3D structure of the protein file generated based on a few options and using py3Dmol package

Alphafold generates top n ranked model estimates of the protein structure. Here we have 5 ranked structures. The lower the rank the higher the accuracy and quality of the predicted model.

Here we can display the structure with various color schemas: chair, lDDT, and rainbow. We also have options to show the sidechain and mainchains of the protein's structure.

Let's import a few important visualization libraries and set the visualization variables.

In [ ]:
```
#@title Display 3D structure {run: "auto"}
import py3Dmol
import glob
import matplotlib.pyplot as plt
from colabfold.colabfold import plot_plddt_legend
from colabfold.colabfold import pymol_color_list, alphabet_list
rank_num = 1 #@param ["1", "2", "3", "4", "5"] {type:"raw"}
color = "lDDT" #@param ["chain", "lDDT", "rainbow"]
show_sidechains = False #@param {type:"boolean"}
show_mainchains = False #@param {type:"boolean"}

tag = results["rank"][0][rank_num - 1]
jobname_prefix = ".custom" if msa_mode == "custom" else ""
pdb_filename = f"{jobname}/{jobname}{jobname_prefix}_unrelaxed_{tag}.pdb"
pdb_file = glob.glob(pdb_filename)
```

Capture the protein structure

Let us store the file path to the generated protein structure so that we can utilize it in the second part of our tutorial for pocket finding and protein-ligand interactions!

In [ ]:
```
print("Files generated: ", pdb_filename, pdb_file)
pdb_filename_captured = pdb_filename
```

```
Files generated:  test_a5e17_0/test_a5e17_0_unrelaxed_rank_001_alphafold2_ptm_model_3_seed_000.pdb ['test_a5e17_
0/test_a5e17_0_unrelaxed_rank_001_alphafold2_ptm_model_3_seed_000.pdb']
```

Now lets define the visualization function show_pdb using py3Dmol. This function takes in the pdb and various visualization parameters such as rank number, sidechains, and mainchains and visualizes the protein accordingly Here we have passed "lDDT" for the color parameter of the function. lDDT is in short for local Distance Difference Test. We display the color based on the score for each part of the protein as mentioned in the key of the .

In [ ]:
```
def show_pdb(rank_num=1, show_sidechains=False, show_mainchains=False, color="lDDT"):
  model_name = f"rank_{rank_num}"
  view = py3Dmol.view(js='https://3dmol.org/build/3Dmol.js',)
  view.addModel(open(pdb_file[0],'r').read(),'pdb')

  if color == "lDDT":
    view.setStyle({'cartoon': {'colorscheme': {'prop':'b','gradient': 'roygb','min':50,'max':90}}})
  elif color == "rainbow":
    view.setStyle({'cartoon': {'color':'spectrum'}})
  elif color == "chain":
    chains = len(queries[0][1]) + 1 if is_complex else 1
    for n,chain,color in zip(range(chains),alphabet_list,pymol_color_list):
       view.setStyle({'chain':chain},{'cartoon': {'color':color}})

  if show_sidechains:
    BB = ['C','O','N']
    view.addStyle({'and':[{'resn':["GLY","PRO"],'invert':True},{'atom':BB,'invert':True}]},
                  {'stick':{'colorscheme':f"WhiteCarbon",'radius':0.3}})
    view.addStyle({'and':[{'resn':"GLY"},{'atom':'CA'}]},
                  {'sphere':{'colorscheme':f"WhiteCarbon",'radius':0.3}})
    view.addStyle({'and':[{'resn':"PRO"},{'atom':['C','O'],'invert':True}]},
                  {'stick':{'colorscheme':f"WhiteCarbon",'radius':0.3}})
  if show_mainchains:
    BB = ['C','O','N','CA']
    view.addStyle({'atom':BB},{'stick':{'colorscheme':f"WhiteCarbon",'radius':0.3}})

  view.zoomTo()
  return view

show_pdb(rank_num, show_sidechains, show_mainchains, color).show()
if color == "lDDT":
  plot_plddt_legend().show()
```



PLOTS of Alphafold

The following 3 types of plots are generated with Alphafold.

1. **PAE** (Predicted Average Error): It measures the average error in the predicted atomic positions of a protein's 3D structure compared to the true positions. In our experiment below, the PAE plot of the top 5 ranked sturctures is mostly blue which means that we have low errors.

2. **COV** (Sequence Coverage): It indicates the percentage of aminoacid sequence for which Alphafold has provided structural prediction and to what degree. In the plot below, the x-axis represents the amino acid sequence and the y-axis represents the number sequences for that amino acid.

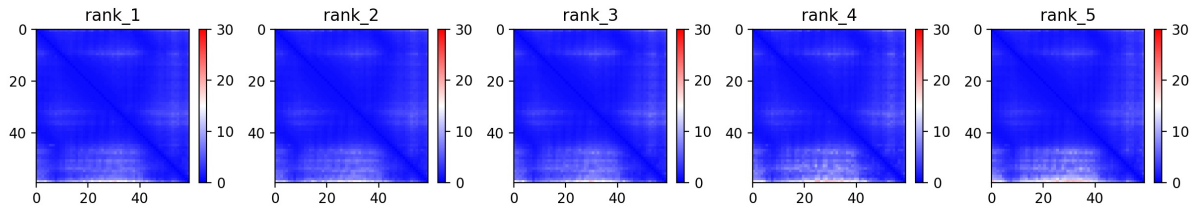
3. **iDDT** (local Distance Difference Test) or lDDT: It provides a per-residue measure of the predicted model's confidence. It assigns a score to each residue in the protein structure, indicating the reliability of the prediction at that location. For our protein sequence the lDDT is high throughout the amino acid sequence but tends to get lower towards the right end.

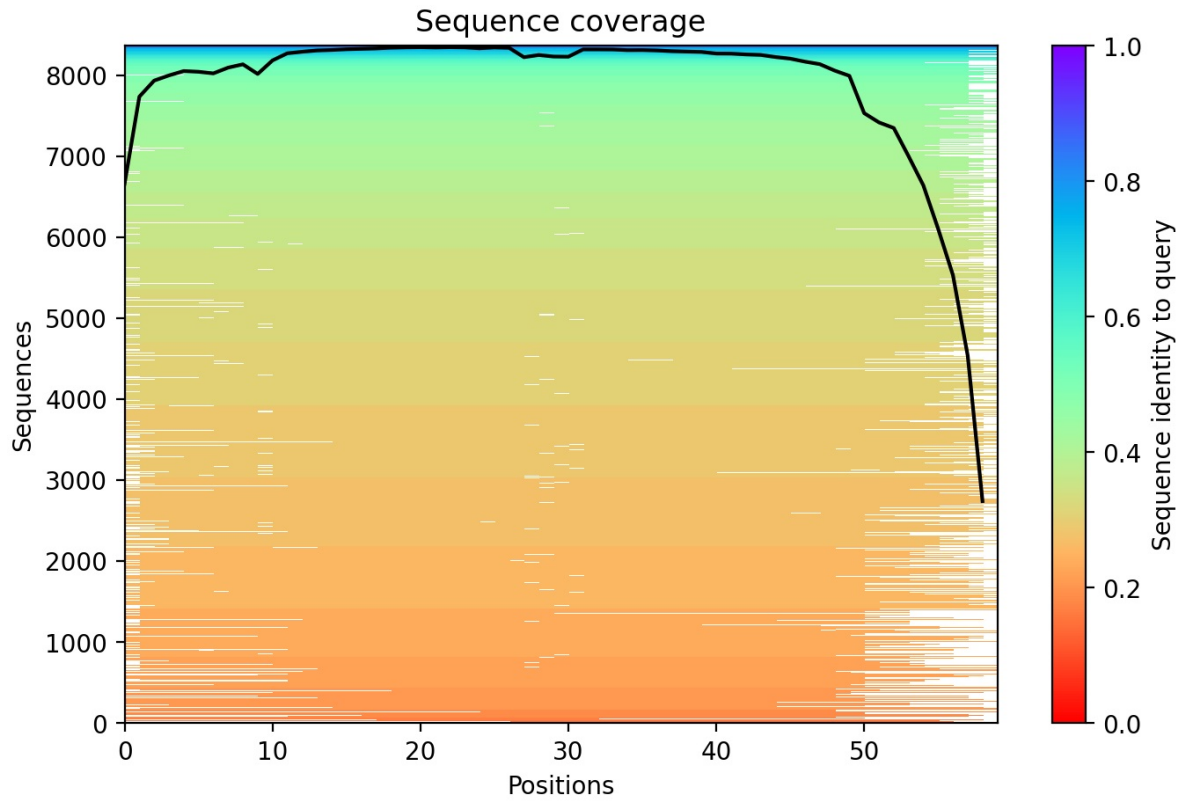
In [ ]:
```
#@title Plots {run: "auto"}
from IPython.display import display, HTML
import base64
from html import escape

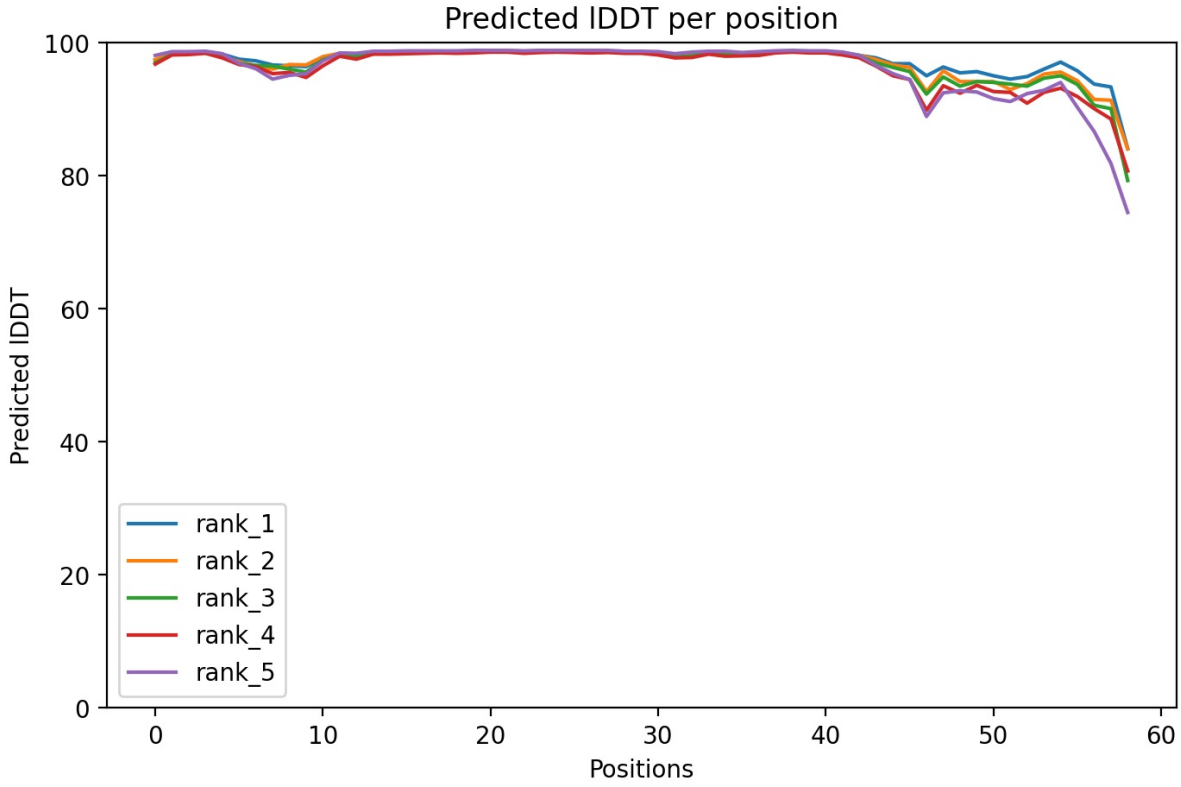
# see: https://stackoverflow.com/a/53688522
def image_to_data_url(filename):
  ext = filename.split('.')[-1]
  prefix = f''
  with open(filename, 'rb') as f:
    img = f.read()
  return prefix + base64.b64encode(img).decode('utf-8')

pae = image_to_data_url(os.path.join(jobname,f"{jobname}{jobname_prefix}_pae.png"))
cov = image_to_data_url(os.path.join(jobname,f"{jobname}{jobname_prefix}_coverage.png"))
plddt = image_to_data_url(os.path.join(jobname,f"{jobname}{jobname_prefix}_plddt.png"))
display(HTML(f"""
<style>
  img {{
    float:left;
  }}
  .full {{
    max-width:100%;
  }}
  .half {{
    max-width:50%;
  }}
  @media (max-width:640px) {{
    .half {{
      max-width:100%;
    }}
  }}
</style>
<div style="max-width:90%; padding:2em;">
  <h1>Plots for {escape(jobname)}</h1>
  <img src="{pae}" class="full" />
  <img src="{cov}" class="half" />
  <img src="{plddt}" class="half" />
</div>
"""))
```

Plots for test_d08f8







Package and download results(optional)

The below code can be used to download the generated zip folder to your local system or saved to google drive.

In [ ]:
```
if msa_mode == "custom":
  print("Don't forget to cite your custom MSA generation method.")

files.download(f"{jobname}.result.zip")

if save_to_google_drive == True and drive:
  uploaded = drive.CreateFile({'title': f"{jobname}.result.zip"})
  uploaded.SetContentFile(f"{jobname}.result.zip")
  uploaded.Upload()
  print(f"Uploaded {jobname}.result.zip to Google Drive with ID {uploaded.get('id')}")
```

Part 2: Estimate binding affinities between the predicted

protein and a sample set of ligands

Let's import deepchem utility functions and rdkit which helps us in visualization.

In [ ]:
```
import os
import numpy as np
import pandas as pd

import tempfile

from rdkit import Chem
from rdkit.Chem import AllChem

import deepchem as dc
from deepchem.utils import download_url, load_from_disk
```

```
2023-11-14 00:45:56,266 Enabling RDKit 2023.09.1 jupyter extensions
2023-11-14 00:46:00,469 Skipped loading modules with pytorch-geometric dependency, missing a dependency. No modu
le named 'torch_geometric'
2023-11-14 00:46:00,471 Skipped loading modules with pytorch-geometric dependency, missing a dependency. cannot
import name 'DMPNN' from 'deepchem.models.torch_models' (/usr/local/lib/python3.10/site-packages/deepchem/models
/torch_models/__init__.py)
2023-11-14 00:46:00,473 Skipped loading modules with pytorch-lightning dependency, missing a dependency. No modu
le named 'pytorch_lightning'
```

To sample a set of ligands we will use PDBind. We will download the respective dataset file and store it in a variable called raw_dataset.

In [ ]:
```
data_dir = dc.utils.get_data_dir()
dataset_file = os.path.join(data_dir, "pdbbind_core_df.csv.gz")

if not os.path.exists(dataset_file):
    print('File does not exist. Downloading file...')
    download_url("https://s3-us-west-1.amazonaws.com/deepchem.io/datasets/pdbbind_core_df.csv.gz")
    print('File downloaded...')

raw_dataset = load_from_disk(dataset_file)
raw_dataset = raw_dataset[['pdb_id', 'smiles', 'label']]


from openmm.app import PDBFile
from pdbfixer import PDBFixer

from deepchem.utils.vina_utils import prepare_inputs
```

```
File does not exist. Downloading file...
File downloaded...
```

Pocket finding in the predicted protein structure

By utilizing DeepChem's docking modules we can find the number of pockets in the generated structure. We do so by utilizing the 3D-ConvexHullPocketFinder.

In [ ]:
```
ligands10 = raw_dataset['smiles'].iloc[0:10]

# %%time
import os
#'test_a5e17/test_a5e17_unrelaxed_rank_001_alphafold2_ptm_model_3_seed_000.pdb'
generated_pdb = pdb_filename_captured
generated_pdb_no_extension = os.path.splitext(os.path.basename(generated_pdb))[0]

finder = dc.dock.binding_pocket.ConvexHullPocketFinder()
pockets = finder.find_pockets(generated_pdb)

print("Pockets for protein: "  + str(len(pockets)) ) # number of identified pockets
```

```
Pockets for protein: 8
```

Docking with Vina Pose Generator

For docking the protein and ligand, we will be using DeepChem's VinaPoseGenerator which automatically installs AutoDock Vina engine and allows us to seamlessly use it. And before we dock a given ligand and protein we have to ensure that the inputs are reasonable for docking which is also handled by a DeepChem's utility function called prepare_inputs. It fixes the protein geometry, optimizes ligand geometry, and sanitizes the molecules. Next, we generate the complex combining the protein and ligand and then display the scores. Analysis of the scores of the combined complex is printed below. For the sake of simplicity, we will be iterating over 3 ligands and comparing the complex scores.

In [ ]:

```
import locale
locale.getpreferredencoding = lambda: "UTF-8"
!mkdir results

vpg = dc.dock.pose_generation.VinaPoseGenerator()
count=0
scores_matrix =[]
complex_mol_array = []
for count in range(0,3):
  print("Docking ligand "+str(count))
  ligand = ligands10[count]
  p, m = None, None

  vpg = dc.dock.pose_generation.VinaPoseGenerator()

  try:
      p, m = prepare_inputs('%s' % (generated_pdb), ligand)
  except:
      print('%s failed PDB fixing' % (generated_pdb))

  if p and m:  # protein and molecule are readable by RDKit
      print(generated_pdb, p.GetNumAtoms())
      Chem.rdmolfiles.MolToPDBFile(p, 'results/protein_%s.pdb' % (count))
      Chem.rdmolfiles.MolToPDBFile(m, 'results/ligand_%s.pdb' % (count))

  complexes, scores = vpg.generate_poses(molecular_complex=('results/protein_%s.pdb' % (count),'results/ligand_%
                                         out_dir='vina_test',
                                         generate_scores=True
                                        )
  complex_mol = Chem.CombineMols(complexes[0][0], complexes[0][1])
  complex_mol_array.append(complex_mol)
  print(scores)
  scores_matrix.append(scores)
```

```
Docking ligand 0
<ipython-input-42-a86da5d11cfe>:17: DeprecationWarning: Call to deprecated function prepare_inputs. Please use t
he corresponding function in deepchem.utils.docking_utils.
  p, m = prepare_inputs('%s' % (generated_pdb), ligand)
[00:47:58] UFFTYPER: Unrecognized atom type: S_5+4 (7)
test_a5e17_0/test_a5e17_0_unrelaxed_rank_001_alphafold2_ptm_model_3_seed_000.pdb 448
2023-11-14 00:48:01,498 Pockets not specified. Will use whole protein to dock
2023-11-14 00:48:03,344 Docking in pocket 1/1
2023-11-14 00:48:03,345 Docking with center: [0.28462623 1.04385902 1.65269617]
2023-11-14 00:48:03,345 Box dimensions: [45.593 35.786 38.447]
2023-11-14 00:48:03,346 About to call Vina
/usr/local/lib/python3.10/site-packages/vina/vina.py:260: DeprecationWarning: `np.int` is a deprecated alias for
the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is
safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If yo
u wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#dep
recations
  self._voxels = np.ceil(np.array(box_size) / self._spacing).astype(np.int)
<ipython-input-42-a86da5d11cfe>:17: DeprecationWarning: Call to deprecated function prepare_inputs. Please use t
he corresponding function in deepchem.utils.docking_utils.
  p, m = prepare_inputs('%s' % (generated_pdb), ligand)
[-4.321, -4.142, -4.135, -4.109, -4.083, -4.069, -4.046, -4.036, -3.993]
Docking ligand 1
test_a5e17_0/test_a5e17_0_unrelaxed_rank_001_alphafold2_ptm_model_3_seed_000.pdb 448
2023-11-14 00:49:13,834 Pockets not specified. Will use whole protein to dock
2023-11-14 00:49:15,455 Docking in pocket 1/1
2023-11-14 00:49:15,457 Docking with center: [0.28062951 1.0434776  1.64895082]
2023-11-14 00:49:15,457 Box dimensions: [45.662 35.909 38.417]
2023-11-14 00:49:15,458 About to call Vina
/usr/local/lib/python3.10/site-packages/vina/vina.py:260: DeprecationWarning: `np.int` is a deprecated alias for
the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is
safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If yo
u wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#dep
recations
  self._voxels = np.ceil(np.array(box_size) / self._spacing).astype(np.int)
<ipython-input-42-a86da5d11cfe>:17: DeprecationWarning: Call to deprecated function prepare_inputs. Please use t
he corresponding function in deepchem.utils.docking_utils.
  p, m = prepare_inputs('%s' % (generated_pdb), ligand)
[-6.083, -6.022, -5.811, -5.797, -5.796, -5.73, -5.689, -5.654, -5.643]
Docking ligand 2
test_a5e17_0/test_a5e17_0_unrelaxed_rank_001_alphafold2_ptm_model_3_seed_000.pdb 448
2023-11-14 00:55:55,258 Pockets not specified. Will use whole protein to dock
2023-11-14 00:55:56,761 Docking in pocket 1/1
2023-11-14 00:55:56,762 Docking with center: [0.28300874 1.0426     1.64975847]
2023-11-14 00:55:56,762 Box dimensions: [45.657 35.819 38.429]
2023-11-14 00:55:56,763 About to call Vina
[-5.96, -5.9, -5.791, -5.733, -5.704, -5.662, -5.617, -5.605, -5.591]
```

```
/usr/local/lib/python3.10/site-packages/vina/vina.py:260: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  self._voxels = np.ceil(np.array(box_size) / self._spacing).astype(np.int)
```

Visualize the Protein ligand Complex!

We will use nglview to display the complex which we have stored previously. We add output.enable_custom_widget_manager() so that we can run nglview in colab.

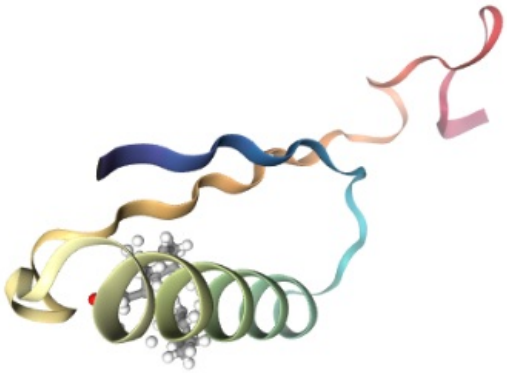
In [ ]:
```
from google.colab import output

import mdtraj as md
import nglview

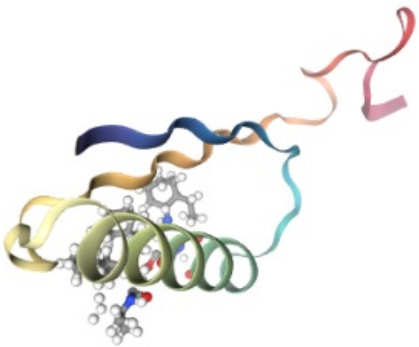
from IPython.display import display, Image
output.enable_custom_widget_manager()
```

Now lets visualize the first 3 protein ligand complexes which we have stored in the complex_mol_array

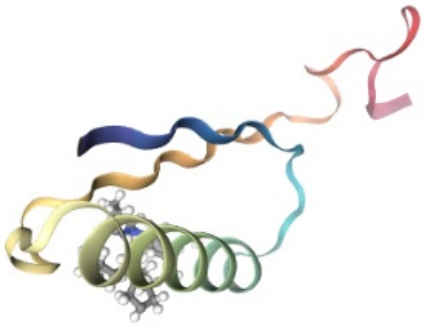
In [ ]:
```
v = nglview.show_rdkit(complex_mol_array[0])
display(v)
```



In [ ]:
```
v = nglview.show_rdkit(complex_mol_array[1])
display(v)
```



In [ ]:
```
v = nglview.show_rdkit(complex_mol_array[2])
display(v)
```



In [ ]:
```
print(scores_matrix)
```

```
[[-4.321, -4.142, -4.135, -4.109, -4.083, -4.069, -4.046, -4.036, -3.993], [-6.083, -6.022, -5.811, -5.797, -5.7
96, -5.73, -5.689, -5.654, -5.643], [-5.96, -5.9, -5.791, -5.733, -5.704, -5.662, -5.617, -5.605, -5.591]]
```

Next, we can see that all the scores generated from Vina Pose Generator for the respective complexes are negative. This is because protein–ligand binding occurs only when the change in Gibbs free energy (ΔG) of the system is negative and more negative the free energy is the more stable the complex would be as show in Ref. Additionally, molecular docking evaluation based on the paper here showed that the binding affinities of all the derivatives range from (- 3.2 and -18.5 kcal/mol).

Hence based on our experiment we can successfully predict the potential affinity between a protein sequence and a ligand even of we just have the protein sequence!

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

```
@manual{DeepChemXAlphafold,
 title={Applications of DeepChem with Alphafold: Docking and protein-ligand interaction from
protein sequence},
 organization={DeepChem},
 author={Bellamkonda, Sriphani Vardhan},
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/DeepChemXAlphafold.ipynb}}

 year={2023},
}
```

UniProt data pre-processing for binding site prediction downstream task

This notebook guides you through:

- **Downloading Data**: Retrieve information from the UniProt website, including details on protein families, binding sites, active sites, and amino acid sequences.
- **Processing Data**: Handle special symbols (angle brackets and question marks) in binding/active site information and convert this data into binary labels. Each amino acid position in the protein sequences is marked as 1 (binding/active site) or 0 (non-binding/active site).
- ✂ **Splitting Data**: Divide amino acid sequences and their labels into stratified train/test sets based on UniProt protein families.
- **Chunking Sequences**: Split sequences and their labels into non-overlapping chunks of a specified length to define a context window for the ESM-2 model.

This tutorial is made to run without any GPU support, and can be used in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Download from UniProt

Let's first download a dataset of proteins from UniProt. We will obtain a TSV (Tab-Separated Values) file with specific columns such as Protein families, Binding site, Active site, and Sequence. You can achieve this following these steps:

- Go to the UniProt website and perform a search to query for the proteins of interest (you can search by organism, protein name, function, etc). Filter your results with the filters on the left-hand side to refine your results further if necessary. Here I performed the search: (organism_id:9606) AND (family:kinase) AND (existence:1 OR existence:2) in UniProtKB.

- Select columns: Above the search results, there is an option to select the columns you want to be included in your download. Click on the 'Columns' button and a dropdown menu will appear.

- Customize columns: In the dropdown menu, you can check the boxes next to the columns you want to include in your TSV file. Look for the 'Protein families', 'Binding site', 'Active site', and 'Sequence' options. I also added further info such as entry name, protein name, gene name, organism, sequence length and whether the entry has been reviewed.

- Download the file: After selecting the desired columns, click the 'Download' button located above the search results. Choose the 'Tab-separated' format from the list of available formats. You may also have the option to select the number of entries you want to download (e.g., all entries, displayed entries, or a custom range). Click on the 'Download' button to start the download process and your browser will prompt you to save the TSV file.

Process data

Now, let's process the downloaded UniProt TSV file with columns (Protein families, Binding site, Active site, Sequence). If the family annotation or binding sites are missing, the code will filter out this sequence. If the Active site annotation is missing, the sequence will be included without issue. Missing sequences are not handled by this notebook.

But first, let's set up the environment:

In [ ]:
```
!pip install pandas
!pip install numpy
!pip install requests
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (
2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.25.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2-
>pandas) (1.16.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.25.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
s) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests) (2.
0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests) (20
24.7.4)
```

In [ ]:
```python
# I/O
import pandas as pd
import numpy as np
import re
import random
import pickle
import os
import requests
import xml.etree.ElementTree as ET
# set seed
random.seed(42)
np.random.seed(42)
```

If you upload the downloaded file from UniProt to Google Drive, you should be able to access it by first mounting your Google Drive and then loading it:

In [ ]:
```python
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

In [ ]:
```python
# Load the dataset
file_path = "/content/gdrive/MyDrive/ESMbind/data/uniprotkb_data_2024_05_29.tsv"
data = pd.read_csv(file_path, sep='\t')
data.head()
```

Out[ ]:

	Entry	Reviewed	Entry Name	Protein names	Gene Names	Organism	Protein families	
0	A0A087WV00	unreviewed	A0A087WV00_HUMAN	Diacylglycerol kinase (DAG kinase) (EC 2.7.1.107)	DGKI	Homo sapiens (Human)	Eukaryotic diacylglycerol kinase family	MDAAGRGCHLI
1	A0A090N7W4	unreviewed	A0A090N7W4_HUMAN	Cell division protein kinase 5	CDK5 hCG_18690 tcag7.772	Homo sapiens (Human)	Protein kinase superfamily, CMGC Ser/Thr prote...	MQKYEKLEKI(
2	A0A0S2Z310	unreviewed	A0A0S2Z310_HUMAN	Serine/threonine-protein kinase receptor (EC 2...	ACVRL1	Homo sapiens (Human)	Protein kinase superfamily, TKL Ser/Thr protei...	MTLGSPRKGLI
3	A0A0S2Z4D1	unreviewed	A0A0S2Z4D1_HUMAN	non-specific serine/threonine protein kinase (...	STK11	Homo sapiens (Human)	Protein kinase superfamily, CAMK Ser/Thr prote...	MEVVDPQQL
4	A0A2P9DU05	unreviewed	A0A2P9DU05_HUMAN	Rho-associated protein kinase (EC 2.7.11.1)	ROCK2	Homo sapiens (Human)	Protein kinase superfamily, AGC Ser/Thr protei...	MSRPPPTGKM

Now let's extract the required information for the purposes of this task: Protein families, Binding site, Active site, Sequence. Also, let's filter out entries without binding site or protein families information.

In [ ]:
```python
data["Binding site"]
```

Out[ ]:
```
0                                                    NaN
1       BINDING 33; /ligand="ATP"; /ligand_id="ChEBI:C...
2       BINDING 229; /ligand="ATP"; /ligand_id="ChEBI:...
3       BINDING 78; /ligand="ATP"; /ligand_id="ChEBI:C...
4       BINDING 121; /ligand="ATP"; /ligand_id="ChEBI:...
                              ...
2186                                                 NaN
2187                                                 NaN
2188                                                 NaN
2189    BINDING 73; /ligand="ATP"; /ligand_id="ChEBI:C...
2190    BINDING 165; /ligand="ATP"; /ligand_id="ChEBI:...
Name: Binding site, Length: 2191, dtype: object
```

In [ ]:
```
data = data[["Entry", "Protein families", "Binding site", "Active site", "Sequence"]]
# Filter out rows with NaN values in the 'Protein families' column nor the 'Binding site' column nor the 'Seque
data = data[pd.notna(data['Protein families']) & pd.notna(data['Binding site']) & pd.notna(data['Sequence'])]
print(data.shape)
data.head()
```

```
(1406, 5)
```

Out[ ]:

	Entry	Protein families	Binding site	Active site	Seque
1	A0A090N7W4	Protein kinase superfamily, CMGC Ser/Thr prote...	BINDING 33; /ligand="ATP"; /ligand_id="ChEBI:C...	NaN	MQKYEKLEKIGEGTYGTVFKAKNRETHEIVALKRVRLDDDDEGVP
2	A0A0S2Z310	Protein kinase superfamily, TKL Ser/Thr protei...	BINDING 229; /ligand="ATP"; /ligand_id="ChEBI:...	NaN	MTLGSPRKGLLMLLMALVTQGDPVKPSRGPLVTCTCESPHCKGPT
3	A0A0S2Z4D1	Protein kinase superfamily, CAMK Ser/Thr prote...	BINDING 78; /ligand="ATP"; /ligand_id="ChEBI:C...	NaN	MEVVDPQQLGMFTEGELMSVGMDTFIHRIDSTEVIYQPRRKRAK
4	A0A2P9DU05	Protein kinase superfamily, AGC Ser/Thr protei...	BINDING 121; /ligand="ATP"; /ligand_id="ChEBI:...	ACT_SITE 214; /note="Proton acceptor"; /eviden...	MSRPPPTGKMPGAPETAPGDGAGASRQRKLEALIRDPRSPINVES
5	A3QNQ0	Protein kinase superfamily, TKL Ser/Thr protei...	BINDING 250..258; /ligand="ATP"; /ligand_id="C...	ACT_SITE 379; /note="Proton acceptor"; /eviden...	MGRGLLRGLWPLHIVLWTRIASTIPPHVQKSVNNDMIVTDNNGAV

So we have a dataset of 1406 proteins, all having a binding site and information of the aminoacids sequence and the protein family. We download proteins proteins from human and kinase family, however there may still exist subgroups of protein families:

In [ ]:
```
# Group the data by 'Protein families' and get the size of each group
family_sizes = data.groupby('Protein families').size()
print(family_sizes.sort_values(ascending=False))

# Create a new column with the size of each family and sort by 'Family size' in descending order and then by 'P
data['Family size'] = data['Protein families'].map(family_sizes)
data = data.sort_values(by=['Family size', 'Protein families'], ascending=[False, True])
data.drop(columns='Family size', inplace=True) # Drop the 'Family size' column as it is no longer needed
data
```

```
Protein families
Protein kinase superfamily                                                                  164
Protein kinase superfamily, CMGC Ser/Thr protein kinase family, CDC2/CDKX subfamily          96
Protein kinase superfamily, STE Ser/Thr protein kinase family, STE20 subfamily              78
Protein kinase superfamily, Tyr protein kinase family, Insulin receptor subfamily          73
Protein kinase superfamily, CAMK Ser/Thr protein kinase family                              56
                                                                                          ...
GHMP kinase family, Mevalonate kinase subfamily                                              1
Protein kinase superfamily, TKL Ser/Thr protein kinase family, ROCO subfamily                1
Glutamate 5-kinase family; Gamma-glutamyl phosphate reductase family                         1
Guanylate kinase family                                                                      1
GHMP kinase family                                                                           1
Length: 126, dtype: int64
```

Out[ ]:

	Entry	Protein families	Binding site	Active site	S
359	Q504Y2	Protein kinase superfamily	BINDING 144..152; /ligand="ATP"; /ligand_id="C...	ACT_SITE 278; /note="Proton acceptor"; /eviden...	MRRRRAAVAAGFCASFLLGSVLNVLFAPGSEPPRPGQSPEI
414	Q8IWB6	Protein kinase superfamily	BINDING 233..241; /ligand="ATP"; /ligand_id="C...	NaN	MSRAVRLPVPCPVQLGTLRNDSLEAQLHEYVKQGNYVKVK
427	Q8NB16	Protein kinase superfamily	BINDING 209..217; /ligand="ATP"; /ligand_id="C...	NaN	MENLKHIITLGQVIHKRCEEMKYCKKQCRRLGHRVLGLIKF
778	A0A7P0T838	Protein kinase superfamily	BINDING 71; /ligand="ATP"; /ligand_id="ChEBI:C...	NaN	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVG
779	A0A7P0T952	Protein kinase superfamily	BINDING 71; /ligand="ATP"; /ligand_id="ChEBI:C...	NaN	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVG
...	...	...	...	...	
1770	M1VPF4	Protein kinase superfamily, Tyr protein kinase...	BINDING 358; /ligand="ATP"; /ligand_id="ChEBI:...	NaN	MMEAIKKKMQMLKLDKENALDRAEQAEAEQKQAEERSKQLE
21	O00764	Pyridoxine kinase family	BINDING 12; /ligand="pyridoxal"; /ligand_id="C...	ACT_SITE 235; /note="Proton acceptor"; /eviden...	MEEECRVLSIQSHVIRGYVGNRAATFPLQVLGFEIDAVNSV
1017	M1V485	SLC34A transporter family; Protein kinase supe...	BINDING 906; /ligand="ATP"; /ligand_id="ChEBI:...	NaN	MAPWPELGDAQPNPDKYLEGAAGQQPTAPDKSKETNKTDN
82	P04183	Thymidine kinase family	BINDING 26..33; /ligand="ATP"; /ligand_id="ChE...	ACT_SITE 98; /note="Proton acceptor"; /evidenc...	MSCINLPTVLPGSPSKTRGQIQVILGPMFSGKSTELMRRV
542	Q9NVE7	Type II pantothenate kinase family; Damage-con...	BINDING 196; /ligand="acetyl-CoA"; /ligand_id=...	NaN	MAECGASGSGSSGDSLDKSITLPPDEIFRNLENAKRFAID

1406 rows × 5 columns

Now let's make the binding and active sites information clearer:

In [ ]:

```python
# Extract the location from the binding and active site columns
def extract_location(site_info):
    if pd.isnull(site_info):
        return None
    locations = []
    for info in site_info.split(';'):
        if 'BINDING' in info or 'ACT_SITE' in info:
            locations.append(info.split()[1])
```

```
    return '; '.join(locations)

# Apply the function to the 'Binding site' and 'Active site' columns to extract the locations
data['Binding site'] = data['Binding site'].apply(extract_location)
data['Active site'] = data['Active site'].apply(extract_location)

# Display the first few rows of the modified dataframe
data.head()
```

Out[ ]:

	Entry	Protein families	Binding site	Active site	Sequence
359	Q504Y2	Protein kinase superfamily	144..152; 166	278	MRRRRAAVAAGFCASFLLGSVLNVLFAPGSEPPRPGQSPEPSPAPG...
414	Q8IWB6	Protein kinase superfamily	233..241; 273	None	MSRAVRLPVPCPVQLGTLRNDSLEAQLHEYVKQGNYVKVKKILKKG...
427	Q8NB16	Protein kinase superfamily	209..217; 230	None	MENLKHIITLGQVIHKRCEEMKYCKKQCRRLGHRVLGLIKPLEMLQ...
778	A0A7P0T838	Protein kinase superfamily	71	None	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...
779	A0A7P0T952	Protein kinase superfamily	71	None	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...

In [ ]:

```
# Create a new column that combines the 'Binding site' and 'Active site' columns
data['Binding-Active site'] = data['Binding site'].astype(str) + '; ' + data['Active site'].astype(str)
# Replace 'nan' values with None
data['Binding-Active site'] = data['Binding-Active site'].replace('nan; nan', None)

data.head()
```

Out[ ]:

	Entry	Protein families	Binding site	Active site	Sequence	Binding-Active site
359	Q504Y2	Protein kinase superfamily	144..152; 166	278	MRRRRAAVAAGFCASFLLGSVLNVLFAPGSEPPRPGQSPEPSPAPG...	144..152; 166; 278
414	Q8IWB6	Protein kinase superfamily	233..241; 273	None	MSRAVRLPVPCPVQLGTLRNDSLEAQLHEYVKQGNYVKVKKILKKG...	233..241; 273; None
427	Q8NB16	Protein kinase superfamily	209..217; 230	None	MENLKHIITLGQVIHKRCEEMKYCKKQCRRLGHRVLGLIKPLEMLQ...	209..217; 230; None
778	A0A7P0T838	Protein kinase superfamily	71	None	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...	71; None
779	A0A7P0T952	Protein kinase superfamily	71	None	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...	71; None

Angle bracket symbols in Binding/Active site

In biological databases like UniProt, you may encounter entries in the "Binding site" or "Active site" columns (or any other feature-related columns) that contain symbols like '<' or '>', these typically indicate positional uncertainty or boundaries that are outside the range of the sequence currently being annotated:

- '<': This symbol is used to indicate that the feature (such as a binding or active site) starts before the position given. For example, if you see "<5" in the context of a binding site, it suggests that the binding site starts before amino acid position 5 in the protein sequence.
- '>': Conversely, this symbol is used to show that the feature extends beyond the position given. If you see ">200" for an active site, it implies that the active site extends beyond amino acid position 200.

These annotations provide information about the location of certain functional sites within a protein, but with an acknowledgment of some level of uncertainty or incompleteness in the data that could be due to various reasons, such as limitations in experimental data, partial protein sequences, or predictions based on related proteins rather than direct evidence.

We will filter out entries containing these symbols so as to work with a dataset with certainty on the binding/active sites.

In [ ]:
```
# Find entries containing '<' or '>'
entries_angles = data['Binding-Active site'].str.contains('<|>', na=False)
print(f"Number of entries with angle brackets: {entries_angles.sum()}")

# Remove all rows where the "Binding-Active site" column contains '<' or '>'
data = data[~entries_angles]
print(f"Number of remaining rows: {data.shape[0]}")
```

```
Number of entries with angle brackets: 0
Number of remaining rows: 1406
```

Question mark ("?") symbols in Binding/Active site

In biological databases like UniProt, a question mark ("?") in the "Binding site" or "Active site" columns typically indicates uncertainty or incomplete information regarding the feature in question. It might mean the exact position of the binding or active site within the protein sequence may not be clearly determined, or it may be a predicted feature based on computational models or inferred from homologous proteins, but not yet experimentally verified. It can also be due to conflicting data or interpretations about the presence or characteristics of the site, or the annotation process just being incomplete.

In [ ]:
```
# Find rows where the "Binding-Active site" column contains the character "?", treating "?" as a literal charac
entries_question_mark = data[data['Binding-Active site'].str.contains('\?', na=False, regex=True)]
print(f"Number of entries with angle brackets: {entries_question_mark.shape[0]}")

# Remove all rows containing '?' in the "Binding-Active site" column
data = data.drop(entries_question_mark.index)
print(f"Number of remaining rows: {data.shape[0]}")
```

```
Number of entries with angle brackets: 0
Number of remaining rows: 1406
```

Binding/active sites labels

Now let's define all aminoacids involved in binding/active sites by expanding the ranges to especify all amino acid indexes that are a binding/active site:

In [ ]:
```
def expand_ranges(s):
    """Expand ranges into a comma-separated string."""
    return re.sub(r'(\d+)\.\.(\d+)', lambda m: ', '.join(map(str, range(int(m.group(1)), int(m.group(2))+1))), s

data['Binding-Active site'] = data['Binding-Active site'].apply(expand_ranges)
print(data.head())
```

```
         Entry            Protein families   Binding site Active site  \
359      Q504Y2  Protein kinase superfamily  144..152; 166         278
414      Q8IWB6  Protein kinase superfamily  233..241; 273        None
427      Q8NB16  Protein kinase superfamily  209..217; 230        None
778  A0A7P0T838  Protein kinase superfamily             71        None
779  A0A7P0T952  Protein kinase superfamily             71        None

                                          Sequence  \
359  MRRRRAAVAAGFCASFLLGSVLNVLFAPGSEPPRPGQSPEPSPAPG...
414  MSRAVRLPVPCPVQLGTLRNDSLEAQLHEYVKQGNYVKVKKILKKG...
427  MENLKHIITLGQVIHKRCEEMKYCKKQCRRLGHRVLGLIKPLEMLQ...
778  MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...
779  MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...

                                Binding-Active site
359  144, 145, 146, 147, 148, 149, 150, 151, 152; 1...
414  233, 234, 235, 236, 237, 238, 239, 240, 241; 2...
427  209, 210, 211, 212, 213, 214, 215, 216, 217; 2...
778                                         71; None
779                                         71; None
```

You can now convert the binding/active sites information into a binary label: 1s where there is a binding/active site; 0s where there is not. Retrieve the indices in 'Bindig/active site' column, and set their corresponding positions in the protein sequence to 1. All other aminoacids of the sequence are set to 0:

In [ ]:
```
def convert_to_binary_list(binding_active_str, sequence_len):
    """Convert a Binding-Active site string to a binary list based on the sequence length."""
    binary_list = [0] * sequence_len
    # Retrieve the indices in bindig/active sites and set their corresponding positions to 1
    if pd.notna(binding_active_str):
        indices = [int(x) - 1 for segment in binding_active_str.split(';') for x in segment.split(',') if x.str
        for idx in indices:
            if 0 <= idx < sequence_len: # Ensure the index is within the valid range
                binary_list[idx] = 1

    return binary_list
```

```
# Apply the function to both datasets
data['Binding-Active site'] = data.apply(lambda row: convert_to_binary_list(row['Binding-Active site'], len(row
data.head()
```

Out[ ]:

	Entry	Protein families	Binding site	Active site	Sequence	Binding-Active site
359	Q504Y2	Protein kinase superfamily	144..152; 166	278	MRRRRAAVAAGFCASFLLGSVLNVLFAPGSEPPRPGQSPEPSPAPG...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
414	Q8IWB6	Protein kinase superfamily	233..241; 273	None	MSRAVRLPVPCPVQLGTLRNDSLEAQLHEYVKQGNYVKVKKILKKG...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
427	Q8NB16	Protein kinase superfamily	209..217; 230	None	MENLKHIITLGQVIHKRCEEMKYCKKQCRRLGHRVLGLIKPLEMLQ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
778	A0A7P0T838	Protein kinase superfamily	71	None	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
779	A0A7P0T952	Protein kinase superfamily	71	None	MPRVKAAQAGRQSSAKRHLAEQFAVGEIITDMAKKEWKVGLPIGQG...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...

Split train/test sets

Let's create a split of the data into training and test sets based on UniProt protein families, such that it ensures entire protein families are either in the training set or the test set. The goal is that the test set will contain completely "new" families of proteins that are not seen in the training set, so the evaluation represents the model's ability to generalize to entirely new families of proteins that it has not seen during training.

Notably, this is different from the traiditional stratified split, which aims to preserve the distribution of classes across both sets.

In [ ]:
```
# Get the number of distinct protein families
num_families = data['Protein families'].nunique()
print(f"Number of distinct protein families: {num_families}")
```

```
Number of distinct protein families: 126
```

In [ ]:
```
def split_data_by_family(data, test_ratio=0.20):
    """
    Splits the dataset into train and test sets by entire protein families (not a family-stratified split!).

    Parameters:
    - data: pandas DataFrame containing the dataset with a 'Protein families' column.
    - test_ratio: float, the proportion of the dataset to include in the test split.

    Returns:
    - test_df: pandas DataFrame containing the test set.
    - train_df: pandas DataFrame containing the training set.
    """
    # Get unique protein families and shuffle them to randomize the selection
    unique_families = data['Protein families'].unique()
    np.random.shuffle(unique_families)

    # Loop through the shuffled families and add rows to the test set
    test_rows = []
    current_test_rows = 0
    for family in unique_families:
```

```
        family_rows = data[data['Protein families'] == family].index.tolist()
        if current_test_rows + len(family_rows) <= int(test_ratio * data.shape[0]):
            test_rows.extend(family_rows)
            current_test_rows += len(family_rows)
        else:
            # If adding the current family exceeds the target, stop adding
            test_rows.extend(family_rows)
            break

    # Create the test and train datasets
    train_rows = [i for i in data.index if i not in test_rows]
    test_df = data.loc[test_rows]
    train_df = data.loc[train_rows]

    return test_df, train_df

test_df, train_df = split_data_by_family(data, test_ratio=0.20)
print(test_df.shape[0], train_df.shape[0])
```

```
392 1014
```

In [ ]:
```
test_df.head()
```

Out[ ]:

	Entry	Protein families	Binding site	Active site	Sequence	B
39	O43252	APS kinase family; Sulfate adenylyltransferase...	62..67; 89..92; 101; 106..109; 132..133; 171; ...	None	MEIPGSLCKKVKLSNNAQNWGMQRATNVTYQAHHVSRNKRGQVVGT...	
68	O95340	APS kinase family; Sulfate adenylyltransferase...	52..57; 79..82; 91; 96..99; 122..123; 161; 174...	None	MSGIKKQKTENQQKSTNVVYQAHHVSRNKRGQVVGTRGGFRGCTVW...	
4	A0A2P9DU05	Protein kinase superfamily, AGC Ser/Thr protei...	121	214	MSRPPPTGKMPGAPETAPGDGAGASRQRKLEALIRDPRSPINVESL...	
12	O00141	Protein kinase superfamily, AGC Ser/Thr protei...	104..112; 127	222	MTVKTEAAKGTLTYSRMRGMVAILIAFMKQRRMGLNDFIQKIANNS...	
22	O14578	Protein kinase superfamily, AGC Ser/Thr protei...	103..111; 126	221	MLKFKYGARNPLDAGAAEPIASRASRLNLFFQGKPPFMTQQQMSPL...	

In case you don't want to keep the entire train/test datasets, you can create a smaller version (with a random representation of the original dataset). Uncomment the code below if that is the case:

In [ ]:
```
# # Percentage of data you want to keep
# k = 0.05  # for keeping 5% of the data

# # Generate random indices representing a percentage of each dataset
# train_df = train_df.sample(frac=k, random_state=42)
# test_df = test_df.sample(frac=k, random_state=42)
```

Split sequences into chunks

Sequences aren’t always of the same length. We will split the longer protein sequences and their lables into non-overlapping chunks of certain length or less to account for a given context window of ESM-2 models. Most protein sequences are on average 350 or so residues, so having longer context windows is often unnecessary, but keep in mind this will effect training time and batch size. Here, we pick a context of 1000.

In [ ]:
```python
def split_into_chunks(sequences, labels, chunk_size = 1000):
    """Split sequences and labels into chunks of size "chunk_size" or less."""
    new_sequences = []
    new_labels = []
    for seq, lbl in zip(sequences, labels):
        if len(seq) > chunk_size:
            # Split the sequence and labels into chunks of size "chunk_size" or less
            for i in range(0, len(seq), chunk_size):
                new_sequences.append(seq[i:i+chunk_size])
                new_labels.append(lbl[i:i+chunk_size])
        else:
            new_sequences.append(seq)
            new_labels.append(lbl)

    return new_sequences, new_labels
```

In [ ]:
```python
# Create lists of sequences and labels
test_seq = test_df['Sequence'].tolist()
test_labels = test_df['Binding-Active site'].tolist()
train_seq = train_df['Sequence'].tolist()
train_labels = train_df['Binding-Active site'].tolist()
```

In [ ]:
```python
# Apply the function to create new datasets with chunks of size "chunk_size" or less
chunk_size = 1000
test_seq_chunked, test_labels_chunked = split_into_chunks(test_seq, test_labels)
train_seq_chunked, train_labels_chunked = split_into_chunks(train_seq, train_labels)
```

The resulting train and test files will be exported to the same path where the input data file was located:

In [ ]:
```python
filename = os.path.splitext(os.path.basename(file_path))[0]
dir = os.path.dirname(file_path)

# Paths to save the new chunked pickle files
test_labels_path =  os.path.join(dir, filename + "_test_labels_chunked_" + str(chunk_size) + ".pkl")
test_seq_path = os.path.join(dir, filename + "_test_sequences_chunked_" + str(chunk_size) + ".pkl")
train_labels_path = os.path.join(dir, filename + "_train_labels_chunked_" + str(chunk_size) + ".pkl")
train_seq_path = os.path.join(dir, filename + "_train_sequences_chunked_" + str(chunk_size) + ".pkl")

# Save the chunked datasets as new pickle files
with open(test_labels_path, 'wb') as file:
    pickle.dump(test_labels_chunked, file)
with open(test_seq_path, 'wb') as file:
    pickle.dump(test_seq_chunked, file)
with open(train_labels_path, 'wb') as file:
    pickle.dump(train_labels_chunked, file)
with open(train_seq_path, 'wb') as file:
    pickle.dump(train_seq_chunked, file)

test_labels_path, test_seq_path, train_labels_path, train_seq_path
```

Out[ ]:
```
('/content/gdrive/MyDrive/ESMbind/data/uniprotkb_data_2024_05_29_test_labels_chunked_1000.pkl',
 '/content/gdrive/MyDrive/ESMbind/data/uniprotkb_data_2024_05_29_test_sequences_chunked_1000.pkl',
 '/content/gdrive/MyDrive/ESMbind/data/uniprotkb_data_2024_05_29_train_labels_chunked_1000.pkl',
 '/content/gdrive/MyDrive/ESMbind/data/uniprotkb_data_2024_05_29_train_sequences_chunked_1000.pkl')
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing this tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

```
@manual{Bioinformatics,
 title={UniProt data pre-processing for binding site prediction downstream task},
 organization={DeepChem},
 author={Gómez de Lope, Elisa},
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/UniProt_Data_Preprocessing

 year={2024},
}
```

Exploring Quantum Chemistry with GDB1k

Most of the tutorials we've walked you through so far have focused on applications to the drug discovery realm, but DeepChem's tool suite works for molecular design problems generally. In this tutorial, we're going to walk through an example of how to train a simple molecular machine learning for the task of predicting the atomization energy of a molecule. (Remember that the atomization energy is the energy required to form 1 mol of gaseous atoms from 1 mol of the molecule in its standard state under standard conditions).

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

With our setup in place, let's do a few standard imports to get the ball rolling.

In [1]:
```
import deepchem as dc
from sklearn.ensemble import RandomForestRegressor
from sklearn.kernel_ridge import KernelRidge
```

The ntext step we want to do is load our dataset. We're using a small dataset we've prepared that's pulled out of the larger GDB benchmarks. The dataset contains the atomization energies for 1K small molecules.

In [2]:
```
tasks = ["atomization_energy"]
dataset_file = "../../datasets/gdb1k.sdf"
smiles_field = "smiles"
mol_field = "mol"
```

We now need a way to transform molecules that is useful for prediction of atomization energy. This representation draws on foundational work [1] that represents a molecule's 3D electrostatic structure as a 2D matrix
C
of distances scaled by charges, where the
ij
-th element is represented by the following charge structure.

$$C_{ij} = \frac{q_i q_j}{r_{ij}^2}$$

If you're observing carefully, you might ask, wait doesn't this mean that molecules with different numbers of atoms generate matrices of different sizes? In practice the trick to get around this is that the matrices are "zero-padded." That is, if you're making coulomb matrices for a set of molecules, you pick a maximum number of atoms
N
, make the matrices
$N \times N$
and set to zero all the extra entries for this molecule. (There's a couple extra tricks that are done under the hood beyond this. Check out reference [1] or read the source code in DeepChem!)

DeepChem has a built in featurization class `dc.feat.CoulombMatrixEig` that can generate these featurizations for you.

In [3]:
```
featurizer = dc.feat.CoulombMatrixEig(23, remove_hydrogens=False)
```

Note that in this case, we set the maximum number of atoms to
$N = 23$
. Let's now load our dataset file into DeepChem. As in the previous tutorials, we use a `Loader` class, in particular `dc.data.SDFLoader` to load our `.sdf` file into DeepChem. The following snippet shows how we do this:

In [4]:
```
loader = dc.data.SDFLoader(
      tasks=["atomization_energy"],
      featurizer=featurizer)
dataset = loader.create_dataset(dataset_file)
```

```
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
RDKit WARNING: [17:25:11] Warning: molecule is tagged as 3D, but all Z coords are zero
/Users/peastman/workspace/deepchem/deepchem/feat/molecule_featurizers/coulomb_matrices.py:141: RuntimeWarning: d
ivide by zero encountered in true_divide
  m = np.outer(z, z) / d
```

For the purposes of this tutorial, we're going to do a random split of the dataset into training, validation, and test. In general, this split is weak and will considerably overestimate the accuracy of our models, but for now in this simple tutorial isn't a bad place to get started.

In [5]:
```
random_splitter = dc.splits.RandomSplitter()
train_dataset, valid_dataset, test_dataset = random_splitter.train_valid_test_split(dataset)
```

One issue that Coulomb matrix featurizations have is that the range of entries in the matrix
C
can be large. The charge
$q_1 q_2 /$
term can range very widely. In general, a wide range of values for inputs can throw off learning for the neural network. For this, a common fix is to normalize the input values so that they fall into a more standard range. Recall that the normalization transform applies to each feature
X_i
of datapoint
X

$\hat{X}_i$ =

where
μ_i
and
σ_i
are the mean and standard deviation of the
i
-th feature. This transformation enables the learning to proceed smoothly. A second point is that the atomization energies also fall across a wide range. So we apply an analogous transformation normalization transformation to the output to scale the energies better. We use DeepChem's transformation API to make this happen:

In [6]:
```
transformers = [
    dc.trans.NormalizationTransformer(transform_X=True, dataset=train_dataset),
    dc.trans.NormalizationTransformer(transform_y=True, dataset=train_dataset)]

for dataset in [train_dataset, valid_dataset, test_dataset]:
  for transformer in transformers:
      dataset = transformer.transform(dataset)
```

Now that we have the data cleanly transformed, let's do some simple machine learning. We'll start by constructing a random forest on top of the data. We'll use DeepChem's hyperparameter tuning module to do this.

In [7]:
```
def rf_model_builder(model_dir, **model_params):
  sklearn_model = RandomForestRegressor(**model_params)
  return dc.models.SklearnModel(sklearn_model, model_dir)
params_dict = {
    "n_estimators": [10, 100],
```

```
    "max_features": ["auto", "sqrt", "log2", None],
}

metric = dc.metrics.Metric(dc.metrics.mean_absolute_error)
optimizer = dc.hyper.GridHyperparamOpt(rf_model_builder)
best_rf, best_rf_hyperparams, all_rf_results = optimizer.hyperparam_search(
    params_dict, train_dataset, valid_dataset, output_transformers=transformers,
    metric=metric, use_max=False)
for key, value in all_rf_results.items():
    print(f'{key}: {value}')
print('Best hyperparams:', best_rf_hyperparams)
```

```
_max_featuresauto_n_estimators_10: 91166.92046422893
_max_featuressqrt_n_estimators_10: 90145.02789928475
_max_featureslog2_n_estimators_10: 85589.77206099383
_max_featuresNone_n_estimators_10: 86870.06019336461
_max_featuresauto_n_estimators_100: 86385.9006447343
_max_featuressqrt_n_estimators_100: 85051.76415912053
_max_featureslog2_n_estimators_100: 86443.79468510246
_max_featuresNone_n_estimators_100: 85464.79840440316
Best hyperparams: (100, 'sqrt')
```

Let's build one more model, a kernel ridge regression, on top of this raw data.

In [8]:
```
def krr_model_builder(model_dir, **model_params):
  sklearn_model = KernelRidge(**model_params)
  return dc.models.SklearnModel(sklearn_model, model_dir)

params_dict = {
    "kernel": ["laplacian"],
    "alpha": [0.0001],
    "gamma": [0.0001]
}

metric = dc.metrics.Metric(dc.metrics.mean_absolute_error)
optimizer = dc.hyper.GridHyperparamOpt(krr_model_builder)
best_krr, best_krr_hyperparams, all_krr_results = optimizer.hyperparam_search(
    params_dict, train_dataset, valid_dataset, output_transformers=transformers,
    metric=metric, use_max=False)
for key, value in all_krr_results.items():
    print(f'{key}: {value}')
print('Best hyperparams:', best_krr_hyperparams)
```

```
_alpha_0.000100_gamma_0.000100_kernellaplacian: 94056.64820129865
Best hyperparams: ('laplacian', 0.0001, 0.0001)
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Bibliography:

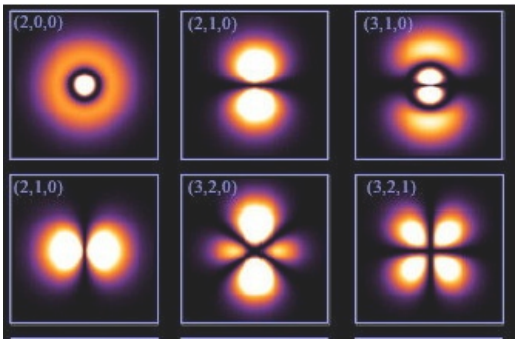
[1] https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.98.146401

DeepQMC tutorial

Background:

The electrons in a molecule are quantum mechanical in nature, meaning they do not follow classical physical laws. Quantum mechanics only gives a probability of where the electrons will be found and will not tell exactly where it can be found. This probability is given by the squared magnitude of a property of molecule system called the wavefunction and is different in each case of different molecule.

For many purposes, the nuclei of the atoms in a molecule can be considered to be stationary, and then we solve to get the wavefunction of the electrons. The probabilites when modelled into 3 dimensional space, takes the shape of the orbitals. Like shown in these images below which is taken via electron microscope.



(From the top left, oribtals in order - 2s, 2p(y),3p(y),2p(x),3d(z^2),3d(x^2-y^2))

Don't worry if you cannot remember or relate to the concept of orbitals, just remember that these are the space where electrons are found with more probability.

Using these wavefunctions, the electronic structure(a model containing electrons at its most probable positions) of a system can be obtained which can be used to calculate the energy at ground state. This value, then can be used to calculate various properties like ionization energy, electron affinity, etc.

The wavefunction of simple one electron systems like hydrogen atom, helium cation can be found easily, but for heavier atoms and molecules, electron-electron repulsion comes into act and makes it hard to compute the wavefunctions due to these interactions. Calculating these wavefunctions exactly will need a lot of computing resources and time which cannot be feasible to get them. Hence, other various different techniques for to approximate the wavefunction have been introduced, where there is a different tradeoff between speed and accuracy of the solution. One such method is the variational Monte Carlo which aims to include the effects of electron correlation in the solution without it.

Since Deep learning act as universal function approximators, it can be used to approximate wavefunction as well!! One such approach is the DNN based Variational Monte Carlo called paulinet. In this tutorial we will be looking into how to use Paulinet which is a part of an application called DeepQMC.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

Open in Colab

Setup:

First, we have download the DeepQMC application

In [ ]:
```
!pip install -U deepqmc[wf,train,cli]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting deepqmc[cli,train,wf]
  Downloading deepqmc-0.3.1.tar.gz (53 kB)
     |████████████████████████████████| 53 kB 1.9 MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Collecting uncertainties<4.0.0,>=3.1.2
  Downloading uncertainties-3.1.6-py2.py3-none-any.whl (98 kB)
     |████████████████████████████████| 98 kB 3.1 MB/s
Requirement already satisfied: numpy<2.0,>=1.16 in /usr/local/lib/python3.7/dist-packages (from deepqmc[cli,trai
```

```
n,wf]) (1.21.6)
Requirement already satisfied: torch<2.0,>=1.2 in /usr/local/lib/python3.7/dist-packages (from deepqmc[cli,train
,wf]) (1.11.0+cu113)
Collecting toml<0.11.0,>=0.10.0
  Downloading toml-0.10.2-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: tqdm<5,>=4 in /usr/local/lib/python3.7/dist-packages (from deepqmc[cli,train,wf])
(4.64.0)
Requirement already satisfied: tensorboard<3,>=2 in /usr/local/lib/python3.7/dist-packages (from deepqmc[cli,tra
in,wf]) (2.8.0)
Requirement already satisfied: Pillow<9,>=7 in /usr/local/lib/python3.7/dist-packages (from deepqmc[cli,train,wf
]) (7.1.2)
Requirement already satisfied: scipy<2,>=1 in /usr/local/lib/python3.7/dist-packages (from deepqmc[cli,train,wf]
) (1.4.1)
Collecting pyscf<3,>=2
  Downloading pyscf-2.0.1-cp37-cp37m-manylinux1_x86_64.whl (37.5 MB)
     |████████████████████████████████| 37.5 MB 1.2 MB/s
Collecting tomlkit<0.8,>=0.7
  Downloading tomlkit-0.7.2-py2.py3-none-any.whl (32 kB)
Collecting click<9,>=8
  Downloading click-8.1.3-py3-none-any.whl (96 kB)
     |████████████████████████████████| 96 kB 6.6 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from click<9,>=8->d
eepqmc[cli,train,wf]) (4.11.4)
Requirement already satisfied: h5py>=2.7 in /usr/local/lib/python3.7/dist-packages (from pyscf<3,>=2->deepqmc[cl
i,train,wf]) (3.1.0)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-packages (from h5py>=2.7->pyscf<
3,>=2->deepqmc[cli,train,wf]) (1.5.2)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2
->deepqmc[cli,train,wf]) (3.17.3)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from ten
sorboard<3,>=2->deepqmc[cli,train,wf]) (1.8.1)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3
,>=2->deepqmc[cli,train,wf]) (2.23.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard
<3,>=2->deepqmc[cli,train,wf]) (1.35.0)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,
>=2->deepqmc[cli,train,wf]) (57.4.0)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2-
>deepqmc[cli,train,wf]) (1.46.3)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>
=2->deepqmc[cli,train,wf]) (1.0.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from
tensorboard<3,>=2->deepqmc[cli,train,wf]) (0.4.6)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2->de
epqmc[cli,train,wf]) (0.37.1)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2
->deepqmc[cli,train,wf]) (3.3.7)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/python3.7/dist-packages (
from tensorboard<3,>=2->deepqmc[cli,train,wf]) (0.6.1)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-packages (from tensorboard<3,>=2->d
eepqmc[cli,train,wf]) (1.0.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from absl-py>=0.4->tensorboard<3,>
=2->deepqmc[cli,train,wf]) (1.15.0)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-aut
h<3,>=1.6.3->tensorboard<3,>=2->deepqmc[cli,train,wf]) (4.2.4)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<3,>=1.6
.3->tensorboard<3,>=2->deepqmc[cli,train,wf]) (4.8)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth
<3,>=1.6.3->tensorboard<3,>=2->deepqmc[cli,train,wf]) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-a
uth-oauthlib<0.5,>=0.4.1->tensorboard<3,>=2->deepqmc[cli,train,wf]) (1.3.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->cli
ck<9,>=8->deepqmc[cli,train,wf]) (3.8.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importli
b-metadata->click<9,>=8->deepqmc[cli,train,wf]) (4.2.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modul
es>=0.2.1->google-auth<3,>=1.6.3->tensorboard<3,>=2->deepqmc[cli,train,wf]) (0.4.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.
21.0->tensorboard<3,>=2->deepqmc[cli,train,wf]) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages
(from requests<3,>=2.21.0->tensorboard<3,>=2->deepqmc[cli,train,wf]) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0-
>tensorboard<3,>=2->deepqmc[cli,train,wf]) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2
.21.0->tensorboard<3,>=2->deepqmc[cli,train,wf]) (2022.5.18.1)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib
>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<3,>=2->deepqmc[cli,train,wf]) (3.2.0)
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from uncertainties<4.0.0,>=3.1.
2->deepqmc[cli,train,wf]) (0.16.0)
Building wheels for collected packages: deepqmc
  Building wheel for deepqmc (PEP 517) ... done
  Created wheel for deepqmc: filename=deepqmc-0.3.1-py3-none-any.whl size=65279 sha256=39c71efd64d51f518b7a6e5d8
30590ce9825ba25dd056c32339bc77f058762a1
```

```
  Stored in directory: /root/.cache/pip/wheels/d7/4b/e0/591aae685c3c53756116cacf763dd230a46c63dac2cd55b9d6
Successfully built deepqmc
Installing collected packages: uncertainties, toml, tomlkit, pyscf, deepqmc, click
  Attempting uninstall: click
    Found existing installation: click 7.1.2
    Uninstalling click-7.1.2:
      Successfully uninstalled click-7.1.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
flask 1.1.4 requires click<8.0,>=5.1, but you have click 8.1.3 which is incompatible.
Successfully installed click-8.1.3 deepqmc-0.3.1 pyscf-2.0.1 toml-0.10.2 tomlkit-0.7.2 uncertainties-3.1.6
```

The Hydrogen Molecule

In this section, let us look at a simple molecule to compute for. That is the Hydrogen molecule (2 electrons and 2 nucleus containg each 1 proton) and let us try varying the inter-nuclear distance and its effect on the total energy.

Then, we create our own custom 'Molecule dataset', which is the format in which DeepQMC accepts the parameters of the molecule system. It shoud contain the list of coordinates of each nucleon(coord), the list of number of protons in each nucleon(charges), the total ionic charge(charge) and the spin.

Since we are testing out each molecule with different distance, we can loop the Molecule with keeping one nuclei at the origin and varying the other nuclei x-axis position as given below. Also, the coordinates when given should be in the magnitude of bohr, where one bohr is equal to 0.52917721092 angstroms.

Then the molecule is loaded and given to train. Here we have modified a particular set of parameters in training to get a solution with reasonable accuracy in short time. Here we have altered n_steps(the number of steps in which electrons are sampled), batch_size(number of samples in a single step) and the epoch size(number of steps between sampling from wavefunction).

Now after training the model, we now will evaluate it, which means that the model will be run from first with the weights and biases of the neural network which one gets after the training. The result of this evaluation is given as \ uncertainity data type like '3.14±0.01', for the sake of graphing, we ignore the uncertainity and take the principal integer called the nominal value(i.e 3.14 in previous case). To do this, we use the 'uncertainities' library.

In [ ]:
```
angstroms_to_bohr=1/(0.52917721092)
```

In [ ]:
```
from deepqmc import train
from deepqmc import evaluate
from deepqmc.wf import PauliNet
from deepqmc import Molecule
from uncertainties import ufloat
import uncertainties


distances = [0.4, 0.5, 0.6, 0.7, 0.9, 1.1, 1.3, 1.5]
energies = []
for distance in distances:
    mol = Molecule(
        coords=[[0, 0, 0], [distance*angstroms_to_bohr, 0, 0]],
        charges=[1, 1],
        charge=0,
        spin=0,
    )
    net = PauliNet.from_hf(mol).cuda()
    train(net, n_steps=200, batch_size=200, epoch_size=50)
    molecule = evaluate(net)
    energies.append(uncertainties.nominal_value(molecule['energy']))
```

```
converged SCF energy = -0.937361804612381
Reducing cusp-correction cutoffs due to overlaps
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allocated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.06010365319165
Reducing cusp-correction cutoffs due to overlaps
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
```

```
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.11127760265224
Reducing cusp-correction cutoffs due to overlaps
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.12727489687867
Reducing cusp-correction cutoffs due to overlaps
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.11346928280952
Reducing cusp-correction cutoffs due to overlaps
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.07787156827504
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.03805266783613
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
converged SCF energy = -1.00006529201883
equilibrating: 0it [00:00, ?it/s]
training:   0%|          | 0/200 [00:00<?, ?it/s]
/usr/local/lib/python3.7/dist-packages/torch/cuda/memory.py:274: FutureWarning: torch.cuda.reset_max_memory_allo
cated now calls torch.cuda.reset_peak_memory_stats, which resets /all/ peak memory stats.
  FutureWarning)
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
sampling:   0%|          | 0/5 [00:00<?, ?it/s]
equilibrating: 0it [00:00, ?it/s]
```

Let's plot the results and draw some conclusions from it. The results of the evaluation happens to be a dictionary with key "energy" and contains the values of the type uncertainities and hence uncertainities library has been used to deal

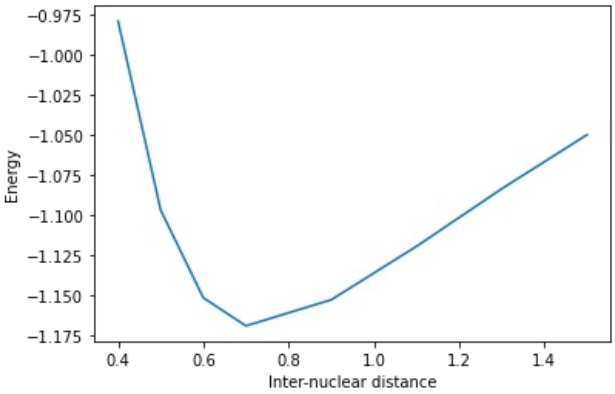
with

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

plt.ylabel("Energy")
plt.xlabel("Inter-nuclear distance")


xpoints = np.array([0.4,0.5,0.6,0.7,0.9,1.1,1.3,1.5])
ypoints = np.array(energies)
#nominal value refers to the principal value excluding the error

plt.plot(xpoints, ypoints)
plt.show()
```



Here we examined the stability of different hypothetical molecule by differing the coordinates of the nucleus of a molecule. The one with lowest ground state energy is the most stable one. The inter-nuclear distance so lies at approximately between 0.7 and 0.8 angstroms as it has a visible minima there.

As you can see here, this result has a lot of applications!! If you calculate the ground state energy for the hydrogen molecule cation (H2+), then difference in their energy gives you the ionization energy, same can be done if you calculate the hyrogen moleculet anion to calculate the electron affinity, all from doing simulations!! Also, via this method, molecular electronic structure can also be determined, which can be used to examine various properties like conductivity, optical and chemical nature. So, this helps in finding better materials for specific applications.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Introduction to Bioinformatics

So far in this tutorial, we've primarily worked on the problems of cheminformatics. We've been interested in seeing how we can use the techniques of machine learning to make predictions about the properties of molecules. In this tutorial, we're going to shift a bit and see how we can use classical computer science techniques and machine learning to tackle problems in bioinformatics.

For this, we're going to use the venerable biopython library to do some basic bioinformatics. A lot of the material in this notebook is adapted from the extensive official [Biopython tutorial]http://biopython.org/DIST/docs/tutorial/Tutorial.html). We strongly recommend checking out the official tutorial after you work through this notebook!

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following cell of installation commands. This will take about 5 minutes to run to completion and install your environment.

In [ ]:
```
!curl -Lo conda_installer.py https://raw.githubusercontent.com/deepchem/deepchem/master/scripts/colab_install.py
import conda_installer
conda_installer.install()
!/root/miniconda/bin/conda info -e
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  3457  100  3457    0     0  11516      0 --:--:-- --:--:-- --:--:-- 11523
add /root/miniconda/lib/python3.10/site-packages to PYTHONPATH
INFO:conda_installer:add /root/miniconda/lib/python3.10/site-packages to PYTHONPATH
python version: 3.10.12
INFO:conda_installer:python version: 3.10.12
fetching installer from https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
INFO:conda_installer:fetching installer from https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.
sh
done
INFO:conda_installer:done
installing miniconda to /root/miniconda
INFO:conda_installer:installing miniconda to /root/miniconda
done
INFO:conda_installer:done
installing openmm, pdbfixer
INFO:conda_installer:installing openmm, pdbfixer
added conda-forge to channels
INFO:conda_installer:added conda-forge to channels
done
INFO:conda_installer:done
conda packages installation finished!
INFO:conda_installer:conda packages installation finished!
# conda environments:
#
base                  /root/miniconda
```

In [2]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

```
Collecting deepchem
  Downloading deepchem-2.7.2.dev20230730200710-py3-none-any.whl (827 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 827.4/827.4 kB 7.3 MB/s eta 0:00:00
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.3.1)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.22.4)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.5.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.2.2)
Requirement already satisfied: scipy>=1.10.1 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.10.1)
Collecting rdkit (from deepchem)
  Downloading rdkit-2023.3.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (29.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 29.7/29.7 MB 46.3 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem) (
2022.7.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from rdkit->deepchem) (9.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n->deepchem) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1-
>pandas->deepchem) (1.16.0)
Installing collected packages: rdkit, deepchem
Successfully installed deepchem-2.7.2.dev20230730200710 rdkit-2023.3.2
WARNING:deepchem.feat.molecule_featurizers.rdkit_descriptors:No normalization for AvgIpc. Feature removed!
WARNING:deepchem.models.torch_models:Skipped loading modules with pytorch-geometric dependency, missing a depend
ency. No module named 'torch_geometric'
WARNING:deepchem.models.torch_models:Skipped loading modules with transformers dependency. No module named 'tran
sformers'
WARNING:deepchem.models:cannot import name 'HuggingFaceModel' from 'deepchem.models.torch_models' (/usr/local/li
b/python3.10/dist-packages/deepchem/models/torch_models/__init__.py)
WARNING:deepchem.models:Skipped loading modules with pytorch-geometric dependency, missing a dependency. cannot
import name 'DMPNN' from 'deepchem.models.torch_models' (/usr/local/lib/python3.10/dist-packages/deepchem/models
/torch_models/__init__.py)
WARNING:deepchem.models:Skipped loading modules with pytorch-lightning dependency, missing a dependency. No modu
le named 'pytorch_lightning'
WARNING:deepchem.models:Skipped loading some Jax models, missing a dependency. No module named 'haiku'
```

Out[2]:
```
'2.7.2.dev'
```

We'll use `pip` to install `biopython`

In [3]:
```
!pip install biopython
```

```
Collecting biopython
  Downloading biopython-1.81-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 12.1 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from biopython) (1.22.4)
Installing collected packages: biopython
Successfully installed biopython-1.81
```

In [4]:
```
import Bio
Bio.__version__
```

Out[4]:
```
'1.81'
```

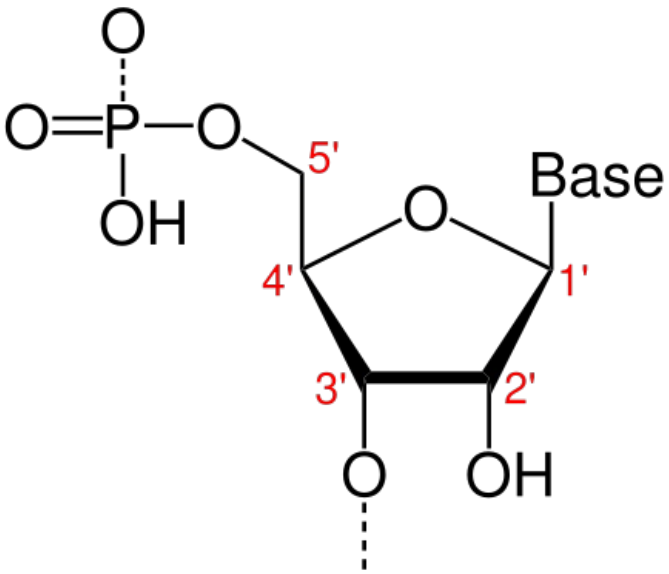
In [5]:
```
from Bio.Seq import Seq
my_seq = Seq("AGTACACATTG")
my_seq
```

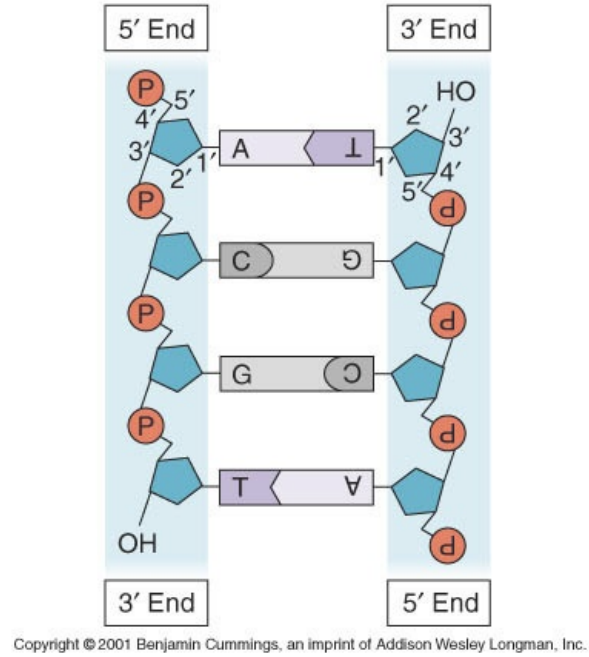
Out[5]:
```
Seq('AGTACACATTG')
```

The complement() method in Biopython's Seq object returns the complement of a DNA sequence. It replaces each base with its complement according to the Watson-Crick base pairing rules. Adenine (A) is complemented by thymine (T), and guanine (G) is complemented by cytosine (C).

The reverse_complement() method in Biopython's Seq object returns the reverse complement of a DNA sequence. It first reverses the sequence and then replaces each base with its complement according to the Watson-Crick base pairing rules.

But why is direction important? Many cellular processes occur only along a particular direction. To understand what gives a sense of directionality to a strand of DNA, take a look at the pictures below. Carbon atoms in the backbone of DNA are numbered from 1' to 5' (usually pronounced as "5 prime") in a clockwise direction. One might notice that the strand on the left has the 5' carbon above the 3' carbon in every nucleotide, resulting in a strand starting with a 5' end and ending with a 3' end. The strand on the right runs from the 3' end to the 5' end. As hinted earlier, reading of a DNA strand during replication and transcription only occurs from the 3' end to the 5' end.





In [6]:
```
my_seq.complement()
```

Out[6]:
```
Seq('TCATGTGTAAC')
```

In [7]:
```
my_seq.reverse_complement()
```

Out[7]:
```
Seq('CAATGTGTACT')
```

Parsing Sequence Records

We're going to download a sample `fasta` file from the Biopython tutorial to use in some exercises. This file is a set of hits for a sequence (of lady slipper orcid genes).It is basically the DNA sequences of a subfamily of plants belonging to Orchids(a diverse group of flowering plants) encoding the ribosomal RNA(rRNA).

In [8]:
```
!wget https://raw.githubusercontent.com/biopython/biopython/master/Doc/examples/ls_orchid.fasta
```

```
--2023-08-02 14:46:07--  https://raw.githubusercontent.com/biopython/biopython/master/Doc/examples/ls_orchid.fas
ta
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108
.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 76480 (75K) [text/plain]
Saving to: ‘ls_orchid.fasta’


ls_orchid.fasta       0%[                    ]       0  --.-KB/s
ls_orchid.fasta     100%[===================>]  74.69K  --.-KB/s    in 0.01s

2023-08-02 14:46:07 (5.24 MB/s) - ‘ls_orchid.fasta’ saved [76480/76480]
```

Let's take a look at what the contents of this file look like:

1. List item
2. List item

In [9]:
```
from Bio import SeqIO
for seq_record in SeqIO.parse('ls_orchid.fasta', 'fasta'):
    print(seq_record.id)
    print(repr(seq_record.seq))
    print(len(seq_record))
```

```
gi|2765658|emb|Z78533.1|CIZ78533
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGATGAGACCGTGG...CGC')
740
gi|2765657|emb|Z78532.1|CCZ78532
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAACAG...GGC')
753
gi|2765656|emb|Z78531.1|CFZ78531
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TAA')
748
gi|2765655|emb|Z78530.1|CMZ78530
```

```
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAAAACAACAT...CAT')
744
gi|2765654|emb|Z78529.1|CLZ78529
Seq('ACGGCGAGCTGCCGAAGGACATTGTTGAGACAGCAGAATATACGATTGAGTGAA...AAA')
733
gi|2765652|emb|Z78527.1|CYZ78527
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...CCC')
718
gi|2765651|emb|Z78526.1|CGZ78526
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...TGT')
730
gi|2765650|emb|Z78525.1|CAZ78525
Seq('TGTTGAGATAGCAGAATATACATCGAGTGAATCCGGAGGACCTGTGGTTATTCG...GCA')
704
gi|2765649|emb|Z78524.1|CFZ78524
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATAGTAG...AGC')
740
gi|2765648|emb|Z78523.1|CHZ78523
Seq('CGTAACCAGGTTTCCGTAGGTGAACCTGCGGCAGGATCATTGTTGAGACAGCAG...AAG')
709
gi|2765647|emb|Z78522.1|CMZ78522
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...GAG')
700
gi|2765646|emb|Z78521.1|CCZ78521
Seq('GTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAGAATATATGATCGAGT...ACC')
726
gi|2765645|emb|Z78520.1|CSZ78520
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGCAG...TTT')
753
gi|2765644|emb|Z78519.1|CPZ78519
Seq('ATATGATCGAGTGAATCTGGTGGACTTGTGGTTACTCAGCTCGCCATAGGCTTT...TTA')
699
gi|2765643|emb|Z78518.1|CRZ78518
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGGAGGATCATTGTTGAGATAGTAG...TCC')
658
gi|2765642|emb|Z78517.1|CFZ78517
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAG...AGC')
752
gi|2765641|emb|Z78516.1|CPZ78516
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACAGTAT...TAA')
726
gi|2765640|emb|Z78515.1|MXZ78515
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGCTGAGACCGTAG...AGC')
765
gi|2765639|emb|Z78514.1|PSZ78514
Seq('CGTAACAAGGTTTCCGTAGGTGGACCTTCGGGAGGATCATTTTTGAAGCCCCCA...CTA')
755
gi|2765638|emb|Z78513.1|PBZ78513
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCGCCA...GAG')
742
gi|2765637|emb|Z78512.1|PWZ78512
Seq('CGTAACAAGGTTTCCGTAGGTGGACCTTCGGGAGGATCATTTTTGAAGCCCCCA...AGC')
762
gi|2765636|emb|Z78511.1|PEZ78511
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTTCGGAAGGATCATTGTTGAGACCCCCA...GGA')
745
gi|2765635|emb|Z78510.1|PCZ78510
Seq('CTAACCAGGGTTCCGAGGTGACCTTCGGGAGGATTCCTTTTTAAGCCCCCGAAA...TTA')
750
gi|2765634|emb|Z78509.1|PPZ78509
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCGCCA...GGA')
731
gi|2765633|emb|Z78508.1|PLZ78508
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCGCCA...TGA')
741
gi|2765632|emb|Z78507.1|PLZ78507
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCCCCA...TGA')
740
gi|2765631|emb|Z78506.1|PLZ78506
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCGCAA...TGA')
727
gi|2765630|emb|Z78505.1|PSZ78505
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCGCCA...TTT')
711
gi|2765629|emb|Z78504.1|PKZ78504
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTTCGGAAGGATCATTGTTGAGACCGCAA...TAA')
743
gi|2765628|emb|Z78503.1|PCZ78503
Seq('CGTAACCAGGTTTCCGTAGGTGAACCTCCGGAAGGATCCTTGTTGAGACCGCCA...TAA')
727
gi|2765627|emb|Z78502.1|PBZ78502
Seq('CGTAACCAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGACCGCCA...CGC')
757
```

```
gi|2765626|emb|Z78501.1|PCZ78501
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGACCGCAA...AGA')
770
gi|2765625|emb|Z78500.1|PWZ78500
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGCTCATTGTTGAGACCGCAA...AAG')
767
gi|2765624|emb|Z78499.1|PMZ78499
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAGGGATCATTGTTGAGATCGCAT...ACC')
759
gi|2765623|emb|Z78498.1|PMZ78498
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAAGGTCATTGTTGAGATCACAT...AGC')
750
gi|2765622|emb|Z78497.1|PDZ78497
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGC')
788
gi|2765621|emb|Z78496.1|PAZ78496
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCGCAT...AGC')
774
gi|2765620|emb|Z78495.1|PEZ78495
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGATCACAT...GTG')
789
gi|2765619|emb|Z78494.1|PNZ78494
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGGTCGCAT...AAG')
688
gi|2765618|emb|Z78493.1|PGZ78493
Seq('CGTAACAAGGATTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCGCAT...CCC')
719
gi|2765617|emb|Z78492.1|PBZ78492
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCGCAT...ATA')
743
gi|2765616|emb|Z78491.1|PCZ78491
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCGCAT...AGC')
737
gi|2765615|emb|Z78490.1|PFZ78490
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGA')
728
gi|2765614|emb|Z78489.1|PDZ78489
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GGC')
740
gi|2765613|emb|Z78488.1|PTZ78488
Seq('CTGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACGCAATAATTGATCGA...GCT')
696
gi|2765612|emb|Z78487.1|PHZ78487
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TAA')
732
gi|2765611|emb|Z78486.1|PBZ78486
Seq('CGTCACGAGGTTTCCGTAGGTGAATCTGCGGGAGGATCATTGTTGAGATCACAT...TGA')
731
gi|2765610|emb|Z78485.1|PHZ78485
Seq('CTGAACCTGGTGTCCGAAGGTGAATCTGCGGATGGATCATTGTTGAGATATCAT...GTA')
735
gi|2765609|emb|Z78484.1|PCZ78484
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGGGGAAGGATCATTGTTGAGATCACAT...TTT')
720
gi|2765608|emb|Z78483.1|PVZ78483
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GCA')
740
gi|2765607|emb|Z78482.1|PEZ78482
Seq('TCTACTGCAGTGACCGAGATTTGCCATCGAGCCTCCTGGGAGCTTTCTTGCTGG...GCA')
629
gi|2765606|emb|Z78481.1|PIZ78481
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGA')
572
gi|2765605|emb|Z78480.1|PGZ78480
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGA')
587
gi|2765604|emb|Z78479.1|PPZ78479
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGT')
700
gi|2765603|emb|Z78478.1|PVZ78478
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCAGTGTTGAGATCACAT...GGC')
636
gi|2765602|emb|Z78477.1|PVZ78477
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGC')
716
gi|2765601|emb|Z78476.1|PGZ78476
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...CCC')
592
gi|2765600|emb|Z78475.1|PSZ78475
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GGT')
716
gi|2765599|emb|Z78474.1|PKZ78474
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACGT...CTT')
```

```
733
gi|2765598|emb|Z78473.1|PSZ78473
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGG')
626
gi|2765597|emb|Z78472.1|PLZ78472
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGC')
737
gi|2765596|emb|Z78471.1|PDZ78471
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGC')
740
gi|2765595|emb|Z78470.1|PPZ78470
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GTT')
574
gi|2765594|emb|Z78469.1|PHZ78469
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GTT')
594
gi|2765593|emb|Z78468.1|PAZ78468
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCGCAT...GTT')
610
gi|2765592|emb|Z78467.1|PSZ78467
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGA')
730
gi|2765591|emb|Z78466.1|PPZ78466
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...CCC')
641
gi|2765590|emb|Z78465.1|PRZ78465
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGC')
702
gi|2765589|emb|Z78464.1|PGZ78464
Seq('CGTAACAAGGTTTCCGTAGGTGAGCGGAAGGGTCATTGTTGAGATCACATAATA...AGC')
733
gi|2765588|emb|Z78463.1|PGZ78463
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGTTCATTGTTGAGATCACAT...AGC')
738
gi|2765587|emb|Z78462.1|PSZ78462
Seq('CGTCACGAGGTCTCCGGATGTGACCCTGCGGAAGGATCATTGTTGAGATCACAT...CAT')
736
gi|2765586|emb|Z78461.1|PWZ78461
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGATCACAT...TAA')
732
gi|2765585|emb|Z78460.1|PCZ78460
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGATCACAT...TTA')
745
gi|2765584|emb|Z78459.1|PDZ78459
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TTT')
744
gi|2765583|emb|Z78458.1|PHZ78458
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TTG')
738
gi|2765582|emb|Z78457.1|PCZ78457
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGATCACAT...GAG')
739
gi|2765581|emb|Z78456.1|PTZ78456
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGC')
740
gi|2765580|emb|Z78455.1|PJZ78455
Seq('CGTAACCAGGTTTCCGTAGGTGGACCTTCGGGAGGATCATTTTTGAGATCACAT...GCA')
745
gi|2765579|emb|Z78454.1|PFZ78454
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AAC')
695
gi|2765578|emb|Z78453.1|PSZ78453
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GCA')
745
gi|2765577|emb|Z78452.1|PBZ78452
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...GCA')
743
gi|2765576|emb|Z78451.1|PHZ78451
Seq('CGTAACAAGGTTTCCGTAGGTGTACCTCCGGAAGGATCATTGTTGAGATCACAT...AGC')
730
gi|2765575|emb|Z78450.1|PPZ78450
Seq('GGAAGGATCATTGCTGATATCACATAATAATTGATCGAGTTAAGCTGGAGGATC...GAG')
706
gi|2765574|emb|Z78449.1|PMZ78449
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGC')
744
gi|2765573|emb|Z78448.1|PAZ78448
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGG')
742
gi|2765572|emb|Z78447.1|PVZ78447
Seq('CGTAACAAGGATTCCGTAGGTGAACCTGCGGGAGGATCATTGTTGAGATCACAT...AGC')
694
gi|2765571|emb|Z78446.1|PAZ78446
```

```
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTCCGGAAGGATCATTGTTGAGATCACAT...CCC')
712
gi|2765570|emb|Z78445.1|PUZ78445
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...TGT')
715
gi|2765569|emb|Z78444.1|PAZ78444
Seq('CGTAACAAGGTTTCCGTAGGGTGAACTGCGGAAGGATCATTGTTGAGATCACAT...ATT')
688
gi|2765568|emb|Z78443.1|PLZ78443
Seq('CGTAACAAGGTTTCCGTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACAT...AGG')
784
gi|2765567|emb|Z78442.1|PBZ78442
Seq('GTAGGTGAACCTGCGGAAGGATCATTGTTGAGATCACATAATAATTGATCGAGT...AGT')
721
gi|2765566|emb|Z78441.1|PSZ78441
Seq('GGAAGGTCATTGCCGATATCACATAATAATTGATCGAGTTAATCTGGAGGATCT...GAG')
703
gi|2765565|emb|Z78440.1|PPZ78440
Seq('CGTAACAAGGTTTCCGTAGGTGGACCTCCGGGAGGATCATTGTTGAGATCACAT...GCA')
744
gi|2765564|emb|Z78439.1|PBZ78439
Seq('CATTGTTGAGATCACATAATAATTGATCGAGTTAATCTGGAGGATCTGTTTACT...GCC')
592
```

Sequence Objects

A large part of the biopython infrastructure deals with tools for handling sequences. These could be DNA sequences, RNA sequences, amino acid sequences or even more exotic constructs. Generally, 'Seq' object can be treated like a normal python string.

In [10]:
```
from Bio.Seq import Seq
my_seq = Seq("ACAGTAGAC")
print(my_seq)
my_seq
```

```
ACAGTAGAC
```

Out[10]:
```
Seq('ACAGTAGAC')
```

If we want to code a protein sequence, we can do that just as easily.

In [11]:
```
my_prot = Seq("AAAAA")
my_prot
```

Out[11]:
```
Seq('AAAAA')
```

We can take the length of sequences and index into them like strings.

In [12]:
```
print(len(my_prot))
```

```
5
```

In [13]:
```
my_prot[0]
```

Out[13]:
```
'A'
```

You can also use slice notation on sequences to get subsequences.

In [14]:
```
my_prot[0:3]
```

Out[14]:
```
Seq('AAA')
```

You can concatenate sequences if they have the same type so this works.

In [15]:
```
my_prot + my_prot
```

Out[15]:
```
Seq('AAAAAAAAAA')
```

Biopython automatically handles the concatenation as both sequences are of the generic alphabet.

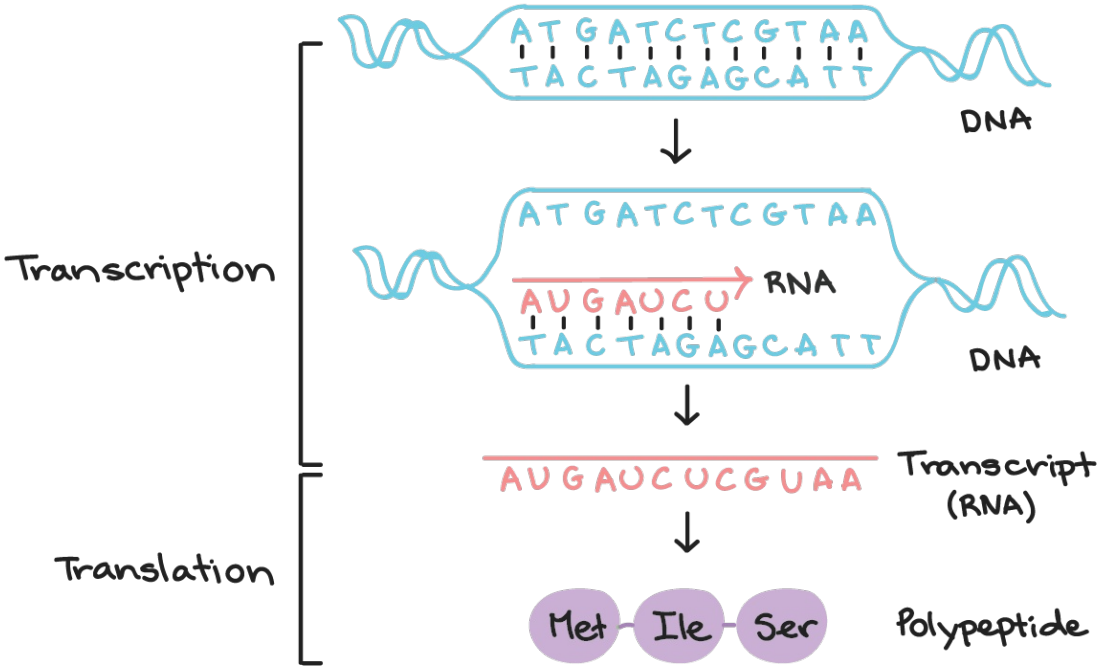
In [16]:
```
my_prot + my_seq
```

Out[16]:
```
Seq('AAAAAACAGTAGAC')
```

Transcription

Transcription is the process by which a DNA sequence is converted into messenger RNA. Remember that this is part of the "central dogma" of biology in which DNA engenders messenger RNA which engenders proteins. Here's a nice

representation of this cycle borrowed from a Khan academy lesson.



Note from the image above that DNA has two strands. The top strand is typically called the coding strand, and the bottom the template strand. The template strand is used for the actual transcription process of conversion into messenger RNA, but in bioinformatics, it's more common to work with the coding strand because this strand has the same sequence as the RNA transcript (except that RNA has uracil (U) instead of thymine (T)). Let's now see how we can execute a transcription computationally using Biopython.

In [17]:
```
from Bio.Seq import Seq
coding_dna = Seq("ATGATCTCGTAA")
print(coding_dna)
```

```
ATGATCTCGTAA
```

In [18]:
```
template_dna = coding_dna.reverse_complement()
template_dna
```

Out[18]:
```
Seq('TTACGAGATCAT')
```

Note that these sequences match those in the image below. You might be confused about why the `template_dna` sequence is shown reversed. The reason is that by convention, the template strand is read in the reverse direction.

Let's now see how we can transcribe our `coding_dna` strand into messenger RNA. This will only swap 'T' for 'U' and change the alphabet for our object.

In [19]:
```
messenger_rna = coding_dna.transcribe()
messenger_rna
```

Out[19]:
```
Seq('AUGAUCUCGUAA')
```

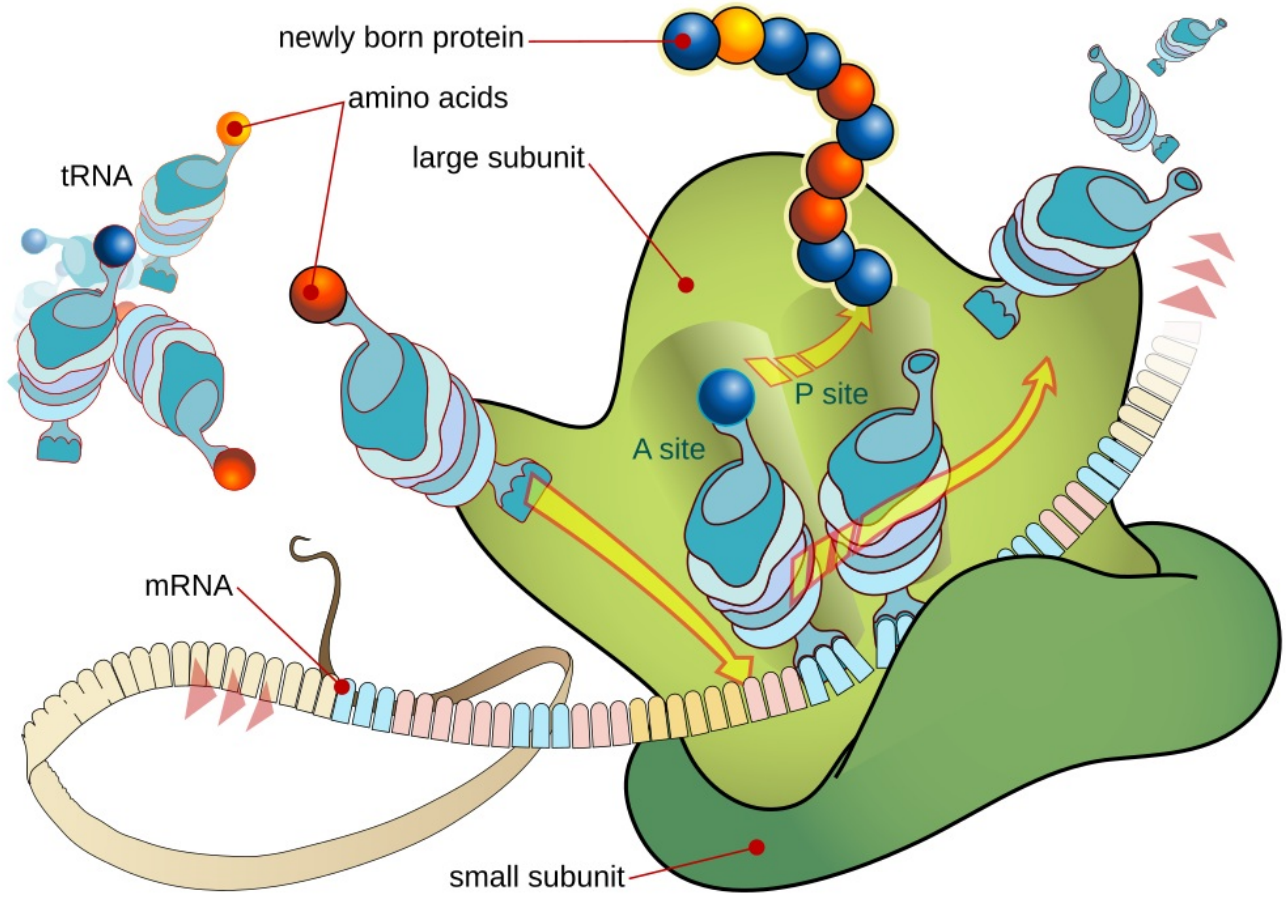
We can also perform a "back-transcription" to recover the original coding strand from the messenger RNA.

In [20]:
```
messenger_rna.back_transcribe()
```

Out[20]:
```
Seq('ATGATCTCGTAA')
```

Translation

Translation is the next step in the process, whereby a messenger RNA is transformed into a protein sequence. Here's a beautiful diagram from Wikipedia#/media/File:Ribosome_mRNA_translation_en.svg) that lays out the basics of this process.



Note how 3 nucleotides at a time correspond to one new amino acid added to the growing protein chain. A set of 3 nucleotides which codes for a given amino acid is called a "codon." We can use the `translate()` method on the messenger rna to perform this transformation in code.

messenger_rna.translate()

The translation can also be performed directly from the coding sequence DNA

In [21]:
```
coding_dna.translate()
```

Out[21]:
```
Seq('MIS*')
```

Let's now consider a longer genetic sequence that has some more interesting structure for us to look at.

In [22]:
```
coding_dna = Seq("ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
coding_dna.translate()
```

Out[22]:
```
Seq('MAIVMGR*KGAR*')
```

In both of the sequences above, '*' represents the stop codon. A stop codon is a sequence of 3 nucleotides that turns off the protein machinery. In DNA, the stop codons are 'TGA', 'TAA', 'TAG'. Note that this latest sequence has multiple stop codons. It's possible to run the machinery up to the first stop codon and pause too.

In [23]:
```
coding_dna.translate(to_stop=True)
```

Out[23]:
```
Seq('MAIVMGR')
```

We're going to introduce a bit of terminology here. A complete coding sequence CDS is a nucleotide sequence of messenger RNA which is made of a whole number of codons (that is, the length of the sequence is a multiple of 3), starts with a "start codon" and ends with a "stop codon". A start codon is basically the opposite of a stop codon and is mostly commonly the sequence "AUG", but can be different (especially if you're dealing with something like bacterial DNA).

Let's see how we can translate a complete CDS of bacterial messenger RNA.

In [24]:
```
from Bio.Seq import Seq
gene = Seq(
    "GTGAAAAAGATGCAATCTATCGTACTCGCACTTTCCCTGGTTCTGGTCGCTCCCATGGCA"
    "GCACAGGCTGCGGAAATTACGTTAGTCCCGTCAGTAAAATTACAGATAGGCGATCGTGAT"
    "AATCGTGGCTATTACTGGGATGGAGGTCACTGGCGCGACCACGGCTGGTGGAAACAACAT"
    "TATGAATGGCGAGGCAATCGCTGGCACCTACACGGACCGCCGCCACCGCCGCGCCACCAT"
    "AAGAAAGCTCCTCATGATCATCACGGCGGTCATGGTCCAGGCAAACATCACCGCTAA"
```

```
)
protein_sequence = gene.translate(table="Bacterial")
print(protein_sequence)
```

```
VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDHGWWKQHYEWRGNRWHLHGPPPPPRHHKKAPHDHHGGHGPGKHHR*
```

In [25]:
```
gene.translate(table="Bacterial", to_stop=True)
```

Out[25]:
```
Seq('VKKMQSIVLALSLVLVAPMAAQAAEITLVPSVKLQIGDRDNRGYYWDGGHWRDH...HHR')
```

Handling Annotated Sequences

Sometimes it will be useful for us to be able to handle annotated sequences where there's richer annotations, as in GenBank or EMBL files. For these purposes, we'll want to use the `SeqRecord` class.

In [26]:
```
from Bio.SeqRecord import SeqRecord
help(SeqRecord)
```

```
Help on class SeqRecord in module Bio.SeqRecord:

class SeqRecord(builtins.object)
 |  SeqRecord(seq, id='<unknown id>', name='<unknown name>', description='<unknown description>', dbxrefs=None,
features=None, annotations=None, letter_annotations=None)
 |
 |  A SeqRecord object holds a sequence and information about it.
 |
 |  Main attributes:
 |   - id          - Identifier such as a locus tag (string)
 |   - seq         - The sequence itself (Seq object or similar)
 |
 |  Additional attributes:
 |   - name        - Sequence name, e.g. gene name (string)
 |   - description - Additional text (string)
 |   - dbxrefs     - List of database cross references (list of strings)
 |   - features    - Any (sub)features defined (list of SeqFeature objects)
 |   - annotations - Further information about the whole sequence (dictionary).
 |     Most entries are strings, or lists of strings.
 |   - letter_annotations - Per letter/symbol annotation (restricted
 |     dictionary). This holds Python sequences (lists, strings
 |     or tuples) whose length matches that of the sequence.
 |     A typical use would be to hold a list of integers
 |     representing sequencing quality scores, or a string
 |     representing the secondary structure.
 |
 |  You will typically use Bio.SeqIO to read in sequences from files as
 |  SeqRecord objects.  However, you may want to create your own SeqRecord
 |  objects directly (see the __init__ method for further details):
 |
 |  >>> from Bio.Seq import Seq
 |  >>> from Bio.SeqRecord import SeqRecord
 |  >>> record = SeqRecord(Seq("MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF"),
 |  ...                    id="YP_025292.1", name="HokC",
 |  ...                    description="toxic membrane protein")
 |  >>> print(record)
 |  ID: YP_025292.1
 |  Name: HokC
 |  Description: toxic membrane protein
 |  Number of features: 0
 |  Seq('MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF')
 |
 |  If you want to save SeqRecord objects to a sequence file, use Bio.SeqIO
 |  for this.  For the special case where you want the SeqRecord turned into
 |  a string in a particular file format there is a format method which uses
 |  Bio.SeqIO internally:
 |
 |  >>> print(record.format("fasta"))
 |  >YP_025292.1 toxic membrane protein
 |  MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF
 |  <BLANKLINE>
 |
 |  You can also do things like slicing a SeqRecord, checking its length, etc
 |
 |  >>> len(record)
 |  44
 |  >>> edited = record[:10] + record[11:]
 |  >>> print(edited.seq)
 |  MKQHKAMIVAIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF
 |  >>> print(record.seq)
 |  MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF
 |
 |  Methods defined here:
```

```
|
|  __add__(self, other)
|      Add another sequence or string to this sequence.
|
|      The other sequence can be a SeqRecord object, a Seq object (or
|      similar, e.g. a MutableSeq) or a plain Python string. If you add
|      a plain string or a Seq (like) object, the new SeqRecord will simply
|      have this appended to the existing data. However, any per letter
|      annotation will be lost:
|
|      >>> from Bio import SeqIO
|      >>> record = SeqIO.read("Quality/solexa_faked.fastq", "fastq-solexa")
|      >>> print("%s %s" % (record.id, record.seq))
|      slxa_0001_1_0001_01 ACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNN
|      >>> print(list(record.letter_annotations))
|      ['solexa_quality']
|
|      >>> new = record + "ACT"
|      >>> print("%s %s" % (new.id, new.seq))
|      slxa_0001_1_0001_01 ACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNNACT
|      >>> print(list(new.letter_annotations))
|      []
|
|      The new record will attempt to combine the annotation, but for any
|      ambiguities (e.g. different names) it defaults to omitting that
|      annotation.
|
|      >>> from Bio import SeqIO
|      >>> with open("GenBank/pBAD30.gb") as handle:
|      ...     plasmid = SeqIO.read(handle, "gb")
|      >>> print("%s %i" % (plasmid.id, len(plasmid)))
|      pBAD30 4923
|
|      Now let's cut the plasmid into two pieces, and join them back up the
|      other way round (i.e. shift the starting point on this plasmid, have
|      a look at the annotated features in the original file to see why this
|      particular split point might make sense):
|
|      >>> left = plasmid[:3765]
|      >>> right = plasmid[3765:]
|      >>> new = right + left
|      >>> print("%s %i" % (new.id, len(new)))
|      pBAD30 4923
|      >>> str(new.seq) == str(right.seq + left.seq)
|      True
|      >>> len(new.features) == len(left.features) + len(right.features)
|      True
|
|      When we add the left and right SeqRecord objects, their annotation
|      is all consistent, so it is all conserved in the new SeqRecord:
|
|      >>> new.id == left.id == right.id == plasmid.id
|      True
|      >>> new.name == left.name == right.name == plasmid.name
|      True
|      >>> new.description == plasmid.description
|      True
|      >>> new.annotations == left.annotations == right.annotations
|      True
|      >>> new.letter_annotations == plasmid.letter_annotations
|      True
|      >>> new.dbxrefs == left.dbxrefs == right.dbxrefs
|      True
|
|      However, we should point out that when we sliced the SeqRecord,
|      any annotations dictionary or dbxrefs list entries were lost.
|      You can explicitly copy them like this:
|
|      >>> new.annotations = plasmid.annotations.copy()
|      >>> new.dbxrefs = plasmid.dbxrefs[:]
|
|  __bool__(self)
|      Boolean value of an instance of this class (True).
|
|      This behaviour is for backwards compatibility, since until the
|      __len__ method was added, a SeqRecord always evaluated as True.
|
|      Note that in comparison, a Seq object will evaluate to False if it
|      has a zero length sequence.
|
|      WARNING: The SeqRecord may in future evaluate to False when its
|      sequence is of zero length (in order to better match the Seq
|      object behaviour)!
```

```
|
|  __bytes__(self)
|
|  __contains__(self, char)
|      Implement the 'in' keyword, searches the sequence.
|
|      e.g.
|
|      >>> from Bio import SeqIO
|      >>> record = SeqIO.read("Fasta/sweetpea.nu", "fasta")
|      >>> "GAATTC" in record
|      False
|      >>> "AAA" in record
|      True
|
|      This essentially acts as a proxy for using "in" on the sequence:
|
|      >>> "GAATTC" in record.seq
|      False
|      >>> "AAA" in record.seq
|      True
|
|      Note that you can also use Seq objects as the query,
|
|      >>> from Bio.Seq import Seq
|      >>> Seq("AAA") in record
|      True
|
|      See also the Seq object's __contains__ method.
|
|  __eq__(self, other)
|      Define the equal-to operand (not implemented).
|
|  __format__(self, format_spec)
|      Return the record as a string in the specified file format.
|
|      This method supports the Python format() function and f-strings.
|      The format_spec should be a lower case string supported by
|      Bio.SeqIO as a text output file format. Requesting a binary file
|      format raises a ValueError. e.g.
|
|      >>> from Bio.Seq import Seq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> record = SeqRecord(Seq("MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF"),
|      ...                    id="YP_025292.1", name="HokC",
|      ...                    description="toxic membrane protein")
|      ...
|      >>> format(record, "fasta")
|      '>YP_025292.1 toxic membrane protein\nMKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF\n'
|      >>> print(f"Here is {record.id} in FASTA format:\n{record:fasta}")
|      Here is YP_025292.1 in FASTA format:
|      >YP_025292.1 toxic membrane protein
|      MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF
|      <BLANKLINE>
|
|      See also the SeqRecord's format() method.
|
|  __ge__(self, other)
|      Define the greater-than-or-equal-to operand (not implemented).
|
|  __getitem__(self, index)
|      Return a sub-sequence or an individual letter.
|
|      Slicing, e.g. my_record[5:10], returns a new SeqRecord for
|      that sub-sequence with some annotation preserved as follows:
|
|      * The name, id and description are kept as-is.
|      * Any per-letter-annotations are sliced to match the requested
|        sub-sequence.
|      * Unless a stride is used, all those features which fall fully
|        within the subsequence are included (with their locations
|        adjusted accordingly). If you want to preserve any truncated
|        features (e.g. GenBank/EMBL source features), you must
|        explicitly add them to the new SeqRecord yourself.
|      * With the exception of any molecule type, the annotations
|        dictionary and the dbxrefs list are not used for the new
|        SeqRecord, as in general they may not apply to the
|        subsequence. If you want to preserve them, you must explicitly
|        copy them to the new SeqRecord yourself.
|
|      Using an integer index, e.g. my_record[5] is shorthand for
|      extracting that letter from the sequence, my_record.seq[5].
|
```

```
|      For example, consider this short protein and its secondary
|      structure as encoded by the PDB (e.g. H for alpha helices),
|      plus a simple feature for its histidine self phosphorylation
|      site:
|
|      >>> from Bio.Seq import Seq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> from Bio.SeqFeature import SeqFeature, SimpleLocation
|      >>> rec = SeqRecord(Seq("MAAGVKQLADDRTLLMAGVSHDLRTPLTRIRLAT"
|      ...                     "EMMSEQDGYLAESINKDIEECNAIIEQFIDYLR"),
|      ...                 id="1JOY", name="EnvZ",
|      ...                 description="Homodimeric domain of EnvZ from E. coli")
|      >>> rec.letter_annotations["secondary_structure"] = "  S  SSSSSSHHHHHTTTHHHHHHHHHHHHHHHHHHHHHHHHTHHHHHHHHH
HHHHHHHHHHHHTT  "
|      >>> rec.features.append(SeqFeature(SimpleLocation(20, 21),
|      ...                     type = "Site"))
|
|      Now let's have a quick look at the full record,
|
|      >>> print(rec)
|      ID: 1JOY
|      Name: EnvZ
|      Description: Homodimeric domain of EnvZ from E. coli
|      Number of features: 1
|      Per letter annotation for: secondary_structure
|      Seq('MAAGVKQLADDRTLLMAGVSHDLRTPLTRIRLATEMMSEQDGYLAESINKDIEE...YLR')
|      >>> rec.letter_annotations["secondary_structure"]
|      '  S  SSSSSSHHHHHTTTHHHHHHHHHHHHHHHHHHHHHHHHTHHHHHHHHHHHHHHHHHHHHTT  '
|      >>> print(rec.features[0].location)
|      [20:21]
|
|      Now let's take a sub sequence, here chosen as the first (fractured)
|      alpha helix which includes the histidine phosphorylation site:
|
|      >>> sub = rec[11:41]
|      >>> print(sub)
|      ID: 1JOY
|      Name: EnvZ
|      Description: Homodimeric domain of EnvZ from E. coli
|      Number of features: 1
|      Per letter annotation for: secondary_structure
|      Seq('RTLLMAGVSHDLRTPLTRIRLATEMMSEQD')
|      >>> sub.letter_annotations["secondary_structure"]
|      'HHHHHTTTHHHHHHHHHHHHHHHHHHHHHH'
|      >>> print(sub.features[0].location)
|      [9:10]
|
|      You can also of course omit the start or end values, for
|      example to get the first ten letters only:
|
|      >>> print(rec[:10])
|      ID: 1JOY
|      Name: EnvZ
|      Description: Homodimeric domain of EnvZ from E. coli
|      Number of features: 0
|      Per letter annotation for: secondary_structure
|      Seq('MAAGVKQLAD')
|
|      Or for the last ten letters:
|
|      >>> print(rec[-10:])
|      ID: 1JOY
|      Name: EnvZ
|      Description: Homodimeric domain of EnvZ from E. coli
|      Number of features: 0
|      Per letter annotation for: secondary_structure
|      Seq('IIEQFIDYLR')
|
|      If you omit both, then you get a copy of the original record (although
|      lacking the annotations and dbxrefs):
|
|      >>> print(rec[:])
|      ID: 1JOY
|      Name: EnvZ
|      Description: Homodimeric domain of EnvZ from E. coli
|      Number of features: 1
|      Per letter annotation for: secondary_structure
|      Seq('MAAGVKQLADDRTLLMAGVSHDLRTPLTRIRLATEMMSEQDGYLAESINKDIEE...YLR')
|
|      Finally, indexing with a simple integer is shorthand for pulling out
|      that letter from the sequence directly:
|
|      >>> rec[5]
```

```
 |      'K'
 |      >>> rec.seq[5]
 |      'K'
 |
 |  __gt__(self, other)
 |      Define the greater-than operand (not implemented).
 |
 |  __init__(self, seq, id='<unknown id>', name='<unknown name>', description='<unknown description>', dbxrefs=N
one, features=None, annotations=None, letter_annotations=None)
 |      Create a SeqRecord.
 |
 |      Arguments:
 |       - seq         - Sequence, required (Seq or MutableSeq)
 |       - id          - Sequence identifier, recommended (string)
 |       - name        - Sequence name, optional (string)
 |       - description - Sequence description, optional (string)
 |       - dbxrefs     - Database cross references, optional (list of strings)
 |       - features    - Any (sub)features, optional (list of SeqFeature objects)
 |       - annotations - Dictionary of annotations for the whole sequence
 |       - letter_annotations - Dictionary of per-letter-annotations, values
 |         should be strings, list or tuples of the same length as the full
 |         sequence.
 |
 |      You will typically use Bio.SeqIO to read in sequences from files as
 |      SeqRecord objects.  However, you may want to create your own SeqRecord
 |      objects directly.
 |
 |      Note that while an id is optional, we strongly recommend you supply a
 |      unique id string for each record.  This is especially important
 |      if you wish to write your sequences to a file.
 |
 |      You can create a 'blank' SeqRecord object, and then populate the
 |      attributes later.
 |
 |  __iter__(self)
 |      Iterate over the letters in the sequence.
 |
 |      For example, using Bio.SeqIO to read in a protein FASTA file:
 |
 |      >>> from Bio import SeqIO
 |      >>> record = SeqIO.read("Fasta/loveliesbleeding.pro", "fasta")
 |      >>> for amino in record:
 |      ...     print(amino)
 |      ...     if amino == "L": break
 |      X
 |      A
 |      G
 |      L
 |      >>> print(record.seq[3])
 |      L
 |
 |      This is just a shortcut for iterating over the sequence directly:
 |
 |      >>> for amino in record.seq:
 |      ...     print(amino)
 |      ...     if amino == "L": break
 |      X
 |      A
 |      G
 |      L
 |      >>> print(record.seq[3])
 |      L
 |
 |      Note that this does not facilitate iteration together with any
 |      per-letter-annotation.  However, you can achieve that using the
 |      python zip function on the record (or its sequence) and the relevant
 |      per-letter-annotation:
 |
 |      >>> from Bio import SeqIO
 |      >>> rec = SeqIO.read("Quality/solexa_faked.fastq", "fastq-solexa")
 |      >>> print("%s %s" % (rec.id, rec.seq))
 |      slxa_0001_1_0001_01 ACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNN
 |      >>> print(list(rec.letter_annotations))
 |      ['solexa_quality']
 |      >>> for nuc, qual in zip(rec, rec.letter_annotations["solexa_quality"]):
 |      ...     if qual > 35:
 |      ...         print("%s %i" % (nuc, qual))
 |      A 40
 |      C 39
 |      G 38
 |      T 37
 |      A 36
 |
```

```
 |      You may agree that using zip(rec.seq, ...) is more explicit than using
 |      zip(rec, ...) as shown above.
 |
 |  __le__(self, other)
 |      Define the less-than-or-equal-to operand (not implemented).
 |
 |  __len__(self)
 |      Return the length of the sequence.
 |
 |      For example, using Bio.SeqIO to read in a FASTA nucleotide file:
 |
 |      >>> from Bio import SeqIO
 |      >>> record = SeqIO.read("Fasta/sweetpea.nu", "fasta")
 |      >>> len(record)
 |      309
 |      >>> len(record.seq)
 |      309
 |
 |  __lt__(self, other)
 |      Define the less-than operand (not implemented).
 |
 |  __ne__(self, other)
 |      Define the not-equal-to operand (not implemented).
 |
 |  __radd__(self, other)
 |      Add another sequence or string to this sequence (from the left).
 |
 |      This method handles adding a Seq object (or similar, e.g. MutableSeq)
 |      or a plain Python string (on the left) to a SeqRecord (on the right).
 |      See the __add__ method for more details, but for example:
 |
 |      >>> from Bio import SeqIO
 |      >>> record = SeqIO.read("Quality/solexa_faked.fastq", "fastq-solexa")
 |      >>> print("%s %s" % (record.id, record.seq))
 |      slxa_0001_1_0001_01 ACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNN
 |      >>> print(list(record.letter_annotations))
 |      ['solexa_quality']
 |
 |      >>> new = "ACT" + record
 |      >>> print("%s %s" % (new.id, new.seq))
 |      slxa_0001_1_0001_01 ACTACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNN
 |      >>> print(list(new.letter_annotations))
 |      []
 |
 |  __repr__(self)
 |      Return a concise summary of the record for debugging (string).
 |
 |      The python built in function repr works by calling the object's __repr__
 |      method.  e.g.
 |
 |      >>> from Bio.Seq import Seq
 |      >>> from Bio.SeqRecord import SeqRecord
 |      >>> rec = SeqRecord(Seq("MASRGVNKVILVGNLGQDPEVRYMPNGGAVANITLATSESWRDKAT"
 |      ...                     "GEMKEQTEWHRVVLFGKLAEVASEYLRKGSQVYIEGQLRTRKWTDQ"
 |      ...                     "SGQDRYTTEVVVNVGGTMQMLGGRQGGGAPAGGNIGGGQPQGGWGQ"
 |      ...                     "PQQPQGGNQFSGGAQSRPQQSAPAAPSNEPPMDFDDDIPF"),
 |      ...                 id="NP_418483.1", name="b4059",
 |      ...                 description="ssDNA-binding protein",
 |      ...                 dbxrefs=["ASAP:13298", "GI:16131885", "GeneID:948570"])
 |      >>> print(repr(rec))
 |      SeqRecord(seq=Seq('MASRGVNKVILVGNLGQDPEVRYMPNGGAVANITLATSESWRDKATGEMKEQTE...IPF'), id='NP_418483.1', nam
e='b4059', description='ssDNA-binding protein', dbxrefs=['ASAP:13298', 'GI:16131885', 'GeneID:948570'])
 |
 |      At the python prompt you can also use this shorthand:
 |
 |      >>> rec
 |      SeqRecord(seq=Seq('MASRGVNKVILVGNLGQDPEVRYMPNGGAVANITLATSESWRDKATGEMKEQTE...IPF'), id='NP_418483.1', nam
e='b4059', description='ssDNA-binding protein', dbxrefs=['ASAP:13298', 'GI:16131885', 'GeneID:948570'])
 |
 |      Note that long sequences are shown truncated. Also note that any
 |      annotations, letter_annotations and features are not shown (as they
 |      would lead to a very long string).
 |
 |  __str__(self)
 |      Return a human readable summary of the record and its annotation (string).
 |
 |      The python built in function str works by calling the object's __str__
 |      method.  e.g.
 |
 |      >>> from Bio.Seq import Seq
 |      >>> from Bio.SeqRecord import SeqRecord
 |      >>> record = SeqRecord(Seq("MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF"),
 |      ...                    id="YP_025292.1", name="HokC",
```

```
|      ...                   description="toxic membrane protein, small")
|      >>> print(str(record))
|      ID: YP_025292.1
|      Name: HokC
|      Description: toxic membrane protein, small
|      Number of features: 0
|      Seq('MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF')
|
|      In this example you don't actually need to call str explicitly, as the
|      print command does this automatically:
|
|      >>> print(record)
|      ID: YP_025292.1
|      Name: HokC
|      Description: toxic membrane protein, small
|      Number of features: 0
|      Seq('MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF')
|
|      Note that long sequences are shown truncated.
|
|  count(self, sub, start=None, end=None)
|      Return the number of non-overlapping occurrences of sub in seq[start:end].
|
|      Optional arguments start and end are interpreted as in slice notation.
|      This method behaves as the count method of Python strings.
|
|  format(self, format)
|      Return the record as a string in the specified file format.
|
|      The format should be a lower case string supported as an output
|      format by Bio.SeqIO, which is used to turn the SeqRecord into a
|      string.  e.g.
|
|      >>> from Bio.Seq import Seq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> record = SeqRecord(Seq("MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF"),
|      ...                   id="YP_025292.1", name="HokC",
|      ...                   description="toxic membrane protein")
|      >>> record.format("fasta")
|      '>YP_025292.1 toxic membrane protein\nMKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF\n'
|      >>> print(record.format("fasta"))
|      >YP_025292.1 toxic membrane protein
|      MKQHKAMIVALIVICITAVVAALVTRKDLCEVHIRTGQTEVAVF
|      <BLANKLINE>
|
|      The Python print function automatically appends a new line, meaning
|      in this example a blank line is shown.  If you look at the string
|      representation you can see there is a trailing new line (shown as
|      slash n) which is important when writing to a file or if
|      concatenating multiple sequence strings together.
|
|      Note that this method will NOT work on every possible file format
|      supported by Bio.SeqIO (e.g. some are for multiple sequences only,
|      and binary formats are not supported).
|
|  islower(self)
|      Return True if all ASCII characters in the record's sequence are lowercase.
|
|      If there are no cased characters, the method returns False.
|
|  isupper(self)
|      Return True if all ASCII characters in the record's sequence are uppercase.
|
|      If there are no cased characters, the method returns False.
|
|  lower(self)
|      Return a copy of the record with a lower case sequence.
|
|      All the annotation is preserved unchanged. e.g.
|
|      >>> from Bio import SeqIO
|      >>> record = SeqIO.read("Fasta/aster.pro", "fasta")
|      >>> print(record.format("fasta"))
|      >gi|3298468|dbj|BAA31520.1| SAMIPF
|      GGHVNPAVTFGAFVGGNITLLRGIVYIIAQLLGSTVACLLLKFVTNDMAVGVFSLSAGVG
|      VTNALVFEIVMTFGLVYTVYATAIDPKKGSLGTIAPIAIGFIVGANI
|      <BLANKLINE>
|      >>> print(record.lower().format("fasta"))
|      >gi|3298468|dbj|BAA31520.1| SAMIPF
|      gghvnpavtfgafvggnitllrgivyiiaqllgstvacllkfvtndmavgvfslsagvg
|      vtnalvfeivmtfglvytvyataidpkkgslgtiapiaigfivgani
|      <BLANKLINE>
|
```

```
 |      To take a more annotation rich example,
 |
 |      >>> from Bio import SeqIO
 |      >>> old = SeqIO.read("EMBL/TRBG361.embl", "embl")
 |      >>> len(old.features)
 |      3
 |      >>> new = old.lower()
 |      >>> len(old.features) == len(new.features)
 |      True
 |      >>> old.annotations["organism"] == new.annotations["organism"]
 |      True
 |      >>> old.dbxrefs == new.dbxrefs
 |      True
 |
 |  reverse_complement(self, id=False, name=False, description=False, features=True, annotations=False, letter_a
nnotations=True, dbxrefs=False)
 |      Return new SeqRecord with reverse complement sequence.
 |
 |      By default the new record does NOT preserve the sequence identifier,
 |      name, description, general annotation or database cross-references -
 |      these are unlikely to apply to the reversed sequence.
 |
 |      You can specify the returned record's id, name and description as
 |      strings, or True to keep that of the parent, or False for a default.
 |
 |      You can specify the returned record's features with a list of
 |      SeqFeature objects, or True to keep that of the parent, or False to
 |      omit them. The default is to keep the original features (with the
 |      strand and locations adjusted).
 |
 |      You can also specify both the returned record's annotations and
 |      letter_annotations as dictionaries, True to keep that of the parent,
 |      or False to omit them. The default is to keep the original
 |      annotations (with the letter annotations reversed).
 |
 |      To show what happens to the pre-letter annotations, consider an
 |      example Solexa variant FASTQ file with a single entry, which we'll
 |      read in as a SeqRecord:
 |
 |      >>> from Bio import SeqIO
 |      >>> record = SeqIO.read("Quality/solexa_faked.fastq", "fastq-solexa")
 |      >>> print("%s %s" % (record.id, record.seq))
 |      slxa_0001_1_0001_01 ACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNN
 |      >>> print(list(record.letter_annotations))
 |      ['solexa_quality']
 |      >>> print(record.letter_annotations["solexa_quality"])
 |      [40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15,
14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5]
 |
 |      Now take the reverse complement, here we explicitly give a new
 |      identifier (the old identifier with a suffix):
 |
 |      >>> rc_record = record.reverse_complement(id=record.id + "_rc")
 |      >>> print("%s %s" % (rc_record.id, rc_record.seq))
 |      slxa_0001_1_0001_01_rc NNNNNNACGTACGTACGTACGTACGTACGTACGTACGTACGT
 |
 |      Notice that the per-letter-annotations have also been reversed,
 |      although this may not be appropriate for all cases.
 |
 |      >>> print(rc_record.letter_annotations["solexa_quality"])
 |      [-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 2
3, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40]
 |
 |      Now for the features, we need a different example. Parsing a GenBank
 |      file is probably the easiest way to get an nice example with features
 |      in it...
 |
 |      >>> from Bio import SeqIO
 |      >>> with open("GenBank/pBAD30.gb") as handle:
 |      ...     plasmid = SeqIO.read(handle, "gb")
 |      >>> print("%s %i" % (plasmid.id, len(plasmid)))
 |      pBAD30 4923
 |      >>> plasmid.seq
 |      Seq('GCTAGCGGAGTGTATACTGGCTTACTATGTTGGCACTGATGAGGGTGTCAGTGA...ATG')
 |      >>> len(plasmid.features)
 |      13
 |
 |      Now, let's take the reverse complement of this whole plasmid:
 |
 |      >>> rc_plasmid = plasmid.reverse_complement(id=plasmid.id+"_rc")
 |      >>> print("%s %i" % (rc_plasmid.id, len(rc_plasmid)))
 |      pBAD30_rc 4923
 |      >>> rc_plasmid.seq
```

```
|      Seq('CATGGGCAAATATTATACGCAAGGCGACAAGGTGCTGATGCCGCTGGCGATTCA...AGC')
|      >>> len(rc_plasmid.features)
|      13
|
|      Let's compare the first CDS feature - it has gone from being the
|      second feature (index 1) to the second last feature (index -2), its
|      strand has changed, and the location switched round.
|
|      >>> print(plasmid.features[1])
|      type: CDS
|      location: [1081:1960](-)
|      qualifiers:
|          Key: label, Value: ['araC']
|          Key: note, Value: ['araC regulator of the arabinose BAD promoter']
|          Key: vntifkey, Value: ['4']
|      <BLANKLINE>
|      >>> print(rc_plasmid.features[-2])
|      type: CDS
|      location: [2963:3842](+)
|      qualifiers:
|          Key: label, Value: ['araC']
|          Key: note, Value: ['araC regulator of the arabinose BAD promoter']
|          Key: vntifkey, Value: ['4']
|      <BLANKLINE>
|
|      You can check this new location, based on the length of the plasmid:
|
|      >>> len(plasmid) - 1081
|      3842
|      >>> len(plasmid) - 1960
|      2963
|
|      Note that if the SeqFeature annotation includes any strand specific
|      information (e.g. base changes for a SNP), this information is not
|      amended, and would need correction after the reverse complement.
|
|      Note trying to reverse complement a protein SeqRecord raises an
|      exception:
|
|      >>> from Bio.Seq import Seq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> protein_rec = SeqRecord(Seq("MAIVMGR"), id="Test",
|      ...                         annotations={"molecule_type": "protein"})
|      >>> protein_rec.reverse_complement()
|      Traceback (most recent call last):
|         ...
|      ValueError: Proteins do not have complements!
|
|      If you have RNA without any U bases, it must be annotated as RNA
|      otherwise it will be treated as DNA by default with A mapped to T:
|
|      >>> from Bio.Seq import Seq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> rna1 = SeqRecord(Seq("ACG"), id="Test")
|      >>> rna2 = SeqRecord(Seq("ACG"), id="Test", annotations={"molecule_type": "RNA"})
|      >>> print(rna1.reverse_complement(id="RC", description="unk").format("fasta"))
|      >RC unk
|      CGT
|      <BLANKLINE>
|      >>> print(rna2.reverse_complement(id="RC", description="RNA").format("fasta"))
|      >RC RNA
|      CGU
|      <BLANKLINE>
|
|      Also note you can reverse complement a SeqRecord using a MutableSeq:
|
|      >>> from Bio.Seq import MutableSeq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> rec = SeqRecord(MutableSeq("ACGT"), id="Test")
|      >>> rec.seq[0] = "T"
|      >>> print("%s %s" % (rec.id, rec.seq))
|      Test TCGT
|      >>> rc = rec.reverse_complement(id=True)
|      >>> print("%s %s" % (rc.id, rc.seq))
|      Test ACGA
|
|  translate(self, table='Standard', stop_symbol='*', to_stop=False, cds=False, gap=None, id=False, name=False,
description=False, features=False, annotations=False, letter_annotations=False, dbxrefs=False)
|      Return new SeqRecord with translated sequence.
|
|      This calls the record's .seq.translate() method (which describes
|      the translation related arguments, like table for the genetic code),
|
```

```
|      By default the new record does NOT preserve the sequence identifier,
|      name, description, general annotation or database cross-references -
|      these are unlikely to apply to the translated sequence.
|
|      You can specify the returned record's id, name and description as
|      strings, or True to keep that of the parent, or False for a default.
|
|      You can specify the returned record's features with a list of
|      SeqFeature objects, or False (default) to omit them.
|
|      You can also specify both the returned record's annotations and
|      letter_annotations as dictionaries, True to keep that of the parent
|      (annotations only), or False (default) to omit them.
|
|      e.g. Loading a FASTA gene and translating it,
|
|      >>> from Bio import SeqIO
|      >>> gene_record = SeqIO.read("Fasta/sweetpea.nu", "fasta")
|      >>> print(gene_record.format("fasta"))
|      >gi|3176602|gb|U78617.1|LOU78617 Lathyrus odoratus phytochrome A (PHYA) gene, partial cds
|      CAGGCTGCGCGGTTTCTATTTATGAAGAACAAGGTCCGTATGATAGTTGATTGTCATGCA
|      AAACATGTGAAGGTTCTTCAAGACGAAAAACTCCCATTTGATTTGACTCTGTGCGGTTCG
|      ACCTTAAGAGCTCCACATAGTTGCCATTTGCAGTACATGGCTAACATGGATTCAATTGCT
|      TCATTGGTTATGGCAGTGGTCGTCAATGACAGCGATGAAGATGGAGATAGCCGTGACGCA
|      GTTCTACCACAAAAGAAAAAGAGACTTTGGGGTTTGGTAGTTTGTCATAACACTACTCCG
|      AGGTTTGTT
|      <BLANKLINE>
|
|      And now translating the record, specifying the new ID and description:
|
|      >>> protein_record = gene_record.translate(table=11,
|      ...                                         id="phya",
|      ...                                         description="translation")
|      >>> print(protein_record.format("fasta"))
|      >phya translation
|      QAARFLFMKNKVRMIVDCHAKHVKVLQDEKLPFDLTLCGSTLRAPHSCHLQYMANMDSIA
|      SLVMAVVVNDSDEDGDSRDAVLPQKKKRLWGLVVCHNTTPRFV
|      <BLANKLINE>
|
|  upper(self)
|      Return a copy of the record with an upper case sequence.
|
|      All the annotation is preserved unchanged. e.g.
|
|      >>> from Bio.Seq import Seq
|      >>> from Bio.SeqRecord import SeqRecord
|      >>> record = SeqRecord(Seq("acgtACGT"), id="Test",
|      ...                   description = "Made up for this example")
|      >>> record.letter_annotations["phred_quality"] = [1, 2, 3, 4, 5, 6, 7, 8]
|      >>> print(record.upper().format("fastq"))
|      @Test Made up for this example
|      ACGTACGT
|      +
|      "#$%&'()
|      <BLANKLINE>
|
|      Naturally, there is a matching lower method:
|
|      >>> print(record.lower().format("fastq"))
|      @Test Made up for this example
|      acgtacgt
|      +
|      "#$%&'()
|      <BLANKLINE>
|
|  ----------------------------------------------------------------------
|  Data descriptors defined here:
|
|  __dict__
|      dictionary for instance variables (if defined)
|
|  __weakref__
|      list of weak references to the object (if defined)
|
|  letter_annotations
|      Dictionary of per-letter-annotation for the sequence.
|
|      For example, this can hold quality scores used in FASTQ or QUAL files.
|      Consider this example using Bio.SeqIO to read in an example Solexa
|      variant FASTQ file as a SeqRecord:
|
|      >>> from Bio import SeqIO
|      >>> record = SeqIO.read("Quality/solexa_faked.fastq", "fastq-solexa")
```

```
 |      >>> print("%s %s" % (record.id, record.seq))
 |      slxa_0001_1_0001_01 ACGTACGTACGTACGTACGTACGTACGTACGTACGTNNNNNN
 |      >>> print(list(record.letter_annotations))
 |      ['solexa_quality']
 |      >>> print(record.letter_annotations["solexa_quality"])
 |      [40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15,
14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5]
 |
 |      The letter_annotations get sliced automatically if you slice the
 |      parent SeqRecord, for example taking the last ten bases:
 |
 |      >>> sub_record = record[-10:]
 |      >>> print("%s %s" % (sub_record.id, sub_record.seq))
 |      slxa_0001_1_0001_01 ACGTNNNNNN
 |      >>> print(sub_record.letter_annotations["solexa_quality"])
 |      [4, 3, 2, 1, 0, -1, -2, -3, -4, -5]
 |
 |      Any python sequence (i.e. list, tuple or string) can be recorded in
 |      the SeqRecord's letter_annotations dictionary as long as the length
 |      matches that of the SeqRecord's sequence.  e.g.
 |
 |      >>> len(sub_record.letter_annotations)
 |      1
 |      >>> sub_record.letter_annotations["dummy"] = "abcdefghij"
 |      >>> len(sub_record.letter_annotations)
 |      2
 |
 |      You can delete entries from the letter_annotations dictionary as usual:
 |
 |      >>> del sub_record.letter_annotations["solexa_quality"]
 |      >>> sub_record.letter_annotations
 |      {'dummy': 'abcdefghij'}
 |
 |      You can completely clear the dictionary easily as follows:
 |
 |      >>> sub_record.letter_annotations = {}
 |      >>> sub_record.letter_annotations
 |      {}
 |
 |      Note that if replacing the record's sequence with a sequence of a
 |      different length you must first clear the letter_annotations dict.
 |
 |  seq
 |      The sequence itself, as a Seq or MutableSeq object.
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  __hash__ = None
```

Let's write a bit of code involving `SeqRecord` and see how it comes out looking.

In [27]:
```
from Bio.SeqRecord import SeqRecord

simple_seq = Seq("GATC")
simple_seq_r = SeqRecord(simple_seq)
```

In [28]:
```
simple_seq_r.id = "AC12345"
simple_seq_r.description = "Made up sequence"
print(simple_seq_r.id)
print(simple_seq_r.description)
```

```
AC12345
Made up sequence
```

Let's now see how we can use `SeqRecord` to parse a large fasta file. We'll pull down a file hosted on the biopython site.

In [29]:
```
!wget https://raw.githubusercontent.com/biopython/biopython/master/Tests/GenBank/NC_005816.fna
```

```
--2023-08-02 14:46:08--  https://raw.githubusercontent.com/biopython/biopython/master/Tests/GenBank/NC_005816.fn
a
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110
.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9853 (9.6K) [text/plain]
Saving to: ‘NC_005816.fna’

NC_005816.fna       100%[===================>]   9.62K  --.-KB/s    in 0s

2023-08-02 14:46:08 (78.8 MB/s) - ‘NC_005816.fna’ saved [9853/9853]
```

In [30]:
```
from Bio import SeqIO

record = SeqIO.read("NC_005816.fna", "fasta")
record
```

Out[30]:
```
SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG'), id='gi|45478711|ref|NC_005816.1|', name='gi|45478711|ref|NC_005816.1|', description='gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete sequence', dbxrefs=[])
```

Note how there's a number of annotations attached to the `SeqRecord` object!

Let's take a closer look.

In [31]:
```
record.id
```

Out[31]:
```
'gi|45478711|ref|NC_005816.1|'
```

In [32]:
```
record.name
```

Out[32]:
```
'gi|45478711|ref|NC_005816.1|'
```

In [33]:
```
record.description
```

Out[33]:
```
'gi|45478711|ref|NC_005816.1| Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete sequence'
```

Let's now look at the same sequence, but downloaded from GenBank. We'll download the hosted file from the biopython tutorial website as before.

In [34]:
```
!wget https://raw.githubusercontent.com/biopython/biopython/master/Tests/GenBank/NC_005816.gb
```

```
--2023-08-02 14:46:08--  https://raw.githubusercontent.com/biopython/biopython/master/Tests/GenBank/NC_005816.gb
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 31838 (31K) [text/plain]
Saving to: ‘NC_005816.gb’

NC_005816.gb        100%[===================>]  31.09K  --.-KB/s    in 0.002s

2023-08-02 14:46:08 (12.7 MB/s) - ‘NC_005816.gb’ saved [31838/31838]
```

In [35]:
```
from Bio import SeqIO

record = SeqIO.read("NC_005816.gb", "genbank")
record
```

Out[35]:
```
SeqRecord(seq=Seq('TGTAACGAACGGTGCAATAGTGATCCACACCCAACGCCTGAAATCAGATCCAGG...CTG'), id='NC_005816.1', name='NC_005816', description='Yersinia pestis biovar Microtus str. 91001 plasmid pPCP1, complete sequence', dbxrefs=['Project:58037'])
```

SeqIO Objects

.count() Method

The .count() method in Biopython's Seq object behaves similar to the .count() method of Python strings. It returns the number of non-overlapping occurrences of a specific subsequence within the sequence.

In [36]:
```
from Bio.Seq import Seq
my_seq = Seq("AGTACACATTG")
count_a = my_seq.count('A')
count_tg = my_seq.count('TG')
print(count_a)   # Output: 3
print(count_tg)  # Output: 1
```

```
4
1
```

MutableSeq objects

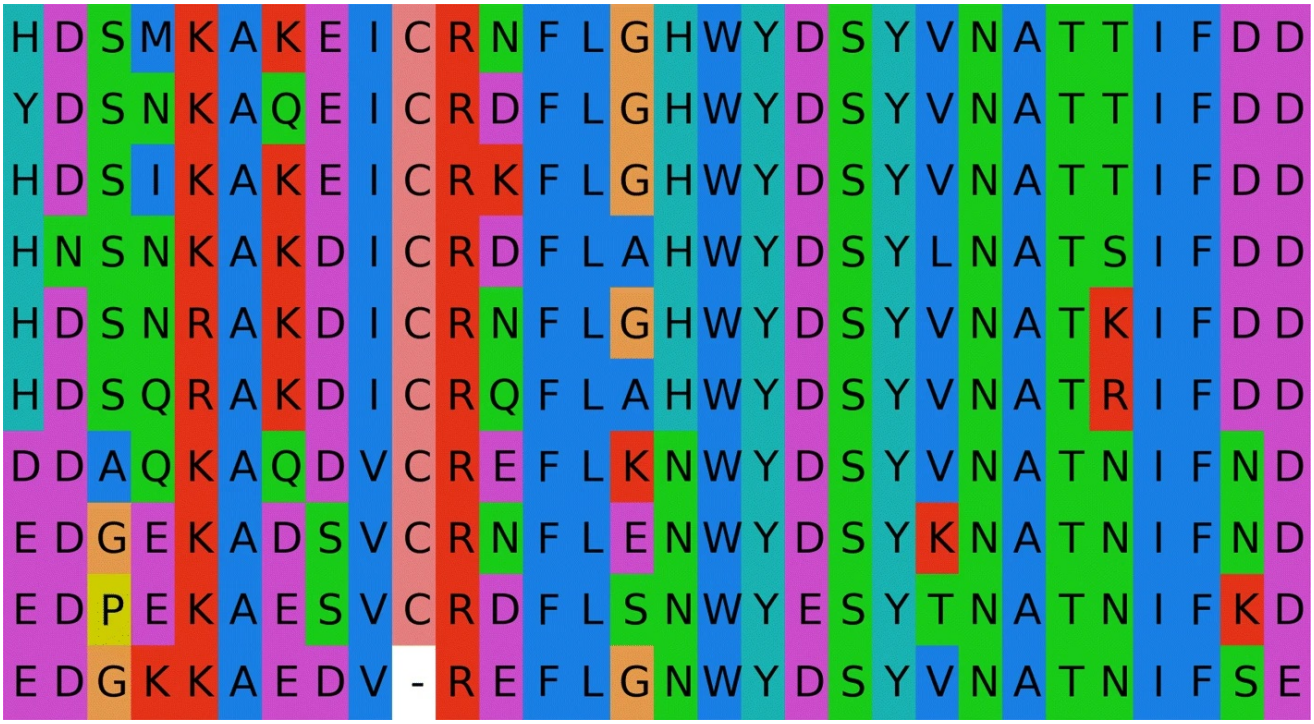
Just like the normal Python string, the Seq object is “read only”, or in Python terminology, immutable. Apart from wanting the Seq object to act like a string, this is also a useful default since in many biological applications you want to ensure you are not changing your sequence data: you can convert it into a mutable sequence (a MutableSeq object) and do pretty much anything you want with it

In [37]:
```
from Bio.Seq import MutableSeq
mutable_seq = MutableSeq("GCCATTGTAATGGGCCGCTGAAAGGGTGCCCGA")
```

References: [1]https://www.khanacademy.org/science/ap-biology/gene-expression-and-regulation/transcription-and-rna-processing/a/overview-of-transcription [2]From DNA to RNA. https://www.ncbi.nlm.nih.gov/books/NBK26887/

Multisequence Alignment (MSA)

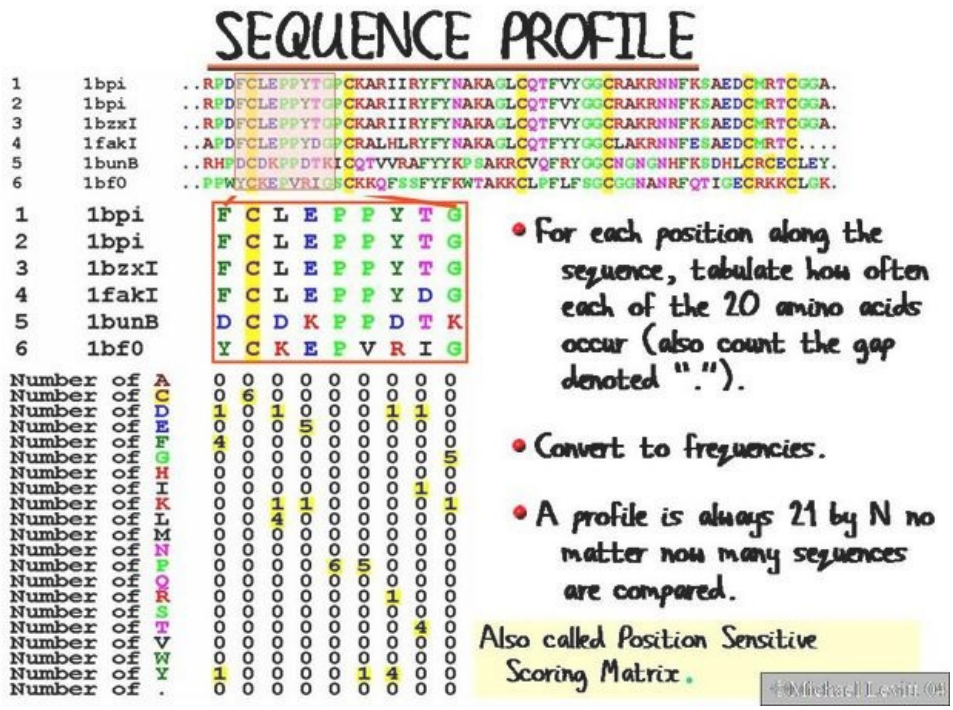
Proteins are made up of sequences of amino acids chained together. Their amino acid sequence determines their structure and function. Finding proteins with similar sequences, or homologous proteins, is very useful in identifying the structures and functions of newly discovered proteins as well as identifying their ancestry. Below is an example of what a protein amino acid multisequence alignment may look like, taken from [2].



To understand Multiple Sequence Alignment (MSA), it's helpful to first grasp pairwise sequence alignment. Pairwise sequence alignment is a hypothesis about how two sequences may have evolved from a common ancestor through events such as mutation, insertion, and deletion. When a nucleotide is aligned with a gap, it represents an indel event—either a deletion in one sequence or an insertion in the other. When two different nucleotides are aligned, this is typically interpreted as a substitution or mutation event introduced in one or both of the lineages since the time they diverged from one another.If identical nucleotides are aligned, it suggests a conserved region, which may indicate functional importance and possibly homology—evidence that the sequences share a common ancestor. Pairwise alignment also provides an optimal alignment of two sequences by strategically introducing gaps, making it useful for comparing sequences and identifying conserved regions.The alignment is an optimal hypothesis but may not reflect the actual evolutionary path.

Using profile-sequence comparison instead of just sequence-sequence comparison when constructing multiple sequence alignment utilizes more information. Sequence profiles are based on frequencies of each of the 20 amino acids at each position in a sequence. Sequence profiles (or PSSMs) tell us how likely it is that a particular amino acid (or nucleotide in DNA/RNA sequences) at a specific position is due to conservation rather than random chance.This is achieved through the use of log-odds scores that compare the observed frequency of an amino acid at a particular position in the multiple sequence alignment (MSA) to its expected frequency under random conditions (i.e., the background probability).

Here is how the position frequence matrix looks. Sequence profile is constructed through the log odds score of these



frequencies.

A Profile Hidden Markov Model is a probabilistic model that represents the sequence conservation at each position (including insertions and deletions) and the likelihood of transitioning between different sequence states.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.

Open in Colab

HH-suite

This tutorial will show you the basics of how to use hh-suite. hh-suite is an open source package for searching protein sequence alignments for homologous proteins. It is the current state of the art for building highly accurate multisequence alignments (MSA) from a single sequence or from MSAs.

HH-suite leverages profile HMMs to improve the accuracy of detecting remote homologs (sequences that share a common ancestor but are highly diverged).

Instead of comparing a single sequence against a database of sequences, it aligns one profile HMM against another profile HMM.The idea is that by comparing two profile HMMs, it can detect relationships between sequence families that are not apparent when comparing just sequences. This is particularly useful when dealing with highly diverged proteins or detecting remote homologs.

Setup

Let's start by importing the deepchem sequence_utils module and downloading a database to compare our query sequence to.

hh-suite provides a set of HMM databases that will work with the software, which you can find here: http://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs

dbCAN is a good one for this tutorial because it is a relatively smaller download.

In [34]:
```
from deepchem.utils import sequence_utils
```

In [36]:
```
%%bash
mkdir hh
cd hh
mkdir databases; cd databases
wget http://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs/dbCAN-fam-V9.tar.gz
tar xzvf dbCAN-fam-V9.tar.gz
```

```
dbCAN-fam-V9_a3m.ffdata
dbCAN-fam-V9_a3m.ffindex
dbCAN-fam-V9_hhm.ffdata
dbCAN-fam-V9_hhm.ffindex
dbCAN-fam-V9_cs219.ffdata
dbCAN-fam-V9_cs219.ffindex
dbCAN-fam-V9.md5sum
```

```
--2022-02-11 12:47:57--  http://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs/dbCAN-fam-V9.tar.gz
Resolving wwwuser.gwdg.de (wwwuser.gwdg.de)... 134.76.10.111
Connecting to wwwuser.gwdg.de (wwwuser.gwdg.de)|134.76.10.111|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 25882327 (25M) [application/x-gzip]
Saving to: ‘dbCAN-fam-V9.tar.gz’

     0K .......... .......... .......... .......... ..........  0%  195K 2m10s
    50K .......... .......... .......... .......... ..........  0%  391K 97s
   100K .......... .......... .......... .......... ..........  0% 55.2M 65s
   150K .......... .......... .......... .......... ..........  0%  392K 64s
   200K .......... .......... .......... .......... ..........  0%  127M 51s
   250K .......... .......... .......... .......... ..........  1% 50.8M 43s
   300K .......... .......... .......... .......... ..........  1% 85.5M 37s
   350K .......... .......... .......... .......... ..........  1%  395K 40s
   400K .......... .......... .......... .......... ..........  1%  102M 35s
   450K .......... .......... .......... .......... ..........  1% 55.2M 32s
   500K .......... .......... .......... .......... ..........  2%  392K 35s
   550K .......... .......... .......... .......... ..........  2% 74.8M 32s
   600K .......... .......... .......... .......... ..........  2%  102M 29s
   650K .......... .......... .......... .......... ..........  2%  399K 32s
   700K .......... .......... .......... .......... ..........  2% 23.6M 29s
   750K .......... .......... .......... .......... ..........  3% 39.5M 28s
   800K .......... .......... .......... .......... ..........  3% 1.26M 27s
   850K .......... .......... .......... .......... ..........  3%  571K 28s
   900K .......... .......... .......... .......... ..........  3% 35.9M 26s
   950K .......... .......... .......... .......... ..........  3% 52.9M 25s
  1000K .......... .......... .......... .......... ..........  4%  401K 27s
  1050K .......... .......... .......... .......... ..........  4% 17.2M 25s
  1100K .......... .......... .......... .......... ..........  4% 64.7M 24s
  1150K .......... .......... .......... .......... ..........  4%  406K 26s
  1200K .......... .......... .......... .......... ..........  4% 14.1M 25s
  1250K .......... .......... .......... .......... ..........  5% 45.9M 24s
  1300K .......... .......... .......... .......... ..........  5% 1.30M 23s
  1350K .......... .......... .......... .......... ..........  5%  572K 24s
  1400K .......... .......... .......... .......... ..........  5% 21.6M 23s
  1450K .......... .......... .......... .......... ..........  5% 39.8M 22s
  1500K .......... .......... .......... .......... ..........  6%  405K 24s
  1550K .......... .......... .......... .......... ..........  6% 16.8M 23s
  1600K .......... .......... .......... .......... ..........  6% 28.4M 22s
  1650K .......... .......... .......... .......... ..........  6%  413K 23s
  1700K .......... .......... .......... .......... ..........  6% 16.0M 22s
  1750K .......... .......... .......... .......... ..........  7% 19.0M 22s
  1800K .......... .......... .......... .......... ..........  7% 31.5M 21s
  1850K .......... .......... .......... .......... ..........  7%  407K 22s
  1900K .......... .......... .......... .......... ..........  7% 14.5M 21s
  1950K .......... .......... .......... .......... ..........  7% 40.5M 21s
  2000K .......... .......... .......... .......... ..........  8%  410K 22s
  2050K .......... .......... .......... .......... ..........  8% 10.1M 21s
  2100K .......... .......... .......... .......... ..........  8% 66.8M 21s
  2150K .......... .......... .......... .......... ..........  8% 44.7M 20s
  2200K .......... .......... .......... .......... ..........  8%  409K 21s
  2250K .......... .......... .......... .......... ..........  9% 11.5M 21s
  2300K .......... .......... .......... .......... ..........  9% 64.3M 20s
  2350K .......... .......... .......... .......... ..........  9%  389K 21s
  2400K .......... .......... .......... .......... ..........  9%  111M 20s
  2450K .......... .......... .......... .......... ..........  9% 63.7M 20s
  2500K .......... .......... .......... .......... .......... 10%  110M 19s
  2550K .......... .......... .......... .......... .......... 10%  392K 20s
  2600K .......... .......... .......... .......... .......... 10%  114M 20s
  2650K .......... .......... .......... .......... .......... 10% 52.9M 19s
  2700K .......... .......... .......... .......... .......... 10%  398K 20s
  2750K .......... .......... .......... .......... .......... 11% 31.7M 20s
  2800K .......... .......... .......... .......... .......... 11% 79.2M 19s
  2850K .......... .......... .......... .......... .......... 11% 71.9M 19s
  2900K .......... .......... .......... .......... .......... 11%  399K 19s
  2950K .......... .......... .......... .......... .......... 11% 26.2M 19s
  3000K .......... .......... .......... .......... .......... 12% 75.7M 19s
  3050K .......... .......... .......... .......... .......... 12% 1.46M 19s
  3100K .......... .......... .......... .......... .......... 12%  537K 19s
  3150K .......... .......... .......... .......... .......... 12% 34.0M 19s
  3200K .......... .......... .......... .......... .......... 12% 78.4M 18s
  3250K .......... .......... .......... .......... .......... 13%  402K 19s
  3300K .......... .......... .......... .......... .......... 13% 26.9M 18s
  3350K .......... .......... .......... .......... .......... 13% 30.2M 18s
  3400K .......... .......... .......... .......... .......... 13% 52.6M 18s
```

```
3450K ..... ..... ..... ..... ..... 13%  400K 18s
3500K ..... ..... ..... ..... ..... 14% 41.8M 18s
3550K ..... ..... ..... ..... ..... 14% 46.1M 18s
3600K ..... ..... ..... ..... ..... 14% 1.44M 18s
3650K ..... ..... ..... ..... ..... 14%  544K 18s
3700K ..... ..... ..... ..... ..... 14% 40.0M 18s
3750K ..... ..... ..... ..... ..... 15% 35.8M 17s
3800K ..... ..... ..... ..... ..... 15%  402K 18s
3850K ..... ..... ..... ..... ..... 15% 46.9M 18s
3900K ..... ..... ..... ..... ..... 15% 19.3M 17s
3950K ..... ..... ..... ..... ..... 15% 82.5M 17s
4000K ..... ..... ..... ..... ..... 16%  404K 17s
4050K ..... ..... ..... ..... ..... 16% 43.3M 17s
4100K ..... ..... ..... ..... ..... 16% 17.1M 17s
4150K ..... ..... ..... ..... ..... 16% 81.5M 17s
4200K ..... ..... ..... ..... ..... 16%  398K 17s
4250K ..... ..... ..... ..... ..... 17% 50.2M 17s
4300K ..... ..... ..... ..... ..... 17% 45.0M 17s
4350K ..... ..... ..... ..... ..... 17% 1.48M 17s
4400K ..... ..... ..... ..... ..... 17%  539K 17s
4450K ..... ..... ..... ..... ..... 17% 42.5M 17s
4500K ..... ..... ..... ..... ..... 18% 32.2M 16s
4550K ..... ..... ..... ..... ..... 18% 1.51M 16s
4600K ..... ..... ..... ..... ..... 18%  539K 16s
4650K ..... ..... ..... ..... ..... 18% 36.4M 16s
4700K ..... ..... ..... ..... ..... 18% 34.1M 16s
4750K ..... ..... ..... ..... ..... 18%  440K 16s
4800K ..... ..... ..... ..... ..... 19% 4.12M 16s
4850K ..... ..... ..... ..... ..... 19% 38.7M 16s
4900K ..... ..... ..... ..... ..... 19% 44.3M 16s
4950K ..... ..... ..... ..... ..... 19%  406K 16s
5000K ..... ..... ..... ..... ..... 19% 21.4M 16s
5050K ..... ..... ..... ..... ..... 20% 39.1M 16s
5100K ..... ..... ..... ..... ..... 20% 43.1M 16s
5150K ..... ..... ..... ..... ..... 20%  407K 16s
5200K ..... ..... ..... ..... ..... 20% 13.1M 16s
5250K ..... ..... ..... ..... ..... 20% 43.6M 15s
5300K ..... ..... ..... ..... ..... 21% 62.2M 15s
5350K ..... ..... ..... ..... ..... 21%  399K 16s
5400K ..... ..... ..... ..... ..... 21% 58.5M 15s
5450K ..... ..... ..... ..... ..... 21% 38.1M 15s
5500K ..... ..... ..... ..... ..... 21% 55.0M 15s
5550K ..... ..... ..... ..... ..... 22%  401K 15s
5600K ..... ..... ..... ..... ..... 22% 27.5M 15s
5650K ..... ..... ..... ..... ..... 22% 50.0M 15s
5700K ..... ..... ..... ..... ..... 22% 47.0M 15s
5750K ..... ..... ..... ..... ..... 22%  403K 15s
5800K ..... ..... ..... ..... ..... 23% 21.3M 15s
5850K ..... ..... ..... ..... ..... 23% 29.3M 15s
5900K ..... ..... ..... ..... ..... 23% 59.4M 15s
5950K ..... ..... ..... ..... ..... 23%  406K 15s
6000K ..... ..... ..... ..... ..... 23% 19.4M 15s
6050K ..... ..... ..... ..... ..... 24% 34.9M 15s
6100K ..... ..... ..... ..... ..... 24% 39.6M 14s
6150K ..... ..... ..... ..... ..... 24%  408K 15s
6200K ..... ..... ..... ..... ..... 24% 18.4M 14s
6250K ..... ..... ..... ..... ..... 24% 36.5M 14s
6300K ..... ..... ..... ..... ..... 25% 29.7M 14s
6350K ..... ..... ..... ..... ..... 25%  420K 14s
6400K ..... ..... ..... ..... ..... 25% 9.03M 14s
6450K ..... ..... ..... ..... ..... 25% 26.0M 14s
6500K ..... ..... ..... ..... ..... 25% 37.5M 14s
6550K ..... ..... ..... ..... ..... 26%  453K 14s
6600K ..... ..... ..... ..... ..... 26% 3.58M 14s
6650K ..... ..... ..... ..... ..... 26% 18.8M 14s
6700K ..... ..... ..... ..... ..... 26% 40.3M 14s
6750K ..... ..... ..... ..... ..... 26%  455K 14s
6800K ..... ..... ..... ..... ..... 27% 3.64M 14s
6850K ..... ..... ..... ..... ..... 27% 15.0M 14s
6900K ..... ..... ..... ..... ..... 27% 49.0M 14s
6950K ..... ..... ..... ..... ..... 27%  733K 14s
7000K ..... ..... ..... ..... ..... 27%  896K 14s
7050K ..... ..... ..... ..... ..... 28% 23.0M 13s
7100K ..... ..... ..... ..... ..... 28% 27.6M 13s
7150K ..... ..... ..... ..... ..... 28% 1.59M 13s
7200K ..... ..... ..... ..... ..... 28%  537K 13s
7250K ..... ..... ..... ..... ..... 28% 24.2M 13s
7300K ..... ..... ..... ..... ..... 29% 23.2M 13s
7350K ..... ..... ..... ..... ..... 29% 68.7M 13s
7400K ..... ..... ..... ..... ..... 29%  408K 13s
7450K ..... ..... ..... ..... ..... 29% 29.0M 13s
7500K ..... ..... ..... ..... ..... 29% 13.4M 13s
7550K ..... ..... ..... ..... ..... 30% 46.7M 13s
```

```
 7600K ..... ..... ..... ..... ..... 30%  413K 13s
 7650K ..... ..... ..... ..... ..... 30% 19.8M 13s
 7700K ..... ..... ..... ..... ..... 30% 29.5M 13s
 7750K ..... ..... ..... ..... ..... 30% 17.0M 13s
 7800K ..... ..... ..... ..... ..... 31%  463K 13s
 7850K ..... ..... ..... ..... ..... 31% 3.53M 13s
 7900K ..... ..... ..... ..... ..... 31% 19.9M 13s
 7950K ..... ..... ..... ..... ..... 31% 13.7M 12s
 8000K ..... ..... ..... ..... ..... 31% 1.72M 12s
 8050K ..... ..... ..... ..... ..... 32%  539K 12s
 8100K ..... ..... ..... ..... ..... 32% 13.4M 12s
 8150K ..... ..... ..... ..... ..... 32% 21.9M 12s
 8200K ..... ..... ..... ..... ..... 32% 43.8M 12s
 8250K ..... ..... ..... ..... ..... 32%  415K 12s
 8300K ..... ..... ..... ..... ..... 33% 14.1M 12s
 8350K ..... ..... ..... ..... ..... 33% 25.5M 12s
 8400K ..... ..... ..... ..... ..... 33% 23.7M 12s
 8450K ..... ..... ..... ..... ..... 33%  416K 12s
 8500K ..... ..... ..... ..... ..... 33% 45.8M 12s
 8550K ..... ..... ..... ..... ..... 34% 13.0M 12s
 8600K ..... ..... ..... ..... ..... 34% 19.6M 12s
 8650K ..... ..... ..... ..... ..... 34%  765K 12s
 8700K ..... ..... ..... ..... ..... 34%  895K 12s
 8750K ..... ..... ..... ..... ..... 34% 14.2M 12s
 8800K ..... ..... ..... ..... ..... 35% 18.2M 12s
 8850K ..... ..... ..... ..... ..... 35% 43.9M 12s
 8900K ..... ..... ..... ..... ..... 35%  416K 12s
 8950K ..... ..... ..... ..... ..... 35% 15.8M 12s
 9000K ..... ..... ..... ..... ..... 35% 32.0M 11s
 9050K ..... ..... ..... ..... ..... 36% 12.4M 11s
 9100K ..... ..... ..... ..... ..... 36%  439K 12s
 9150K ..... ..... ..... ..... ..... 36% 7.92M 11s
 9200K ..... ..... ..... ..... ..... 36% 14.8M 11s
 9250K ..... ..... ..... ..... ..... 36% 11.0M 11s
 9300K ..... ..... ..... ..... ..... 36% 1.85M 11s
 9350K ..... ..... ..... ..... ..... 37%  537K 11s
 9400K ..... ..... ..... ..... ..... 37% 16.8M 11s
 9450K ..... ..... ..... ..... ..... 37% 25.4M 11s
 9500K ..... ..... ..... ..... ..... 37% 12.7M 11s
 9550K ..... ..... ..... ..... ..... 37%  422K 11s
 9600K ..... ..... ..... ..... ..... 38% 38.8M 11s
 9650K ..... ..... ..... ..... ..... 38% 14.7M 11s
 9700K ..... ..... ..... ..... ..... 38% 11.1M 11s
 9750K ..... ..... ..... ..... ..... 38% 1.84M 11s
 9800K ..... ..... ..... ..... ..... 38%  538K 11s
 9850K ..... ..... ..... ..... ..... 39% 15.3M 11s
 9900K ..... ..... ..... ..... ..... 39% 31.6M 11s
 9950K ..... ..... ..... ..... ..... 39% 12.4M 11s
10000K ..... ..... ..... ..... ..... 39%  422K 11s
10050K ..... ..... ..... ..... ..... 39% 14.1M 11s
10100K ..... ..... ..... ..... ..... 40% 30.8M 11s
10150K ..... ..... ..... ..... ..... 40% 12.6M 10s
10200K ..... ..... ..... ..... ..... 40% 1.84M 10s
10250K ..... ..... ..... ..... ..... 40%  532K 10s
10300K ..... ..... ..... ..... ..... 40% 22.0M 10s
10350K ..... ..... ..... ..... ..... 41% 25.4M 10s
10400K ..... ..... ..... ..... ..... 41% 11.9M 10s
10450K ..... ..... ..... ..... ..... 41%  471K 10s
10500K ..... ..... ..... ..... ..... 41% 3.24M 10s
10550K ..... ..... ..... ..... ..... 41% 17.7M 10s
10600K ..... ..... ..... ..... ..... 42% 13.3M 10s
10650K ..... ..... ..... ..... ..... 42% 1.92M 10s
10700K ..... ..... ..... ..... ..... 42%  532K 10s
10750K ..... ..... ..... ..... ..... 42% 12.6M 10s
10800K ..... ..... ..... ..... ..... 42% 39.9M 10s
10850K ..... ..... ..... ..... ..... 43% 12.8M 10s
10900K ..... ..... ..... ..... ..... 43%  803K 10s
10950K ..... ..... ..... ..... ..... 43%  849K 10s
11000K ..... ..... ..... ..... ..... 43% 23.2M 10s
11050K ..... ..... ..... ..... ..... 43% 22.8M 10s
11100K ..... ..... ..... ..... ..... 44% 11.8M 10s
11150K ..... ..... ..... ..... ..... 44%  431K 10s
11200K ..... ..... ..... ..... ..... 44% 12.4M 10s
11250K ..... ..... ..... ..... ..... 44% 17.2M 10s
11300K ..... ..... ..... ..... ..... 44% 14.0M 9s
11350K ..... ..... ..... ..... ..... 45% 1.87M 9s
11400K ..... ..... ..... ..... ..... 45%  533K 9s
11450K ..... ..... ..... ..... ..... 45% 13.6M 9s
11500K ..... ..... ..... ..... ..... 45% 43.6M 9s
11550K ..... ..... ..... ..... ..... 45% 14.5M 9s
11600K ..... ..... ..... ..... ..... 46%  819K 9s
11650K ..... ..... ..... ..... ..... 46%  839K 9s
11700K ..... ..... ..... ..... ..... 46% 16.7M 9s
```

```
11750K .......... .......... .......... .......... .......... 46% 40.2M 9s
11800K .......... .......... .......... .......... .......... 46% 11.6M 9s
11850K .......... .......... .......... .......... .......... 47%  453K 9s
11900K .......... .......... .......... .......... .......... 47% 4.96M 9s
11950K .......... .......... .......... .......... .......... 47% 18.2M 9s
12000K .......... .......... .......... .......... .......... 47% 17.6M 9s
12050K .......... .......... .......... .......... .......... 47% 1.87M 9s
12100K .......... .......... .......... .......... .......... 48%  533K 9s
12150K .......... .......... .......... .......... .......... 48% 11.7M 9s
12200K .......... .......... .......... .......... .......... 48% 35.4M 9s
12250K .......... .......... .......... .......... .......... 48% 15.3M 9s
12300K .......... .......... .......... .......... .......... 48% 1.88M 9s
12350K .......... .......... .......... .......... .......... 49%  533K 9s
12400K .......... .......... .......... .......... .......... 49% 12.8M 9s
12450K .......... .......... .......... .......... .......... 49% 22.3M 9s
12500K .......... .......... .......... .......... .......... 49% 21.2M 8s
12550K .......... .......... .......... .......... .......... 49%  806K 8s
12600K .......... .......... .......... .......... .......... 50%  859K 8s
12650K .......... .......... .......... .......... .......... 50% 15.8M 8s
12700K .......... .......... .......... .......... .......... 50% 14.4M 8s
12750K .......... .......... .......... .......... .......... 50% 21.4M 8s
12800K .......... .......... .......... .......... .......... 50%  484K 8s
12850K .......... .......... .......... .......... .......... 51% 3.11M 8s
12900K .......... .......... .......... .......... .......... 51% 12.4M 8s
12950K .......... .......... .......... .......... .......... 51% 18.9M 8s
13000K .......... .......... .......... .......... .......... 51% 20.2M 8s
13050K .......... .......... .......... .......... .......... 51%  429K 8s
13100K .......... .......... .......... .......... .......... 52% 11.1M 8s
13150K .......... .......... .......... .......... .......... 52% 26.1M 8s
13200K .......... .......... .......... .......... .......... 52% 17.4M 8s
13250K .......... .......... .......... .......... .......... 52% 1.98M 8s
13300K .......... .......... .......... .......... .......... 52%  532K 8s
13350K .......... .......... .......... .......... .......... 53% 8.56M 8s
13400K .......... .......... .......... .......... .......... 53% 18.6M 8s
13450K .......... .......... .......... .......... .......... 53% 37.9M 8s
13500K .......... .......... .......... .......... .......... 53% 1.83M 8s
13550K .......... .......... .......... .......... .......... 53%  518K 8s
13600K .......... .......... .......... .......... .......... 54% 53.3M 8s
13650K .......... .......... .......... .......... .......... 54% 17.0M 8s
13700K .......... .......... .......... .......... .......... 54% 32.4M 8s
13750K .......... .......... .......... .......... .......... 54% 1.88M 7s
13800K .......... .......... .......... .......... .......... 54%  518K 8s
13850K .......... .......... .......... .......... .......... 54% 35.2M 7s
13900K .......... .......... .......... .......... .......... 55% 20.6M 7s
13950K .......... .......... .......... .......... .......... 55% 26.7M 7s
14000K .......... .......... .......... .......... .......... 55% 1.90M 7s
14050K .......... .......... .......... .......... .......... 55%  508K 7s
14100K .......... .......... .......... .......... .......... 55%  111M 7s
14150K .......... .......... .......... .......... .......... 56% 21.6M 7s
14200K .......... .......... .......... .......... .......... 56% 43.4M 7s
14250K .......... .......... .......... .......... .......... 56% 1.90M 7s
14300K .......... .......... .......... .......... .......... 56%  509K 7s
14350K .......... .......... .......... .......... .......... 56% 47.6M 7s
14400K .......... .......... .......... .......... .......... 57% 29.2M 7s
14450K .......... .......... .......... .......... .......... 57% 36.3M 7s
14500K .......... .......... .......... .......... .......... 57% 1.07M 7s
14550K .......... .......... .......... .......... .......... 57%  639K 7s
14600K .......... .......... .......... .......... .......... 57% 19.1M 7s
14650K .......... .......... .......... .......... .......... 58%  113M 7s
14700K .......... .......... .......... .......... .......... 58% 40.6M 7s
14750K .......... .......... .......... .......... .......... 58% 1.08M 7s
14800K .......... .......... .......... .......... .......... 58%  638K 7s
14850K .......... .......... .......... .......... .......... 58% 16.9M 7s
14900K .......... .......... .......... .......... .......... 59% 50.2M 7s
14950K .......... .......... .......... .......... .......... 59% 16.4M 7s
15000K .......... .......... .......... .......... .......... 59% 2.03M 7s
15050K .......... .......... .......... .......... .......... 59%  514K 7s
15100K .......... .......... .......... .......... .......... 59% 18.5M 7s
15150K .......... .......... .......... .......... .......... 60% 38.8M 6s
15200K .......... .......... .......... .......... .......... 60% 12.3M 6s
15250K .......... .......... .......... .......... .......... 60% 2.15M 6s
15300K .......... .......... .......... .......... .......... 60%  528K 6s
15350K .......... .......... .......... .......... .......... 60% 6.73M 6s
15400K .......... .......... .......... .......... .......... 61%  110M 6s
15450K .......... .......... .......... .......... .......... 61% 19.5M 6s
15500K .......... .......... .......... .......... .......... 61% 2.14M 6s
15550K .......... .......... .......... .......... .......... 61%  528K 6s
15600K .......... .......... .......... .......... .......... 61% 6.48M 6s
15650K .......... .......... .......... .......... .......... 62% 51.5M 6s
15700K .......... .......... .......... .......... .......... 62%  114M 6s
15750K .......... .......... .......... .......... .......... 62% 2.36M 6s
15800K .......... .......... .......... .......... .......... 62%  530K 6s
15850K .......... .......... .......... .......... .......... 62% 4.16M 6s
```

```
15900K .......... .......... .......... .......... .......... 63% 70.9M 6s
15950K .......... .......... .......... .......... .......... 63% 50.3M 6s
16000K .......... .......... .......... .......... .......... 63% 2.44M 6s
16050K .......... .......... .......... .......... .......... 63%  531K 6s
16100K .......... .......... .......... .......... .......... 63% 4.72M 6s
16150K .......... .......... .......... .......... .......... 64% 20.3M 6s
16200K .......... .......... .......... .......... .......... 64% 28.9M 6s
16250K .......... .......... .......... .......... .......... 64% 54.9M 6s
16300K .......... .......... .......... .......... .......... 64%  476K 6s
16350K .......... .......... .......... .......... .......... 64% 2.72M 6s
16400K .......... .......... .......... .......... .......... 65% 20.8M 6s
16450K .......... .......... .......... .......... .......... 65% 38.4M 6s
16500K .......... .......... .......... .......... .......... 65% 33.5M 5s
16550K .......... .......... .......... .......... .......... 65%  914K 5s
16600K .......... .......... .......... .......... .......... 65%  766K 5s
16650K .......... .......... .......... .......... .......... 66% 9.17M 5s
16700K .......... .......... .......... .......... .......... 66% 70.4M 5s
16750K .......... .......... .......... .......... .......... 66% 18.1M 5s
16800K .......... .......... .......... .......... .......... 66% 2.35M 5s
16850K .......... .......... .......... .......... .......... 66%  513K 5s
16900K .......... .......... .......... .......... .......... 67% 8.14M 5s
16950K .......... .......... .......... .......... .......... 67% 44.1M 5s
17000K .......... .......... .......... .......... .......... 67% 16.1M 5s
17050K .......... .......... .......... .......... .......... 67% 2.90M 5s
17100K .......... .......... .......... .......... .......... 67%  532K 5s
17150K .......... .......... .......... .......... .......... 68% 4.43M 5s
17200K .......... .......... .......... .......... .......... 68% 17.9M 5s
17250K .......... .......... .......... .......... .......... 68% 17.0M 5s
17300K .......... .......... .......... .......... .......... 68% 35.4M 5s
17350K .......... .......... .......... .......... .......... 68%  490K 5s
17400K .......... .......... .......... .......... .......... 69% 2.72M 5s
17450K .......... .......... .......... .......... .......... 69% 14.4M 5s
17500K .......... .......... .......... .......... .......... 69% 35.5M 5s
17550K .......... .......... .......... .......... .......... 69% 16.3M 5s
17600K .......... .......... .......... .......... .......... 69%  967K 5s
17650K .......... .......... .......... .......... .......... 70%  772K 5s
17700K .......... .......... .......... .......... .......... 70% 7.62M 5s
17750K .......... .......... .......... .......... .......... 70% 24.5M 5s
17800K .......... .......... .......... .......... .......... 70% 15.2M 5s
17850K .......... .......... .......... .......... .......... 70% 2.99M 5s
17900K .......... .......... .......... .......... .......... 71%  515K 5s
17950K .......... .......... .......... .......... .......... 71% 5.63M 5s
18000K .......... .......... .......... .......... .......... 71% 16.3M 4s
18050K .......... .......... .......... .......... .......... 71% 59.2M 4s
18100K .......... .......... .......... .......... .......... 71% 15.1M 4s
18150K .......... .......... .......... .......... .......... 72%  499K 4s
18200K .......... .......... .......... .......... .......... 72% 2.69M 4s
18250K .......... .......... .......... .......... .......... 72% 14.5M 4s
18300K .......... .......... .......... .......... .......... 72% 18.4M 4s
18350K .......... .......... .......... .......... .......... 72% 14.8M 4s
18400K .......... .......... .......... .......... .......... 72% 1.39M 4s
18450K .......... .......... .......... .......... .......... 73%  628K 4s
18500K .......... .......... .......... .......... .......... 73% 7.11M 4s
18550K .......... .......... .......... .......... .......... 73% 20.1M 4s
18600K .......... .......... .......... .......... .......... 73% 19.7M 4s
18650K .......... .......... .......... .......... .......... 73% 2.87M 4s
18700K .......... .......... .......... .......... .......... 74%  534K 4s
18750K .......... .......... .......... .......... .......... 74% 5.43M 4s
18800K .......... .......... .......... .......... .......... 74% 11.6M 4s
18850K .......... .......... .......... .......... .......... 74% 22.7M 4s
18900K .......... .......... .......... .......... .......... 74% 13.0M 4s
18950K .......... .......... .......... .......... .......... 75%  998K 4s
19000K .......... .......... .......... .......... .......... 75%  776K 4s
19050K .......... .......... .......... .......... .......... 75% 6.50M 4s
19100K .......... .......... .......... .......... .......... 75% 31.0M 4s
19150K .......... .......... .......... .......... .......... 75% 19.4M 4s
19200K .......... .......... .......... .......... .......... 76% 2.86M 4s
19250K .......... .......... .......... .......... .......... 76%  530K 4s
19300K .......... .......... .......... .......... .......... 76% 4.17M 4s
19350K .......... .......... .......... .......... .......... 76% 22.7M 4s
19400K .......... .......... .......... .......... .......... 76% 32.2M 4s
19450K .......... .......... .......... .......... .......... 77% 12.6M 3s
19500K .......... .......... .......... .......... .......... 77%  555K 3s
19550K .......... .......... .......... .......... .......... 77% 2.00M 3s
19600K .......... .......... .......... .......... .......... 77% 6.86M 3s
19650K .......... .......... .......... .......... .......... 77% 24.7M 3s
19700K .......... .......... .......... .......... .......... 78% 33.6M 3s
19750K .......... .......... .......... .......... .......... 78% 2.84M 3s
19800K .......... .......... .......... .......... .......... 78%  530K 3s
19850K .......... .......... .......... .......... .......... 78% 6.11M 3s
19900K .......... .......... .......... .......... .......... 78% 7.70M 3s
19950K .......... .......... .......... .......... .......... 79% 47.5M 3s
20000K .......... .......... .......... .......... .......... 79% 11.6M 3s
```

```
20050K ..... ..... ..... ..... ..... 79% 1024K 3s
20100K ..... ..... ..... ..... ..... 79%  776K 3s
20150K ..... ..... ..... ..... ..... 79% 5.19M 3s
20200K ..... ..... ..... ..... ..... 80% 81.5M 3s
20250K ..... ..... ..... ..... ..... 80% 43.4M 3s
20300K ..... ..... ..... ..... ..... 80% 2.88M 3s
20350K ..... ..... ..... ..... ..... 80%  529K 3s
20400K ..... ..... ..... ..... ..... 80% 6.19M 3s
20450K ..... ..... ..... ..... ..... 81% 6.76M 3s
20500K ..... ..... ..... ..... ..... 81% 58.5M 3s
20550K ..... ..... ..... ..... ..... 81% 12.3M 3s
20600K ..... ..... ..... ..... ..... 81% 1.51M 3s
20650K ..... ..... ..... ..... ..... 81%  619K 3s
20700K ..... ..... ..... ..... ..... 82% 5.38M 3s
20750K ..... ..... ..... ..... ..... 82% 40.0M 3s
20800K ..... ..... ..... ..... ..... 82% 51.5M 3s
20850K ..... ..... ..... ..... ..... 82% 13.3M 3s
20900K ..... ..... ..... ..... ..... 82%  509K 3s
20950K ..... ..... ..... ..... ..... 83% 2.89M 3s
21000K ..... ..... ..... ..... ..... 83% 6.67M 3s
21050K ..... ..... ..... ..... ..... 83% 40.1M 2s
21100K ..... ..... ..... ..... ..... 83% 16.8M 2s
21150K ..... ..... ..... ..... ..... 83% 3.17M 2s
21200K ..... ..... ..... ..... ..... 84%  528K 2s
21250K ..... ..... ..... ..... ..... 84% 6.20M 2s
21300K ..... ..... ..... ..... ..... 84% 8.17M 2s
21350K ..... ..... ..... ..... ..... 84% 17.7M 2s
21400K ..... ..... ..... ..... ..... 84% 24.9M 2s
21450K ..... ..... ..... ..... ..... 85% 1.00M 2s
21500K ..... ..... ..... ..... ..... 85%  771K 2s
21550K ..... ..... ..... ..... ..... 85% 5.64M 2s
21600K ..... ..... ..... ..... ..... 85% 12.3M 2s
21650K ..... ..... ..... ..... ..... 85%  112M 2s
21700K ..... ..... ..... ..... ..... 86% 3.08M 2s
21750K ..... ..... ..... ..... ..... 86%  597K 2s
21800K ..... ..... ..... ..... ..... 86% 2.91M 2s
21850K ..... ..... ..... ..... ..... 86% 6.65M 2s
21900K ..... ..... ..... ..... ..... 86% 12.6M 2s
21950K ..... ..... ..... ..... ..... 87% 53.0M 2s
22000K ..... ..... ..... ..... ..... 87% 3.17M 2s
22050K ..... ..... ..... ..... ..... 87%  531K 2s
22100K ..... ..... ..... ..... ..... 87% 5.84M 2s
22150K ..... ..... ..... ..... ..... 87% 8.21M 2s
22200K ..... ..... ..... ..... ..... 88% 10.0M 2s
22250K ..... ..... ..... ..... ..... 88%  111M 2s
22300K ..... ..... ..... ..... ..... 88% 3.32M 2s
22350K ..... ..... ..... ..... ..... 88%  531K 2s
22400K ..... ..... ..... ..... ..... 88% 4.38M 2s
22450K ..... ..... ..... ..... ..... 89% 17.7M 2s
22500K ..... ..... ..... ..... ..... 89% 10.1M 2s
22550K ..... ..... ..... ..... ..... 89% 29.9M 2s
22600K ..... ..... ..... ..... ..... 89% 1.03M 2s
22650K ..... ..... ..... ..... ..... 89%  779K 2s
22700K ..... ..... ..... ..... ..... 90% 5.55M 1s
22750K ..... ..... ..... ..... ..... 90% 24.9M 1s
22800K ..... ..... ..... ..... ..... 90% 10.3M 1s
22850K ..... ..... ..... ..... ..... 90% 52.2M 1s
22900K ..... ..... ..... ..... ..... 90%  512K 1s
22950K ..... ..... ..... ..... ..... 90% 3.46M 1s
23000K ..... ..... ..... ..... ..... 91% 5.22M 1s
23050K ..... ..... ..... ..... ..... 91% 14.6M 1s
23100K ..... ..... ..... ..... ..... 91% 18.8M 1s
23150K ..... ..... ..... ..... ..... 91% 3.36M 1s
23200K ..... ..... ..... ..... ..... 91%  541K 1s
23250K ..... ..... ..... ..... ..... 92% 5.19M 1s
23300K ..... ..... ..... ..... ..... 92% 8.97M 1s
23350K ..... ..... ..... ..... ..... 92% 12.7M 1s
23400K ..... ..... ..... ..... ..... 92% 13.2M 1s
23450K ..... ..... ..... ..... ..... 92% 3.80M 1s
23500K ..... ..... ..... ..... ..... 93%  540K 1s
23550K ..... ..... ..... ..... ..... 93% 4.97M 1s
23600K ..... ..... ..... ..... ..... 93% 9.17M 1s
23650K ..... ..... ..... ..... ..... 93% 13.6M 1s
23700K ..... ..... ..... ..... ..... 93% 8.62M 1s
23750K ..... ..... ..... ..... ..... 94% 4.47M 1s
23800K ..... ..... ..... ..... ..... 94%  536K 1s
23850K ..... ..... ..... ..... ..... 94% 5.21M 1s
23900K ..... ..... ..... ..... ..... 94% 9.52M 1s
23950K ..... ..... ..... ..... ..... 94% 6.19M 1s
24000K ..... ..... ..... ..... ..... 95% 24.5M 1s
24050K ..... ..... ..... ..... ..... 95% 4.77M 1s
24100K ..... ..... ..... ..... ..... 95%  519K 1s
24150K ..... ..... ..... ..... ..... 95% 8.05M 1s
```

```
 24200K .......... .......... .......... .......... .......... 95% 8.30M 1s
 24250K .......... .......... .......... .......... .......... 96% 6.79M 1s
 24300K .......... .......... .......... .......... .......... 96% 14.7M 1s
 24350K .......... .......... .......... .......... .......... 96% 1.96M 1s
 24400K .......... .......... .......... .......... .......... 96%  621K 0s
 24450K .......... .......... .......... .......... .......... 96% 6.38M 0s
 24500K .......... .......... .......... .......... .......... 97% 10.8M 0s
 24550K .......... .......... .......... .......... .......... 97% 6.81M 0s
 24600K .......... .......... .......... .......... .......... 97% 13.8M 0s
 24650K .......... .......... .......... .......... .......... 97% 1.18M 0s
 24700K .......... .......... .......... .......... .......... 97%  755K 0s
 24750K .......... .......... .......... .......... .......... 98% 9.82M 0s
 24800K .......... .......... .......... .......... .......... 98% 10.3M 0s
 24850K .......... .......... .......... .......... .......... 98% 6.37M 0s
 24900K .......... .......... .......... .......... .......... 98% 16.3M 0s
 24950K .......... .......... .......... .......... .......... 98% 1.18M 0s
 25000K .......... .......... .......... .......... .......... 99%  755K 0s
 25050K .......... .......... .......... .......... .......... 99% 10.5M 0s
 25100K .......... .......... .......... .......... .......... 99% 9.45M 0s
 25150K .......... .......... .......... .......... .......... 99% 6.48M 0s
 25200K .......... .......... .......... .......... .......... 99% 15.5M 0s
 25250K .......... .......... .....                           100% 3.71M=14s

2022-02-11 12:48:12 (1.72 MB/s) - ‘dbCAN-fam-V9.tar.gz’ saved [25882327/25882327]
```

Using hhsearch

hhblits and hhsearch are the main functions in hhsuite which identify homologous proteins. They do this by calculating a profile hidden Markov model (HMM) from a given alignment and searching over a reference HMM proteome database using the Viterbi algorithm. Then the most similar HMMs are realigned and output to the user. To learn more, check out the original paper in the references above.

Run a function from hhsuite with no parameters to read its documentation.

In [37]:
```
!hhsearch
```

```
HHsearch 3.3.0
Search a database of HMMs with a query alignment or query HMM
(c) The HH-suite development team
Steinegger M, Meier M, Mirdita M, Vöhringer H, Haunsberger S J, and Söding J (2019)
HH-suite3 for fast remote homology detection and deep protein annotation.
BMC Bioinformatics, doi:10.1186/s12859-019-3019-7

Usage: hhsearch -i query -d database [options]
 -i <file>      input/query multiple sequence alignment (a2m, a3m, FASTA) or HMM
Options:
 -d <name>      database name (e.g. uniprot20_29Feb2012)
                Multiple databases may be specified with '-d <db1> -d <db2> ...'
 -e     [0,1]   E-value cutoff for inclusion in result alignment (def=0.001)

Input alignment format:
 -M a2m         use A2M/A3M (default): upper case = Match; lower case = Insert;
               '-' = Delete; '.' = gaps aligned to inserts (may be omitted)
 -M first       use FASTA: columns with residue in 1st sequence are match states
 -M [0,100]     use FASTA: columns with fewer than X% gaps are match states
 -tags/-notags  do NOT / do neutralize His-, C-myc-, FLAG-tags, and trypsin
                recognition sequence to background distribution (def=-notags)

Output options:
 -o <file>      write results in standard format to file (default=<infile.hhr>)
 -oa3m <file>   write result MSA with significant matches in a3m format
 -blasttab <name> write result in tabular BLAST format (compatible to -m 8 or -outfmt 6 output)
                  1     2      3           4      5         6        7      8    9      10   11   12
                  query target #match/tLen alnLen #mismatch #gapOpen qstart qend tstart tend eval score
 -add_cons      generate consensus sequence as master sequence of query MSA (default=don't)
 -hide_cons     don't show consensus sequence in alignments (default=show)
 -hide_pred     don't show predicted 2ndary structure in alignments (default=show)
 -hide_dssp     don't show DSSP 2ndary structure in alignments (default=show)
 -show_ssconf   show confidences for predicted 2ndary structure in alignments
Filter options applied to query MSA, database MSAs, and result MSA
 -all           show all sequences in result MSA; do not filter result MSA
 -id   [0,100]  maximum pairwise sequence identity (def=90)
 -diff [0,inf[  filter MSAs by selecting most diverse set of sequences, keeping
                at least this many seqs in each MSA block of length 50
                Zero and non-numerical values turn off the filtering. (def=100)
 -cov  [0,100]  minimum coverage with master sequence (%) (def=0)
 -qid  [0,100]  minimum sequence identity with master sequence (%) (def=0)
 -qsc  [0,100]  minimum score per column with master sequence (default=-20.0)
 -neff [1,inf]  target diversity of multiple sequence alignment (default=off)
 -mark          do not filter out sequences marked by ">@"in their name line

HMM-HMM alignment options:
 -norealign          do NOT realign displayed hits with MAC algorithm (def=realign)
 -ovlp <int>         banded alignment: forbid <ovlp> largest diagonals |i-j| of DP matrix (def=0)
 -mact [0,1[         posterior prob threshold for MAC realignment controlling greedi-
                     ness at alignment ends: 0:global >0.1:local (default=0.35)
 -glob/-loc          use global/local alignment mode for searching/ranking (def=local)
Other options:
 -v <int>       verbose mode: 0:no screen output  1:only warnings  2: verbose (def=2)
 -cpu <int>     number of CPUs to use (for shared memory SMPs) (default=2)

An extended list of options can be obtained by calling 'hhblits -h all'

Example: hhsearch -i a.1.1.1.a3m -d scop70_1.71

Download databases from <http://wwwuser.gwdg.de/~compbiol/data/hhsuite/databases/hhsuite_dbs/>.
- 12:48:13.127 ERROR: Database is missing (see -d)!
```

Let's do an example. Say we have a protein which we want to compare to a MSA in order to identify any homologous regions. For this we can use hhsearch.

Now let's take some protein sequence and search through the dbCAN database to see if we can find any potential homologous regions. First we will specify the sequence and save it as a FASTA file or a3m file in order to be readable by hhsearch. I pulled this sequence from the example query.a3m in the hhsuite data directory.

In [38]:
```
with open('protein.fasta', 'w') as f:
    f.write("""
>Uncharacterized bovine protein (Fragment)
--PAGGQCtgiWHLLTRPLRP--QGRLPGLRVKYVFLVWLGVFAGSWMAYTHYSSYAELCRGHICQVVICDQFRKGIISGSICQDLCHLHQVEWRTCLSSVPGQQVYSGLW
""")
```

Then we can call hhsearch, specifying the query sequence with the -i flag, the database to search through with -d, and the output with -o.

In [39]:
```
from deepchem.utils import sequence_utils
dataset_path = 'protein.fasta'
```

```
data_dir = 'hh/databases'
results = sequence_utils.hhsearch(dataset_path,database='dbCAN-fam-V9', data_dir=data_dir)
```

```
- 12:48:13.301 INFO: Search results will be written to /home/tony/github/deepchem/examples/tutorials/protein.hhr

- 12:48:13.331 INFO: /home/tony/github/deepchem/examples/tutorials/protein.fasta is in A2M, A3M or FASTA format

- 12:48:13.331 WARNING: Input alignment /home/tony/github/deepchem/examples/tutorials/protein.fasta looks like a
ligned FASTA instead of A2M/A3M format. Consider using '-M first' or '-M 50'

- 12:48:13.331 INFO: NOTE: Use the '-add_cons' option to calculate a consensus sequence as first sequence of the
alignment with hhconsensus or hhmake.

- 12:48:13.331 INFO: Searching 683 database HHMs without prefiltering

- 12:48:13.332 INFO: Iteration 1

- 12:48:13.420 INFO: Scoring 683 HMMs using HMM-HMM Viterbi alignment

- 12:48:13.460 INFO: Alternative alignment: 0

- 12:48:13.611 INFO: 683 alignments done

- 12:48:13.612 INFO: Alternative alignment: 1

- 12:48:13.625 INFO: 38 alignments done

- 12:48:13.625 INFO: Alternative alignment: 2

- 12:48:13.629 INFO: 3 alignments done

- 12:48:13.629 INFO: Alternative alignment: 3

- 12:48:13.655 INFO: Premerge done

- 12:48:13.656 INFO: Realigning 10 HMM-HMM alignments using Maximum Accuracy algorithm

- 12:48:13.692 INFO: 0 sequences belonging to 0 database HMMs found with an E-value < 0.001

- 12:48:13.692 INFO: Number of effective sequences of resulting query HMM: Neff = 1
```

In [40]:
```
#open the results and print them
f = open("protein.hhr", "r")
print(f.read())
```

```
Query         Uncharacterized bovine protein (Fragment)
Match_columns 431
No_of_seqs    1 out of 1
Neff          1
Searched_HMMs 683
Date          Fri Feb 11 12:48:13 2022
Command       hhsearch -i /home/tony/github/deepchem/examples/tutorials/protein.fasta -d hh/databases/dbCAN-fam-
V9 -oa3m /home/tony/github/deepchem/examples/tutorials/results.a3m -cpu 4 -e 0.001

 No Hit                             Prob E-value P-value  Score    SS Cols Query HMM  Template HMM
  1 ABJ15796.1|231-344|9.6e-33       8.2     2.9  0.0042   25.2   0.0   13  224-236    40-52  (116)
  2 lcl|consensus                    5.1     5.2  0.0076   17.1   0.0   14  182-195     1-14  (21)
  3 ABW08129.1|GT4|GT97||563-891     4.8     5.7  0.0084   26.6   0.0   46  104-150    93-140 (329)
  4 AEO62162.1|AA13||19-250          4.6       6  0.0087   25.5   0.0   18  330-347   139-156 (232)
  5 BAF49076.1|GH5_26.hmm|8.3e-11|   2.4      13    0.02   21.9   0.0   12  287-298    45-56  (141)
  6 BBD44721.1 Hypothetical protei   2.3      14    0.02   25.7   0.0   81  110-221   326-406 (552)
  7 AAU92474.1|CBM2|2-82|1.9e-23     2.3      14    0.02   19.1   0.0   19  222-240    13-33  (104)
  8 BAX82587.1 hypothetical protei   2.3      14   0.021   25.7   0.0   25  104-128   466-490 (656)
  9 AHE46274.1|GH13_13.hmm|1.6e-20   2.0      16   0.024   24.1   0.0   45  143-199    99-143 (393)
 10 ACF55060.1|GH13_13.hmm|2.5e-47   1.9      17   0.025   23.2   0.0   22  144-165    74-95  (330)

No 1
>ABJ15796.1|231-344|9.6e-33
Probab=8.16  E-value=2.9  Score=25.22  Aligned_cols=13  Identities=46%  Similarity=0.795  Sum_probs=10.2  Templa
te_Neff=3.400

Q Uncharacterize  223 YCGDLYVTEGVPL  235 (430)
Q Consensus       224 ycgdlyvtegvpl  236 (431)
                      --||||.||||--
T Consensus        40 I~Gnlyi~eGVG~   52 (116)
T ABJ15796.1|231   40 INGNLYIAEGVGE   52 (116)
Confidence            3599999999853


No 2
>lcl|consensus
Probab=5.13  E-value=5.2  Score=17.13  Aligned_cols=14  Identities=29%  Similarity=0.437  Sum_probs=10.2  Templa
te_Neff=4.300
```

```
Q Uncharacterize  181 DFNKDNRVSLAEAK  194 (430)
Q Consensus       182 dfnkdnrvslaeak  195 (431)
                      |.|.|++|+-.++-
T Consensus         1 DvN~DG~Vna~D~~   14 (21)
T lcl|consensus_    1 DVNGDGKVNALDLA   14 (21)
Confidence            67888888766553


No 3
>ABW08129.1|GT4|GT97||563-891
Probab=4.78  E-value=5.7  Score=26.58  Aligned_cols=46  Identities=20%  Similarity=0.367  Sum_probs=28.5  Templa
te_Neff=1.500

Q Uncharacterize  103 YSGLWQGKEVTIKCGIEESLN--SKAGSDGAPRRELVLFDKPSRGTSIK  149 (430)
Q Consensus       104 ysglwqgkevtikcgieesln--skagsdgaprrelvlfdkpsrgtsik  150 (431)
                      ..|+|. |...+.-.|.-...  .+..-|.+|..|-++||-|.||-+-+
T Consensus        93 ~~G~W~-~~~~~~~~~i~~~~DheG~r~m~~~~~~T~i~e~~Rk~~~~~  140 (329)
T ABW08129.1|GT4   93 FTGKWE-KHFQTSPKIDYRFDHEGKRSMDDVFSEETFIMEFPRKNGIDK  140 (329)
Confidence            457774 33333333433333  455567788888899999888887654


No 4
>AEO62162.1|AA13||19-250
Probab=4.61  E-value=6  Score=25.50  Aligned_cols=18  Identities=39%  Similarity=0.936  Sum_probs=14.8  Template
_Neff=1.600

Q Uncharacterize  329 RGRRCEHSADCTYGRDCR  346 (430)
Q Consensus       330 rgrrcehsadctygrdcr  347 (431)
                      .|..|..|+||+-|..|-
T Consensus       139 ~Gq~C~y~pDC~~gq~C~  156 (232)
T AEO62162.1|AA1  139 SGQTCGYSPDCSPGQPCW  156 (232)
Confidence            467899999999998774


No 5
>BAF49076.1|GH5_26.hmm|8.3e-11|182-335
Probab=2.39  E-value=13  Score=21.92  Aligned_cols=12  Identities=33%  Similarity=0.720  Sum_probs=9.5  Template
_Neff=1.900

Q Uncharacterize  286 HGAYGNFYMCET  297 (430)
Q Consensus       287 hgaygnfymcet  298 (431)
                      .|+|++|||-..
T Consensus        45 ~G~yn~~Y~l~s   56 (141)
T BAF49076.1|GH5   45 QGTYNGNYMLTS   56 (141)
Confidence            478999998654


No 6
>BBD44721.1 Hypothetical protein PEIBARAKI_4714 [Petrimonas sp. IBARAKI]
Probab=2.34  E-value=14  Score=25.75  Aligned_cols=81  Identities=23%  Similarity=0.240  Sum_probs=46.6  Templat
e_Neff=3.400

Q Uncharacterize  109 GKEVTIKCGIEESLNSKAGSDGAPRRELVLFDKPSRGTSIKEFREMTLSFLKANLGDLPSLPALVGRVLLMADFNKDNRV  188 (430
)
Q Consensus       110 gkevtikcgieeslnskagsdgaprrelvlfdkpsrgtsikefremtlsflkanlgdlpslpalvgrvllmadfnkdnrv  189 (431
)
                      |..|+|-.|+|+|..+-+.             +++-.-.|+.-+|-+.+++|.+--.          .|-+   ||-.+-+
T Consensus       326 g~~V~Iya~l~~~~~~~~~~~~~~~~~~~~~S~~~~Rg~Aa~~L~rGAd----------GIyl---FN~f~~~  379 (552
)
T BBD44721.1_con  326 GTGVKIYAGLEDARAPDP------------STRRETNSLEAYRGRAANALSRGAD----------GIYL---FNYFYPP  379 (552
)
Confidence            777888888888744332             44555668888888888765322          2333   4443332


Q Uncharacterize  189 SLAEAKSVWALLQRNEFLLLLSLQEKEHASRL  220 (430)
Q Consensus       190 slaeaksvwallqrnefllllslqekehasrl  221 (431)
                      ..     --.|||.-.=+-.|.-|+|.|+-..
T Consensus       380 ~~-----~~~llrelgd~~~L~~~~K~y~~s~  406 (552)
T BBD44721.1_con  380 QM-----RSPLLRELGDLETLATQEKLYALSI  406 (552)
Confidence            11     12334433334455667888765 43


No 7
>AAU92474.1|CBM2|2-82|1.9e-23
Probab=2.33  E-value=14  Score=19.07  Aligned_cols=19  Identities=26%  Similarity=0.562  Sum_probs=14.3  Templat
e_Neff=6.600

Q Uncharacterize  221 LGYCGDLYVT--EGVPLSSWP  239 (430)
Q Consensus       222 lgycgdlyvt--egvplsswp  240 (431)
                      -||++++-|+  ...+++.|-
T Consensus        13 ~Gf~~~v~vtN~~~~~i~~W~   33 (104)
T AAU92474.1|CBM   13 GGFQANVTVTNTGSSAISGWT   33 (104)
```

```
Confidence            3789998888  567777774


No 8
>BAX82587.1 hypothetical protein ALGA_4297 [Marinifilaceae bacterium SPP2]
Probab=2.28  E-value=14  Score=25.65  Aligned_cols=25  Identities=40%  Similarity=0.314  Sum_probs=21.4  Templat
e_Neff=1.500

Q Uncharacterize  103 YSGLWQGKEVTIKCGIEESLNSKAG  127 (430)
Q Consensus       104 ysglwqgkevtikcgieeslnskag  128 (431)
                      -+.+||+|.+.||..||.|-|-+--
T Consensus       466 ~kd~~~tk~~sik~kietSenFtl~  490 (656)
T BAX82587.1_con  466 NKDLNQTKQVSIKTKIETSENFTLS  490 (656)
Confidence            57899999999999999876543


No 9
>AHE46274.1|GH13_13.hmm|1.6e-201|415-835
Probab=2.04  E-value=16  Score=24.14  Aligned_cols=45  Identities=29%  Similarity=0.332  Sum_probs=25.7  Templat
e_Neff=3.900

Q Uncharacterize  142 PSRGTSIKEFREMTLSFLKANLGDLPSLPALVGRVLLMADFNKDNRVSLAEAKSVWA  198 (430)
Q Consensus       143 psrgtsikefremtlsflkanlgdlpslpalvgrvllmadfnkdnrvslaeaksvwa  199 (431)
                      |.-.+-|+|||+|.-+.           -+.=-||.+=.=||--+-.-+ .++||..
T Consensus        99 ~~g~~Ri~EfR~MV~al----------h~~GlrVv~DVVyNHT~~sg~-~~~~SVlD  143 (393)
T AHE46274.1|GH1   99 PDGVARIKEFRAMVQAL----------HAMGLRVVMDVVYNHTAASGQ-YDNSVLD  143 (393)
Confidence            34445589999998653           222235655555676655544 3345543


No 10
>ACF55060.1|GH13_13.hmm|2.5e-47|336-542
Probab=1.94  E-value=17  Score=23.24  Aligned_cols=22  Identities=23%  Similarity=0.371  Sum_probs=17.6  Templat
e_Neff=4.500

Q Uncharacterize  143 SRGTSIKEFREMTLSFLKANLG  164 (430)
Q Consensus       144 srgtsikefremtlsflkanlg  165 (431)
                      +.-+.|+||++|...+=++.++
T Consensus        74 dp~~RI~E~K~mI~~lH~~GI~   95 (330)
T ACF55060.1|GH1   74 DPYGRIREFKQMIQALHDAGIR   95 (330)
Confidence            44567999999988887765
```

Two files are output and saved to the dataset directory, results.hhr and results.a3m. results.hhr is the hhsuite results file, which is a summary of the results. results.a3m is the actual MSA file.

In the hhr file, the 'Prob' column describes the estimated probability of the query sequence being at least partially homologous to the template. Probabilities of 95% or more are nearly certain, and probabilities of 30% or more call for closer consideration. The E value tells you how many random matches with a better score would be expected if the searched database was unrelated to the query sequence. These results show that none of the sequences align well with our randomly chosen protein, which is to be expected because our query sequence was chosen at random.

Now let's check the results if we use a sequence that we know will align with something in the dbCAN database. I pulled this protein from the dockerin.faa file in dbCAN.

In [41]:
```
with open('protein2.fasta', 'w') as f:
    f.write(""">dockerin,22,NCBI-Bacteria,gi|125972715|ref|YP_001036625.1|,162-245,0.033
SCADLNGDGKITSSDYNLLKRYILHLIDKFPIGNDETDEGINDGFNDETDEDINDSFIEANSKFAFDIFKQISKDEQGKNVFIS
""")

dataset_path = 'protein2.fasta'
sequence_utils.hhsearch(dataset_path,database='dbCAN-fam-V9', data_dir=data_dir)

#open the results and print them
f = open("protein2.hhr", "r")
print(f.read())
```

```
- 12:48:13.823 INFO: Search results will be written to /home/tony/github/deepchem/examples/tutorials/protein2.hh
r

- 12:48:13.851 INFO: /home/tony/github/deepchem/examples/tutorials/protein2.fasta is in A2M, A3M or FASTA format

- 12:48:13.852 INFO: Searching 683 database HHMs without prefiltering

- 12:48:13.852 INFO: Iteration 1

- 12:48:13.873 INFO: Scoring 683 HMMs using HMM-HMM Viterbi alignment

- 12:48:13.913 INFO: Alternative alignment: 0

- 12:48:13.979 INFO: 683 alignments done

- 12:48:13.979 INFO: Alternative alignment: 1

- 12:48:13.982 INFO: 10 alignments done

- 12:48:13.982 INFO: Alternative alignment: 2

- 12:48:13.984 INFO: 3 alignments done

- 12:48:13.984 INFO: Alternative alignment: 3

- 12:48:13.986 INFO: 3 alignments done
```

```
Query         dockerin,22,NCBI-Bacteria,gi|125972715|ref|YP_001036625.1|,162-245,0.033
Match_columns 84
No_of_seqs    1 out of 1
Neff          1
Searched_HMMs 683
Date          Fri Feb 11 12:48:14 2022
Command       hhsearch -i /home/tony/github/deepchem/examples/tutorials/protein2.fasta -d hh/databases/dbCAN-fam
-V9 -oa3m /home/tony/github/deepchem/examples/tutorials/results.a3m -cpu 4 -e 0.001

 No Hit                             Prob E-value P-value  Score    SS Cols Query HMM  Template HMM
  1 lcl|consensus                   97.0 5.9E-08 8.7E-11   43.5   0.0   21    4-24      1-21  (21)
  2 ABN51673.1|GH124|2-334|2.6e-21  92.5 0.00033 4.8E-07   45.5   0.0   68    1-75     21-88  (318)
  3 AAK20911.1|PL11|47-657|0        15.7     1.1  0.0017   27.6   0.0   14    1-14    329-342 (606)
  4 AGE62576.1|PL11_1.hmm|0|1-596   10.2     2.1  0.0031   26.0   0.0   13    1-13    118-130 (602)
  5 AAZ21803.1|GH103|26-328|1.7e-8   9.3     2.4  0.0035   22.4   0.0   10    4-13    175-184 (293)
  6 AGE62576.1|PL11_1.hmm|0|1-596    5.5     4.8   0.007   23.9   0.0   12    1-12    329-340 (602)
  7 AAK20911.1|PL11|47-657|0         5.5     4.8   0.007   23.8   0.0   13    1-13    118-130 (606)
  8 APU21542.1|PL11_2.hmm|1.4e-162   4.9     5.6  0.0082   23.5   0.0   14    2-15    318-331 (579)
  9 AAK20911.1|PL11|47-657|0         4.7     5.8  0.0084   23.4   0.0   10    3-12    184-193 (606)
 10 AGE62576.1|PL11_1.hmm|0|1-596    4.6       6  0.0088   23.3   0.0    7    4-10    185-191 (602)

No 1
>lcl|consensus
Probab=97.03  E-value=5.9e-08  Score=43.48  Aligned_cols=21  Identities=57%  Similarity=1.061  Sum_probs=20.1  T
emplate_Neff=4.300

Q dockerin,22,NC    4 DLNGDGKITSSDYNLLKRYIL   24 (84)
Q Consensus         4 dlngdgkitssdynllkryil   24 (84)
                      |+|+||+|+..|+.++|||+|
T Consensus         1 DvN~DG~Vna~D~~~l~~~l~   21 (21)
T lcl|consensus_    1 DVNGDGKVNALDLALLKKYLL   21 (21)
Confidence            899999999999999999986


No 2
>ABN51673.1|GH124|2-334|2.6e-219
Probab=92.52  E-value=0.00033  Score=45.54  Aligned_cols=68  Identities=31%  Similarity=0.523  Sum_probs=51.6  T
emplate_Neff=1.400

Q dockerin,22,NC    1 SCADLNGDGKITSSDYNLLKRYILHLIDKFPIGNDETDEGINDGFNDETDEDINDSFIEANSKFAFDIFKQISKD   75 (84)
Q Consensus         1 scadlngdgkitssdynllkryilhlidkfpigndetdegindgfndetdedindsfieanskfafdifkqiskd   75 (84)
                      ++||+||||+|+||||+|||| ||++|++||+++|+|++|..      .-|+|.--.+-.++.....+.+.|.
T Consensus        21 v~GD~n~dgvv~isd~vl~k~~~l~~~a~~~a~~d~w~g~vN~d------d~I~D~d~~~~kryll~mir~~pk~   88 (318)
T ABN51673.1|GH1   21 VIGDVNADGVVNISDYVLMKR-ILRIIADFPADDDMWVGDVNGD------DVINDIDCNYLKRYLLHMIREFPKN   88 (318)
Confidence            489999999999999999 99999999999999999854      333333333344444444444443


No 3
>AAK20911.1|PL11|47-657|0
Probab=15.69  E-value=1.1  Score=27.56  Aligned_cols=14  Identities=50%  Similarity=0.641  Sum_probs=10.4  Templ
ate_Neff=3.500

Q dockerin,22,NC    1 SCADLNGDGKITSS   14 (84)
Q Consensus         1 scadlngdgkitss   14 (84)
                      |++|+|+||+=.|-
T Consensus       329 svaDVDgDGkDEIi  342 (606)
```

```
T AAK20911.1|PL1  329 SVADVDGDGKDEII  342 (606)
Confidence            57889998886553


No 4
>AGE62576.1|PL11_1.hmm|0|1-596
Probab=10.22  E-value=2.1  Score=26.01  Aligned_cols=13  Identities=46%  Similarity=0.772  Sum_probs=10.8  Templ
ate_Neff=3.300

Q dockerin,22,NC    1 SCADLNGDGKITS   13 (84)
Q Consensus         1 scadlngdgkits   13 (84)
                      |+||||+||...|
T Consensus       118 SVGDLDGDG~YEi  130 (602)
T AGE62576.1|PL1  118 SVGDLDGDGEYEI  130 (602)
Confidence            6899999998654


No 5
>AAZ21803.1|GH103|26-328|1.7e-83
Probab=9.26  E-value=2.4  Score=22.41  Aligned_cols=10  Identities=40%  Similarity=0.833  Sum_probs=9.2  Templat
e_Neff=5.600

Q dockerin,22,NC    4 DLNGDGKITS   13 (84)
Q Consensus         4 dlngdgkits   13 (84)
                      |.|+||++++
T Consensus       175 D~DgDG~~Dl  184 (293)
T AAZ21803.1|GH1  175 DFDGDGRRDL  184 (293)
Confidence            8899999997


No 6
>AGE62576.1|PL11_1.hmm|0|1-596
Probab=5.50  E-value=4.8  Score=23.90  Aligned_cols=12  Identities=58%  Similarity=0.847  Sum_probs=7.5  Templat
e_Neff=3.300

Q dockerin,22,NC    1 SCADLNGDGKIT   12 (84)
Q Consensus         1 scadlngdgkit   12 (84)
                      |+||+|+||+=.
T Consensus       329 svaDVDgDG~DE  340 (602)
T AGE62576.1|PL1  329 SVADVDGDGKDE  340 (602)
Confidence            467777777633


No 7
>AAK20911.1|PL11|47-657|0
Probab=5.47  E-value=4.8  Score=23.84  Aligned_cols=13  Identities=46%  Similarity=0.772  Sum_probs=10.6  Templa
te_Neff=3.500

Q dockerin,22,NC    1 SCADLNGDGKITS   13 (84)
Q Consensus         1 scadlngdgkits   13 (84)
                      |+||||+||...+
T Consensus       118 SVGDLDGDG~yEi  130 (606)
T AAK20911.1|PL1  118 SVGDLDGDGEYEI  130 (606)
Confidence            6899999998654


No 8
>APU21542.1|PL11_2.hmm|1.4e-162|44-417
Probab=4.86  E-value=5.6  Score=23.51  Aligned_cols=14  Identities=50%  Similarity=0.715  Sum_probs=9.4  Templat
e_Neff=2.600

Q dockerin,22,NC    2 CADLNGDGKITSSD   15 (84)
Q Consensus         2 cadlngdgkitssd   15 (84)
                      +.|+|+||+=.+++
T Consensus       318 ~~DvD~DG~DEi~~  331 (579)
T APU21542.1|PL1  318 IVDVDGDGKDEISD  331 (579)
Confidence            45777777766655


No 9
>AAK20911.1|PL11|47-657|0
Probab=4.74  E-value=5.8  Score=23.38  Aligned_cols=10  Identities=50%  Similarity=0.896  Sum_probs=5.6  Templat
e_Neff=3.500

Q dockerin,22,NC    3 ADLNGDGKIT   12 (84)
Q Consensus         3 adlngdgkit   12 (84)
                      -|+|+|||-.
T Consensus       184 yD~DGDGkAE  193 (606)
T AAK20911.1|PL1  184 YDFDGDGKAE  193 (606)
Confidence            3666666543


No 10
```

```
>AGE62576.1|PL11_1.hmm|0|1-596
Probab=4.58  E-value=6  Score=23.30  Aligned_cols=7  Identities=71%  Similarity=1.426  Sum_probs=0.0  Template_N
eff=3.300

Q dockerin,22,NC    4 DLNGDGK   10 (84)
Q Consensus         4 dlngdgk   10 (84)
                      |||||||
T Consensus       185 D~DGDGk  191 (602)
T AGE62576.1|PL1  185 DFDGDGK  191 (602)
```

```
- 12:48:14.063 INFO: Premerge done

- 12:48:14.063 INFO: Realigning 10 HMM-HMM alignments using Maximum Accuracy algorithm

- 12:48:14.084 INFO: 4 sequences belonging to 4 database HMMs found with an E-value < 0.001

- 12:48:14.084 INFO: Number of effective sequences of resulting query HMM: Neff = 1.39047
```

As you can see, there are 2 sequences which are a match for our query sequence.

Using hhblits

hhblits works in much the same way as hhsearch, but it is much faster and slightly less sensitive. This would be more suited to searching very large databases, or producing a MSA with multiple sequences instead of just one. Let's make use of that by using our query sequence to create an MSA. We could then use that MSA, with its family of proteins, to search a larger database for potential matches. This will be much more effective than searching a large database with a single sequence.

We will use the same dbCAN database. I will pull a glycoside hydrolase protein from UnipProt, so it will likely be related to some proteins in dbCAN, which has carbohydrate-active enzymes.

The option -oa3m will tell hhblits to output an MSA as an a3m file. The -n option specifies the number of iterations. This is recommended to keep between 1 and 4, we will try 2.

In [42]:
```
!wget -O protein3.fasta https://www.uniprot.org/uniprot/G8M3C3.fasta

dataset_path = 'protein3.fasta'
sequence_utils.hhblits(dataset_path,database='dbCAN-fam-V9', data_dir=data_dir)

#open the results and print them
f = open("protein3.hhr", "r")
print(f.read())
```

```
--2022-02-11 12:48:14--  https://www.uniprot.org/uniprot/G8M3C3.fasta
Resolving www.uniprot.org (www.uniprot.org)... 193.62.193.81
Connecting to www.uniprot.org (www.uniprot.org)|193.62.193.81|:443... connected.
HTTP request sent, awaiting response... 200
Length: 897 [text/plain]
Saving to: ‘protein3.fasta’

protein3.fasta      100%[===================>]     897  --.-KB/s    in 0s

2022-02-11 12:48:15 (1.70 GB/s) - ‘protein3.fasta’ saved [897/897]
```

```
- 12:48:15.242 WARNING: Ignoring unknown option -n

- 12:48:15.242 WARNING: Ignoring unknown option 2

- 12:48:15.242 INFO: Search results will be written to /home/tony/github/deepchem/examples/tutorials/protein3.hh
r

- 12:48:15.270 INFO: /home/tony/github/deepchem/examples/tutorials/protein3.fasta is in A2M, A3M or FASTA format

- 12:48:15.271 INFO: Searching 683 database HHMs without prefiltering

- 12:48:15.271 INFO: Iteration 1

- 12:48:15.424 INFO: Scoring 683 HMMs using HMM-HMM Viterbi alignment

- 12:48:15.465 INFO: Alternative alignment: 0

- 12:48:15.658 INFO: 683 alignments done

- 12:48:15.659 INFO: Alternative alignment: 1

- 12:48:15.697 INFO: 92 alignments done


- 12:48:15.697 INFO: Alternative alignment: 2

- 12:48:15.710 INFO: 7 alignments done

- 12:48:15.710 INFO: Alternative alignment: 3
```

```
Query         tr|G8M3C3|G8M3C3_HUNCD Dockerin-like protein OS=Hungateiclostridium clariflavum (strain DSM 19732
/ NBRC 101661 / EBR45) OX=720554 GN=Clocl_4007 PE=4 SV=1
Match_columns 728
No_of_seqs    1 out of 1
Neff          1
Searched_HMMs 683
Date          Fri Feb 11 12:48:16 2022
Command       hhsearch -i /home/tony/github/deepchem/examples/tutorials/protein3.fasta -d hh/databases/dbCAN-fam
-V9 -oa3m /home/tony/github/deepchem/examples/tutorials/results.a3m -cpu 4 -n 2 -e 0.001

 No Hit                             Prob E-value P-value  Score    SS Cols Query HMM  Template HMM
  1 AAA91086.1|GH48|150-238|4.7e-1 100.0  7E-195  1E-197 1475.1   0.0  608   31-644     1-619 (620)
  2 lcl|consensus                   91.8 0.00051 7.4E-07   37.5   0.0   20  668-687     1-20  (21)
  3 ABN51673.1|GH124|2-334|2.6e-21  52.5   0.096 0.00014   40.1   0.0   66  663-728    19-85  (318)
  4 CAR68154.1|GH88|62-388|4.9e-13  10.5       2   0.003   30.7   0.0   43  421-463   181-223 (329)
  5 ACY49347.1|GH105|46-385|1.1e-1   6.4       4  0.0058   28.2   0.0   60  324-383   169-228 (329)
  6 QGI59602.1|GH16_22|78-291        5.4     4.9  0.0072   27.6   0.0   10  391-400    33-42  (224)
  7 QGI59602.1|GH16_22|78-291        5.3       5  0.0073   27.5   0.0   18  581-598   204-221 (224)
  8 AQA16748.1|GH5_51.hmm|7.4e-189   4.9     5.5  0.0081   28.6   0.0   37  644-680   253-291 (351)
  9 CCF60459.1|GH5_12.hmm|1.2e-238   3.3     9.1   0.013   28.3   0.0   27  357-383   298-324 (541)
 10 ACI55886.1|GH25|58-236|2.7e-60   3.0      10   0.015   22.2   0.0   41  594-634    18-61  (174)

No 1
>AAA91086.1|GH48|150-238|4.7e-10
Probab=100.00  E-value=6.7e-195  Score=1475.15  Aligned_cols=608  Identities=60%  Similarity=1.105  Sum_probs=60
4.0  Template_Neff=2.700

Q tr|G8M3C3|G8M3   31 FKDRFNYMYNKIHDPANGYFDSEGIPYHSVETLCVEAPDYGHESTSEAASYYAWLEAVNGKLNGKWSGLTEAWNVVEKYF  110 (728
)
Q Consensus        31 fkdrfnymynkihdpangyfdsegipyhsvetlcveapdyghestseaasyyawleavngklngkwsglteawnvvekyf  110 (728
)
                      |.||||+||+|||||+|||||++||||||||||||||||||||||++|||||||+|||||++|++||++||++||
T Consensus         1 Y~~rFl~lY~kI~dp~nGYFS~~GiPYHsvETlivEAPDyGHeTTSEA~SY~~WLeAmyg~itgd~s~~~~AW~~mE~y~   80 (620
)
T AAA91086.1|GH4    1 YKQRFLELYNKIHDPANGYFSPEGIPYHSVETLIVEAPDYGHETTSEAYSYYVWLEAMYGKITGDWSGFNKAWDTMEKYI   80 (620
)
Confidence            78999999999999999999999999999999999999999999999999999999999999999999999999999999


Q tr|G8M3C3|G8M3  111 IPSESIQKGMNRYNPSSPAGYADEFPLPDDYPAQIQSNVTVGQDPIHQELVSAYNTYAMYGMHWLVDVDNWYGYGT----  186 (728
)
Q Consensus       111 ipsesiqkgmnrynpsspagyadefplpddypaqiqsnvtvgqdpihqelvsayntyamygmhwlvdvdnwygygt----  186 (728
)
                      ||++++||+|+|.|||++|||||+++|++||++|+++|||++|||+++|||+|||+|||||||||||||||||||+
T Consensus        81 IP~~~~Qp~~~~Ynp~~pAtya~E~~~P~~YPs~l~~~~~vG~DPi~~eL~saYgt~~iY~MHWLlDVDN~YGfG~~g~~  160 (620
)
T AAA91086.1|GH4   81 IPSHQDQPTMSSYNPSSPATYAPEYDTPSQYPSQLDFNVPVGQDPIANELKSAYGTDDIYGMHWLLDVDNWYGFGNLGDG  160 (620
)
Confidence            99999999999999999999999999999999999999999999999999999999999999999999999999999999


Q tr|G8M3C3|G8M3  187 GTNCTFINTYQRGEQESVFETVPHPSIEEFKYGGRQGFSDLFTAG-ETQPKWAFTIASDADGRLIQVQYWANKWAKEQGQ  265 (728
)
```

```
Q Consensus       187 gtnctfintyqrgeqesvfetvphpsieefkyggrqgfsdlftag-etqpkwaftiasdadgrliqvqywankwakeqgq  265 (728
)
                      +++|+|||||||+|||||||||||||+|||||+||||+||++| ++++|||||||||||||||+|||++||+||+
T Consensus       161 ~~~psyINTfQRG~qESvWeTvp~P~~d~fk~Gg~nGfldlFt~d~~ya~QwkYTnApDADARavQa~YwA~~Wa~e~G~  240 (620
)
T AAA91086.1|GH4  161 TSGPSYINTFQRGEQESVWETVPHPSCEEFKYGGPNGFLDLFTKDSSYAKQWRYTNAPDADARAVQAAYWANQWAKEQGK  240 (620
)
Confidence            899999999999999999999999999999999999999999 999999999999999999999999999999999


Q tr|G8M3C3|G8M3  266 --NLSTLNAKAAKLGDYLRYSMFDKYFMKIG--AQGKTPASGYDSCHYLLAWYYAWGGAIAG-DWSWKIGCSHVHWGYQA  340 (728
)
Q Consensus       266 --nlstlnakaaklgdylrysmfdkyfmkig--aqgktpasgydschyllawyyawggaiag-dwswkigcshvhwgyqa  340 (728
)
                        +|+++++||+||||||+||||||||  +.+|++|+|||||||||||++|||++++ +|+|||||+||||||
T Consensus       241 ~~~is~~~~KAaKmGDyLRY~mfDKYfkkiG~~~~s~~ag~GkdSaHYLlsWY~aWGG~~~~~~WaWrIG~Sh~H~GYQN  320 (620
)
T AAA91086.1|GH4  241 ESEISSTVAKAAKMGDYLRYAMFDKYFKKIGVGPSSCPAGTGKDSAHYLLSWYYAWGGALDGSGWAWRIGSSHAHFGYQN  320 (620
)
Confidence              99999999999999999999999999  899999999999999999999999999 99999999999999999


Q tr|G8M3C3|G8M3  341 PLAAYALANDPDLKPKSANGAKDWNSSFKRQVELYAWLQSAEGAIAGGVTNSVGGQYKSY-NGASTFYDMAYTYAPVYAD  419 (728
)
Q Consensus       341 plaayalandpdlkpksangakdwnssfkrqvelyawlqsaegaiaggvtnsvggqyksy-ngastfydmaytyapvyad  419 (728
)
                      ||||||||++++|||||+|+++||++||+||||||+||||+||+|||||||||+|+|+| +|++||||||+|+++|||||
T Consensus       321 P~AAyaLs~~~~lkPks~ta~~DW~~SL~RQlEfy~wLQS~eG~iAGGaTNSW~G~Y~~~Psg~~TFygM~Yd~~PVY~D  400 (620
)
T AAA91086.1|GH4  321 PLAAYALSNDSDLKPKSPTAASDWAKSLDRQLEFYQWLQSAEGAIAGGATNSWNGRYETYPSGTSTFYGMAYDEHPVYHD  400 (620
)
Confidence            99999999999999999999999999999999999999999999999999999999999999 9999999999999999


Q tr|G8M3C3|G8M3  420 PPSNNWFGMQAWSMQRMCEVYYETGDSLAKEICDKWVAWAESVCEADIEAGTWKIPATLEWSGQPDTWRGTKPSNNNLHC  499 (728
)
Q Consensus       420 ppsnnwfgmqawsmqrmcevyyetgdslakeicdkwvawaesvceadieagtwkipatlewsgqpdtwrgtkpsnnnlhc  499 (728
)
                      ||||+|||||+|+|||||||||+|||++||+||||||+|++++|+|+ ++|+|+|||++|+|+|||||||+|++++|+||
T Consensus       401 PpSN~WfG~Q~Wsm~RvAeyYY~tGD~~ak~ildKWv~W~~~~~~~~~-~dg~~~iPs~L~WsGqPDtW~gs~~~N~~lhv  479 (620
)
T AAA91086.1|GH4  401 PPSNRWFGMQAWSMQRVAEYYYVTGDARAKAILDKWVAWVKSNTTVN-SDGTFQIPSTLEWSGQPDTWNGSYTGNPNLHV  479 (620
)
Confidence            99999999999999999999999999999999999999999999 889999999999999999999999999999


Q tr|G8M3C3|G8M3  500 KVVNYGNDIGITGSLANAFLFYDQATQRWNGNTTLGKKAADKALAMLQVVWDTCRDQYGVGVKETNESLNRIFTQEVFIP  579 (728
)
Q Consensus       500 kvvnygndigitgslanaflfydqatqrwngnttlgkkaadkalamlqvvwdtcrdqygvgvketneslnriftqevfip  579 (728
)
                      +|+++|+|||||+||||||+||||++|++ |    ++||++||+|||+||++|||++|+|||+|+|+||||++||||
T Consensus       480 ~V~~yg~dvGva~s~A~tL~yYAa~sg~~~d-----~~ak~~Ak~LLD~~w~~~~d~~Gvs~~E~r~dy~Rf~d~~VYiP  554 (620
)
T AAA91086.1|GH4  480 TVTDYGQDVGVAASLAKTLMYYAAASGKYG-D----TAAKNLAKQLLDAMWKNYRDDKGVSTPETRGDYKRFFDQEVYIP  554 (620
)
Confidence            99999999999999999999999999888 7    889999999999999999999999999999999999999999


Q tr|G8M3C3|G8M3  580 AGWTGKMPNGDVIQQGVKFIDIRSKYKDDPWYEGLKKQAEQGIPFEYTLHRFWHQVDYAVALGIA  644 (728)
Q Consensus       580 agwtgkmpngdviqqgvkfidirskykddpwyeglkkqaeqgipfeytlhrfwhqvdyavalgia  644 (728)
                      +||+|+|||||+|++|+|||||||+||+||+|++||++|++|++|+|+|+|+|+|+|+|+|+|+.
T Consensus       555 ~gwtG~mPnGD~I~~g~tFl~IRs~Yk~Dp~w~kvq~~l~gG~~p~f~YHRFWaQ~diA~A~g~y  619 (620)
T AAA91086.1|GH4  555 SGWTGTMPNGDVIKSGATFLDIRSKYKQDPDWPKVEAYLNGGAAPEFTYHRFWAQADIAMANGTY  619 (620)
Confidence            99999999999999999999999999999999999999999999999999999999999863


No 2
>lcl|consensus
Probab=91.79  E-value=0.00051  Score=37.45  Aligned_cols=20  Identities=55%  Similarity=0.811  Sum_probs=11.9  T
emplate_Neff=4.300

Q tr|G8M3C3|G8M3  668 DINFDGDINSIDYALLKAHL  687 (728)
Q Consensus       668 dinfdgdinsidyallkahl  687 (728)
                      |+|-||.+|++|++++|.++
T Consensus         1 DvN~DG~Vna~D~~~l~~~l   20 (21)
T lcl|consensus_    1 DVNGDGKVNALDLALLKKYL   20 (21)
Confidence            45566666666666665554


No 3
>ABN51673.1|GH124|2-334|2.6e-219
Probab=52.47  E-value=0.096  Score=40.07  Aligned_cols=66  Identities=35%  Similarity=0.533  Sum_probs=48.5  Tem
plate_Neff=1.400
```

```
Q tr|G8M3C3|G8M3  663 DIKLGDINFDGDINSIDYALLKAHLLGINKLSGDAL-KAADVDQNGDVNSIDYAKMKSYLLGISKDF  728 (728)
Q Consensus       663 diklgdinfdgdinsidyallkahllginklsgdal-kaadvdqngdvnsidyakmksyllgiskdf  728 (728)
                      .+..||.|-||-+|--||.|+|..|.-|.+...+.- -..+++....++.+|-.-+|.|||.+-++|
T Consensus        19 kav~GD~n~dgvv~isd~vl~k~~l~~~a~~~a~~d~w~g~vN~dd~I~D~d~~~~kryll~mir~~   85 (318)
T ABN51673.1|GH1   19 KAVIGDVNADGVVNISDYVLMKRILRIIADFPADDDMWVGDVNGDDVINDIDCNYLKRYLLHMIREF   85 (318)
Confidence            567899999999999999999977666665433211234444455778888888999999876553


No 4
>CAR68154.1|GH88|62-388|4.9e-137
Probab=10.51  E-value=2  Score=30.74  Aligned_cols=43  Identities=19%  Similarity=0.234  Sum_probs=34.3  Templat
e_Neff=5.400

Q tr|G8M3C3|G8M3  421 PSNNWFGMQAWSMQRMCEVYYETGDSLAKEICDKWVAWAESVC  463 (728)
Q Consensus       421 psnnwfgmqawsmqrmcevyyetgdslakeicdkwvawaesvc  463 (728)
                      .+..|=-=|+|.|==.|..|..|||++--++-.+=+.+++++.
T Consensus       181 d~S~WsRGQAWaiYG~a~~yr~t~d~~yL~~A~~~a~yfl~~l  223 (329)
T CAR68154.1|GH8  181 DDSAWARGQAWAIYGFALAYRYTKDPEYLDTAKKVADYFLNRL  223 (329)
Confidence            367899999999999999999999555555555555555


No 5
>ACY49347.1|GH105|46-385|1.1e-131
Probab=6.37  E-value=4  Score=28.22  Aligned_cols=60  Identities=25%  Similarity=0.330  Sum_probs=50.1  Template
_Neff=6.200

Q tr|G8M3C3|G8M3  324 DWSWKIGCSHVHWGYQAPLAAYALANDPDLKPKSANGAKDWNSSFKRQVELYAWLQSAEG  383 (728)
Q Consensus       324 dwswkigcshvhwgyqaplaayalandpdlkpksangakdwnssfkrqvelyawlqsaeg  383 (728)
                      .|+=.-|.|..+.|==|=-.+.||...-++-|+.........+.|++|++=..=+|+.+|
T Consensus       169 ~wa~~t~~s~~fW~RgnGW~~~aL~~~L~~lP~~~p~r~~l~~~~~~~~~al~~~Qd~~G  228 (329)
T ACY49347.1|GH1  169 NWADPTGGSPAFWGRGNGWVAMALVDVLELLPEDHPDRRFLIDILKEQAAALAKYQDESG  228 (329)
Confidence            444456677777777888899999999999988888899999999999988889997 76


No 6
>QGI59602.1|GH16_22|78-291
Probab=5.38  E-value=4.9  Score=27.58  Aligned_cols=10  Identities=30%  Similarity=0.245  Sum_probs=6.0  Templat
e_Neff=3.000

Q tr|G8M3C3|G8M3  391 NSVGGQYKSY  400 (728)
Q Consensus       391 nsvggqyksy  400 (728)
                      ||-+..|-..
T Consensus        33 NS~nNvyie~   42 (224)
T QGI59602.1|GH1   33 NSPNNVYIEK   42 (224)
Confidence            6666666555


No 7
>QGI59602.1|GH16_22|78-291
Probab=5.33  E-value=5  Score=27.55  Aligned_cols=18  Identities=28%  Similarity=0.730  Sum_probs=9.8  Template_
Neff=3.000

Q tr|G8M3C3|G8M3  581 GWTGKMPNGDVIQQGVKF  598 (728)
Q Consensus       581 gwtgkmpngdviqqgvkf  598 (728)
                      .|+|.|.-|+...-++..
T Consensus       204 ~WsGnM~vg~sa~lqIqW  221 (224)
T QGI59602.1|GH1  204 SWSGNMSVGDSAYLQIQW  221 (224)
Confidence            366666666654444333


No 8
>AQA16748.1|GH5_51.hmm|7.4e-189|58-409
Probab=4.89  E-value=5.5  Score=28.57  Aligned_cols=37  Identities=32%  Similarity=0.524  Sum_probs=28.0  Templa
te_Neff=2.200

Q tr|G8M3C3|G8M3  644 AEIFGYKPPK--GGSGGGETGDIKLGDINFDGDINSIDY  680 (728)
Q Consensus       644 aeifgykppk--ggsgggetgdiklgdinfdgdinsidy  680 (728)
                      |..+||.-|+  |.+|-|||.|.+..|+.-..-.+.++-
T Consensus       253 aHfYgYTGP~htGatg~get~dpRY~Dl~~~~l~~~l~~  291 (351)
T AQA16748.1|GH5  253 AHFYGYTGPNHTGATGIGETHDPRYRDLSPAELAAVLDD  291 (351)
Confidence            5678998775  67778999999999877655555443


No 9
>CCF60459.1|GH5_12.hmm|1.2e-238|14-567
Probab=3.29  E-value=9.1  Score=28.28  Aligned_cols=27  Identities=26%  Similarity=0.540  Sum_probs=17.1  Templa
te_Neff=4.100

Q tr|G8M3C3|G8M3  357 SANGAKDWNSSFKRQVELYAWLQSAEG  383 (728)
Q Consensus       357 sangakdwnssfkrqvelyawlqsaeg  383 (728)
                      .|.+++-|.+.-+|+=.-|-|-.+++-
T Consensus       298 np~G~saWl~~~~~d~~ygw~r~~~w  324 (541)
```

```
T CCF60459.1|GH5  298 NPKGVSAWLSGEERDDKKYGWKRDPEW  324 (541)
Confidence            34566777777777666777655443


No 10
>ACI55886.1|GH25|58-236|2.7e-60
Probab=3.03  E-value=10  Score=22.20  Aligned_cols=41  Identities=20%  Similarity=0.114  Sum_probs=30.1  Templat
e_Neff=7.700

Q tr|G8M3C3|G8M3  594 QGVKFIDIRSKY---KDDPWYEGLKKQAEQGIPFEYTLHRFWHQ  634 (728)
Q Consensus       594 qgvkfidirsky---kddpwyeglkkqaeqgipfeytlhrfwhq  634 (728)
                      +|..|.-||.-+   -.||.|..=-+....-..|.=.||-+.+.
T Consensus        18 ~gi~Fv~ikateG~~~~D~~f~~n~~~a~~aGl~~G~Yhf~~~~   61 (174)
T ACI55886.1|GH2   18 SGVDFVIIKATEGTSYVDPYFASNWAGARAAGLPVGAYHFARPC   61 (174)
Confidence            389999999765   3688887655555555567888889988854


- 12:48:16.066 INFO: Premerge done

- 12:48:16.066 INFO: Realigning 10 HMM-HMM alignments using Maximum Accuracy algorithm

- 12:48:16.115 INFO: 4 sequences belonging to 4 database HMMs found with an E-value < 0.001

- 12:48:16.115 INFO: Number of effective sequences of resulting query HMM: Neff = 2.41642
```

We can see that the exact protein was found in dbCAN in hit 1, but also some highly related proteins were found in hits 1-5. This query.a3m MSA can then be useful if we want to search a larger database like UniProt or Uniclust because it includes this more diverse selection of related protein sequences.

Other hh-suite functions

hhsuite contains other functions which may be useful if you are working with MSA or HMMs. For more detailed information, see the documentation at https://github.com/soedinglab/hh-suite/wiki

hhmake: Build an HMM from an input MSA

hhfilter: Filter an MSA by max sequence identity, coverage, and other criteria

hhalign: Calculate pairwise alignments etc. for two HMMs/MSAs

hhconsensus: Calculate the consensus sequence for an A3M/FASTA input file

reformat.pl: Reformat one or many MSAs

addss.pl: Add PSIPRED predicted secondary structure to an MSA or HHM file

hhmakemodel.pl: Generate MSAs or coarse 3D models from HHsearch or HHblits results

hhmakemodel.py: Generates coarse 3D models from HHsearch or HHblits results and modifies cif files such that they are compatible with MODELLER

hhsuitedb.py: Build HHsuite database with prefiltering, packed MSA/HMM, and index files

splitfasta.pl: Split a multiple-sequence FASTA file into multiple single-sequence files

renumberpdb.pl: Generate PDB file with indices renumbered to match input sequence indices

HHPaths.pm: Configuration file with paths to the PDB, BLAST, PSIPRED etc. mergeali.pl: Merge MSAs in A3M format according to an MSA of their seed sequences

pdb2fasta.pl: Generate FASTA sequence file from SEQRES records of globbed pdb files

cif2fasta.py: Generate a FASTA sequence from the pdbx_seq_one_letter_code entry of the entity_poly of globbed cif files

pdbfilter.pl: Generate representative set of PDB/SCOP sequences from pdb2fasta.pl output

pdbfilter.py: Generate representative set of PDB/SCOP sequences from cif2fasta.py output

References:

[1] Steinegger M, Meier M, Mirdita M, Vöhringer H, Haunsberger S J, and Söding J (2019) HH-suite3 for fast remote homology detection and deep protein annotation, BMC Bioinformatics, 473. doi: 10.1186/s12859-019-3019-7

[2] Kunzmann, P., Mayer, B.E. & Hamacher, K. Substitution matrix based color schemes for sequence alignment visualization. BMC Bioinformatics 21, 209 (2020). https://doi.org/10.1186/s12859-020-3526-6

[3]Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. https://www.researchgate.net/publication/12812078_

[4]https://github.com/soedinglab/hh-suite/wiki#what-are-hmm-hmm-comparisons-and-why-are-they-so-powerful

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

ScanPy

ScanPy is a scalable toolkit for analyzing single-cell gene expression data. It includes methods for preprocessing, visualization, clustering, pseudotime and trajectory inference, differential expression testing, and simulation of gene regulatory networks. There are many advantage of using a Python-based platform to process scRNA-seq data including increased processing efficiency and running speed as well as seamless integration with machine learning frameworks.

ANNDATA was presented alongside ScanPy as a generic class for handling annotated data matrices that can deal with the sparsity inherent in gene expression data.

This tutorial is largely adapted from the original tutorials which can be found in Scanpy's read the docs and from this notebook.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Make sure you've installed Scanpy:

In [2]:
```
%conda install -c conda-forge scanpy python-igraph leidenalg
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
# import necessary packages
import numpy as np
import pandas as pd
import scanpy as sc
```

In [4]:
```
sc.settings.verbosity = 3   # verbosity: errors (0), warnings (1), info (2), hints (3)
```

In [ ]:
```
# see what package versions you have installed
sc.logging.print_versions()
# customize resolution and color of your figures
sc.settings.set_figure_params(dpi=80, facecolor='white')
```

In [ ]:
```
# download the test data for this tutorial
!mkdir data
!wget http://cf.10xgenomics.com/samples/cell-exp/1.1.0/pbmc3k/pbmc3k_filtered_gene_bc_matrices.tar.gz -O data/pl
!cd data; tar -xzf pbmc3k_filtered_gene_bc_matrices.tar.gz
!mkdir write
```

In [7]:
```
results_file = 'write/pbmc3k.h5ad' # the file that will store the analysis results
```

In [8]:
```
adata = sc.read_10x_mtx(
    'data/filtered_gene_bc_matrices/hg19/',  # the directory with the `.mtx` file
    var_names='gene_symbols',                # use gene symbols for the variable names (variables-axis index)
    cache=True)                              # write a cache file for faster subsequent reading
```

```
... reading from cache file cache/data-filtered_gene_bc_matrices-hg19-matrix.h5ad
```

See [anndata-tutorials/getting-started] for a more comprehensive introduction to AnnData.

In [9]:
```
adata.var_names_make_unique()  # this is unnecessary if using `var_names='gene_ids'` in `sc.read_10x_mtx`
```

In [10]:
```
# look at what the AnnData object contains
adata
```

Out[10]:
```
AnnData object with n_obs × n_vars = 2700 × 32738
    var: 'gene_ids'
```

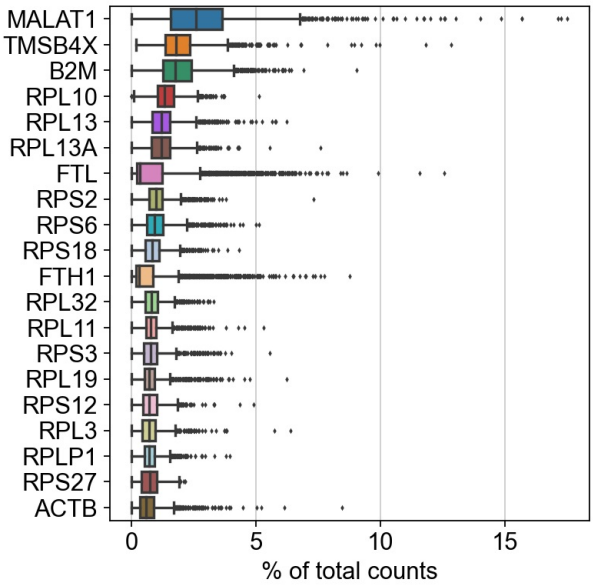
Pre-processing

Check for highly expressed genes

Show genes that yield the highest fraction of counts in each single cell, across all cells. The `sc.pl.highest_expr_genes` command normalizes counts per cell, and plots the genes that are most abundant in each cell.

In [11]:
```
sc.pl.highest_expr_genes(adata, n_top=20, )
```

```
normalizing counts per cell
    finished (0:00:00)
```



Note that MALAT1, a non-coding RNA that is known to be extremely abundant in many cells, ranks at the top.

Basic filtering: remove cells and genes with low expression or missing values.

In [12]:
```
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)
```

```
filtered out 19024 genes that are detected in less than 3 cells
```

Check mitochondrial genes for Quality Control

Let's assemble some information about mitochondrial genes, which are important for quality control.

Citing from "Simple Single Cell" workflows (Lun, McCarthy & Marioni, 2017):

High proportions are indicative of poor-quality cells (Islam et al. 2014; Ilicic et al. 2016), possibly because of loss of cytoplasmic RNA from perforated cells. The reasoning is that mitochondria are larger than individual transcript molecules and less likely to escape through tears in the cell membrane.

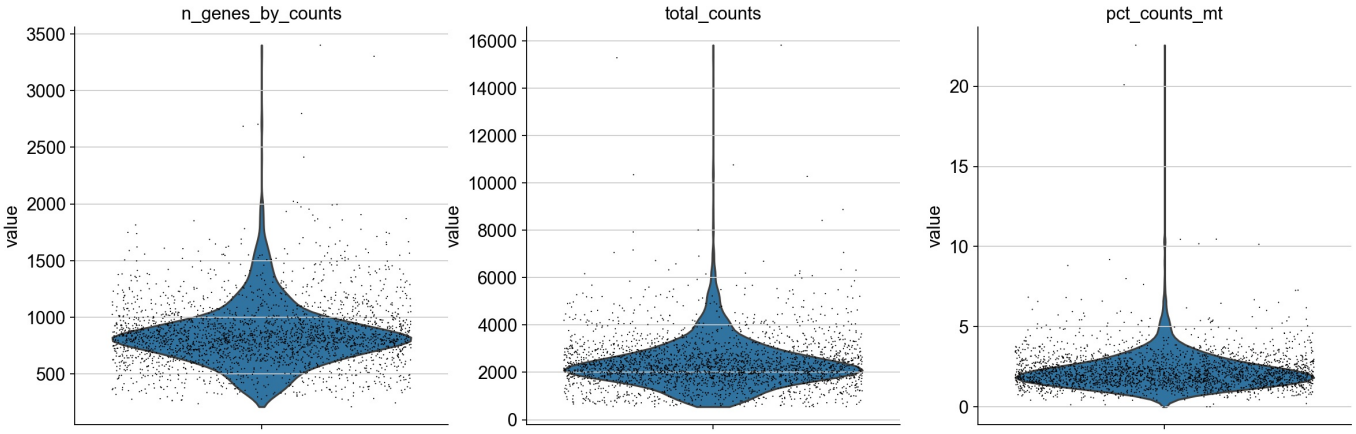
With `pp.calculate_qc_metrics`, we can compute many metrics very efficiently.

In [13]:
```
adata.var['mt'] = adata.var_names.str.startswith('MT-')  # annotate the group of mitochondrial genes as 'mt'
sc.pp.calculate_qc_metrics(adata, qc_vars=['mt'], percent_top=None, log1p=False, inplace=True)
```

A violin plot of some of the computed quality measures:

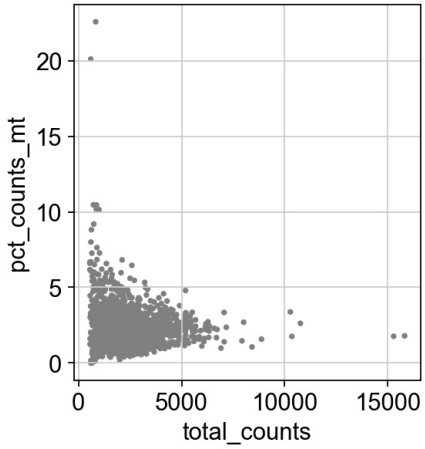
- the number of genes expressed in the count matrix
- the total counts per cell
- the percentage of counts in mitochondrial genes

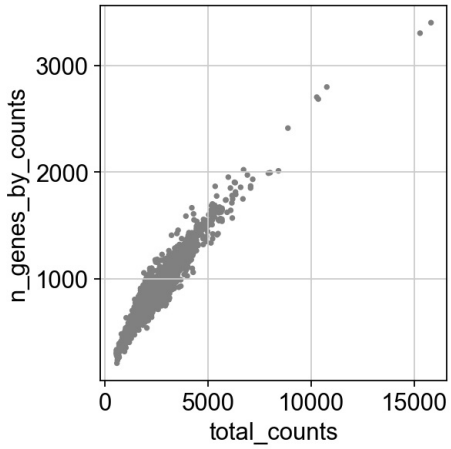
In [14]:
```
sc.pl.violin(adata, ['n_genes_by_counts', 'total_counts', 'pct_counts_mt'],
             jitter=0.4, multi_panel=True)
```



Remove cells that have too many mitochondrial genes expressed or too many total counts. High proportions of mitochondrial genes indicate poor-quality cells, potentially because of loss of cytoplasmic RNA from perforated cells.

In [15]:
```
sc.pl.scatter(adata, x='total_counts', y='pct_counts_mt')
sc.pl.scatter(adata, x='total_counts', y='n_genes_by_counts')
```





Filter data based on QC

Check current datset and filter it by slicing the AnnData object.

In [16]:
```
print(adata)
```

```
AnnData object with n_obs × n_vars = 2700 × 13714
    obs: 'n_genes', 'n_genes_by_counts', 'total_counts', 'total_counts_mt', 'pct_counts_mt'
    var: 'gene_ids', 'n_cells', 'mt', 'n_cells_by_counts', 'mean_counts', 'pct_dropout_by_counts', 'total_counts
'
```

In [17]:
```
# slice the adata object so you only keep genes and cells that pass the QC
adata = adata[adata.obs.n_genes_by_counts < 2500, :]
adata = adata[adata.obs.pct_counts_mt < 5, :]
print(adata)
```

```
View of AnnData object with n_obs × n_vars = 2638 × 13714
    obs: 'n_genes', 'n_genes_by_counts', 'total_counts', 'total_counts_mt', 'pct_counts_mt'
    var: 'gene_ids', 'n_cells', 'mt', 'n_cells_by_counts', 'mean_counts', 'pct_dropout_by_counts', 'total_counts
'
```

Data normalization

To correct differences in library sizes across cells, normalize the total read count of the data matrix to 10,000 reads per cell so that counts become comparable among cells.

In [18]:
```
sc.pp.normalize_total(adata, target_sum=1e4)
```

```
normalizing counts per cell
    finished (0:00:00)
/Users/paulinampaiz/opt/anaconda3/envs/deepchem/lib/python3.10/site-packages/scanpy/preprocessing/_normalization
.py:170: UserWarning: Received a view of an AnnData. Making a copy.
  view_to_actual(adata)
```

Log transform the data for later use in differential gene expression as well as in visualizations. The natural logarithm is used, and `log1p` means that an extra read is added to cells of the count matrix as a pseudo-read. See here for more information on why log scale makes more sense for genomic data.

In [19]:
```
sc.pp.log1p(adata)
```

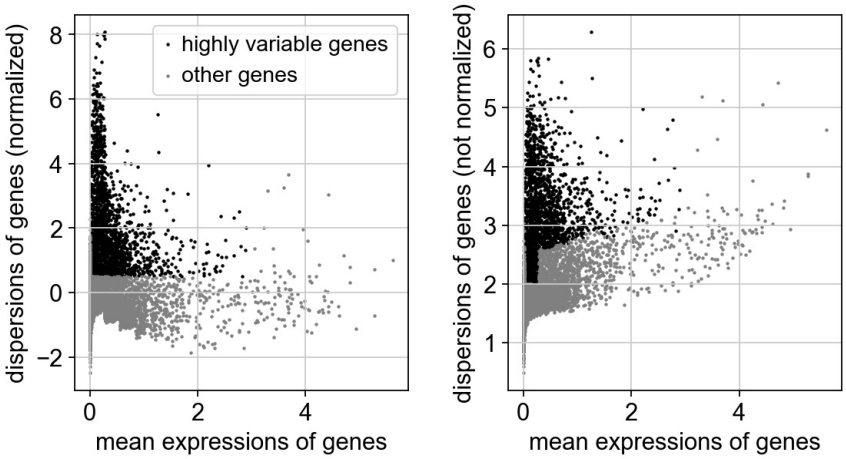
Identify highly-variable genes.

The function `sc.pp.highly_variable_genes` can detect marker genes that can help us identify cells based on a few manually set parameters, including mininum mean expression, maximum mean expression, and minimum dispersion. We will focus our analysis on such genes.

In [20]:
```
sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3, min_disp=0.5)
```

```
extracting highly variable genes
    finished (0:00:00)
--> added
    'highly_variable', boolean vector (adata.var)
    'means', float vector (adata.var)
    'dispersions', float vector (adata.var)
    'dispersions_norm', float vector (adata.var)
```

In [21]:
```
# visualize the highly variable genes with a plot
sc.pl.highly_variable_genes(adata)
```



Set the `.raw` attribute of the AnnData object to the normalized and logarithmized raw gene expression for later use in differential testing and visualizations of gene expression. This simply freezes the state of the AnnData object.

In [22]:
```
adata.raw = adata
```

Filter the `adata` object so that only genes that are highly variable are kept.

In [23]:
```
adata = adata[:, adata.var.highly_variable]
```

Correct for the effects of counts per cell and mitochondrial gene expression

Regress out effects of total counts per cell and the percentage of mitochondrial genes expressed. This can consume some memory and take some time because the input data is sparse.

In [24]:
```
sc.pp.regress_out(adata, ['n_genes_by_counts', 'pct_counts_mt'])
```

```
regressing out ['n_genes_by_counts', 'pct_counts_mt']
    sparse input is densified and may lead to high memory use
    finished (0:00:11)
```

Center the data to zero and scale to unit variance

Use `sc.pp.scale` to center the average expression per gene to zero. Here we are also clipping scaled values that exceed standard deviation of 10.

In [25]:
```
sc.pp.scale(adata, max_value=10)
```

Dimension reduction with PCA

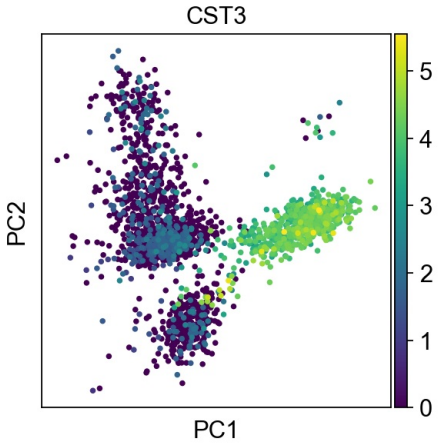
We first use principal component analysis (PCA), a linear dimention-reduction technique, to reveal the main axes of variation and denoise the data.

In [26]:
```
sc.tl.pca(adata, svd_solver='arpack')
```

```
computing PCA
    on highly variable genes
    with n_comps=50
    finished (0:00:00)
```

We can make a scatter plot in the PCA coordinates, but we will not use that later on.

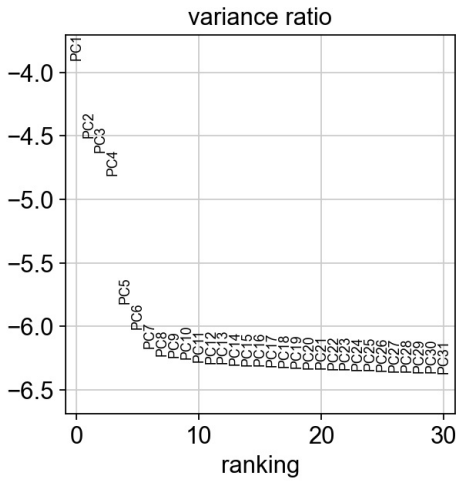
In [27]:
```
sc.pl.pca(adata, color="CST3")
```



The variance ratio plot lists contributions of individual principal components (PC) to the total variance in the data. This piece of information helps us to choose an appropriate number of PCs in order to compute the neighborhood relationships between the cells, for instance, using the clustering method Louvain `sc.tl.louvain()` or the embedding method tSNE `sc.tl.tsne()` for dimension-reduction.

According to the authors of Scanpy, a rough estimate of the number of PCs does fine.

In [28]:
```
sc.pl.pca_variance_ratio(adata, log=True)
```



Save the result up to PCA analysis.

In [29]:
```
! mkdir -p write
adata.write(results_file)
```

Note that our adata object has following elements: observations annotation (obs), variables (var), unstructured annotation (uns), multi-dimensional observations annotation (obsm), and multi-dimensional variables annotation (varm). The meanings of these parameters are documented in the anndata package, available at anndata documentation.

In [30]:
```
adata
```

Out[30]:
```
AnnData object with n_obs × n_vars = 2638 × 1838
    obs: 'n_genes', 'n_genes_by_counts', 'total_counts', 'total_counts_mt', 'pct_counts_mt'
    var: 'gene_ids', 'n_cells', 'mt', 'n_cells_by_counts', 'mean_counts', 'pct_dropout_by_counts', 'total_count
s', 'highly_variable', 'means', 'dispersions', 'dispersions_norm', 'mean', 'std'
    uns: 'log1p', 'hvg', 'pca'
    obsm: 'X_pca'
    varm: 'PCs'
```

Computing and Embedding the neighborhood graph

Use the PCA representation of the data matrix to compute the neighborhood graph of cells.

In [ ]:
```
sc.pp.neighbors(adata, n_neighbors=10, n_pcs=40)
```

The auhours of Scanpy suggest embedding the graph in two dimensions using UMAP (McInnes et al., 2018). UMAP is potentially more faithful to the global connectivity of the manifold than tSNE, i.e., it better preserves trajectories.

In [ ]:
```
sc.tl.umap(adata)
```

In [ ]:
```
sc.pl.umap(adata, color=['CST3', 'NKG7', 'PPBP'])
```

As we set the `.raw` attribute of adata, the previous plots showed the “raw” (normalized, logarithmized, but uncorrected) gene expression. You can also plot the scaled and corrected gene expression by explicitly stating that you don’t want to use `.raw`.

In [ ]:
```
sc.pl.umap(adata, color=['CST3', 'NKG7', 'PPBP'], use_raw=False)
```

In some ocassions, you might still observe disconnected clusters and similar connectivity violations. They can usually be remedied by running:

In [ ]:
```
tl.paga(adata)
pl.paga(adata, plot=False)  # remove `plot=False` if you want to see the coarse-grained graph
tl.umap(adata, init_pos='paga')
```

Clustering the neighborhood graph

As with Seurat and many other frameworks, we recommend the Leiden graph-clustering method (community detection based on optimizing modularity) by Traag *et al.* (2018). Note that Leiden clustering directly clusters the neighborhood graph of cells, which we already computed in the previous section. Compared with the Louvain algorithm, the Leiden algorithm yields communities that are guaranteed to be connected. When applied iteratively, the Leiden algorithm converges to a partition in which all subsets of all communicities are locally optimally assigned. Last but not least, it runs faster.

In [ ]:
```
sc.tl.leiden(adata)
```

Plot the clusters using `sc.pl.umap`. Note that the color parameter accepts both individual genes and the clustering method (leiden in this case).

In [ ]:
```
sc.pl.umap(adata, color=['leiden', 'CST3', 'NKG7'])
```

We save the result again.

In [ ]:
```
adata.write(results_file)
```

Identifying marker genes

Compute a ranking for the highly differential genes in each cluster. For this, by default, the `.raw` attribute of AnnData is used in case it has been initialized before. The simplest and fastest method to do so is the t-test. Other methods include Wilcoxon rank-sum (Mann-Whitney-U) test, MAST, limma, DESeq2, and diffxpy by the Theis lab. The authours of Scanpy reccomend using the Wilcoxon rank-sum test in publications.

The Wilcoxon's test

For simplicity, we start with the Mann-Whitney-U test. The null hypothesis is that the rank of a gene in a cluster is the same as its rank in all cells. The alternative hypothesis is that the rank of a gene in a cluster is much higher than its rank in all cells (one-sided). The function `sc.tl.rank_genes_groups` performs the test, and `sc.pl.rank_genes_groups` plots the top genes.

In [ ]:
```
sc.tl.rank_genes_groups(adata, groupby='leiden', method='wilcoxon')
sc.pl.rank_genes_groups(adata, n_genes=15, sharey=False)
```

In [ ]:
```
sc.settings.verbosity = 2  # reduce the verbosity
```

In [ ]:
```
adata.write(results_file) # write the output to the results file
```

The Student's t-test

An alternative to the non-parametric Wilcoxon test is the t-test.

In [ ]:
```
sc.tl.rank_genes_groups(adata, 'leiden', method='t-test')
sc.pl.rank_genes_groups(adata, n_genes=15, sharey=False)
```

As an alternative, let us rank genes using logistic regression. For instance, this has been suggested by Natranos et al. (2018). The essential difference is that here, we use a multi-variate appraoch whereas conventional differential tests are uni-variate. Clark et al. (2014) has more details.

In [ ]:
```
sc.tl.rank_genes_groups(adata, 'leiden', method="logreg")
sc.pl.rank_genes_groups(adata, n_genes=15, sharey=False)
```

Let us also define a list of marker genes for later reference.

In [ ]:
```
marker_genes = ['IL7R', 'CD79A', 'MS4A1', 'CD8A', 'CD8B', 'LYZ', 'CD14',
                'LGALS3', 'S100A8', 'GNLY', 'NKG7', 'KLRB1',
                'FCGR3A', 'MS4A7', 'FCER1A', 'CST3', 'PPBP']
```

Listing signatures using the results of the Wilcoxon's test

We use the results of the Wilcoxon's test for downstream analysis. Reload the object that has been save with the Wilcoxon Rank-Sum test result.

In [ ]:
```
adata = sc.read(results_file)
```

Show the 10 top ranked genes per cluster 0, 1, ..., 7 in a dataframe.

In [ ]:
```
pd.DataFrame(adata.uns['rank_genes_groups']['names']).head(5)
```

Get a table with the scores and groups.

In [ ]:
```
result = adata.uns['rank_genes_groups']
groups = result['names'].dtype.names
pd.DataFrame(
    {group + '_' + key[:1]: result[key][group]
    for group in groups for key in ['names', 'pvals']}).head(5)
```

Compare to a single cluster:

In [ ]:
```
sc.tl.rank_genes_groups(adata, 'leiden', groups=['0'], reference='1', method='wilcoxon')
sc.pl.rank_genes_groups(adata, groups=['0'], n_genes=20)
```

If we want a more detailed view for a certain group, use `sc.pl.rank_genes_groups_violin`.

In [ ]:
```
sc.pl.rank_genes_groups_violin(adata, groups='0', n_genes=8)
```

If you want to compare a certain gene across groups, use the following.

In [ ]:
```
sc.pl.violin(adata, ['CST3', 'NKG7', 'PPBP'], groupby='leiden')
```

Actually mark the cell types.

In [ ]:
```
new_cluster_names = [
    'CD4 T', 'CD14 Monocytes',
    'B', 'CD8 T',
```

```
    'NK', 'FCGR3A Monocytes',
    'Dendritic', 'Megakaryocytes']
adata.rename_categories('leiden', new_cluster_names)
```

In [ ]:
```
# plot the UMAP
sc.pl.umap(adata, color='leiden', legend_loc='on data', title='', frameon=False, save='.pdf')
```

Now that we annotated the cell types, let us visualize the marker genes.

In [ ]:
```
sc.pl.dotplot(adata, marker_genes, groupby='leiden');
```

There is also a very compact violin plot.

In [ ]:
```
sc.pl.stacked_violin(adata, marker_genes, groupby='leiden', rotation=90);
```

During the course of this analysis, the AnnData accumulated the following annotations.

In [ ]:
```
adata
```

If you want to save your results:

In [ ]:
```
adata.write(results_file, compression='gzip')  # `compression='gzip'` saves disk space, but slows down writing a
```

Get a rough overview of the file using h5ls, which has many options - for more details see here. The file format might still be subject to further optimization in the future. All reading functions will remain backwards-compatible, though.

If you want to share this file with people who merely want to use it for visualization, a simple way to reduce the file size is by removing the dense scaled and corrected data matrix. The file still contains the raw data used in the visualizations in `adata.raw`.

In [ ]:
```
adata.raw.to_adata().write('./write/pbmc3k_withoutX.h5ad')
```

If you want to export to “csv”, you have the following options:

In [ ]:
```
# Export single fields of the annotation of observations
# adata.obs[['n_counts', 'louvain_groups']].to_csv(
#     './write/pbmc3k_corrected_louvain_groups.csv')

# Export single columns of the multidimensional annotation
# adata.obsm.to_df()[['X_pca1', 'X_pca2']].to_csv(
#     './write/pbmc3k_corrected_X_pca.csv')

# Or export everything except the data using `.write_csvs`.
# Set `skip_data=False` if you also want to export the data.
# adata.write_csvs(results_file[:-5], )
```

Deep Probabilistic Analysis of Single-Cell Omics Data

Recordings at single-cell resolution can give us a better understanding of the biological differences in our sample. As sequencing technologies and instruments have become better and cheaper, generating single-cell data is becoming more popular. In order to derive meaningful biological insights, it is important to select reliable analysis tools such as the one we will cover in this tutorial.

scvi-tools (single-cell variational inference tools) is a package for probabilistic modeling and analysis of single-cell omics data, built on top of PyTorch and AnnData that aims to address some of the limitations that arise when developing and implementing probabilistic models. scvi-tools is used in tandem with Scanpy for which Deepchem also offers a tutorial. In the broader analysis pipeline, scVI sits downstream of initial quality control (QC)-driven preprocessing and generates outputs that may be further interpreted via general single-cell analysis tools.

In this introductory tutorial, we go through the different steps of an scvi-tools workflow. While we focus on scVI in this tutorial, the API is consistent across all models. Please note that this tutorial was largely adapted from the one provided by scvi-tools and you can head to their page to find more information.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
# install necessary packages
!pip install --quiet scvi-colab
from scvi_colab import install
install()
```

In [ ]:
```
import scvi
import scanpy as sc
import matplotlib.pyplot as plt

# set preferences for figures and plots
sc.set_figure_params(figsize=(4, 4))

# for white background of figures (only for docs rendering)
%config InlineBackend.print_figure_kwargs={'facecolor' : "w"}
%config InlineBackend.figure_format='retina'
```

```
Global seed set to 0
```

Loading and preparing data

Let us first load a subsampled version of the heart cell atlas dataset described in Litviňuková et al. (2020). scvi-tools has many "built-in" datasets as well as support for loading arbitrary `.csv`, `.loom`, and `.h5ad` (AnnData) files. Please see our tutorial on data loading for more examples.

- Litviňuková, M., Talavera-López, C., Maatz, H., Reichart, D., Worth, C. L., Lindberg, E. L., ... & Teichmann, S. A. (2020). Cells of the adult human heart. Nature, 588(7838), 466-472.

> Important
> All scvi-tools models require AnnData objects as input.

In [ ]:
```
adata = scvi.data.heart_cell_atlas_subsampled()
```

```
INFO     File data/hca_subsampled_20k.h5ad already downloaded
```

Now we preprocess the data to remove, for example, genes that are very lowly expressed and other outliers. For these tasks we prefer the Scanpy preprocessing module.

In [ ]:
```
sc.pp.filter_genes(adata, min_counts=3)
```

In scRNA-seq analysis, it's popular to normalize the data. These values are not used by scvi-tools, but given their popularity in other tasks as well as for visualization, we store them in the anndata object separately (via the `.raw` attribute).

> Important
>
> Unless otherwise specified, scvi-tools models require the raw counts (not log library size normalized).

In [ ]:
```
adata.layers["counts"] = adata.X.copy() # preserve counts
sc.pp.normalize_total(adata, target_sum=1e4)
sc.pp.log1p(adata)
adata.raw = adata # freeze the state in `.raw`
```

Finally, we perform feature selection, to reduce the number of features (genes in this case) used as input to the scvi-tools model. For best practices of how/when to perform feature selection, please refer to the model-specific tutorial. For scVI, we recommend anywhere from 1,000 to 10,000 HVGs, but it will be context-dependent.

In [ ]:
```
sc.pp.highly_variable_genes(
    adata,
    n_top_genes=1200,
    subset=True,
    layer="counts",
    flavor="seurat_v3",
    batch_key="cell_source"
)
```

Now it's time to run `setup_anndata()`, which alerts scvi-tools to the locations of various matrices inside the anndata. It's important to run this function with the correct arguments so scvi-tools is notified that your dataset has batches, annotations, etc. For example, if batches are registered with scvi-tools, the subsequent model will correct for batch effects. See the full documentation for details.

In this dataset, there is a "cell_source" categorical covariate, and within each "cell_source", multiple "donors", "gender" and "age_group". There are also two continuous covariates we'd like to correct for: "percent_mito" and "percent_ribo". These covariates can be registered using the `categorical_covariate_keys` argument. If you only have one categorical covariate, you can also use the `batch_key` argument instead.

In [ ]:
```
scvi.model.SCVI.setup_anndata(
    adata,
    layer="counts",
    categorical_covariate_keys=["cell_source", "donor"],
    continuous_covariate_keys=["percent_mito", "percent_ribo"]
)
```

> Warning
>
> If the adata is modified after running `setup_anndata`, please run `setup_anndata` again, before creating an instance of a model.

Creating and training a model

While we highlight the scVI model here, the API is consistent across all scvi-tools models and is inspired by that of scikit-learn. For a full list of options, see the scvi documentation.

In [ ]:
```
model = scvi.model.SCVI(adata)
```

We can see an overview of the model by printing it.

In [ ]:
```
model
```

```
SCVI Model with the following params:
n_hidden: 128, n_latent: 10, n_layers: 1, dropout_rate: 0.1, dispersion: gene,
gene_likelihood: zinb, latent_distribution: normal
Training status: Not Trained
```

Out[ ]:

> Important
> All scvi-tools models run faster when using a GPU. By default, scvi-tools will use a GPU if one is found to be available. Please see the installation page for more information about installing scvi-tools when a GPU is available.

In [ ]:
```
model.train()
```

```
GPU available: True, used: True
TPU available: False, using: 0 TPU cores
IPU available: False, using: 0 IPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]
Epoch 400/400: 100%|██████████| 400/400 [05:43<00:00,  1.16it/s, loss=284, v_num=1]
```

Saving and loading

Saving consists of saving the model neural network weights, as well as parameters used to initialize the model.

In [ ]:
```
# model.save("my_model/")
```

In [ ]:
```
# model = scvi.model.SCVI.load("my_model/", adata=adata, use_gpu=True)
```

Obtaining model outputs

In [ ]:
```
latent = model.get_latent_representation()
```

It's often useful to store the outputs of scvi-tools back into the original anndata, as it permits interoperability with Scanpy.

In [ ]:
```
adata.obsm["X_scVI"] = latent
```

The `model.get...()` functions default to using the anndata that was used to initialize the model. It's possible to also query a subset of the anndata, or even use a completely independent anndata object as long as the anndata is organized in an equivalent fashion.

In [ ]:
```
adata_subset = adata[adata.obs.cell_type == "Fibroblast"]
latent_subset = model.get_latent_representation(adata_subset)
```

```
INFO     Received view of anndata, making copy.
INFO:scvi.model.base._base_model:Received view of anndata, making copy.
INFO     Input AnnData not setup with scvi-tools. attempting to transfer AnnData setup
INFO:scvi.model.base._base_model:Input AnnData not setup with scvi-tools. attempting to transfer AnnData setup
```

In [ ]:
```
denoised = model.get_normalized_expression(adata_subset, library_size=1e4)
denoised.iloc[:5, :5]
```

```
INFO     Received view of anndata, making copy.
INFO:scvi.model.base._base_model:Received view of anndata, making copy.
INFO     Input AnnData not setup with scvi-tools. attempting to transfer AnnData setup
INFO:scvi.model.base._base_model:Input AnnData not setup with scvi-tools. attempting to transfer AnnData setup
```

Out[ ]:

	ISG15	TNFRSF18	VWA1	HES5	SPSB1
GTCAAGTCATGCCACG-1-HCAHeart7702879	0.343808	0.118372	1.945633	0.062702	4.456603
GAGTCATTCTCCGTGT-1-HCAHeart8287128	1.552080	0.275485	1.457585	0.013442	14.617260
CCTCTGATCGTGACAT-1-HCAHeart7702881	5.157080	0.295140	1.195748	0.143792	2.908867
CGCCATTCATCATCTT-1-H0035_apex	0.352172	0.019281	0.570386	0.105633	6.325405
TCGTAGAGTAGGACTG-1-H0015_septum	0.290155	0.040910	0.400771	0.723409	8.142258

Let's store the normalized values back in the anndata.

In [ ]:
```
adata.layers["scvi_normalized"] = model.get_normalized_expression(
    library_size=10e4
)
```

Interoperability with Scanpy

Scanpy is a powerful python library for visualization and downstream analysis of scRNA-seq data. We show here how to feed the objects produced by scvi-tools into a scanpy workflow.

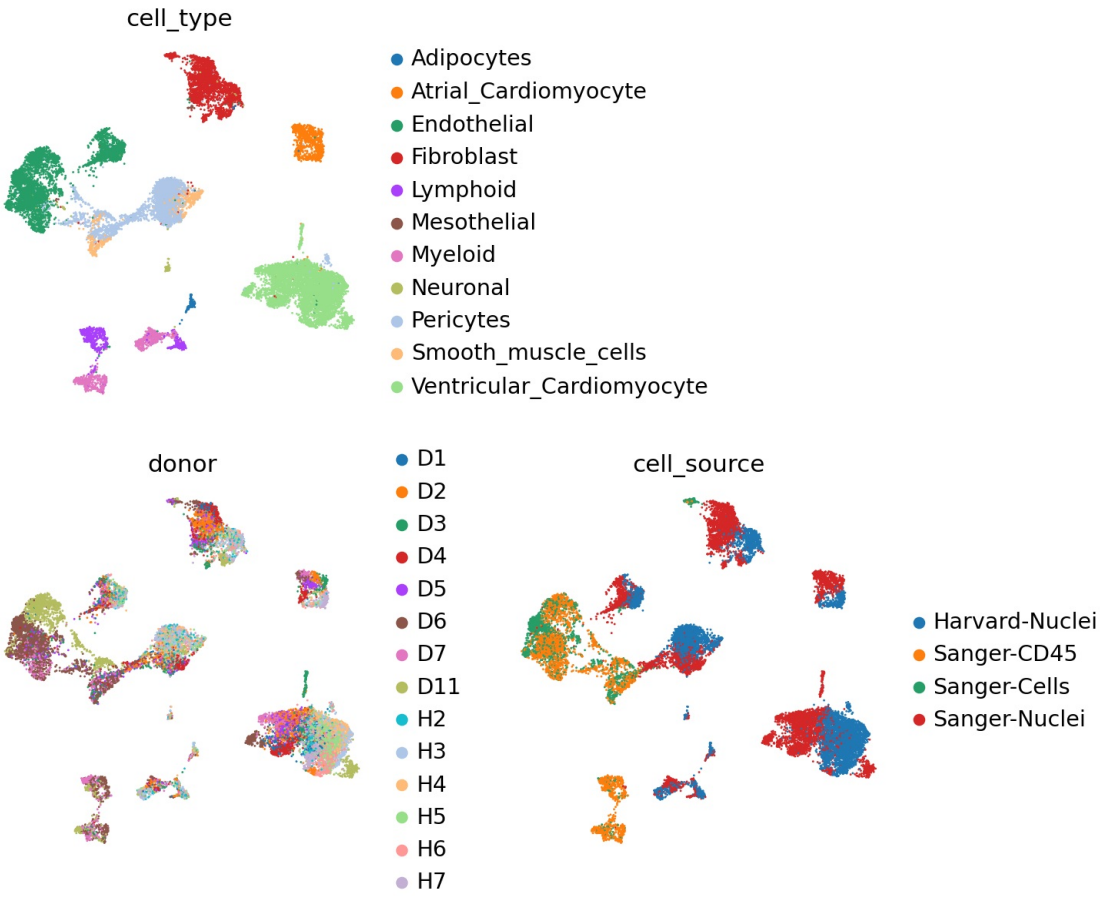
Visualization without batch correction

> Warning
>
> We use UMAP to *qualitatively* assess our low-dimension embeddings of cells. We do not advise using UMAP or any similar approach quantitatively. We do recommend using the embeddings produced by scVI as a plug-in replacement of what you would get from PCA, as we show below.

First, we demonstrate the presence of nuisance variation with respect to nuclei/whole cell, age group, and donor by plotting the UMAP results of the top 30 PCA components for the raw count data.

In [ ]:
```
# run PCA then generate UMAP plots
sc.tl.pca(adata)
sc.pp.neighbors(adata, n_pcs=30, n_neighbors=20)
sc.tl.umap(adata, min_dist=0.3)
```

In [ ]:
```
sc.pl.umap(
    adata,
    color=["cell_type"],
    frameon=False,
)
sc.pl.umap(
    adata,
    color=["donor", "cell_source"],
    ncols=2,
    frameon=False,
)
```



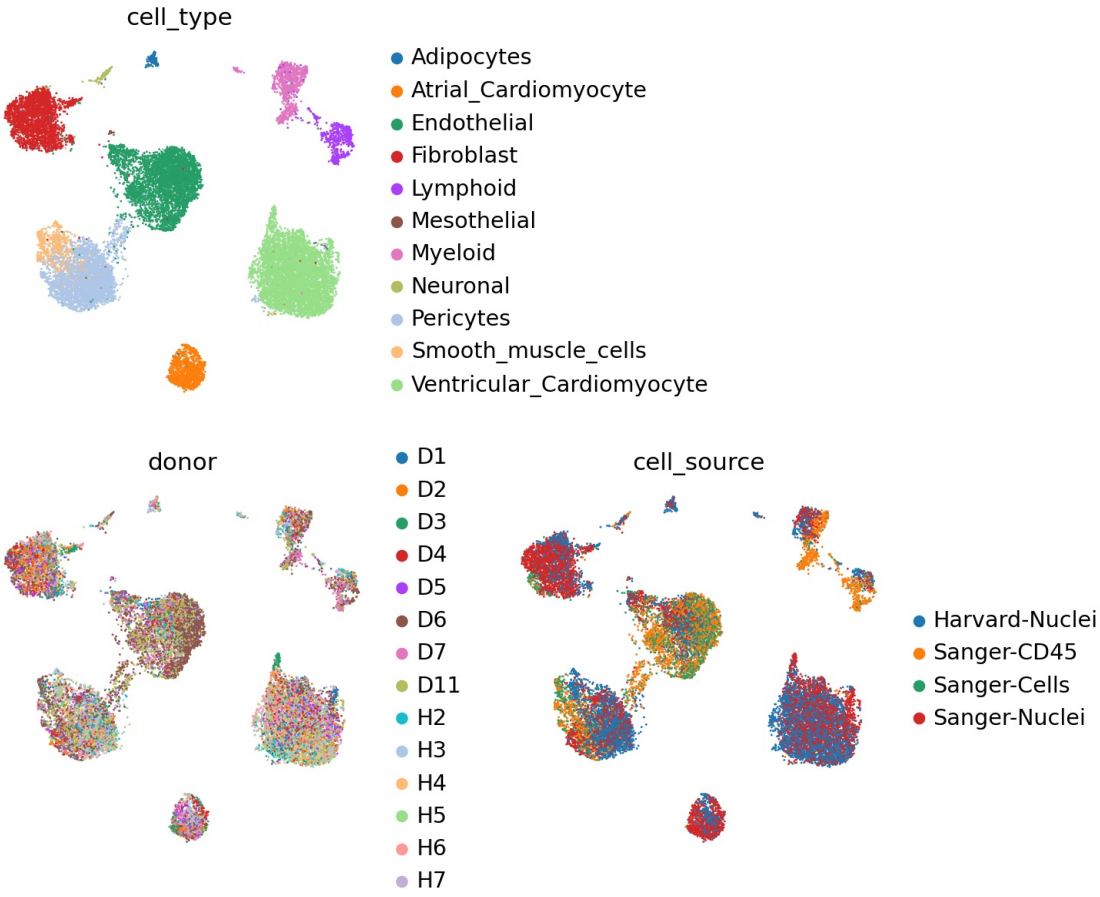
We see that while the cell types are generally well separated, nuisance variation plays a large part in the variation of the data.

Visualization with batch correction (scVI)

Now, let us try using the scVI latent space to generate the same UMAP plots to see if scVI successfully accounts for batch effects in the data.

In [ ]:
```
# use scVI latent space for UMAP generation
sc.pp.neighbors(adata, use_rep="X_scVI")
sc.tl.umap(adata, min_dist=0.3)
```

In [ ]:
```
sc.pl.umap(
    adata,
    color=["cell_type"],
    frameon=False,
)
sc.pl.umap(
    adata,
    color=["donor", "cell_source"],
    ncols=2,
    frameon=False,
)
```



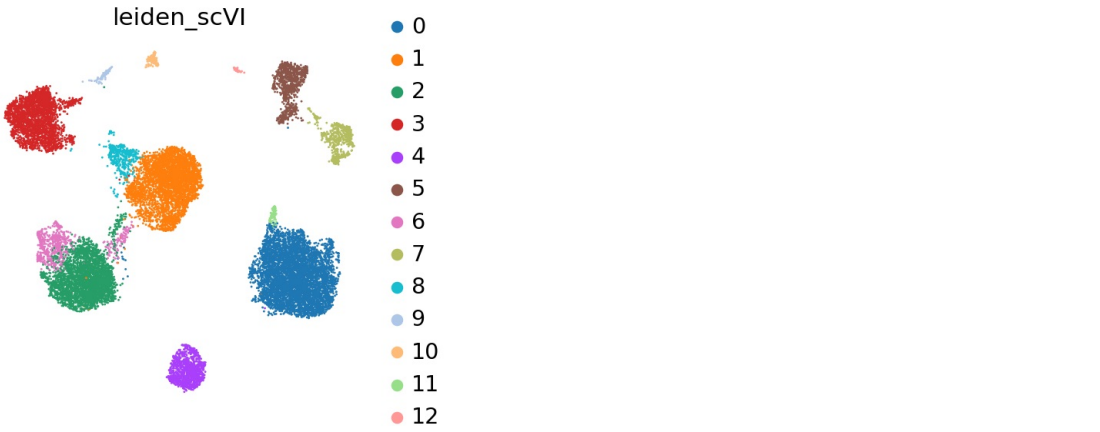
We can see that scVI was able to correct for nuisance variation due to nuclei/whole cell, age group, and donor, while maintaining separation of cell types.

Clustering on the scVI latent space

The user will note that we imported curated labels from the original publication. Our interface with scanpy makes it easy to cluster the data with scanpy from scVI's latent space and then reinject them into scVI (e.g., for differential expression).

In [ ]:
```
# neighbors were already computed using scVI
sc.tl.leiden(adata, key_added="leiden_scVI", resolution=0.5)
```

In [ ]:
```
sc.pl.umap(
    adata,
    color=["leiden_scVI"],
    frameon=False,
)
```



Differential expression

We can also use many scvi-tools models for differential expression. For further details on the methods underlying these functions as well as additional options, please see the API docs.

In [ ]:
```
adata.obs.cell_type.head()
```

Out[ ]:
```
AACTCCCCACGAGAGT-1-HCAHeart7844001                    Myeloid
ATAACGCAGAGCTGGT-1-HCAHeart7829979    Ventricular_Cardiomyocyte
GTCAAGTCATGCCACG-1-HCAHeart7702879                   Fibroblast
GGTGATTCAAATGAGT-1-HCAHeart8102858                  Endothelial
AGAGAATTCTTAGCAG-1-HCAHeart8102863                  Endothelial
Name: cell_type, dtype: category
Categories (11, object): ['Adipocytes', 'Atrial_Cardiomyocyte', 'Endothelial', 'Fibroblast', ..., 'Neuronal', '
Pericytes', 'Smooth_muscle_cells', 'Ventricular_Cardiomyocyte']
```

For example, a 1-vs-1 DE test is as simple as:

In [ ]:
```
de_df = model.differential_expression(
    groupby="cell_type",
    group1="Endothelial",
    group2="Fibroblast"
)
de_df.head()
```

```
DE...: 100%|██████████| 1/1 [00:00<00:00,  3.07it/s]
```

Out[ ]:

	proba_de	proba_not_de	bayes_factor	scale1	scale2	pseudocounts	delta	lfc_mean	lfc_median
SOX17	0.9998	0.0002	8.516943	0.001615	0.000029	0.0	0.25	6.222365	6.216846
SLC9A3R2	0.9996	0.0004	7.823621	0.010660	0.000171	0.0	0.25	5.977907	6.049340
ABCA10	0.9990	0.0010	6.906745	0.000081	0.006355	0.0	0.25	-8.468659	-9.058912
EGFL7	0.9986	0.0014	6.569875	0.008471	0.000392	0.0	0.25	4.751251	4.730982
VWF	0.9984	0.0016	6.436144	0.014278	0.000553	0.0	0.25	5.013347	5.029471

5 rows × 22 columns

We can also do a 1-vs-all DE test, which compares each cell type with the rest of the dataset:

In [ ]:
```
de_df = model.differential_expression(
    groupby="cell_type",
)
de_df.head()
```

```
DE...: 100%|██████████| 11/11 [00:03<00:00,  3.08it/s]
```

Out[ ]:

	proba_de	proba_not_de	bayes_factor	scale1	scale2	pseudocounts	delta	lfc_mean	lfc_median	
CIDEC	0.9988	0.0012	6.724225	0.002336	0.000031	0.0	0.25	7.082959	7.075700	2.6
ADIPOQ	0.9988	0.0012	6.724225	0.003627	0.000052	0.0	0.25	7.722131	7.461277	3.3
GPAM	0.9986	0.0014	6.569875	0.025417	0.000202	0.0	0.25	7.365266	7.381156	2.5
PLIN1	0.9984	0.0016	6.436144	0.004482	0.000048	0.0	0.25	7.818194	7.579515	2.9
GPD1	0.9974	0.0026	5.949637	0.002172	0.000044	0.0	0.25	6.543847	6.023436	2.8

5 rows × 22 columns

We now extract top markers for each cluster using the DE results.

In [ ]:
```
markers = {}
cats = adata.obs.cell_type.cat.categories
for i, c in enumerate(cats):
    cid = "{} vs Rest".format(c)
    cell_type_df = de_df.loc[de_df.comparison == cid]

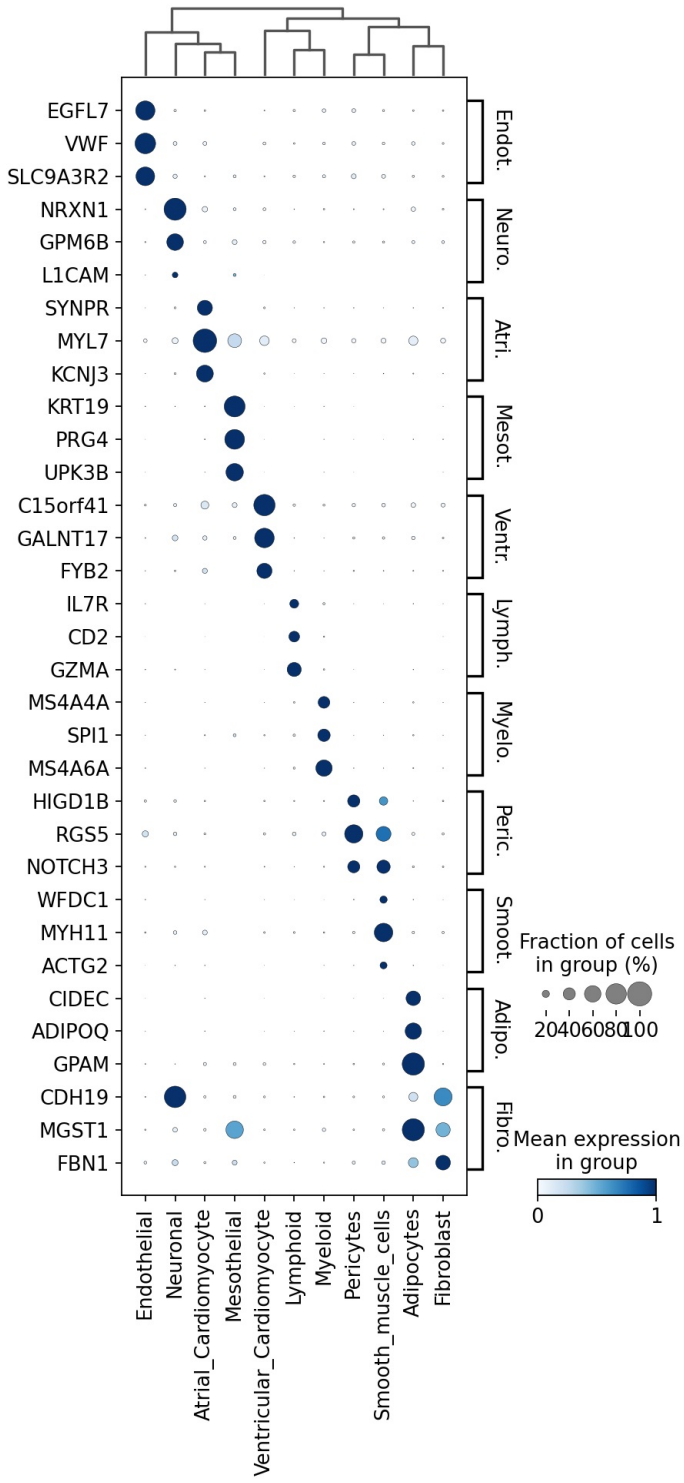
    cell_type_df = cell_type_df[cell_type_df.lfc_mean > 0]

    cell_type_df = cell_type_df[cell_type_df["bayes_factor"] > 3]
    cell_type_df = cell_type_df[cell_type_df["non_zeros_proportion1"] > 0.1]

    markers[c] = cell_type_df.index.tolist()[:3]
```

In [ ]:
```
sc.tl.dendrogram(adata, groupby="cell_type", use_rep="X_scVI")
```

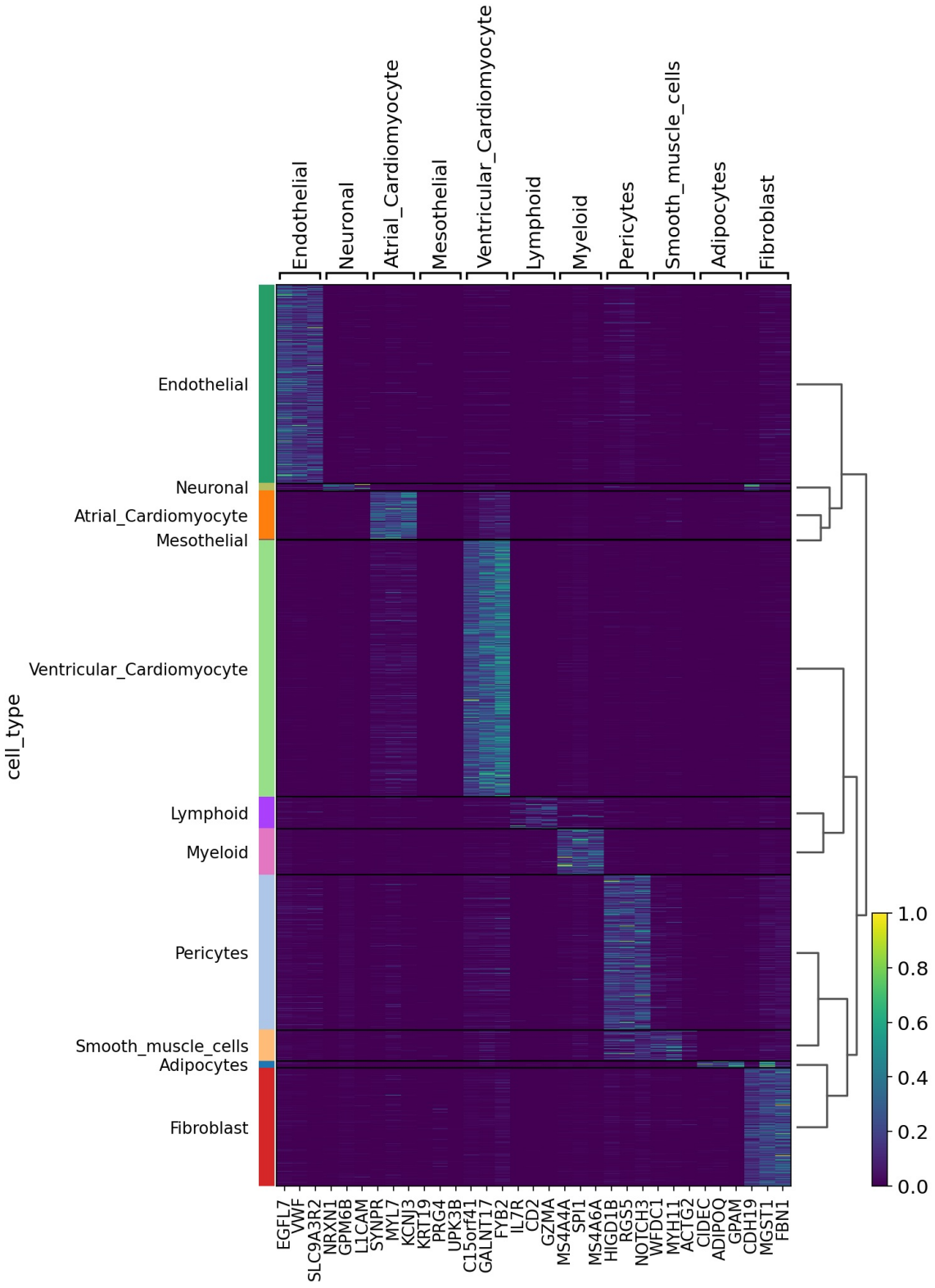
In [ ]:
```
sc.pl.dotplot(
    adata,
    markers,
    groupby='cell_type',
    dendrogram=True,
    color_map="Blues",
    swap_axes=True,
    use_raw=True,
    standard_scale="var",
)
```



We can also visualize the scVI normalized gene expression values with the `layer` option.

In [ ]:
```
sc.pl.heatmap(
    adata,
    markers,
    groupby='cell_type',
    layer="scvi_normalized",
    standard_scale="var",
    dendrogram=True,
```

```
    figsize=(8, 12)
)
```



Logging information

Verbosity varies in the following way:

- `logger.setLevel(logging.WARNING)` will show a progress bar.
- `logger.setLevel(logging.INFO)` will show global logs including the number of jobs done.
- `logger.setLevel(logging.DEBUG)` will show detailed logs for each training (e.g the parameters tested).

This function's behaviour can be customized, please refer to its documentation for information about the different parameters available.

In general, you can use `scvi.settings.verbosity` to set the verbosity of the scvi package. Note that `verbosity` corresponds to the logging levels of the standard python `logging` module. By default, that verbosity level is set to `INFO` (=20). As a reminder the logging levels are:

Level	Numeric value

`CRITICAL`	50
`ERROR`	40
`WARNING`	30
`INFO`	20
`DEBUG`	10
`NOTSET`	0

Reference

If you use scvi-tools in your research, please consider citing

In [ ]:
```
@article{Gayoso2022,
        author={Gayoso, Adam and Lopez, Romain and Xing, Galen and Boyeau, Pierre and Valiollah Pour Amiri, Va
        title={A Python library for probabilistic analysis of single-cell omics data},
        journal={Nature Biotechnology},
        year={2022},
        month={Feb},
        day={07},
        issn={1546-1696},
        doi={10.1038/s41587-021-01206-w},
        url={https://doi.org/10.1038/s41587-021-01206-w}
}
```

along with the publicaton describing the model used.

This tutorial was contributed to Deepchem by:

In [ ]:
```
@manual{Bioinformatics,
 title={Deep Probabilistic Analysis of Single-Cell Omics Data},
 organization={DeepChem},
 author={Paiz, Paulina},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Deep_probabilistic_ana
 year={2022},
}
```

Cell Counting

Cell counting is a fundamental task found in many biological research and medical diagnostic processes. It underlies decisions in cell culture, drug development, and disease analysis. However, traditional manual cell counting methods are often time-consuming and prone to human error. This variability can hinder research progress and lead to inconsistencies across studies.

Although cell counting machines exist, they are expensive and may not be readily available to all researchers. Automating cell counting using machine learning offers a powerful solution to this problem. ML-powered cell counters can quickly and accurately analyze large volumes of cell samples, freeing up researchers' time and minimizing inconsistencies.

Ready to build your own cell counter and revolutionize your research efficiency? This tutorial equips you with the knowledge and skills to create a customized tool that streamlines your cell counting needs.

Colab

This tutorial and the rest in this sequence can be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



Setup

To run DeepChem within Colab, you'll need to run the following installation commands. You can of course run this tutorial locally if you prefer. In that case, don't run these cells since they will download and install DeepChem in your local machine again.

In [16]:
```
!pip install --pre deepchem

import deepchem as dc
dc.__version__
```

```
Requirement already satisfied: deepchem in /usr/local/lib/python3.10/dist-packages (2.7.2.dev20240221173509)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.3.2)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.25.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.5.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.2.2)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.12)
Requirement already satisfied: scipy>=1.10.1 in /usr/local/lib/python3.10/dist-packages (from deepchem) (1.11.4)
Requirement already satisfied: rdkit in /usr/local/lib/python3.10/dist-packages (from deepchem) (2023.9.5)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->d
eepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->deepchem) (
2023.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from rdkit->deepchem) (9.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n->deepchem) (3.3.0)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->deepchem) (1
.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1-
>pandas->deepchem) (1.16.0)
```

Out[16]:
```
'2.7.2.dev'
```

Now we will import all the necessary packages and functions

In [17]:
```
import numpy as np
import matplotlib.pyplot as plt

from deepchem.data import NumpyDataset
from deepchem.models.torch_models import CNN
```

BBBC Datasets

We used the image set BBBC002v1 [Carpenter et al., Genome Biology, 2006] from the Broad Bioimage Benchmark Collection [Ljosa et al., Nature Methods, 2012] for this tutorial.

The Broad Bioimage Benchmark Collection Dataset 002 (BBBC002) contains images of Drosophila Kc167 cells. The ground truth labels consist of cell counts. Full details about this dataset are present at

https://bbbc.broadinstitute.org/BBBC002.

For counting cells, our dataset needs to have images as inputs and the corresponding cell counts as the ground truth labels. We have several BBBC datasets that can be loaded using the deepchem package. These datasets are an extension to `MoleculeNet` and can be accessed through `dc.molnet`.

The `BBBC002` dataset consists of 60 images, each 512x512 pixels in size, which are split into train, validation and test sets in a 80/10/10 split by default.
We also use `splitter='random'` in order to ensure that these images are randomly split into the train, validation and test sets in the above mention ratios.

In [18]:
```
bbbc2_dataset = dc.molnet.load_bbbc002(splitter='random')
tasks, dataset, transforms = bbbc2_dataset
train, val, test = dataset

train_x, train_y, train_w, train_ids = train.X, train.y, train.w, train.ids
val_x, val_y, val_w, val_ids = val.X, val.y, val.w, val.ids
test_x, test_y, test_w, test_ids = test.X, test.y, test.w, test.ids
```

Now that we've loaded the dataset and randomly split it, let's take a look at the data.

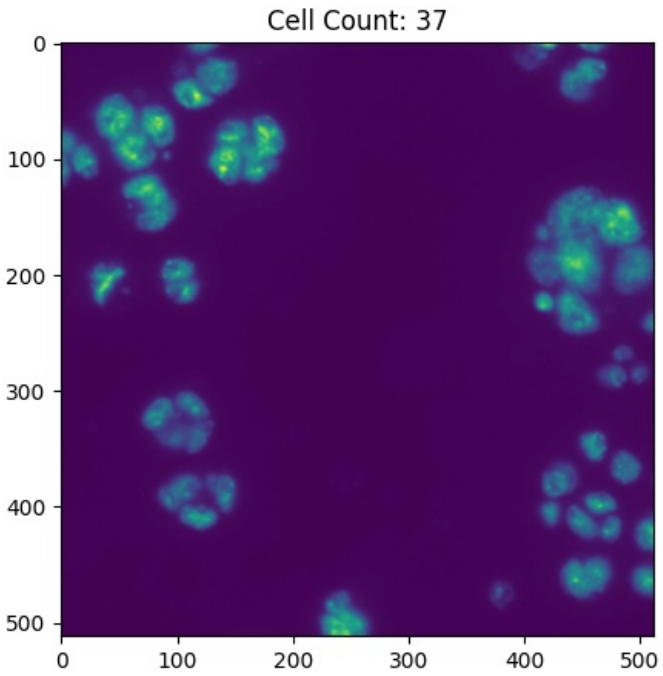
In [19]:
```
print(f"Shape of train data: {train_x.shape}")
print(f"Shape of train labels: {train_y.shape}")
```

```
Shape of train data: (40, 512, 512)
Shape of train labels: (40,)
```

We can confirm that a sample from our dataset is in the form of a 512x512 image. Let's visualize this sample:

In [20]:
```
i = 2

plt.figure(figsize=(5, 5))
plt.imshow(train_x[i])
plt.title(f"Cell Count: {train_y[i]}")
plt.show()
```



Now let's prepare the data for the model.

PyTorch based CNN Models require that images be in the shape of (C, H, W), wherein 'C' is the number of input channels, 'H' is the height of the image and 'W' is the width of the image. So we will reshape the data.

In [21]:
```
train_x = np.array(train_x.reshape(-1, 512, 512, 1), dtype=np.float32)
train_y = np.array(train_y.reshape(-1), dtype=np.float32)

val_x = np.array(val_x.reshape(-1, 512, 512, 1), dtype=np.float32)
val_y = np.array(val_y.reshape(-1), dtype=np.float32)

test_x = np.array(test_x.reshape(-1, 512, 512, 1), dtype=np.float32)
test_y = np.array(test_y.reshape(-1), dtype=np.float32)
```

In [22]:
```
train_data = NumpyDataset(train_x, train_y)
val_data = NumpyDataset(val_x, val_y)
```

```
test_data = NumpyDataset(test_x, test_y)
```

Creating and training our model

We will use the `rms_score` metric for our Validation Callback in order to monitor the performance of the model during training.

For more information on how to use callbacks, refer to this tutorial on Advanced Model Training

We will use the CNN model from the deepchem package. Since cell counting is a relational problem, we will use the `regression` mode.

We will use a 2D CNN model, with 6 hidden layers of the following sizes [32, 64, 128, 128, 64, 32] and a kernel size of 3 across all the filters, you can modify both the kernel size and the number of filters per layer. We have also used average pooling made residual connections and added dropout layers between subsequent layers in order to improve performance. Feel free to experiment with various models.

In [28]:
```
regression_metric = dc.metrics.Metric(dc.metrics.rms_score)

model = CNN(n_tasks=1, n_features=1, dims=2, layer_filters = [32, 64, 128, 128, 64, 32], kernel_size=3, learning
            mode='regression', padding='same', batch_size=4, residual=True, dropouts=0.1, pool_type='average')

callback = dc.models.ValidationCallback(val_data, 10, [regression_metric])

avg_loss = model.fit(train_data, nb_epoch=20, callbacks=callback)
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/modules/lazy.py:180: UserWarning: Lazy modules are a new featur
e under heavy development so changes to the API or functionality can happen at any moment.
  warnings.warn('Lazy modules are a new feature under heavy development '
Step 10 validation: rms_score=43.5554
Step 20 validation: rms_score=61.2546
Step 30 validation: rms_score=39.8772
Step 40 validation: rms_score=52.2475
Step 50 validation: rms_score=42.5802
Step 60 validation: rms_score=38.1957
Step 70 validation: rms_score=72.0341
Step 80 validation: rms_score=35.5798
Step 90 validation: rms_score=46.5774
Step 100 validation: rms_score=31.5153
Step 110 validation: rms_score=38.8215
Step 120 validation: rms_score=35.6907
Step 130 validation: rms_score=29.9797
Step 140 validation: rms_score=28.0428
Step 150 validation: rms_score=44.3926
Step 160 validation: rms_score=37.7657
Step 170 validation: rms_score=34.5076
Step 180 validation: rms_score=26.8319
Step 190 validation: rms_score=26.6618
Step 200 validation: rms_score=26.4627
```

Evaluating the performance of our model

Now let's use `mean_absolute_error` as our test metric and print out the results of our model. We have also created a graph of True vs Predicted values in order to visualize our model's performance

We can see that the model performs fairly well with a test loss of about 14.6. This means that on average, the predicted number of cells for a sample image is off by 14.6 cells when compared to the ground truth. Although this seems like a very high value for test loss, we will see that a difference of about 15 cells is actually not bad for this particular task.

In [32]:
```
test_metric = dc.metrics.Metric(dc.metrics.mean_absolute_error)

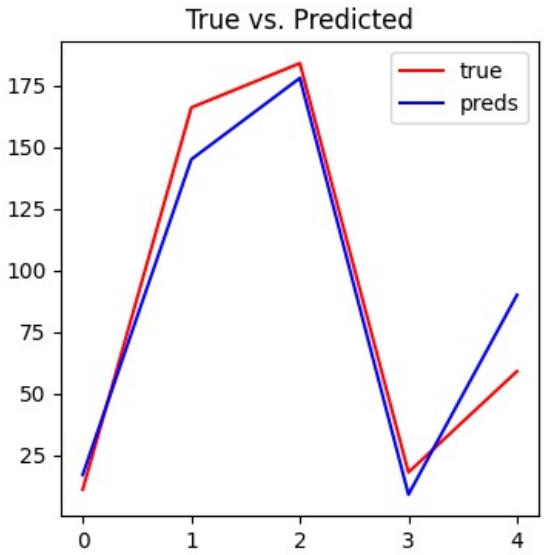
preds = np.array(model.predict(train_data), dtype=np.uint32)
print("Train loss: ", test_metric.compute_metric(train_y, preds))

preds = np.array(model.predict(val_data), dtype=np.uint32)
print("Val Loss: ", test_metric.compute_metric(val_y, preds))

preds = np.array(model.predict(test_data), dtype=np.uint32)
print("Test Loss: ", test_metric.compute_metric(test_y, preds))

plt.figure(figsize=(4, 4))
plt.title("True vs. Predicted")
plt.plot(test_y, color='red', label='true')
plt.plot(preds, color='blue', label='preds')
plt.legend()
plt.show()
```

```
Train loss:  19.05
Val Loss:  22.2
Test Loss:  14.6
```



Let us print out the mean cell count of our predictions and compare them with the ground truth. We will also print out the maximum difference between the ground truth and the prediction from the test set.

In [35]:
```
print(f"Mean of True Values: {np.mean(test_y):.2f}")
print(f"Mean of Predictions: {np.mean(preds):.2f}")

diff = []
for i in range(len(test_y)):
  diff.append(abs(test_y[i] - preds[i]))

print(f"Max of Difference: {np.max(diff)}")
```

```
Mean of True Values: 87.60
Mean of Predictions: 87.80
Max of Difference: 31.0
```

We can observe that the averages of our predictions and the ground truth are very close with a difference of just 0.20. Although we see a maximum difference of 31 cells between the prediction and true value, when we take into account the `Test Loss`, the close proximity of the means of predictions and the true labels, and the small size of our test set, we can say that our model performs fairly well.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

In [ ]:
```
@manual{Bioinformatics,
 title={Cell Counting Tutorial},
 organization={DeepChem},
 author={Menezes, Aaron},
 howpublished = {\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Cell_Counting_Tutoria
 year={2024},
```

```
}
```

Introduction To Material Science

Table of Contents:

- Introduction
- Setup
- Featurizers
  - Crystal Featurizers
  - Compound Featurizers
- Datasets
- Predicting structural properties of a crystal
- Further Reading

Introduction

One of the most exciting applications of machine learning in the recent time is it's application to material science domain. DeepChem helps in development and application of machine learning to solid-state systems. As a starting point of applying machine learning to material science domain, DeepChem provides material science datasets as part of the MoleculeNet suite of datasets, data featurizers and implementation of popular machine learning algorithms specific to material science domain. This tutorial serves as an introduction of using DeepChem for machine learning related tasks in material science domain.

Traditionally, experimental research were used to find and characterize new materials. But traditional methods have high limitations by constraints of required resources and equipments. Material science is one of the booming areas where machine learning is making new in-roads. The discovery of new material properties holds key to lot of problems like climate change, development of new semi-conducting materials etc. DeepChem acts as a toolbox for using machine learning in material science.

This tutorial can also be used in Google colab. If you'd like to open this notebook in colab, you can use the following link. This notebook is made to run without any GPU support.



In [ ]:
```
!pip install --pre deepchem
```

DeepChem for material science will also require the additiona libraries `pymatgen` and `matminer`. These two libraries assist machine learning in material science. For graph neural network models which we will be used in the backend, DeepChem requires `dgl` library. All these can be installed using `pip`. **Note** when using locally, install a higher version of the jupyter notebook (>6.5.5, here on colab).

In [2]:
```
!pip install -q pymatgen==2023.12.18
!pip install -q matminer==0.9.0
!pip install -q dgl
!pip install -q tqdm
```

In [ ]:
```
import deepchem as dc
dc.__version__
```

In [4]:
```
from tqdm import tqdm
import matplotlib.pyplot as plt

import pymatgen as mg
from pymatgen import core as core

import os
os.environ['DEEPCHEM_DATA_DIR'] = os.getcwd()
```

Featurizers

Material Structure Featurizers

Crystal are geometric structures which has to be featurized for using in machine learning algorithms. The following featurizers provided by DeepChem helps in featurizing crystals:

- The SineCoulombMatrix featurizer a crystal by calculating sine coulomb matrix for the crystals. It can be called using `dc.featurizers.SineCoulombMatrix` function. [1]

- The CGCNNFeaturizer calculates structure graph features of crystals. It can be called using `dc.featurizers.CGCNNFeaturizer` function. [2]
- The LCNNFeaturizer calculates the 2-D Surface graph features in 6 different permutations. It can be used using the utility `dc.feat.LCNNFeaturizer`. [3]

[1] Faber et al. “Crystal Structure Representations for Machine Learning Models of Formation Energies”, Inter. J. Quantum Chem. 115, 16, 2015. https://arxiv.org/abs/1503.07406

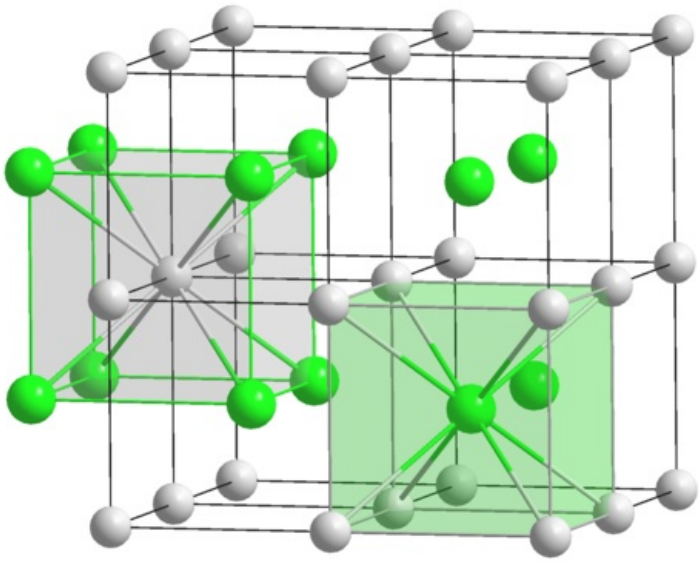
[2] T. Xie and J. C. Grossman, “Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties”, Phys. Rev. Lett. 120, 2018, https://arxiv.org/abs/1710.10324

[3] Jonathan Lym, Geun Ho Gu, Yousung Jung, and Dionisios G. Vlachos, Lattice Convolutional Neural Network Modeling of Adsorbate Coverage Effects, J. Phys. Chem. C 2019 https://pubs.acs.org/doi/10.1021/acs.jpcc.9b03370

Example: Featurizing a crystal

In this part, we will be using `pymatgen` for representing the crystal structure of Caesium Chloride and calculate structure graph features using `CGCNNFeaturizer`.

The `CsCl` crystal is a cubic lattice with the chloride atoms lying upon the lattice points at the edges of the cube, while the caesium atoms lie in the holes in the center of the cubes. The green colored atoms are the caesium atoms in this crystal structure and chloride atoms are the grey ones.



Source: Wikipedia

In [5]:
```
# the lattice paramter of a cubic cell
a = 4.2
lattice = core.Lattice.cubic(a)

# Atoms in a crystal
atomic_species = ["Cs", "Cl"]
# Coordinates of atoms in a crystal
cs_coords = [0, 0, 0]
cl_coords = [0.5, 0.5, 0.5]
structure = mg.core.Structure(lattice, atomic_species, [cs_coords, cl_coords])
structure
```

Out[5]:
```
Structure Summary
Lattice
    abc : 4.2 4.2 4.2
 angles : 90.0 90.0 90.0
 volume : 74.08800000000001
      A : 4.2 0.0 0.0
      B : 0.0 4.2 0.0
      C : 0.0 0.0 4.2
    pbc : True True True
PeriodicSite: Cs (0.0, 0.0, 0.0) [0.0, 0.0, 0.0]
PeriodicSite: Cl (2.1, 2.1, 2.1) [0.5, 0.5, 0.5]
```

In above code sample, we first defined a cubic lattice using the cubic lattice parameter `a`. Then, we created a structure

with atoms in the crystal and their coordinates as features. A nice introduction to crystallographic coordinates can be found here. Once a structure is defined, it can be featurized using CGCNN Featurizer. Featurization of a crystal using `CGCNNFeaturizer` returns a DeepChem GraphData object which can be used for machine learning tasks.

In [6]:
```
featurizer = dc.feat.CGCNNFeaturizer()
features = featurizer.featurize([structure])
features[0]
```

Out[6]:
```
GraphData(node_features=[2, 92], edge_index=[2, 24], edge_features=[24, 41])
```

Material Composition Featurizers

The above part discussed about using DeepChem for featurizing crystal structures. Here, we will be seeing about featurizing material compositions. DeepChem supports the following material composition featurizers:

- The ElementPropertyFingerprint can be used to find fingerprint of elements based on elemental stoichiometry. It can be used using a call to `dc.featurizers.ElementPropertyFingerprint`. [4]
- The ElemNetFeaturizer returns a vector containing fractional compositions of each element in the compound. It can be used using a call to `dc.feat.ElemNetFeaturizer`. [5]

[4] Ward, L., Agrawal, A., Choudhary, A. et al. A general-purpose machine learning framework for predicting properties of inorganic materials. npj Comput Mater 2, 16028 (2016). https://doi.org/10.1038/npjcompumats.2016.28

[5] Jha, D., Ward, L., Paul, A. et al. "ElemNet: Deep Learning the Chemistry of Materials From Only Elemental Composition", Sci Rep 8, 17593 (2018). https://doi.org/10.1038/s41598-018-35934-y

Example: Featurizing a compund

In the below example, we featurize Ferric Oxide (Fe2O3) using `ElementPropertyFingerprint` featurizer . The featurizer returns the compounds elemental stoichoimetry properties as features.

In [7]:
```
comp = core.Composition("Fe2O3")
featurizer = dc.feat.ElementPropertyFingerprint()
features = featurizer.featurize([comp])
features[0]
```

```
/usr/local/lib/python3.10/dist-packages/pymatgen/core/periodic_table.py:186: UserWarning: No data available for electrical_resistivity for O
  warnings.warn(f"No data available for {item} for {self.symbol}")
/usr/local/lib/python3.10/dist-packages/pymatgen/core/periodic_table.py:186: UserWarning: No data available for bulk_modulus for O
  warnings.warn(f"No data available for {item} for {self.symbol}")
/usr/local/lib/python3.10/dist-packages/pymatgen/core/periodic_table.py:186: UserWarning: No data available for coefficient_of_linear_thermal_expansion for O
  warnings.warn(f"No data available for {item} for {self.symbol}")
```

Out[7]:
```
array([1.83000000e+00, 3.44000000e+00, 1.61000000e+00, 2.79600000e+00,
       1.13844192e+00, 2.00000000e+00, 4.00000000e+00, 2.00000000e+00,
       2.80000000e+00, 1.41421356e+00, 8.00000000e+00, 1.60000000e+01,
       8.00000000e+00, 1.28000000e+01, 5.65685425e+00, 2.00000000e+00,
       3.00000000e+00, 1.00000000e+00, 2.40000000e+00, 7.07106781e-01,
       1.59994000e+01, 5.58450000e+01, 3.98456000e+01, 3.19376400e+01,
       2.81750940e+01, 6.00000000e-01, 1.40000000e+00, 8.00000000e-01,
       9.20000000e-01, 5.65685425e-01, 6.10000000e+01, 1.01000000e+02,
       4.00000000e+01, 8.50000000e+01, 2.82842712e+01, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       3.17500000e+02, 4.91000000e+03, 4.59250000e+03, 2.15450000e+03,
       3.24738789e+03, 2.65800000e-02, 8.00000000e+01, 7.99734200e+01,
       3.20159480e+01, 5.65497476e+01, 5.48000000e+01, 1.81100000e+03,
       1.75620000e+03, 7.57280000e+02, 1.24182093e+03, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
       0.00000000e+00])
```

Datasets

DeepChem has the following material properties dataset as part of MoleculeNet suite of datasets. These datasets can be used for a variety of tasks in material science like predicting structure formation energy, metallicity of a compound etc.

- The Band Gap dataset contains 4604 experimentally measured band gaps for inorganic crystal structure compositions. The dataset can be loaded using `dc.molnet.load_bandgap` utility.
- The Perovskite dataset contains 18928 perovskite structures and their formation energies. It can be loaded using a call to `dc.molnet.load_perovskite`.
- The Formation Energy dataset contains 132752 calculated formation energies and inorganic crystal structures from the Materials Project database. It can be loaded using a call to `dc.molnet.load_mp_formation_energy`.

- The Metallicity dataset contains 106113 inorganic crystal structures from the Materials Project database labeled as metals or nonmetals. It can be loaded using `dc.molnet.load_mp_metallicity` utility.

In the below example, we will demonstrate loading *perovskite* dataset and use it to predict *formation energy* of new crystals. Perovskite structures are structures adopted by many oxides. Ideally it is a cubic structure but non-cubic variants also exists. Each datapoint in the `perovskite` dataset contains the lattice structure as a `pymatgen.core.Structure` object and the formation energy of the corresponding structure. It can be used by calling for machine learning tasks by calling `dc.molnet.load_perovskite` utility. The utility takes care of loading, featurizing and splitting the dataset for machine learning tasks.

In [8]:
```
dataset_config = {"reload": True, "featurizer": dc.feat.CGCNNFeaturizer(), "transformers": []}
tasks, datasets, transformers = dc.molnet.load_perovskite(**dataset_config)
train_dataset, valid_dataset, test_dataset = datasets
```

In [9]:
```
train_dataset.get_data_shape
```

Out[9]:

deepchem.data.datasets.DiskDataset.get_data_shape
def get_data_shape() -> Shape

Gets array shape of datapoints in this dataset.

Predicting Formation Energy

Along with the dataset and featurizers, DeepChem also provide implementation of various machine learning algorithms which can be used on the fly for material science applications. For predicting formation energy, we use `CGCNNModel` as described in the paper [1].

In [10]:
```
model = dc.models.CGCNNModel(mode='regression', batch_size=256, learning_rate=0.0008)

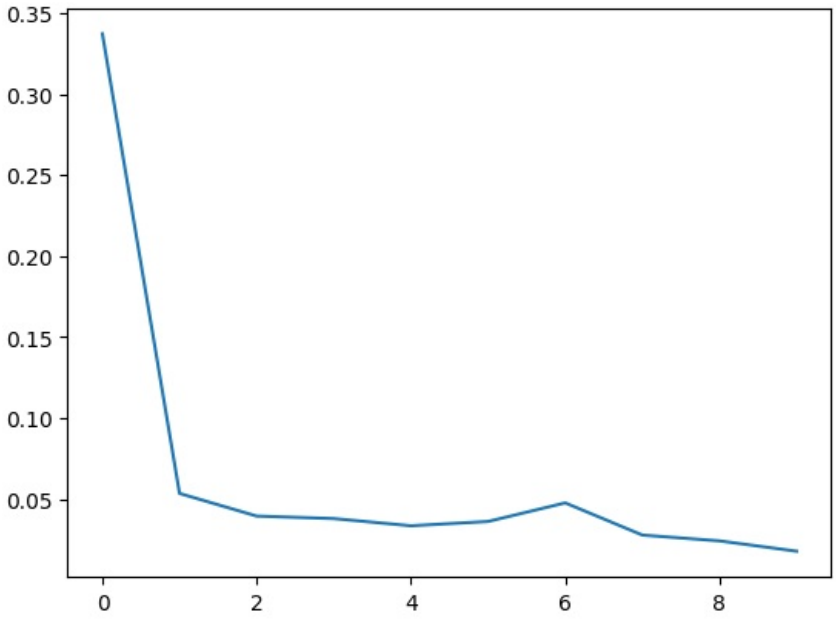
losses = []

for _ in tqdm(range(10), desc="Training"):
    loss = model.fit(train_dataset, nb_epoch=1)
    losses.append(loss)

plt.plot(losses)
```

```
Training: 100%|██████████| 10/10 [11:26<00:00, 68.65s/it]
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x7967a9a126e0>]
```



Once fitting the model, we evaluate the performance of the model using mean squared error metric since it is a regression task. For selection a metric, `dc.metrics.mean_squared_error` function can be used and we evaluate the model by calling `dc.model.evaluate`.`

In [11]:
```
metric = dc.metrics.Metric(dc.metrics.mean_absolute_error)
print("Training set score:", model.evaluate(train_dataset, [metric], transformers))
print("Test set score:", model.evaluate(test_dataset, [metric], transformers))
```

```
Training set score: {'mean_absolute_error': 0.1310973839991837}
Test set score: {'mean_absolute_error': 0.13470105945654667}
```

The original paper achieved a MAE of 0.130 eV/atom on the same dataset (with a 60:20:20 split, instead of the 80:10:10 being used in this tutorial).

Further Reading

For further reading on getting started on using machine learning for material science, here are two great resources:

- Getting Started in Material Informatics
- A Collection of Open Source Material Informatics Resources

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Using Reinforcement Learning to Play Pong

This tutorial demonstrates using reinforcement learning to train an agent to play Pong. This task isn't directly related to chemistry, but video games make an excellent demonstration of reinforcement learning techniques.



Colab

This tutorial and the rest in this sequence can be done in Google Colab (although the visualization at the end doesn't work correctly on Colab, so you might prefer to run this tutorial locally). If you'd like to open this notebook in colab, you can use the following link.



In [1]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

```
Requirement already satisfied: deepchem in c:\users\hp\deepchem_2 (2.8.1.dev20240501183346)
Requirement already satisfied: joblib in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from deepchem) (1.3.
2)
Requirement already satisfied: numpy>=1.21 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from deepchem)
(1.26.4)
Requirement already satisfied: pandas in c:\users\hp\anaconda3\envs\deep\lib\site-packages\pandas-2.2.1-py3.10-w
in-amd64.egg (from deepchem) (2.2.1)
Requirement already satisfied: scikit-learn in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from deepchem)
(1.4.1.post1)
Requirement already satisfied: sympy in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from deepchem) (1.12)
Requirement already satisfied: scipy>=1.10.1 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from deepchem
) (1.12.0)
Requirement already satisfied: rdkit in c:\users\hp\anaconda3\envs\deep\lib\site-packages\rdkit-2023.9.5-py3.10-
win-amd64.egg (from deepchem) (2023.9.5)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from
pandas->deepchem) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\hp\anaconda3\envs\deep\lib\site-packages\pytz-2024.1-py3
.10.egg (from pandas->deepchem) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hp\anaconda3\envs\deep\lib\site-packages\tzdata-2024.1
-py3.10.egg (from pandas->deepchem) (2024.1)
Requirement already satisfied: Pillow in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from rdkit->deepchem
) (10.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from s
cikit-learn->deepchem) (3.3.0)
Requirement already satisfied: mpmath>=0.19 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from sympy->de
epchem) (1.3.0)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from python-dateut
il>=2.8.2->pandas->deepchem) (1.16.0)
No normalization for SPS. Feature removed!
No normalization for AvgIpc. Feature removed!
WARNING:tensorflow:From c:\Users\HP\anaconda3\envs\deep\lib\site-packages\keras\src\losses.py:2976: The name tf.
losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy i
nstead.

WARNING:tensorflow:From c:\Users\HP\anaconda3\envs\deep\lib\site-packages\tensorflow\python\util\deprecation.py:
588: calling function (from tensorflow.python.eager.polymorphic_function.polymorphic_function) with experimental
_relax_shapes is deprecated and will be removed in a future version.
Instructions for updating:
experimental_relax_shapes is deprecated, use reduce_retracing instead
Skipped loading modules with pytorch-geometric dependency, missing a dependency. No module named 'dgl'
Skipped loading modules with transformers dependency. No module named 'transformers'
cannot import name 'HuggingFaceModel' from 'deepchem.models.torch_models' (c:\users\hp\deepchem_2\deepchem\model
s\torch_models\__init__.py)
Skipped loading modules with pytorch-lightning dependency, missing a dependency. No module named 'lightning'
Skipped loading some Jax models, missing a dependency. No module named 'jax'
```

Out[1]:
```
'2.8.1.dev'
```

In [2]:
```
!pip install "gym[atari,accept-rom-license]"
```

```
Requirement already satisfied: gym[accept-rom-license,atari] in c:\users\hp\anaconda3\envs\deep\lib\site-package
s (0.26.2)
Requirement already satisfied: numpy>=1.18.0 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from gym[acce
pt-rom-license,atari]) (1.26.4)
Requirement already satisfied: cloudpickle>=1.2.0 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from gym
[accept-rom-license,atari]) (3.0.0)
Requirement already satisfied: gym-notices>=0.0.4 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from gym
[accept-rom-license,atari]) (0.0.8)
Requirement already satisfied: ale-py~=0.8.0 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from gym[acce
pt-rom-license,atari]) (0.8.1)
Requirement already satisfied: autorom~=0.4.2 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from autorom
[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari]) (0.4.2)
Requirement already satisfied: importlib-resources in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from al
e-py~=0.8.0->gym[accept-rom-license,atari]) (6.4.0)
Requirement already satisfied: typing-extensions in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from ale-
py~=0.8.0->gym[accept-rom-license,atari]) (4.9.0)
Requirement already satisfied: click in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from autorom~=0.4.2->
autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari]) (8.1.7)
Requirement already satisfied: requests in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from autorom~=0.4.
2->autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari]) (2.31.0)
Requirement already satisfied: tqdm in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from autorom~=0.4.2->a
utorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari]) (4.66.2)
Requirement already satisfied: AutoROM.accept-rom-license in c:\users\hp\anaconda3\envs\deep\lib\site-packages (
from autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari]) (0.6.1)
Requirement already satisfied: colorama in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from click->autoro
m~=0.4.2->autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari]) (0.4
.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (fr
om requests->autorom~=0.4.2->autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-l
icense,atari]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from requests-
>autorom~=0.4.2->autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license,atari
]) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hp\anaconda3\envs\deep\lib\site-packages (from req
uests->autorom~=0.4.2->autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-rom-license
,atari]) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\hp\appdata\roaming\python\python310\site-packages
(from requests->autorom~=0.4.2->autorom[accept-rom-license]~=0.4.2; extra == "accept-rom-license"->gym[accept-ro
m-license,atari]) (2022.5.18.1)
```

Reinforcement Learning

Reinforcement learning involves an *agent* that interacts with an *environment*. In this case, the environment is the video game and the agent is the player. By trial and error, the agent learns a *policy* that it follows to perform some task (winning the game). As it plays, it receives *rewards* that give it feedback on how well it is doing. In this case, it receives a positive reward every time it scores a point and a negative reward every time the other player scores a point.

The first step is to create an `Environment` that implements this task. Fortunately, OpenAI Gym already provides an implementation of Pong (and many other tasks appropriate for reinforcement learning). DeepChem's `GymEnvironment` class provides an easy way to use environments from OpenAI Gym. We could just use it directly, but in this case we subclass it and preprocess the screen image a little bit to make learning easier.

In [3]:
```
import deepchem as dc
import numpy as np


class PongEnv(dc.rl.GymEnvironment):
  def __init__(self):
    super(PongEnv, self).__init__('Pong-v4')
    self._state_shape = (80, 80)

  @property
  def state(self):
    # Crop everything outside the play area, reduce the image size,
    # and convert it to black and white.
    state_array = self._state
    cropped = state_array[34:194, :, :]
    reduced = cropped[0:-1:2, 0:-1:2]
    grayscale = np.sum(reduced, axis=2)
    bw = np.zeros(grayscale.shape, dtype=np.float32)
    bw[grayscale != 233] = 1
    return bw

  def __deepcopy__(self, memo):
    return PongEnv()

env = PongEnv()
```

Next we create a model to implement our policy. This model receives the current state of the environment (the pixels

being displayed on the screen at this moment) as its input. Given that input, it decides what action to perform. In Pong there are three possible actions at any moment: move the paddle up, move it down, or leave it where it is. The policy model produces a probability distribution over these actions. It also produces a *value* output, which is interpreted as an estimate of how good the current state is. This turns out to be important for efficient learning.

The model begins with two convolutional layers to process the image. That is followed by a dense (fully connected) layer to provide plenty of capacity for game logic. We also add a small Gated Recurrent Unit (GRU). That gives the network a little bit of memory, so it can keep track of which way the ball is moving. Just from the screen image, you cannot tell whether the ball is moving to the left or to the right, so having memory is important.

We concatenate the dense and GRU outputs together, and use them as inputs to two final layers that serve as the network's outputs. One computes the action probabilities, and the other computes an estimate of the state value function.

We also provide an input for the initial state of the GRU, and return its final state at the end. This is required by the learning algorithm.

In [4]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

class PongPolicy(dc.rl.Policy):
    def __init__(self):
        super(PongPolicy, self).__init__(['action_prob', 'value', 'rnn_state'], [np.zeros(16, dtype=np.float32)

    def create_model(self, **kwargs):
        class TestModel(nn.Module):
            def __init__(self):
                super(TestModel, self).__init__()
                # Convolutional layers
                self.conv1 = nn.Conv2d(1, 16, kernel_size=8, stride=4)
                self.conv2 = nn.Conv2d(16, 32, kernel_size=4, stride=2)
                self.fc1 = nn.Linear(2048, 256)
                self.gru = nn.GRU(256, 16, batch_first = True)
                self.action_prob = nn.Linear(272, env.n_actions)
                self.value = nn.Linear(272, 1)
            def forward(self, inputs):
                state = (torch.from_numpy((inputs[0])))
                rnn_state = (torch.from_numpy(inputs[1]))
                reshaped = state.view(-1, 1, 80, 80)
                conv1 = F.relu(self.conv1(reshaped))
                conv2 = F.relu(self.conv2(conv1))
                conv2 = conv2.view(conv2.size(0), -1)
                x = F.relu(self.fc1(conv2))
                reshaped_x = x.view(1, -1, 256)
                #x = torch.flatten(x, 1)
                gru_out, rnn_final_state = self.gru(reshaped_x, rnn_state.unsqueeze(0))
                rnn_final_state = rnn_final_state.view(-1,16)
                gru_out = gru_out.view(-1, 16)
                concat = torch.cat((x, gru_out), dim=1)
                #concat = concat.view(-1, 272)
                action_prob = F.softmax(self.action_prob(concat), dim=-1)
                value = self.value(concat)
                return action_prob, value, rnn_final_state
        return TestModel()
policy = PongPolicy()
```

We will optimize the policy using the Advantage Actor Critic (A2C) algorithm. There are lots of hyperparameters we could specify at this point, but the default values for most of them work well on this problem. The only one we need to customize is the learning rate.

In [5]:
```
import torch.nn.functional as F
from deepchem.rl.torch_rl.torch_a2c import A2C

from deepchem.models.optimizers import Adam
a2c = A2C(env, policy, model_dir='model', optimizer=Adam(learning_rate=0.0002))
```

Optimize for as long as you have patience to. By 1 million steps you should see clear signs of learning. Around 3 million steps it should start to occasionally beat the game's built in AI. By 7 million steps it should be winning almost every time. Running on my laptop, training takes about 20 minutes for every million steps.

In [6]:
```
# Change this to train as many steps as you have patience for.
a2c.fit(1000)
```

```
c:\Users\HP\anaconda3\envs\deep\lib\site-packages\gym\utils\passive_env_checker.py:233: DeprecationWarning: `np.
bool8` is a deprecated alias for `np.bool_`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
```

Let's watch it play and see how it does!

In [7]:
```
# This code doesn't work well on Colab
env.reset()
while not env.terminated:
    env.env.render()
    env.step(a2c.select_action(env.state))
```

```
c:\Users\HP\anaconda3\envs\deep\lib\site-packages\gym\utils\passive_env_checker.py:289: UserWarning: WARN: No re
nder fps was declared in the environment (env.metadata['render_fps'] is None or not defined), rendering may occu
r at inconsistent fps.
  logger.warn(
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Uncertainty in Deep Learning

A common criticism of deep learning models is that they tend to act as black boxes. A model produces outputs, but doesn't given enough context to interpret them properly. How reliable are the model's predictions? Are some predictions more reliable than others? If a model predicts a value of 5.372 for some quantity, should you assume the true value is between 5.371 and 5.373? Or that it's between 2 and 8? In some fields this situation might be good enough, but not in science. For every value predicted by a model, we also want an estimate of the uncertainty in that value so we can know what conclusions to draw based on it.

DeepChem makes it very easy to estimate the uncertainty of predicted outputs (at least for the models that support it—not all of them do). Let's start by seeing an example of how to generate uncertainty estimates. We load a dataset, create a model, train it on the training set, predict the output on the test set, and then derive some uncertainty estimates.

Colab

This tutorial and the rest in this sequence are designed to be done in Google colab. If you'd like to open this notebook in colab, you can use the following link.



In [ ]:
```
!pip install --pre deepchem
import deepchem
deepchem.__version__
```

We'll use the Delaney dataset from the MoleculeNet suite to run our experiments in this tutorial. Let's load up our dataset for our experiments, and then make some uncertainty predictions.

In [1]:
```
import deepchem as dc
import numpy as np
import matplotlib.pyplot as plot

tasks, datasets, transformers = dc.molnet.load_delaney()
train_dataset, valid_dataset, test_dataset = datasets

model = dc.models.MultitaskRegressor(len(tasks), 1024, uncertainty=True)
model.fit(train_dataset, nb_epoch=20)
y_pred, y_std = model.predict_uncertainty(test_dataset)
```

All of this looks exactly like any other example, with just two differences. First, we add the option `uncertainty=True` when creating the model. This instructs it to add features to the model that are needed for estimating uncertainty. Second, we call `predict_uncertainty()` instead of `predict()` to produce the output. `y_pred` is the predicted outputs. `y_std` is another array of the same shape, where each element is an estimate of the uncertainty (standard deviation) of the corresponding element in `y_pred`. And that's all there is to it! Simple, right?

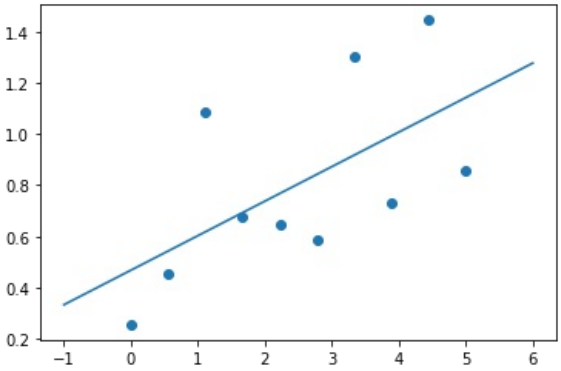
Of course, it isn't really that simple at all. DeepChem is doing a lot of work to come up with those uncertainties. So now let's pull back the curtain and see what is really happening. (For the full mathematical details of calculating uncertainty, see https://arxiv.org/abs/1703.04977)

To begin with, what does "uncertainty" mean? Intuitively, it is a measure of how much we can trust the predictions. More formally, we expect that the true value of whatever we are trying to predict should usually be within a few standard deviations of the predicted value. But uncertainty comes from many sources, ranging from noisy training data to bad modelling choices, and different sources behave in different ways. It turns out there are two fundamental types of uncertainty we need to take into account.

Aleatoric Uncertainty

Consider the following graph. It shows the best fit linear regression to a set of ten data points.

In [2]:
```
# Generate some fake data and plot a regression line.
x = np.linspace(0, 5, 10)
y = 0.15*x + np.random.random(10)
plot.scatter(x, y)
fit = np.polyfit(x, y, 1)
line_x = np.linspace(-1, 6, 2)
plot.plot(line_x, np.poly1d(fit)(line_x))
plot.show()
```



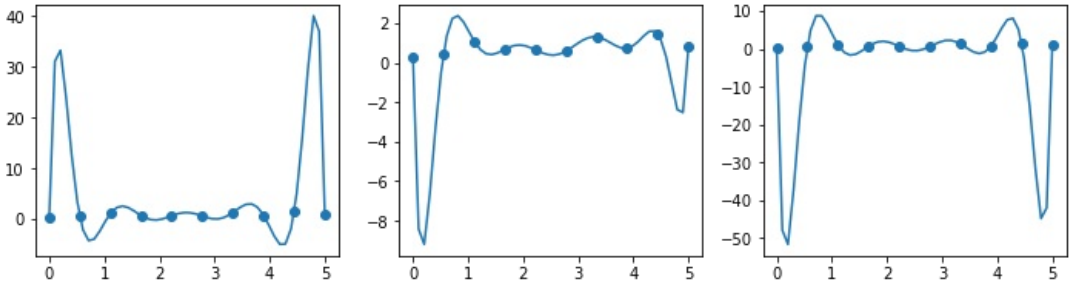
The line clearly does not do a great job of fitting the data. There are many possible reasons for this. Perhaps the measuring device used to capture the data was not very accurate. Perhaps `y` depends on some other factor in addition to `x`, and if we knew the value of that factor for each data point we could predict `y` more accurately. Maybe the relationship between `x` and `y` simply isn't linear, and we need a more complicated model to capture it. Regardless of the cause, the model clearly does a poor job of predicting the training data, and we need to keep that in mind. We cannot expect it to be any more accurate on test data than on training data. This is known as *aleatoric uncertainty*.

How can we estimate the size of this uncertainty? By training a model to do it, of course! At the same time it is learning to predict the outputs, it is also learning to predict how accurately each output matches the training data. For every output of the model, we add a second output that produces the corresponding uncertainty. Then we modify the loss function to make it learn both outputs at the same time.

Epistemic Uncertainty

Now consider these three curves. They are fit to the same data points as before, but this time we are using 10th degree polynomials.

In [3]:
```
plot.figure(figsize=(12, 3))
line_x = np.linspace(0, 5, 50)
for i in range(3):
    plot.subplot(1, 3, i+1)
    plot.scatter(x, y)
    fit = np.polyfit(np.concatenate([x, [3]]), np.concatenate([y, [i]]), 10)
    plot.plot(line_x, np.poly1d(fit)(line_x))
plot.show()
```



Each of them perfectly interpolates the data points, yet they clearly are different models. (In fact, there are infinitely many 10th degree polynomials that exactly interpolate any ten data points.) They make identical predictions for the data we fit them to, but for any other value of `x` they produce different predictions. This is called *epistemic uncertainty*. It means the data does not fully constrain the model. Given the training data, there are many different models we could have found, and those models make different predictions.

The ideal way to measure epistemic uncertainty is to train many different models, each time using a different random seed and possibly varying hyperparameters. Then use all of them for each input and see how much the predictions vary. This is very expensive to do, since it involves repeating the whole training process many times. Fortunately, we can approximate the same effect in a less expensive way: by using dropout.

Recall that when you train a model with dropout, you are effectively training a huge ensemble of different models all at once. Each training sample is evaluated with a different dropout mask, corresponding to a different random subset of the connections in the full model. Usually we only perform dropout during training and use a single averaged mask for prediction. But instead, let's use dropout for prediction too. We can compute the output for lots of different dropout masks, then see how much the predictions vary. This turns out to give a reasonable estimate of the epistemic uncertainty in the outputs.

Uncertain Uncertainty?

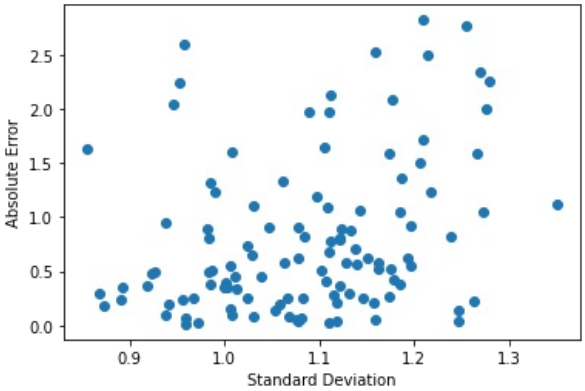
Now we can combine the two types of uncertainty to compute an overall estimate of the error in each output:

$$\sigma_{\text{total}} = \sqrt{\sigma_{\text{aleatoric}}^2 + \sigma_{\text{epistemic}}^2}$$

This is the value DeepChem reports. But how much can you trust it? Remember how I started this tutorial: deep learning models should not be used as black boxes. We want to know how reliable the outputs are. Adding uncertainty estimates does not completely eliminate the problem; it just adds a layer of indirection. Now we have estimates of how reliable the outputs are, but no guarantees that those estimates are themselves reliable.

Let's go back to the example we started with. We trained a model on the SAMPL training set, then generated predictions and uncertainties for the test set. Since we know the correct outputs for all the test samples, we can evaluate how well we did. Here is a plot of the absolute error in the predicted output versus the predicted uncertainty.

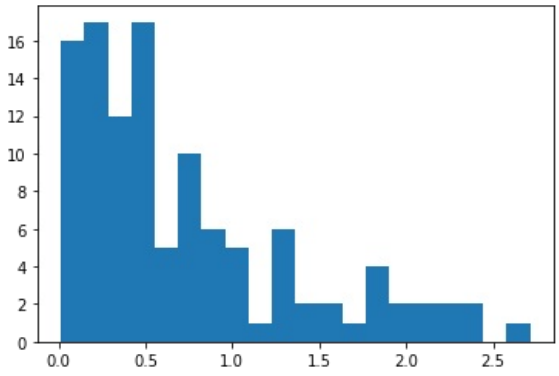
In [4]:
```
abs_error = np.abs(y_pred.flatten()-test_dataset.y.flatten())
plot.scatter(y_std.flatten(), abs_error)
plot.xlabel('Standard Deviation')
plot.ylabel('Absolute Error')
plot.show()
```



The first thing we notice is that the axes have similar ranges. The model clearly has learned the overall magnitude of errors in the predictions. There also is clearly a correlation between the axes. Values with larger uncertainties tend on average to have larger errors. (Strictly speaking, we expect the absolute error to be *less than* the predicted uncertainty. Even a very uncertain number could still happen to be close to the correct value by chance. If the model is working well, there should be more points below the diagonal than above it.)

Now let's see how well the values satisfy the expected distribution. If the standard deviations are correct, and if the errors are normally distributed (which is certainly not guaranteed to be true!), we expect 95% of the values to be within two standard deviations, and 99% to be within three standard deviations. Here is a histogram of errors as measured in standard deviations.

In [5]:
```
plot.hist(abs_error/y_std.flatten(), 20)
plot.show()
```



All the values are in the expected range, and the distribution looks roughly Gaussian although not exactly. Perhaps this indicates the errors are not normally distributed, but it may also reflect inaccuracies in the uncertainties. This is an important reminder: the uncertainties are just estimates, not rigorous measurements. Most of them are pretty good, but you should not put too much confidence in any single value.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

Starring DeepChem on GitHub helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!



In [ ]:
```
!pip install --pre deepchem[jax]
```

```
Collecting deepchem[jax]
  Downloading deepchem-2.6.0.dev20210924223259-py3-none-any.whl (609 kB)
     |▏                               | 10 kB 24.0 MB/s eta 0:00:01
     |▎                               | 20 kB 25.7 MB/s eta 0:00:01
     |▌                               | 30 kB 20.2 MB/s eta 0:00:01
     |▋                               | 40 kB 16.4 MB/s eta 0:00:01
     |▊                               | 51 kB 10.7 MB/s eta 0:00:01
     |▉                               | 61 kB 12.3 MB/s eta 0:00:01
     |█                               | 71 kB 11.7 MB/s eta 0:00:01
     |█                               | 81 kB 12.9 MB/s eta 0:00:01
     |█▏                              | 92 kB 11.7 MB/s eta 0:00:01
     |█▎                              | 102 kB 11.0 MB/s eta 0:00:01
     |█▍                              | 112 kB 11.0 MB/s eta 0:00:01
     |█▌                              | 122 kB 11.0 MB/s eta 0:00:01
     |█▋                              | 133 kB 11.0 MB/s eta 0:00:01
     |█▊                              | 143 kB 11.0 MB/s eta 0:00:01
     |█▉                              | 153 kB 11.0 MB/s eta 0:00:01
     |██                              | 163 kB 11.0 MB/s eta 0:00:01
     |██                              | 174 kB 11.0 MB/s eta 0:00:01
     |██▏                             | 184 kB 11.0 MB/s eta 0:00:01
     |██▎                             | 194 kB 11.0 MB/s eta 0:00:01
     |██▍                             | 204 kB 11.0 MB/s eta 0:00:01
     |██▌                             | 215 kB 11.0 MB/s eta 0:00:01
     |██▋                             | 225 kB 11.0 MB/s eta 0:00:01
     |██▊                             | 235 kB 11.0 MB/s eta 0:00:01
     |██▉                             | 245 kB 11.0 MB/s eta 0:00:01
     |███                             | 256 kB 11.0 MB/s eta 0:00:01
     |███                             | 266 kB 11.0 MB/s eta 0:00:01
     |███▏                            | 276 kB 11.0 MB/s eta 0:00:01
     |███▎                            | 286 kB 11.0 MB/s eta 0:00:01
     |███▍                            | 296 kB 11.0 MB/s eta 0:00:01
     |███▌                            | 307 kB 11.0 MB/s eta 0:00:01
     |███▋                            | 317 kB 11.0 MB/s eta 0:00:01
     |███▊                            | 327 kB 11.0 MB/s eta 0:00:01
     |███▉                            | 337 kB 11.0 MB/s eta 0:00:01
     |████                            | 348 kB 11.0 MB/s eta 0:00:01
     |████                            | 358 kB 11.0 MB/s eta 0:00:01
     |████▏                           | 368 kB 11.0 MB/s eta 0:00:01
     |████▎                           | 378 kB 11.0 MB/s eta 0:00:01
     |████▍                           | 389 kB 11.0 MB/s eta 0:00:01
     |████▌                           | 399 kB 11.0 MB/s eta 0:00:01
     |████▋                           | 409 kB 11.0 MB/s eta 0:00:01
     |████▊                           | 419 kB 11.0 MB/s eta 0:00:01
     |████▉                           | 430 kB 11.0 MB/s eta 0:00:01
     |█████                           | 440 kB 11.0 MB/s eta 0:00:01
     |█████                           | 450 kB 11.0 MB/s eta 0:00:01
     |█████▏                          | 460 kB 11.0 MB/s eta 0:00:01
     |█████▎                          | 471 kB 11.0 MB/s eta 0:00:01
     |█████▍                          | 481 kB 11.0 MB/s eta 0:00:01
     |█████▌                          | 491 kB 11.0 MB/s eta 0:00:01
     |█████▋                          | 501 kB 11.0 MB/s eta 0:00:01
     |█████▊                          | 512 kB 11.0 MB/s eta 0:00:01
     |█████▉                          | 522 kB 11.0 MB/s eta 0:00:01
     |██████                          | 532 kB 11.0 MB/s eta 0:00:01
     |██████                          | 542 kB 11.0 MB/s eta 0:00:01
     |██████▏                         | 552 kB 11.0 MB/s eta 0:00:01
     |██████▎                         | 563 kB 11.0 MB/s eta 0:00:01
     |██████▍                         | 573 kB 11.0 MB/s eta 0:00:01
     |██████▌                         | 583 kB 11.0 MB/s eta 0:00:01
     |██████▋                         | 593 kB 11.0 MB/s eta 0:00:01
     |██████▊                         | 604 kB 11.0 MB/s eta 0:00:01
     |██████████████████████████████| 609 kB 11.0 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (1.4.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (0.22
.2.post1)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (1.19.5)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (1.1.5)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (1.0.1)
Collecting optax
  Downloading optax-0.0.9-py3-none-any.whl (118 kB)
     |████████████████████████████████| 118 kB 52.6 MB/s
Requirement already satisfied: jaxlib in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (0.1.71+cud
a111)
Collecting dm-haiku
  Downloading dm_haiku-0.0.5.dev0-py3-none-any.whl (284 kB)
     |████████████████████████████████| 284 kB 30.1 MB/s
```

```
Requirement already satisfied: jax in /usr/local/lib/python3.7/dist-packages (from deepchem[jax]) (0.2.21)
Requirement already satisfied: tabulate>=0.8.9 in /usr/local/lib/python3.7/dist-packages (from dm-haiku->deepche
m[jax]) (0.8.9)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from dm-haiku->deepc
hem[jax]) (3.7.4.3)
Requirement already satisfied: absl-py>=0.7.1 in /usr/local/lib/python3.7/dist-packages (from dm-haiku->deepchem
[jax]) (0.12.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from absl-py>=0.7.1->dm-haiku->dee
pchem[jax]) (1.15.0)
Requirement already satisfied: opt-einsum in /usr/local/lib/python3.7/dist-packages (from jax->deepchem[jax]) (3
.3.0)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in /usr/local/lib/python3.7/dist-packages (from jaxlib->de
epchem[jax]) (1.12)
Collecting chex>=0.0.4
  Downloading chex-0.0.8-py3-none-any.whl (57 kB)
     |████████████████████████████████| 57 kB 6.0 MB/s
Requirement already satisfied: dm-tree>=0.1.5 in /usr/local/lib/python3.7/dist-packages (from chex>=0.0.4->optax
->deepchem[jax]) (0.1.6)
Requirement already satisfied: toolz>=0.9.0 in /usr/local/lib/python3.7/dist-packages (from chex>=0.0.4->optax->
deepchem[jax]) (0.11.1)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem[jax]) (2.8.2)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem[jax
]) (2018.9)
Installing collected packages: chex, optax, dm-haiku, deepchem
Successfully installed chex-0.0.8 deepchem-2.6.0.dev20210924223259 dm-haiku-0.0.5.dev0 optax-0.0.9
```

In [ ]:
```python
import numpy as np
import functools
try:
  import jax
  import jax.numpy as jnp
  import haiku as hk
  import optax
  from deepchem.models import PINNModel, JaxModel
  from deepchem.data import NumpyDataset
  from deepchem.models.optimizers import Adam
  from jax import jacrev
  has_haiku_and_optax = True
except:
  has_haiku_and_optax = False
```

Given Physical Data

We have a 10 random points between
$x \in [-2\pi, 2\pi]$
and its corresponding value f(x)

We know that data follows an underlying physical rule that

$\frac{df(x)}{dx} = -sin(x)$

In [ ]:
```python
import matplotlib.pyplot as plt

give_size = 10
in_given = np.linspace(-2 * np.pi, 2 * np.pi, give_size)
out_given = np.cos(in_given) + 0.1*np.random.normal(loc=0.0, scale=1, size=give_size)

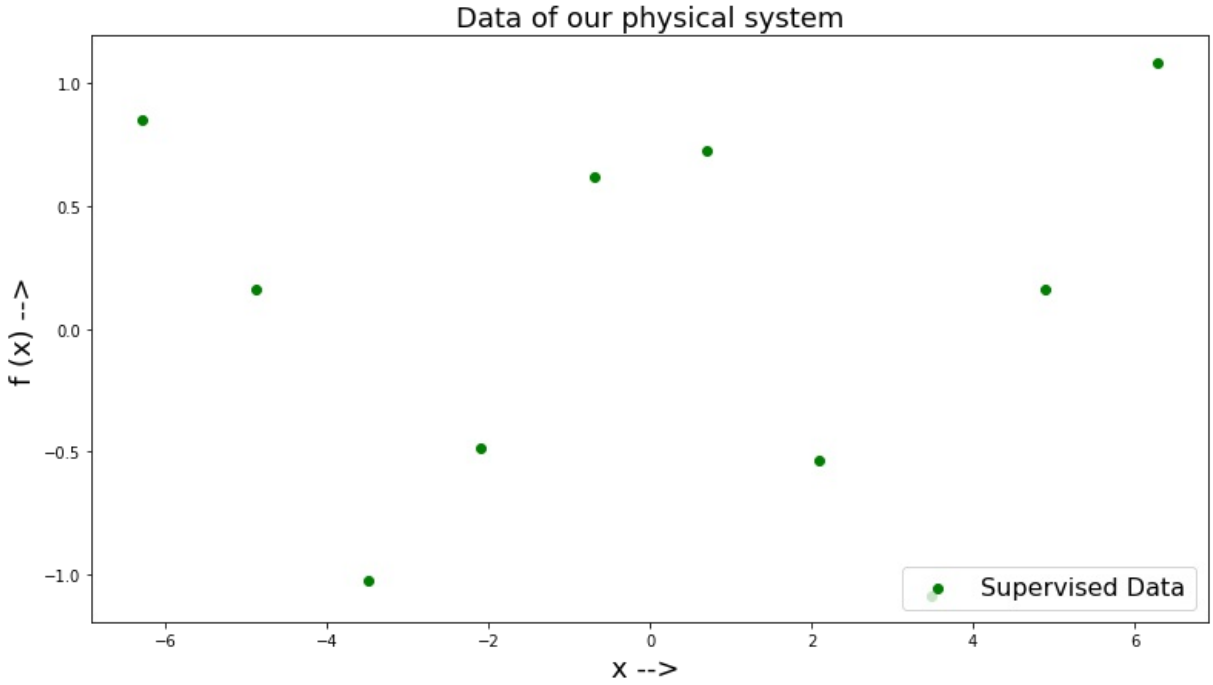


# red for numpy.sin()
plt.figure(figsize=(13, 7))

plt.scatter(in_given, out_given, color = 'green', marker = "o")
plt.xlabel("x --> ", fontsize=18)
plt.ylabel("f (x) -->", fontsize=18)
plt.legend(["Supervised Data"], prop={'size': 16}, loc ="lower right")

plt.title("Data of our physical system", fontsize=18)
```

Out[ ]:
```
Text(0.5, 1.0, 'Data of our physical system')
```



From simple integeration, we can easily solve the diffrential equation and the solution will be -

$f(x) = cos(x)$

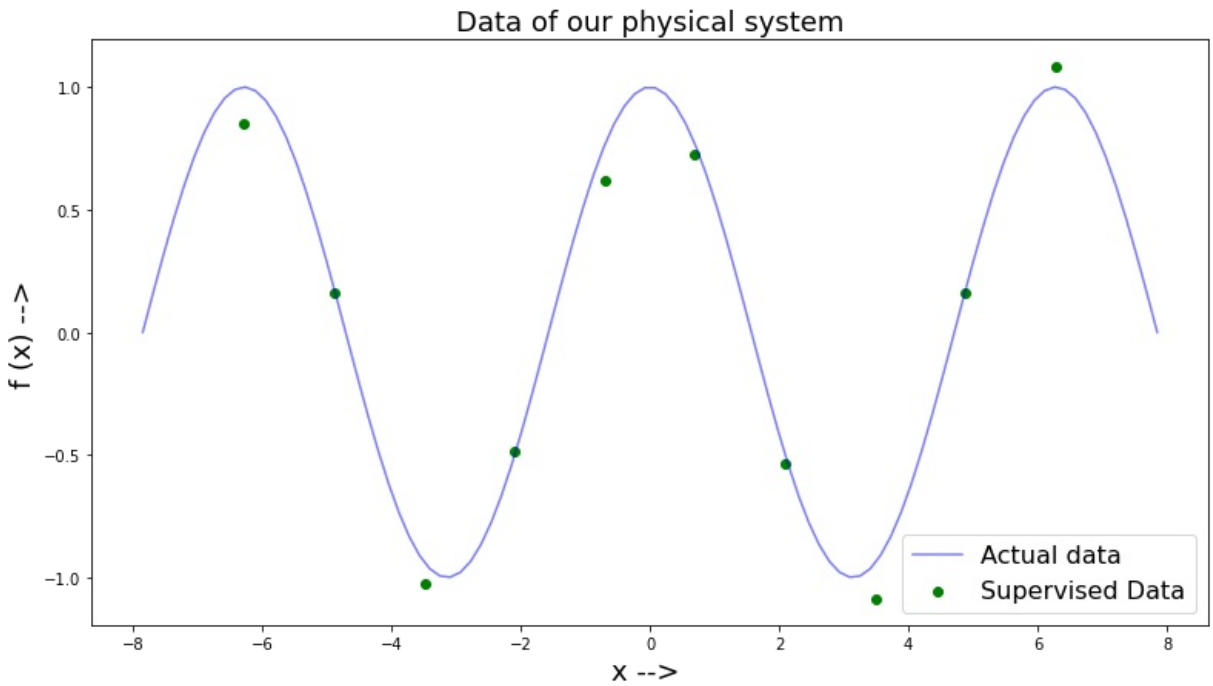
In [ ]:
```
import matplotlib.pyplot as plt

test = np.expand_dims(np.linspace(-2.5 * np.pi, 2.5 * np.pi, 100), 1)
out_array = np.cos(test)

plt.figure(figsize=(13, 7))
plt.plot(test, out_array, color = 'blue', alpha = 0.5)
plt.scatter(in_given, out_given, color = 'green', marker = "o")
plt.xlabel("x --> ", fontsize=18)
plt.ylabel("f (x) -->", fontsize=18)
plt.legend(["Actual data" ,"Supervised Data"], prop={'size': 16}, loc ="lower right")

plt.title("Data of our physical system", fontsize=18)
```

Out[ ]:
```
Text(0.5, 1.0, 'Data of our physical system')
```



Building a Simple Neural Network Model -

We define a simple Feed-forward Neural Network with 2 hidden layers of size 256 & 128 neurons.

In [ ]:
```
# defining the Haiku model

# A neural network is defined as a function of its weights & operations.
# NN(x) = F(x, W)

# forward function defines the F which describes the mathematical operations like Matrix & dot products, Signmo
# W is the init_params

def f(x):
  net = hk.nets.MLP(output_sizes=[256, 128, 1], activation=jax.nn.softplus)
  val = net(x)
  return val

init_params, forward_fn = hk.transform(f)
rng = jax.random.PRNGKey(500)
params = init_params(rng, np.random.rand(1000, 1))
```

```
/usr/local/lib/python3.7/dist-packages/jax/_src/numpy/lax_numpy.py:3634: UserWarning: Explicitly requested dtype
float64 requested in zeros is not available, and will be truncated to dtype float32. To enable more dtypes, set
the jax_enable_x64 configuration option or the JAX_ENABLE_X64 shell environment variable. See https://github.com
/google/jax#current-gotchas for more.
  lax._check_user_dtype_supported(dtype, "zeros")
```

Fitting a simple Neural Network solution to the Physical Data

In [ ]:
```
train_dataset = NumpyDataset(np.expand_dims(in_given, axis=1), np.expand_dims(out_given, axis=1))
rms_loss = lambda pred, tar, w: jnp.mean(optax.l2_loss(pred, tar))
# JaxModel Working
nn_model = JaxModel(
    forward_fn,
    params,
    rms_loss,
    batch_size=100,
    learning_rate=0.001,
    log_frequency=2)
nn_model.fit(train_dataset, nb_epochs=10000, deterministic=True)
```

```
/usr/local/lib/python3.7/dist-packages/deepchem/models/jax_models/jax_model.py:160: UserWarning: JaxModel is sti
ll in active development and all features may not yet be implemented
  'JaxModel is still in active development and all features may not yet be implemented'
```

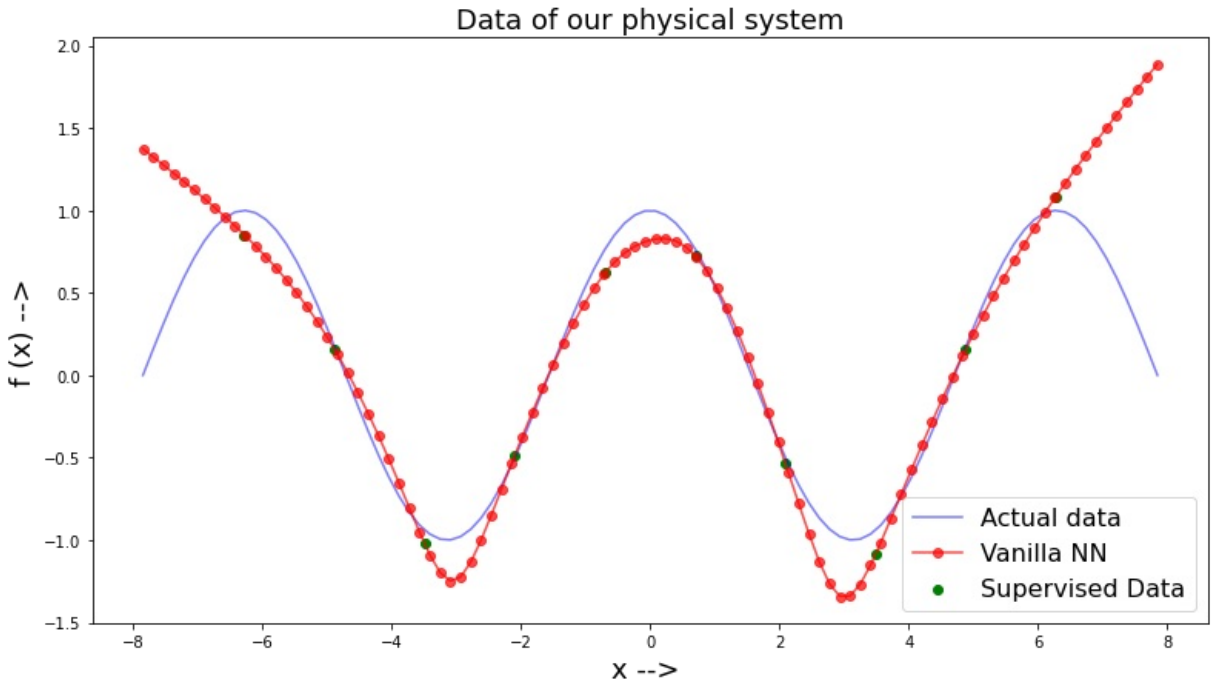
Out[ ]:
```
2.1729056921826473e-11
```

In [ ]:
```
dataset_test = NumpyDataset(test)
nn_output = nn_model.predict(dataset_test)

plt.figure(figsize=(13, 7))
plt.plot(test, out_array, color = 'blue', alpha = 0.5)
plt.scatter(in_given, out_given, color = 'green', marker = "o")
plt.plot(test, nn_output, color = 'red', marker = "o", alpha = 0.7)
plt.xlabel("x --> ", fontsize=18)
plt.ylabel("f (x) -->", fontsize=18)
plt.legend(["Actual data", "Vanilla NN", "Supervised Data"], prop={'size': 16}, loc ="lower right")

plt.title("Data of our physical system", fontsize=18)
```

Out[ ]:
```
Text(0.5, 1.0, 'Data of our physical system')
```



Learning to fit the Data using the underlying Diffrential equation

Lets ensure that final output of the neural network satisfies the diffrential equation as a loss function-

In [ ]:

```
def create_eval_fn(forward_fn, params):
  """
    Calls the function to evaluate the model
  """

  @jax.jit
  def eval_model(x, rng=None):

    bu = forward_fn(params, rng, x)
    return jnp.squeeze(bu)

  return eval_model

def gradient_fn(forward_fn, loss_outputs, initial_data):
  """
  This function calls the gradient function, to implement the backpropagation
  """
  boundary_data = initial_data['X0']
  boundary_target = initial_data['u0']

  @jax.jit
  def model_loss(params, target, weights, rng, x_train):

    @functools.partial(jax.vmap, in_axes=(None, 0))
    def periodic_loss(params, x):
      """
      diffrential equation => grad(f(x)) = - sin(x)
      minimize f(x) := grad(f(x)) + sin(x)
      """
      x = jnp.expand_dims(x, 0)
```

```
      u_x = jacrev(forward_fn, argnums=(2))(params, rng, x)
      return u_x + jnp.sin(x)

    u_pred = forward_fn(params, rng, boundary_data)
    loss_u = jnp.mean((u_pred - boundary_target)**2)

    f_pred = periodic_loss(params, x_train)
    loss_f = jnp.mean((f_pred**2))

    return loss_u + loss_f

  return model_loss
```

In [ ]:
```
initial_data = {
    'X0': jnp.expand_dims(in_given, 1),
    'u0': jnp.expand_dims(out_given, 1)
}
opt = Adam(learning_rate=1e-3)
pinn_model= PINNModel(
    forward_fn=forward_fn,
    params=params,
    initial_data=initial_data,
    batch_size=1000,
    optimizer=opt,
    grad_fn=gradient_fn,
    eval_fn=create_eval_fn,
    deterministic=True,
    log_frequency=1000)

# defining our training data. We feed 100 points between [-2.5pi, 2.5pi] without the labels,
# which will be used as the differential loss(regulariser)
X_f = np.expand_dims(np.linspace(-3 * np.pi, 3 * np.pi, 1000), 1)
dataset = NumpyDataset(X_f)
pinn_model.fit(dataset, nb_epochs=3000)
```

```
/usr/local/lib/python3.7/dist-packages/deepchem/models/jax_models/pinns_model.py:157: UserWarning: PinnModel is still in active development and we could change the design of the API in the future.
  'PinnModel is still in active development and we could change the design of the API in the future.'
/usr/local/lib/python3.7/dist-packages/deepchem/models/jax_models/jax_model.py:160: UserWarning: JaxModel is still in active development and all features may not yet be implemented
  'JaxModel is still in active development and all features may not yet be implemented'
```

Out[ ]:
```
0.026332732232287527
```

In [ ]:
```
import matplotlib.pyplot as plt

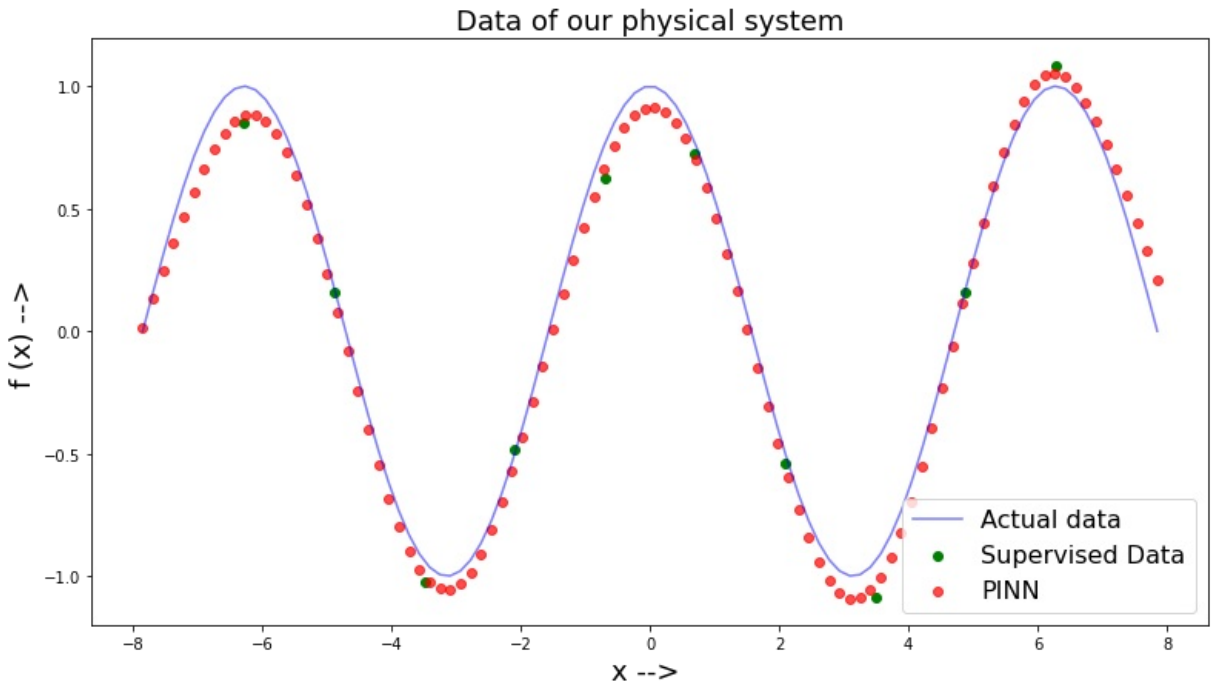
pinn_output = pinn_model.predict(dataset_test)

plt.figure(figsize=(13, 7))
plt.plot(test, out_array, color = 'blue', alpha = 0.5)
plt.scatter(in_given, out_given, color = 'green', marker = "o")
# plt.plot(test, nn_output, color = 'red', marker = "x", alpha = 0.3)
plt.scatter(test, pinn_output, color = 'red', marker = "o", alpha = 0.7)

plt.xlabel("x --> ", fontsize=18)
plt.ylabel("f (x) -->", fontsize=18)
plt.legend(["Actual data" ,"Supervised Data", "PINN"], prop={'size': 16}, loc ="lower right")

plt.title("Data of our physical system", fontsize=18)
```

Out[ ]:
```
Text(0.5, 1.0, 'Data of our physical system')
```



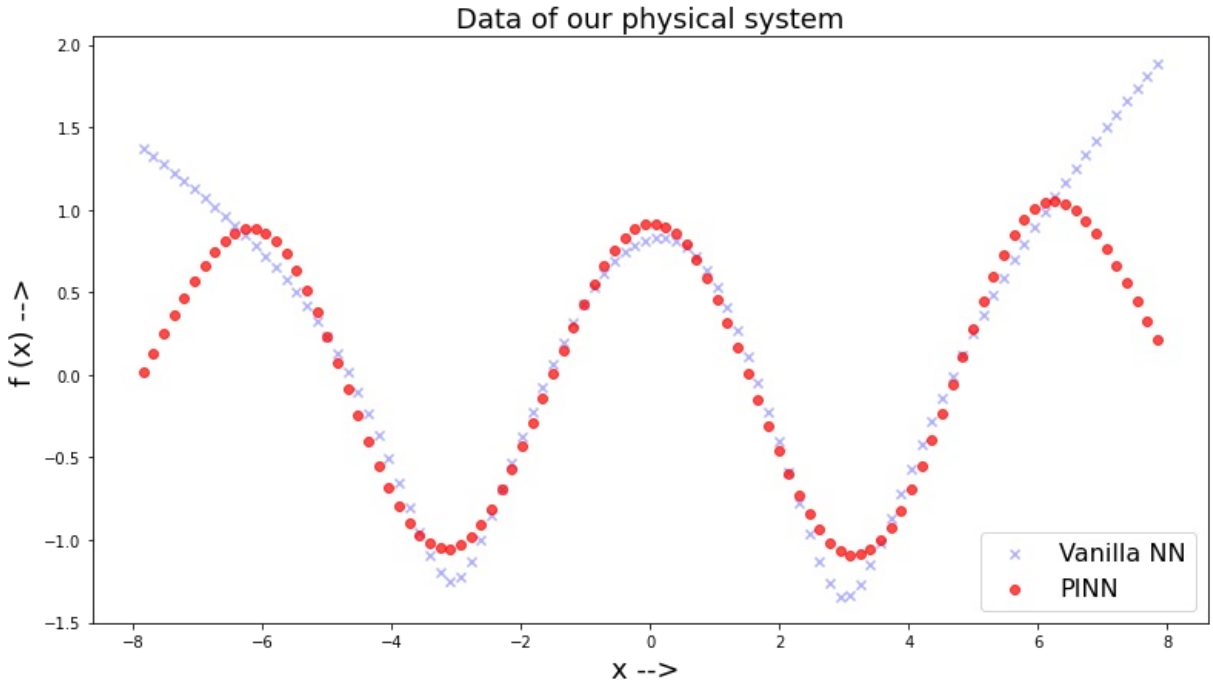
Comparing the results between PINN & Vanilla NN model

In [ ]:
```
plt.figure(figsize=(13, 7))
# plt.plot(test, out_array, color = 'blue', alpha = 0.5)
# plt.scatter(in_given, out_given, color = 'green', marker = "o")
plt.scatter(test, nn_output, color = 'blue', marker = "x", alpha = 0.3)
plt.scatter(test, pinn_output, color = 'red', marker = "o", alpha = 0.7)

plt.xlabel("x --> ", fontsize=18)
plt.ylabel("f (x) -->", fontsize=18)
plt.legend(["Vanilla NN", "PINN"], prop={'size': 16}, loc ="lower right")

plt.title("Data of our physical system", fontsize=18)
```

Out[ ]:
```
Text(0.5, 1.0, 'Data of our physical system')
```


Data of our physical system
f (x) -->
x -->
Vanilla NN
PINN
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−8
−6
−4
−2
0
2
4
6
8

About Neural ODE : Using `Torchdiffeq` with `Deepchem`

Author : Anshuman Mishra : Linkedin

Open in Colab

Before getting our hands dirty with code , let us first understand little bit about what Neural ODEs are ?

NeuralODEs and torchdiffeq

NeuralODE stands for "Neural Ordinary Differential Equation. You heard right. Let me guess . Your first impression of the word is : "Has it something to do with differential equations that we studied in the school ?"

Spot on ! Let's see the formal definition as stated by the original paper :

```
Neural ODEs are a new family of deep neural network models. Instead of specifying a discrete
sequence of
hidden layers, we parameterize the derivative of the hidden state using a neural network.

The output of the network is computed using a blackbox differential equation solver.These are
continuous-depth models that have constant memory
cost, adapt their evaluation strategy to each input, and can explicitly trade numerical
precision for speed.
```

In simple words perceive NeuralODEs as yet another type of layer like Linear, Conv2D, MHA...

In this tutorial we will be using torchdiffeq. This library provides ordinary differential equation (ODE) solvers implemented in PyTorch framework. The library provides a clean API of ODE solvers for usage in deep learning applications. As the solvers are implemented in PyTorch, algorithms in this repository are fully supported to run on the GPU.

What will you learn after completing this tutorial ?

1. How to implement a Neural ODE in a Neural Network ?
2. Using torchdiffeq with deepchem.

Installing Libraries

In [1]:
```
!pip install torchdiffeq
!pip install --pre deepchem
```

```
Collecting torchdiffeq
  Downloading torchdiffeq-0.2.2-py3-none-any.whl (31 kB)
Requirement already satisfied: torch>=1.3.0 in /usr/local/lib/python3.7/dist-packages (from torchdiffeq) (1.10.0
+cu111)
Requirement already satisfied: scipy>=1.4.0 in /usr/local/lib/python3.7/dist-packages (from torchdiffeq) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy>=1.4.0->torch
diffeq) (1.21.5)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch>=1.3.0->t
orchdiffeq) (3.10.0.2)
Installing collected packages: torchdiffeq
Successfully installed torchdiffeq-0.2.2
Collecting deepchem
  Downloading deepchem-2.6.1-py3-none-any.whl (608 kB)
     |████████████████████████████████| 608 kB 8.9 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.4.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.3.5)
Collecting rdkit-pypi
  Downloading rdkit_pypi-2021.9.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (20.6 MB)
     |████████████████████████████████| 20.6 MB 8.2 MB/s
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.1.0)
Requirement already satisfied: numpy>=1.21 in /usr/local/lib/python3.7/dist-packages (from deepchem) (1.21.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages (from pandas->de
epchem) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->deepchem) (2
018.9)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.7.3->
pandas->deepchem) (1.15.0)
Requirement already satisfied: Pillow in /usr/local/lib/python3.7/dist-packages (from rdkit-pypi->deepchem) (7.1
.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn
->deepchem) (3.1.0)
Installing collected packages: rdkit-pypi, deepchem
Successfully installed deepchem-2.6.1 rdkit-pypi-2021.9.4
```

Import Libraries

In [2]:
```python
import torch
import torch.nn as nn

from torchdiffeq import odeint
import math
import numpy as np

import deepchem as dc
import matplotlib.pyplot as plt
```

Before diving into the core of this tutorial , let's first acquaint ourselves with usage of torchdiffeq. Let's solve following differential equation .

$$\frac{dz(t)}{dt} = f(t) = t$$

when
$z(0) = 0$

The process to do it by hand is :

$$\int dz = \int t dt + C \; z(t) = \frac{t^2}{2} + C$$

Let's solve it using ODE Solver called `odeint` from torchdiffeq

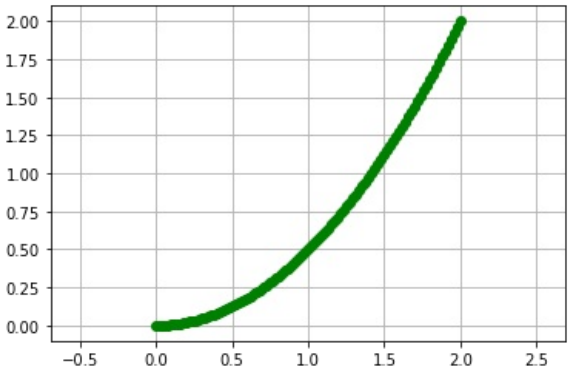
In [3]:
```python
def f(t,z):
  return t

z0 = torch.Tensor([0])
t = torch.linspace(0,2,100)
out = odeint(f, z0, t);
```

Let's plot our result .It should be a parabola (remember general equation of parabola as
$x^2 = 4ay$
)

In [4]:
```python
plt.plot(t, out, 'go--')
plt.axes().set_aspect('equal','datalim')
plt.grid()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: MatplotlibDeprecationWarning: Adding an axes usi
ng the same arguments as a previous axes currently reuses the earlier instance.  In a future version, a new inst
ance will always be created and returned.  Meanwhile, this warning can be suppressed, and the future behavior en
sured, by passing a unique label to each axes instance.
```



What is Neural Differential Equation ?

A neural differential equation is a differential equation using a neural network to parameterize the vector field. The canonical example is a neural ordinary differential equation :

$$y(0) = y_0$$

$$\frac{dy}{dt}(t) = f_\theta(t, y(t))$$

Here θ represents some vector of learnt parameters,
$f_\theta : \mathbb{R}$
is any standard neural architecture and
$y : [0, T] \to \mathbb{R}^{d_1 \times \ldots d_k}$
is the solution. For many applications
f_θ
will just be a simple feedforward network. Here
d_i
is the dimension.

Reference

The central idea now is to use a differential equation solver as part of a learnt differentiable computation graph (the sort of computation graph ubiquitous to deep learning)

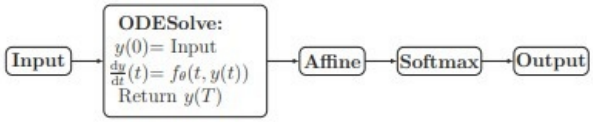


Figure 1.1: Computation graph for a simple neural ODE.

As simple example, suppose we observe some picture
$y_0 \in \mathbb{R}^{3 \times 32 \times 3}$
(RGB and 32x32 pixels), and wish to classify it as a picture of a cat or as a picture of a dog.

With torchdiffeq , we can solve even complex higher order differential equations too. Following is a real world example , a set of differential equations that models a spring - mass damper system

$\dot{x}$ =

$\ddot{x}$ =

$\dddot{x}$ =

with initial state t=0 , x=1

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -\frac{k}{m} & p & 0 \\ 0 & g & -r \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix}$$

The right hand side may be regarded as a particular differentiable computation graph. The parameters may be fitted by setting up a loss between the trajectories of the model and the observed trajectories in the data, backpropagating through the model, and applying stochastic gradient descent.

In [5]:
```
class SystemOfEquations:

  def __init__(self, km, p, g, r):
    self.mat = torch.Tensor([[0,1,0],[-km, p, 0],[0,g,-r]])

  def solve(self, t, x0, dx0, ddx0):
    y0 = torch.cat([x0, dx0, ddx0])
    out = odeint(self.func, y0, t)
    return out

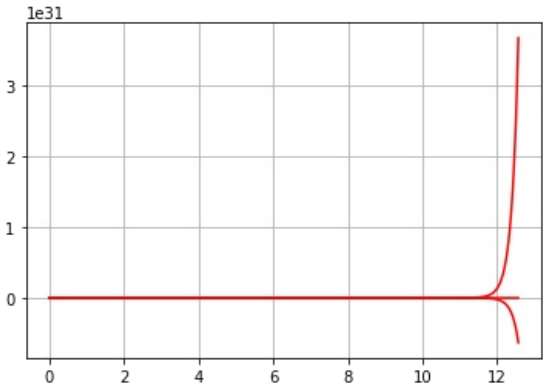
  def func(self, t, y):
    out = y@self.mat
    return out


x0 = torch.Tensor([1])
dx0 = torch.Tensor([0])
ddx0 = torch.Tensor([1])

t = torch.linspace(0, 4*np.pi, 1000)
solver = SystemOfEquations(1,6,3,2)
out = solver.solve(t, x0, dx0, ddx0)

plt.plot(t, out, 'r')
plt.axes()
plt.grid()
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:25: MatplotlibDeprecationWarning: Adding an axes us
ing the same arguments as a previous axes currently reuses the earlier instance.  In a future version, a new ins
tance will always be created and returned.  Meanwhile, this warning can be suppressed, and the future behavior e
nsured, by passing a unique label to each axes instance.
```



This is precisely the same procedure as the more general neural ODEs we introduced earlier. At first glance, the NDE approach of ‘putting a neural network in a differential equation’ may seem unusual, but it is actually in line with standard practice. All that has happened is to change the parameterisation of the vector field.

Model

Let us have a look at how to embed an ODEsolver in a neural network .

In [6]:
```
from torchdiffeq import odeint_adjoint as odeadj

class f(nn.Module):
  def __init__(self, dim):
    super(f, self).__init__()
    self.model = nn.Sequential(
        nn.Linear(dim,124),
        nn.ReLU(),
        nn.Linear(124,124),
        nn.ReLU(),
        nn.Linear(124,dim),
        nn.Tanh()
    )

  def forward(self, t, x):
    return self.model(x)
```

`function` f in above code cell , is wrapped in an `nn.Module` (see codecell below) thus forming the dynamics of $\frac{dy}{dt}(t) = f_\theta(t, y(t))$
embedded within a neural Network. ODE Block treats the received input x as the initial value of the differential equation. The integration interval of ODE Block is fixed at [0, 1]. And it returns the output of the layer at
$t = 1$
.

In [7]:
```
class ODEBlock(nn.Module):

  # This is ODEBlock. Think of it as a wrapper over ODE Solver , so as to easily connect it with our neurons !

  def __init__(self, f):
    super(ODEBlock, self).__init__()
    self.f = f
    self.integration_time = torch.Tensor([0,1]).float()

  def forward(self, x):
    self.integration_time = self.integration_time.type_as(x)
    out = odeadj(
        self.f,
        x,
        self.integration_time
    )

    return out[1]


class ODENet(nn.Module):

  #This is our main neural network that uses ODEBlock within a sequential module

  def __init__(self, in_dim, mid_dim, out_dim):
    super(ODENet, self).__init__()
    fx = f(dim=mid_dim)
    self.fc1 = nn.Linear(in_dim, mid_dim)
    self.relu1 = nn.ReLU(inplace=True)
    self.norm1 = nn.BatchNorm1d(mid_dim)
    self.ode_block = ODEBlock(fx)
    self.dropout = nn.Dropout(0.4)
    self.norm2 = nn.BatchNorm1d(mid_dim)
    self.fc2 = nn.Linear(mid_dim, out_dim)

  def forward(self, x):
    batch_size = x.shape[0]
    x = x.view(batch_size, -1)

    out = self.fc1(x)
    out = self.relu1(out)
    out = self.norm1(out)
    out = self.ode_block(out)
    out = self.norm2(out)
    out = self.dropout(out)
    out = self.fc2(out)

    return out
```

As mentioned before , Neural ODE Networks acts similar (has advantages though) to other neural networks , so we can solve any problem with them as the existing models do. We are gonna reuse the training process mentioned in this deepchem tutorial.

So Rather than demonstrating how to use NeuralODE model with a normal dataset, we shall use the **Delaney solubility dataset** provided under **deepchem** . Our model will learn to predict the solubilities of molecules based on their extended-connectivity fingerprints (ECFPs) . For performance metrics we use pearson_r2_score . Here loss is computed directly from the model's output

In [8]:
```
tasks, dataset, transformers = dc.molnet.load_delaney(featurizer='ECFP', splitter='random')
train_set, valid_set, test_set = dataset
metric = dc.metrics.Metric(dc.metrics.pearson_r2_score)
```

Time to Train

We train our model for 50 epochs, with L2 as Loss Function.

In [9]:
```
# Like mentioned before one can use GPUs with PyTorch and torchdiffeq
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

```
model = ODENet(in_dim=1024, mid_dim=1000, out_dim=1).to(device)
model = dc.models.TorchModel(model, dc.models.losses.L2Loss())

model.fit(train_set, nb_epoch=50)

print('Training set score : ', model.evaluate(train_set,[metric]))
print('Test set score : ', model.evaluate(test_set,[metric]))
```

```
Training set score :  {'pearson_r2_score': 0.9708644701066554}
Test set score :  {'pearson_r2_score': 0.7104556551957734}
```

Neural ODEs are invertible neural nets Reference Invertible neural networks have been a significant thread of research in the ICML community for several years. Such transformations can offer a range of unique benefits:

- They preserve information, allowing perfect reconstruction (up to numerical limits) and obviating the need to store hidden activations in memory for backpropagation.
- They are often designed to track the changes in probability density that applying the transformation induces (as in normalizing flows).
- Like autoregressive models, normalizing flows can be powerful generative models which allow exact likelihood computations; with the right architecture, they can also allow for much cheaper sampling than autoregressive models.

While many researchers are aware of these topics and intrigued by several high-profile papers, few are familiar enough with the technical details to easily follow new developments and contribute. Many may also be unaware of the wide range of applications of invertible neural networks, beyond generative modelling and variational inference.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Differentiation Infrastructure in Deepchem

Author : Rakshit Kr. Singh : Website : LinkedIn : GitHub

Scientific advancement in machine learning hinges on the effective resolution of complex optimization problems. From material property design to drug discovery, these problems often involve numerous variables and intricate relationships. Traditional optimization techniques often face hurdles when addressing such challenges, often resulting in slow convergence or solutions deemed unreliable. We introduce solutions that are differentiable and also seamlessly integrable into machine learning systems, offering a novel approach to resolving these complexities.

This tutorials introduces DeepChem's comprehensive set of differentiable optimisation tools to empower researchers across the physical sciences. DeepChem addresses limitations of conventional methods by offering a diverse set of optimization algorithms. These includes established techniques like Broyden's first and second methods alongside cutting-edge advancements, allowing researchers to select the most effective approach for their specific problem.

Overview of Differentiation Utilities in Deepchem

DeepChem provides a number of optimisation algorithms and Utilities for implementing more algorithms. Some of the optimisation algorithms provided by DeepChem are:

- Broyden's First Method
- Broyden's Second Method
- Anderson Acceleration
- Gradient Descent
- Adam

Along with these optimisation algorithms, DeepChem also provides a number of utilities for implementing more algorithms.

What are Non Linear Equations? and why do they matter?

Nonlinear equations are mathematical expressions where the relationship between the variables is not linear. Unlike linear equations, which have a constant rate of change, nonlinear equations involve terms with higher powers or functions like exponentials, logarithms, trigonometric functions, etc.

Nonlinear equations are essential across various disciplines, including physics, engineering, economics, biology, and finance. They describe complex relationships and phenomena that cannot be adequately modeled with linear equations. From gravitational interactions in celestial bodies to biochemical reactions in living organisms, non-linear equations play a vital role in understanding and predicting real-world systems. Whether it's optimizing structures, analyzing market dynamics, or designing machine learning algorithms.

Some Simple Non Linear Equations:

$f(x) = sin(x)$
, is a trigonometric function defined for all real numbers. It represents the ratio of the length of the side opposite an angle in a right triangle to the length of the hypotenuse.

$f(x) = cos(x)$
, is an another trigonometric function. It represents the ratio of the length of the adjacent side of a right triangle to the length of the hypotenuse when x is the measure of an acute angle.

$f(x) = x^2$
, is a parabola, symmetric around the y-axis, with its vertex at the origin. It represents a mathematical model of quadratic growth or decay. In physical systems, it often describes phenomena where the rate of change is proportional to the square of the quantity involved.

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)
y3 = x**2

fig, axs = plt.subplots(1, 3, figsize=(9, 3))

axs[0].plot(x, y1, color='blue')
```

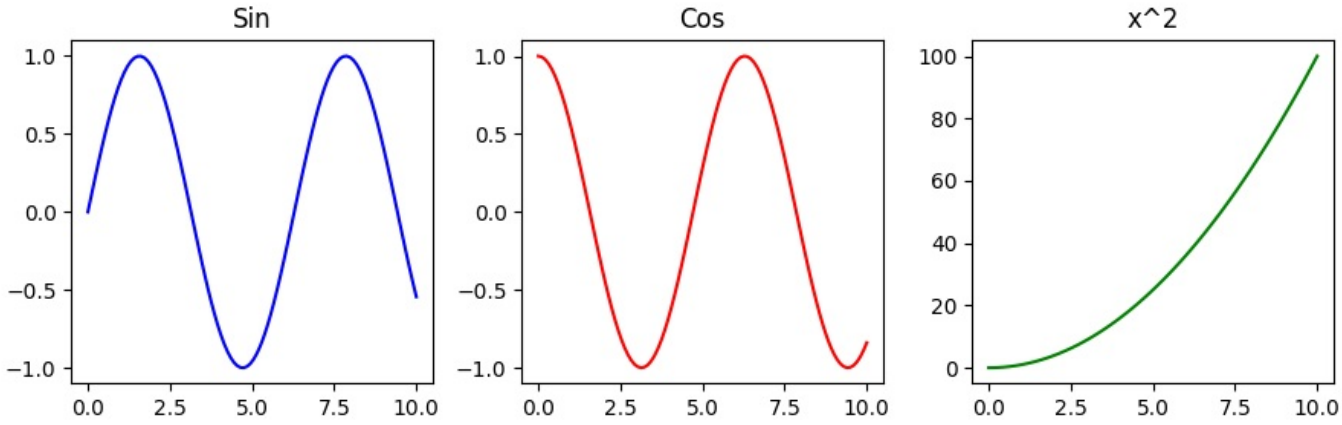
```
axs[0].set_title('Sin')

axs[1].plot(x, y2, color='red')
axs[1].set_title('Cos')

axs[2].plot(x, y3, color='green')
axs[2].set_title('x^2')

plt.tight_layout()

plt.show()
```



Root Finder Methods

`deepchem.utils.differentiation_utils.optimize.rootfinder` provides a collection of algorithms for solving nonlinear equations. These methods are designed to find the roots of functions efficiently, making them indispensable for a wide range of applications in mathematics, physics, engineering, and other fields.

At its core, rootfinding seeks to determine the solutions (roots) of equations, where a function equals zero. This operation plays a pivotal role in numerous real-world applications, making it indispensable in both theoretical and practical domains.

Broyden's First Method:

Broyden's First Method is an iterative numerical method used for solving systems of nonlinear equations. It's particularly useful when the Jacobian matrix (the matrix of partial derivatives of the equations) is difficult or expensive to compute.

Broyden's Method is an extension of the Secant Method for systems of nonlinear equations. It iteratively updates an approximation to the Jacobian matrix using the information from previous iterations. The algorithm converges to the solution by updating the variables in the direction that minimizes the norm of the system of equations.

Steps:

1. Initialize the approximation to the Jacobian matrix.

$$J_k$$

2. Initialize the variables.

$$X_k = f(X_k)$$

3. Compute the function values.

$$f_k = f(X_k)$$

4. Update the variables.

$$X_{k+1} = X_{k+1} - J_k^{-1} f_k$$

5. Compute the change in variables.

$$\Delta X_{k+1} = X_{k+1} - X_k$$

6. Compute the function values.

$$f_{k+1} = f(X_{k+1})$$

7. Update the approximation to the Jacobian matrix.

$$J_{k+1}^{-1} = J_k^{-1} + \frac{(\Delta X_{k+1} - J_k^{-1}\Delta f_{k+1})\Delta X_{k+1}^T J_k^{-1}}{\Delta X_{k+1}^T J_k^{-1} \Delta f_{k+1}}$$

8. Repeat steps 4-7 until convergence criteria are met.

References:

[1] "A class of methods for solving nonlinear simultaneous equations" by Charles G. Broyden

In [3]:
```
import torch
from deepchem.utils.differentiation_utils import rootfinder
def func1(y, A):
    return torch.tanh(A @ y + 0.1) + y / 2.0
A = torch.tensor([[1.1, 0.4], [0.3, 0.8]]).requires_grad_()
y0 = torch.zeros((2,1))

# Broyden's First Method
yroot = rootfinder(func1, y0, params=(A,), method='broyden1')
print("Root By Broyden's First Method:")
print(yroot)
print("Function Value at Calculated Root:")
print(func1(yroot, A))
```

```
Root By Broyden's First Method:
tensor([[-0.0459],
        [-0.0663]], grad_fn=<_RootFinderBackward>)
Function Value at Calculated Root:
tensor([[1.1735e-07],
        [1.7881e-07]], grad_fn=<AddBackward0>)
```

In [28]:
```
from deepchem.utils.differentiation_utils.optimize.rootsolver import broyden1
def fcn(x):
   return x**2 - 4 + torch.tan(x)
x0 = torch.tensor(0.0, requires_grad=True)
x = broyden1(fcn, x0)
x, fcn(x)
```

Out[28]:
```
(tensor(2.2752, grad_fn=<ViewBackward0>),
 tensor(1.7881e-06, grad_fn=<AddBackward0>))
```

Broyden's Second Method:

Broyden's Second Method differs from the first method in how it updates the approximation to the Jacobian matrix. Instead of using the change in variables and function values, it uses the change in the residuals (the difference between the function values and the target values) to update the Jacobian matrix. This approach can be more stable and robust in certain situations.

Steps:

1...6 are same as Broyden's First Method.

7. Update the approximation to the Jacobian matrix.

$$J_{k+1}^{-1} = J_k^{-1} + \frac{(\Delta X_{k+1} - J_k^{-1}\Delta f_{k+1})\Delta f_{k+1}^T}{\Delta f_{k+1}^T \Delta f_{k+1}}$$

8. Repeat steps 4-7 until convergence criteria are met.

In [24]:
```
# Broyden's Second Method
import torch
from deepchem.utils.differentiation_utils import rootfinder
def func1(y, A):
    return torch.tanh(A @ y + 0.1) + y / 2.0
A = torch.tensor([[1.1, 0.4], [0.3, 0.8]]).requires_grad_()
y0 = torch.zeros((2,1))

yroot = rootfinder(func1, y0, params=(A,), method='broyden2')
print("\nRoot by Broyden's Second Method:")
print(yroot)
print("Function Value at Calculated Root:")
print(func1(yroot, A))
```

```
Root by Broyden's Second Method:
tensor([[-0.0459],
        [-0.0663]], grad_fn=<_RootFinderBackward>)
Function Value at Calculated Root:
tensor([[ 1.0300e-06],
        [-3.2783e-07]], grad_fn=<AddBackward0>)
```

Equilibrium Methods (Fixed Point Iteration)

`deepchem.utils.differentiation_utils.optimize.equilibrium` contains algorithms for solving equilibrium problems, where the goal is to find a fixed point of a function. While all the rootfinding methods can be used to solve equilibrium problems, these specialized algorithms are designed to exploit the structure of equilibrium problems for more efficient convergence.

Equilibrium methods are essential in machine learning for optimizing models, ensuring stability and convergence, regularizing parameters, and analyzing strategic interactions in multi-agent systems. By leveraging equilibrium principles and techniques, machine learning practitioners can train more robust and generalizable models capable of addressing a wide range of real-world challenges.

The Fixed-Point Problem:

Given the function
$f : \mathbb{R}^n \to \mathbb{R}^n$
, compute a fixed-point
$x^* \in \mathbb{R}^n$
such that

$$x^* = f(x^*)$$

Classical Approach:

Steps:

1. Initialize the variables.
   x_k
2. Compute the function values.
   $f_k = f(x_k)$
3. Update the variables.
   $x_{k+1} = f_k$
4. Repeat steps 2-3 until convergence criteria are met.

Anderson Acceleration Approach (Anderson Mixing):

Anderson Acceleration is an iterative method for accelerating the convergence of fixed-point iterations. It combines information from previous iterations to construct a better approximation to the fixed-point. The algorithm uses a history of function values and updates to compute a new iterate that minimizes the residual norm.

Steps:

1. **Initialize point:** x_0, fixed-point mapping $f : \mathbb{R}^n \to \mathbb{R}^n$

   for $k = 0, 1, \ldots$ **do**

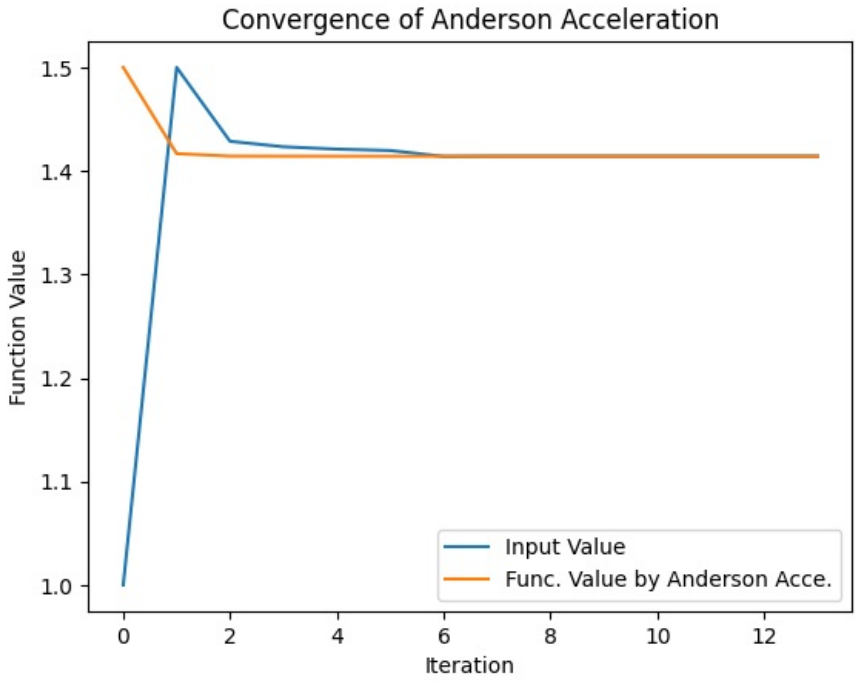
2. - Choose m_k (e.g., $m_k = min(m, k)$ for some integer $m \geq 0$).
   - Select weights α_k^j based on the last m_k iterations satisfying $\sum_{j=0}^{m_k} \alpha_j^k = 1$.
   - $x_{k+1} = \sum_{j=0}^{m_k} \alpha_j^k f(x_{k-m_k+j})$.
3. **end**

for

In [9]:
```
import torch
import matplotlib.pyplot as plt
from deepchem.utils.differentiation_utils.optimize.equilibrium import anderson_acc
x_value, f_value = [], []
def fcn(x, a):
    x_value.append(x.item())
    f_value.append((a/x + x).item()/2)
    return (a/x + x)/2
a = 2.0
x0 = torch.tensor([1.0], requires_grad=True)
x = anderson_acc(fcn, x0, params=[a], maxiter=16)
print("Root by Anderson Acceleration:", x.item())
print("Function Value at Calculated Root:", fcn(x, a).item())

# Plotting the convergence of Anderson Acceleration
plt.plot(x_value, label='Input Value')
plt.plot(f_value, label='Func. Value by Anderson Acce.')
plt.legend(loc='lower right')
plt.xlabel('Iteration')
plt.ylabel('Function Value')
plt.title('Convergence of Anderson Acceleration')
plt.show()
```

```
Root by Anderson Acceleration: 1.4142135381698608
Function Value at Calculated Root: 1.4142135381698608
```



Minimizer

`deepchem.utils.differentiation_utils.optimize.minimizer` provides a collection of algorithms for minimizing functions. These methods are designed to find the minimum of a function efficiently, making them indispensable for a wide range of applications in mathematics, physics, engineering, and other fields.

Minimization algorithms, including variants of gradient descent like ADAM, are fundamental tools in various fields of science, engineering, and optimization.

Gradient Descent

Gradient descent is a method for unconstrained mathematical optimization. It is a first-order iterative algorithm for finding a local minimum of a differentiable multivariate function.

It is used to minimize the cost function in various machine learning and optimization problems. It iteratively updates the parameters in the direction of the negative gradient of the cost function.

Steps:

1. **Initialize Parameters:**
   Denote the parameter vector to be optimized -
   θ

and
θ_0
represent the initial guess.

2. **Compute Gradient:**
Calculate the gradient of the cost function
$J(\theta)$
with respect to each parameter.

$$\nabla J(\theta) = \left[\frac{\partial J(\theta)}{\partial \theta_1}, \frac{\partial J(\theta)}{\partial \theta_2}, \ldots, \frac{\partial J(\theta)}{\partial \theta_n}\right]^T$$

- **Update Parameters:**
Adjust the parameters in the opposite direction of the gradient to minimize the cost function according to the learning rate
α
:

$$\theta = \theta - \alpha \nabla J(\theta)$$

- **Repeat:**
Steps 2 and 3 until the algorithm converges or Stops.

In [15]:
```
import torch
from deepchem.utils.differentiation_utils.optimize.minimizer import gd
def fcn(x):
    return 2 * x + (x - 2) ** 2, 2 * (x - 2) + 2
x0 = torch.tensor(0.0, requires_grad=True)
x = gd(fcn, x0, [])
print("Minimum by Gradient Descent:", x.item())
print("Function Value at Calculated Minimum:", fcn(x)[0].item())
```

```
Minimum by Gradient Descent: 0.9973406791687012
Function Value at Calculated Minimum: (tensor(3.0000), tensor(-0.0053))
```

ADAM (Adaptive Moment Estimation)

ADAM is an optimization algorithm used for training deep learning models. It's an extension of the gradient descent optimization algorithm and combines the ideas of both momentum and RMSProp algorithms.

Steps:

1. **Initialization:**
ADAM initializes two moving average variables:
m
(the first moment, similar to momentum) and
v
(the second moment, similar to RMSProp).

2. **Compute Gradients:**
At each iteration of training, the gradients of the parameters concerning the loss function are computed.

3. **Update Moving Averages:**
The moving averages
m
and
v
are updated using exponential decay, with momentum and RMSProp components respectively:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla J(\theta_t)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla J(\theta_t))^2$$

4. **Bias Correction:**
Due to the initialization of the moving averages to zero vectors, there's a bias towards zero, especially during the initial iterations. To correct this bias, ADAM applies a bias correction step:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

5. **Update Parameters:**
   Finally, the parameters (weights and biases) of the model are updated using the moving averages and the learning rate α:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

In [2]:
```
import torch
from deepchem.utils.differentiation_utils.optimize.minimizer import adam
def fcn(x):
    return 2 * x + (x - 2) ** 2, 2 * (x - 2) + 2
x0 = torch.tensor(10.0, requires_grad=True)
x = adam(fcn, x0, [], maxiter=20000)
print("X at Minimum by Adam:", x.item())
print("Function Value at Calculated Minimum:", fcn(x)[0].item())
```

```
X at Minimum by Adam: 1.0067708492279053
Function Value at Calculated Minimum: 3.0000457763671875
```

Conclusion

Differentiable optimization techniques are essential for many advanced computational experiments involving Environment Simulations like DFT, Physics Informed Neural Networks and as fundamental mathematical foundation for Molecular Simulation like Monte Carlo and Molecular Dynamics.

By integrating deep learning into simulations, we optimize efficiency and accuracy by leveraging trainable neural networks to replace costly or less precise components. This advancement holds immense potential for expediting scientific advancements and addressing longstanding mysteries with greater efficacy.

References

[1] Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational physics (2019)

[2] Muhammad F. Kasim, Sam M. Vinko. Learning the exchange-correlation functional from nature with fully differentiable density functional theory. 2021 American Physical Society

[3] Nathan Argaman, Guy Makov. Density Functional Theory -- an introduction. American Journal of Physics 68 (2000), 69-79

[4] John Ingraham et al. Learning Protein Structure with a Differentiable Simulator. ICLR. 2019.

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

```
@manual{Quantum Chemistry,
 title={Differentiation Infrastructure in Deepchem},
 organization={DeepChem},
 author={Singh, Rakshit kr.},
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/Differentiation_Infrastructure

 year={2024},
}
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Ordinary Differential Equation Solving

Author : Rakshit Kr. Singh : Website : LinkedIn : GitHub

Ordinary Differential Equations (ODEs) are a cornerstone of mathematical modeling, essential for understanding dynamic systems in scientific and engineering fields.

This tutorial aims to introduce you to ODE solving tools in DeepChem.

What are Ordinary Differential Equations?

An ordinary differential equation (ODE) is a type of differential equation that depends on a single independent variable. In contrast to partial differential equations (PDEs), which involve multiple independent variables, ODEs focus on relationships where changes occur with respect to just one variable. The term "ordinary" distinguishes these equations from stochastic differential equations (SDEs), which incorporate random processes.

ODEs consist of unknown functions and their derivatives, establishing relationships that describe how a quantity changes over time or space. These equations are fundamental in expressing the dynamics of systems.

General Form of an ODE -

$$\frac{dy}{dx} = f(x, y)$$

Here,
$dy/$
is the derivative of
y
with regards to
x
, and
$f(x, y)$
is a function of
x
and
y
.

Why we should Care About Ordinary Differential Equations

They are essential because they model how physical quantities change over time. ODEs are used in:

- Physics: To describe the motion of particles, the evolution of wave functions, and more.
- Engineering: To design control systems, signal processing, and electrical circuits.
- Biology: To model population dynamics, the spread of diseases, and biological processes.
- Economics: To analyze growth models, market equilibrium, and financial systems.
- Control Systems and Robotics: In control systems and robotics, ODEs are fundamental in describing the dynamics of systems.

Solving Ordinary Differential Equations

Steps:

- Formulate the ODE: An ODE involves an unknown function
  $y(x)$
  and its derivatives.

$$Example : \frac{dy}{dx} + y = e^x$$

- Find the general solution: The goal is to find the function
  $y(x)$
  that satisfies the ODE. This process often involves integration.

- Apply initial or boundary condition: To find a specific solution, you often need additional information, such as the value of the function and its derivatives at certain points. These are called initial or boundary conditions.

Methods for Solving ODEs in DeepChem

DeepChem boasts a number of methods for solving ODEs. Some of them are:

- Euler's Method
- Mid Point Method
- 3/8 Method
- RK-4 Method

Euler's Method (1st Order Runge-Kutta Method)

- It is the simplest Runge–Kutta method
- Explicit Runge–Kutta method with one stage

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Mid-Point Method (2nd Order Runge-Kutta Method)

- It is also called modified Euler's method
- second-order method with two stages

$$y_{n+1} = y_n + hf\left(t_n + \frac{1}{2}\right.$$

In [2]:
```
from deepchem.utils.differentiation_utils.integrate.explicit_rk import mid_point_ivp, fwd_euler_ivp
import matplotlib.pyplot as plt
import torch

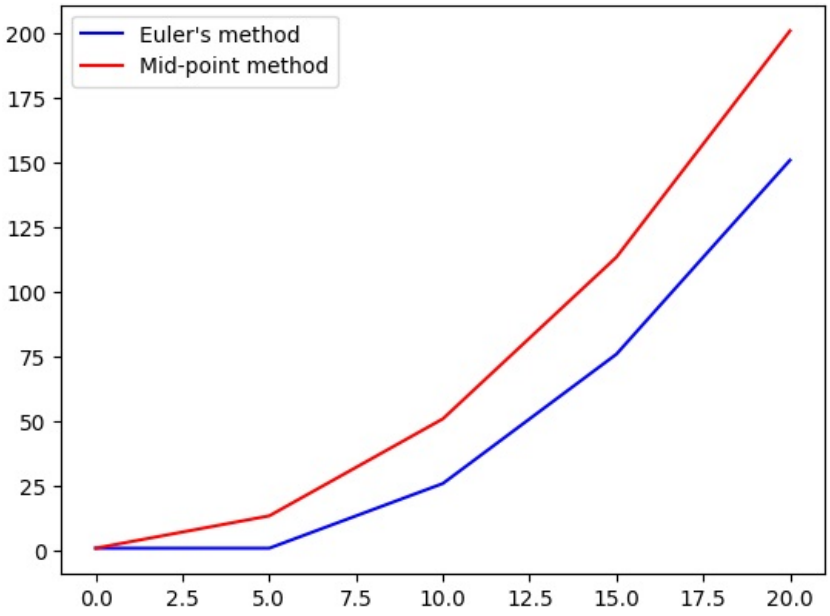
# Simple ODE dy/dt = a*y
def ode(t, y, params):
    a = params[0]
    return y * a

t = torch.linspace(0, 20, 5)
y_0 = torch.tensor([1])
a = torch.tensor([1])

sol_fwd_euler = fwd_euler_ivp(ode, y_0, t, [a])
sol_mid_point = mid_point_ivp(ode, y_0, t, [a])

plt.plot(t, sol_fwd_euler, "-b", label="Euler's method")
plt.plot(t, sol_mid_point, "-r", label="Mid-point method")
plt.legend(loc="upper left")
plt.show()
```

```
No normalization for SPS. Feature removed!
No normalization for AvgIpc. Feature removed!
Skipped loading some Tensorflow models, missing a dependency. No module named 'tensorflow'
Skipped loading modules with pytorch-geometric dependency, missing a dependency. No module named 'dgl'
Skipped loading modules with transformers dependency. No module named 'transformers'
cannot import name 'HuggingFaceModel' from 'deepchem.models.torch_models' (/home/gigavolt/deepchem/deepchem/mode
ls/torch_models/__init__.py)
Skipped loading modules with pytorch-lightning dependency, missing a dependency. No module named 'lightning'
Skipped loading some Jax models, missing a dependency. No module named 'jax'
Skipped loading some PyTorch models, missing a dependency. No module named 'tensorflow'
```



RK4 Method

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$t_{n+1} = t_n + h$$

for n = 0, 1, 2, 3, ...

$$k_1 = f(t_n, y_n)$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_1}{2}\right)$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + h\frac{k_2}{2}\right)$$

$$k_4 = f(t_n + h, y_n + hk_3)$$

Second Order Differential Equation

Example:
$\frac{d^2y}{dt^2} + \frac{dy}{dt} - 6y = 0$

given,
$y(0) = 5$
and
$dy/$
at t=0 is -5

Procedure:

$$\frac{dy}{dt} = z$$

$$\frac{dz}{dt} = 6y - z$$

In [3]:
```
from deepchem.utils.differentiation_utils.integrate.explicit_rk import rk4_ivp
import matplotlib.pyplot as plt
import torch

def sode(variables, t, params):
    y, z = variables

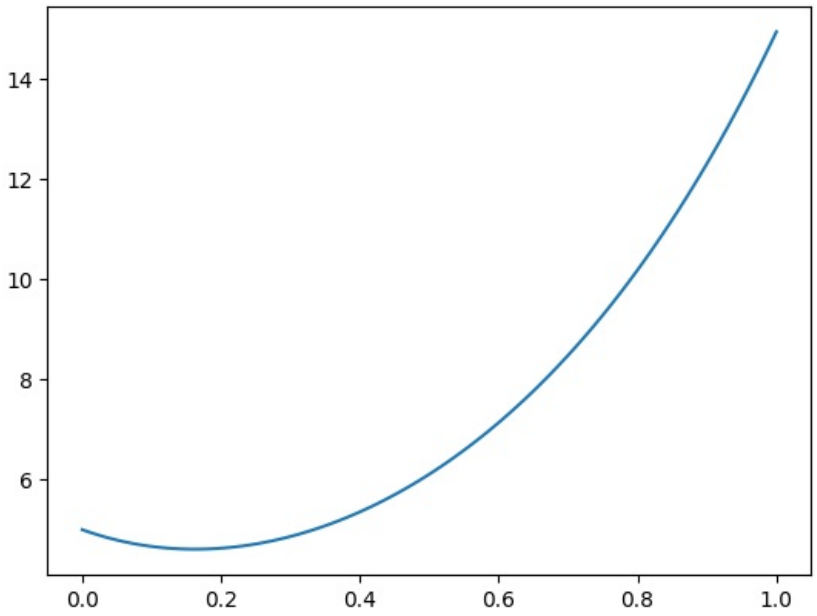
    a = params[0]

    dydt = z
    dzdt = a * y - z

    return torch.tensor([dydt, dzdt])

params = torch.tensor([6])
t = torch.linspace(0, 1, 100)
y0 = torch.tensor([5, -5])
sol = rk4_ivp(sode, y0, t, params)

plt.plot(t, sol[:, 0])
plt.show()
```

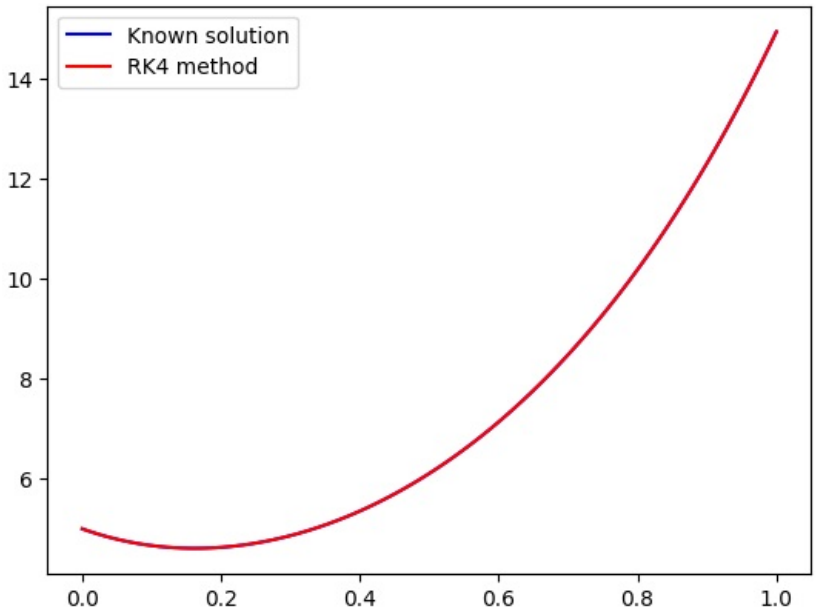


Comparing with Particular Solution

Particular Solution:
$y = 2e^{2x} + 3e^{-3x}$

In [4]:
```
yy = 2 * torch.exp(2 * t) + 3 * torch.exp(-3 * t)

plt.plot(t, yy, "-b", label="Known solution")
plt.plot(t, sol[:, 0], "-r", label="RK4 method")
plt.legend(loc="upper left")
plt.show()
```



Solving Lotka Volterra using Deepchem

The Lotka–Volterra equations, also known as the Lotka–Volterra predator–prey model, are a pair of first-order nonlinear differential equations, frequently used to describe the dynamics of biological systems in which two species interact, one as a predator and the other as prey. The populations change through time according to the pair of equations:

$$\frac{dx}{dt} = \alpha x - \beta xy,$$
$$\frac{dy}{dt} = -\gamma y + \delta xy,$$

The Lotka–Volterra system of equations is an example of a Kolmogorov model, which is a more general framework that can model the dynamics of ecological systems with predator–prey interactions, competition, disease, and mutualism.

In [5]:
```python
from deepchem.utils.differentiation_utils.integrate.explicit_rk import rk38_ivp
import matplotlib.pyplot as plt
import torch

def lotka_volterra(y, x, params):
    y1, y2 = y
    a, b, c, d = params
    return torch.tensor([(a * y1 - b * y1 * y2), (c * y2 * y1 - d * y2)])

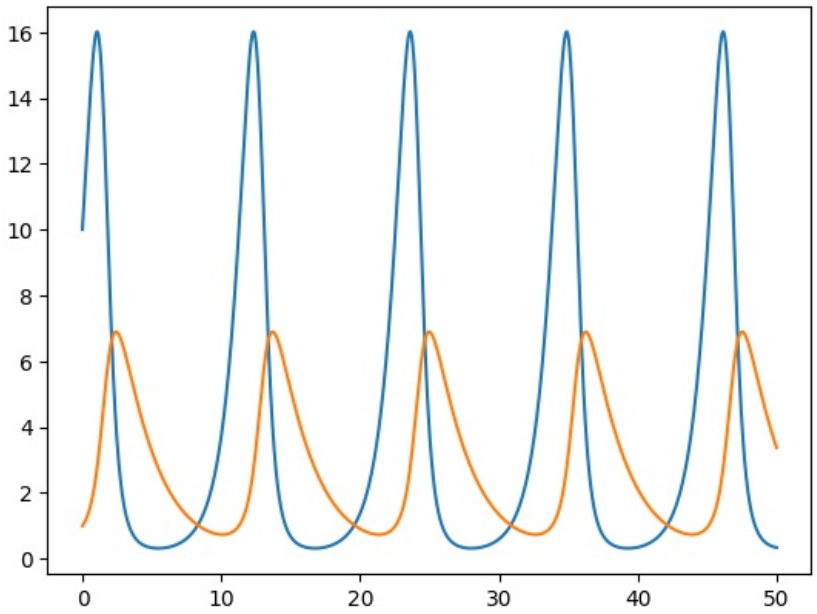
t = torch.linspace(0, 50, 10000)

solver_param = [lotka_volterra,
                torch.tensor([10., 1.]),
                t,
                torch.tensor([1.1, 0.4, 0.1, 0.4])]

sol_rk38 = rk38_ivp(*solver_param)

plt.plot(t, sol_rk38)

plt.show()
```



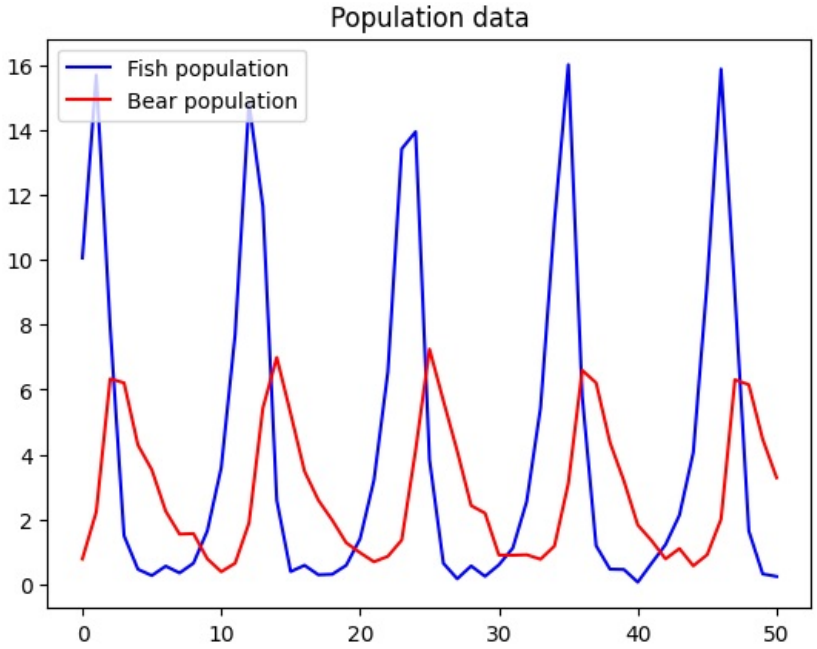
lotka volterra (Parammeter Estimation)

Parameter Estimation is used to estimate the values of the changable parameters in the ODE. The parameters describe an underlying physical setting in such a way that their value affects the distribution of the measured data. An estimator attempts to approximate the unknown parameters using the measurements.

In [6]:
```
import pandas as pd

dataset = pd.read_csv('assets/population_data.csv')
years = torch.tensor(dataset['year'])
fish_pop = torch.tensor(dataset['fish_hundreds'])
bears_pop = torch.tensor(dataset['bears_hundreds'])

plt.plot(fish_pop, "-b", label="Fish population")
plt.plot(bears_pop, "-r", label="Bear population")
plt.legend(loc="upper left")
plt.title("Population data")
plt.show()
```



In [7]:
```
from deepchem.utils.differentiation_utils.integrate.explicit_rk import rk4_ivp
import torch
import matplotlib.pyplot as plt


def lotka_volterra(y, x, params):
    y1, y2 = y
    a, b, c, d = params

    return torch.tensor([a * y1 - b * y1 * y2, c * y2 * y1 - d * y2])
```

```
def loss_function(params, years,fish_pop, bears_pop):

    y0 = torch.tensor([fish_pop[0], bears_pop[0]])

    t = torch.linspace(years[0], years[-1], len(years))

    output = rk4_ivp(lotka_volterra, y0, t, params)

    loss = 0

    for i in range(len(years)):
        data_fish = fish_pop[i]
        model_fish = output[i,0]

        data_bears = bears_pop[i]
        model_bears = output[i,1]

        res = (data_fish - model_fish)**2 + (data_bears - model_bears)**2

        loss += res

    return(loss)

import scipy.optimize

params0 = torch.tensor([1.1, .4, .1, .4])
minimum = scipy.optimize.fmin(loss_function, params0, args=(years,fish_pop, bears_pop))


alpha_fit = minimum[0]
beta_fit = minimum[1]
delta_fit = minimum[2]
gamma_fit = minimum[3]

params = torch.tensor([alpha_fit, beta_fit, delta_fit, gamma_fit])

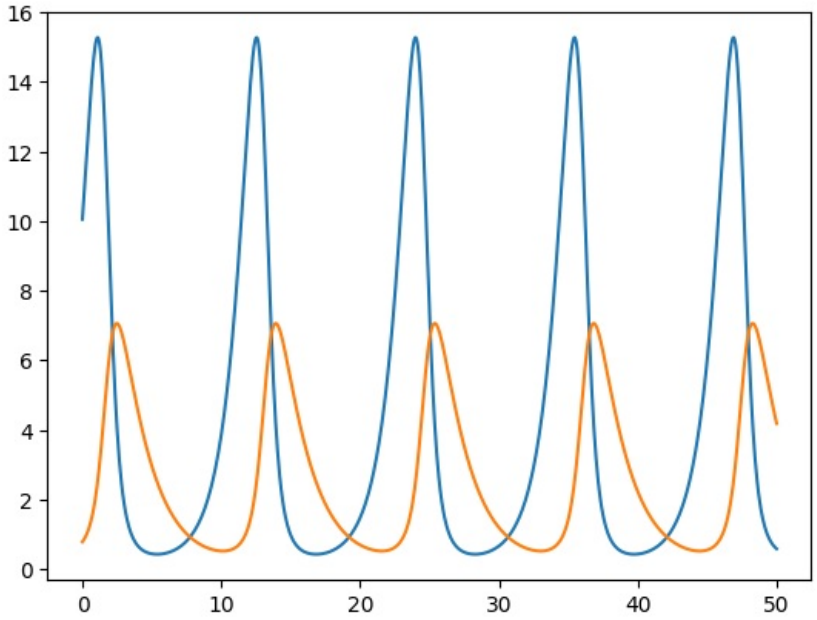
y0 = torch.tensor([fish_pop[0], bears_pop[0]])

t = torch.linspace(years[0], years[-1], 1000)

output = rk4_ivp(lotka_volterra, y0, t, params)

plt.plot(t, output)
plt.show()
```

```
Optimization terminated successfully.
         Current function value: 42.135876
         Iterations: 155
         Function evaluations: 256
```



SIR Epidemiology

The SIR model is one of the simplest compartmental models, and many models are derivatives of this basic form. The model consists of three compartments:

- S: The number of susceptible individuals. When a susceptible and an infectious individual come into "infectious

contact", the susceptible individual contracts the disease and transitions to the infectious compartment.

- I: The number of infectious individuals. These are individuals who have been infected and are capable of infecting susceptible individuals.
- R: The number of removed (and immune) or deceased individuals. These are individuals who have been infected and have either recovered from the disease and entered the removed compartment, or died. It is assumed that the number of deaths is negligible with respect to the total population. This compartment may also be called "recovered" or "resistant".

$$\frac{dS}{dt} = -\frac{\beta IS}{N}$$

$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I$$

$$\frac{dR}{dt} = \gamma I$$

$$N = S + I + R$$

$$R_0 = \frac{\beta}{\gamma}$$

In [12]:
```
import torch
import matplotlib.pyplot as plt
from deepchem.utils.differentiation_utils.integrate.explicit_rk import rk4_ivp

def sim(variables, t, params):
    S, I, R = variables

    N = S + I + R

    beta, gamma = params

    dSdt = - beta * I * S / N
    dIdt = beta * I * S / N - gamma * I
    dRdt = gamma * I

    return torch.tensor([dSdt, dIdt, dRdt])

t = torch.linspace(0, 500, 1000)

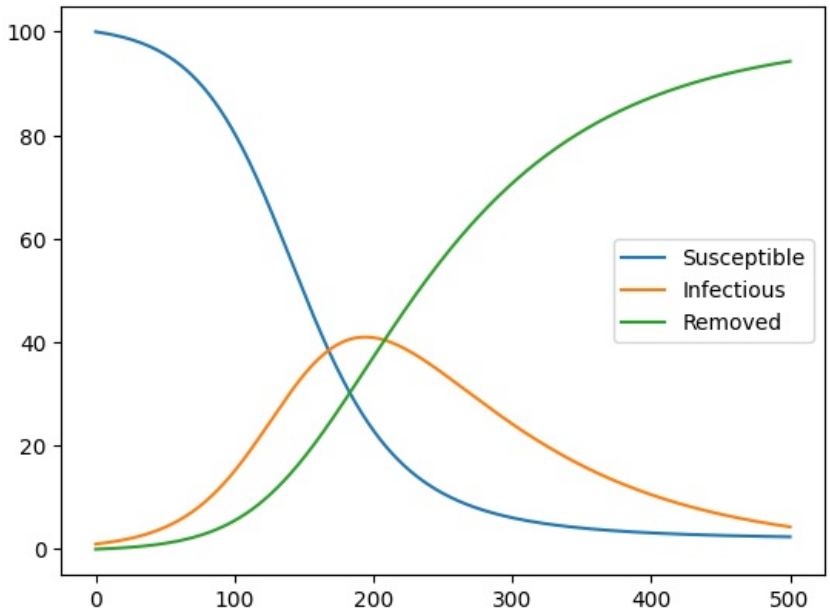
beta = 0.04
gamma = 0.01

params = torch.tensor([beta, gamma])

y0 = torch.tensor([100, 1, 0])

y = rk4_ivp(sim, y0, t, params)

plt.plot(t, y)
plt.legend(["Susceptible", "Infectious", "Removed"])
plt.show()
```



SIS Model

Some infections, for example, those from the common cold and influenza, do not confer any long-lasting immunity. Such infections may give temporary resistance but do not give long-term immunity upon recovery from infection, and individuals become susceptible again.

Model:

$$\frac{dS}{dt} = -\frac{\beta IS}{N} + \gamma I$$
$$\frac{dI}{dt} = \frac{\beta IS}{N} - \gamma I$$

Total Population:

$$N = S(t) + I(t)$$
$$\frac{dR}{dt} = \gamma I$$
$$R_0 = \frac{\beta}{\gamma}$$

In [13]:
```python
import torch
import matplotlib.pyplot as plt
from deepchem.utils.differentiation_utils.integrate.explicit_rk import rk4_ivp

def sim(variables, t, params):
    S, I = variables

    N = S + I

    beta, gamma = params

    dSdt = - beta * I * S / N + gamma * I
    dIdt = beta * I * S / N - gamma * I

    return torch.tensor([dSdt, dIdt])

t = torch.linspace(0, 500, 1000)

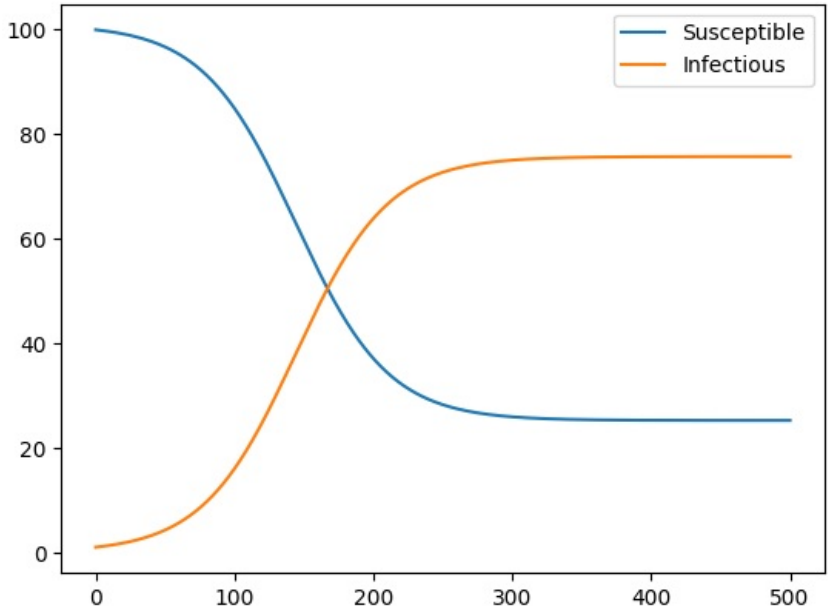
beta = 0.04
gamma = 0.01

params = torch.tensor([beta, gamma])

y0 = torch.tensor([100, 1])

y = rk4_ivp(sim, y0, t, params)

plt.plot(t, y)
plt.legend(["Susceptible", "Infectious"])
plt.show()
```



References

1. More Computational Biology and Python by Mike Saint-Antoine https://www.youtube.com/playlist?list=PLWVKUEZ25V97W2qS7faggHrv5gdhPcgjq
2. Compartmental models in epidemiology. (2024, May 27). In Wikipedia. https://en.wikipedia.org/wiki/Compartmental_models_in_epidemiology
3. Runge–Kutta methods. (2024, June 1). In Wikipedia. https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods

Citing This Tutorial

If you found this tutorial useful please consider citing it using the provided BibTeX.

```
@manual{Differential Equation,
 title={Differentiation Infrastructure in Deepchem},
 organization={DeepChem},
 author={Singh, Rakshit kr. and Ramsundar, Bharath},
 howpublished =
{\url{https://github.com/deepchem/deepchem/blob/master/examples/tutorials/ODE_Solving.ipynb}},
 year={2024},
}
```

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Discord

The DeepChem Discord hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!

Introduction To Equivariance and Equivariant Modeling

Table of Contents:

- Introduction
- What is Equivariance
- Why Do We Need Equivariance
- Example
- References

Introduction

In the preceding sections of this tutorial series, we focused on training models using DeepChem for various applications. However, we haven't yet addressed the important topic of equivariant modeling.

Equivariant modeling ensures that the relationship between input and output remains consistent even when subjected to symmetry operations. By incorporating equivariant modeling techniques, we can effectively analyze and predict diverse properties by leveraging the inherent symmetries present in the data. This is particularly valuable in the fields of cheminformatics, bioinformatics, and material sciences, where understanding the interplay between symmetries and properties of molecules and materials is critical.

This tutorial aims to explore the concept of equivariance and its significance within the domains of chemistry, biology, and material sciences. We will delve into the reasons why equivariant modeling is vital for accurately characterizing and predicting the properties of molecules and materials. By the end, you will have a solid understanding of the importance of equivariance and how it can significantly enhance our modeling capabilities in these areas.

You can follow this tutorial using the Google Colab. If you'd like to open this notebook in colab, you can use the following link.



What is Equivariance

A key aspect of the structure in our data is the presence of certain symmetries. To effectively capture this structure, our model should incorporate our knowledge of these symmetries.Therefore, our model should retain the symmetries of the input data in its outputs. In other words, when we apply a symmetry operation (denoted by σ) to the input and pass it through the model, the result should be the same as applying σ to the output of the model.

Mathematically, we can express this idea as an equation:

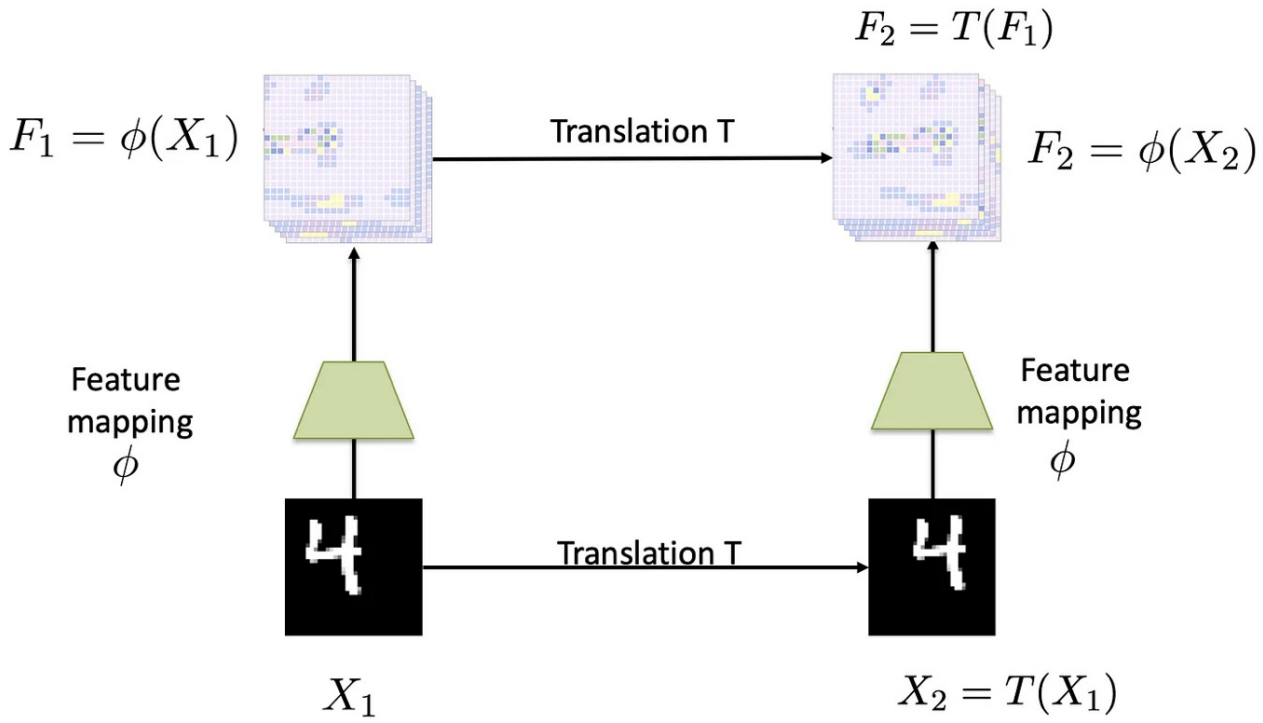
$$f(\sigma(x)) = \sigma(f(x))$$

Here, f represents the function learned by our model. If this equation holds for every symmetry operation in a collection S, we say that f is equivariant with respect to S.

While a precise definition of equivariance involves group theory and allows for differences between the applied symmetry operations on the input and output, we'll focus on the case where they are identical to keep things simpler. Group Equivariant Convolutional Networks exemplify this stricter definition of equivariance.

Interestingly, equivariance shares a similarity with linearity. Just as linear functions are equivariant with respect to scalar multiplication, equivariant functions allow symmetry operations to be applied inside or outside the function.

To gain a better understanding, let's consider Convolutional Neural Networks (CNNs). The image below demonstrates how CNNs exhibit equivariance with respect to translation: a shift in the input image directly corresponds to a shift in the output features.



It is also useful to relate equivariance to the concept of invariance, which is more familiar. If a function f is invariant, its output remains unchanged when σ is applied to the input. In this case, the equation simplifies to:

```
f(σ(x)) = f(x)
```

An equivariant embedding in one layer can be transformed into an invariant embedding in a subsequent layer. The feasibility and meaningfulness of this transformation depend on the implementation of equivariance. Notably, networks with multiple convolutional layers followed by a global average pooling layer (GAP) achieve this conversion. In such cases, everything up to the GAP layer exhibits translation equivariance, while the output of the GAP layer (and the entire network) becomes invariant to translations of the input.

Why Do We Need Equivariance

Equivariance is a critical concept in modeling various types of data, particularly when dealing with structures and symmetries. It provides a powerful framework for capturing and leveraging the inherent properties and relationships present in the data. In this section, we will explore the reasons why equivariance is essential and how it is particularly advantageous when working with graph-structured data.

1. Preserving Structural Information

Data often exhibits inherent structural properties and symmetries. Equivariant models preserve the structural information present in the data, allowing us to analyze and manipulate it while maintaining consistency under symmetry operations. By doing so, equivariant models capture the underlying relationships and patterns, leading to more accurate and meaningful insights.

2. Handling Symmetry and Invariance

Symmetries and invariances are prevalent in many real-world systems. Equivariant models ensure that the learned representations and predictions are consistent under these symmetries and invariances. By explicitly modeling equivariance, we can effectively handle and exploit these properties, leading to robust and reliable models.

3. Improved Generalization

Equivariant models have the advantage of generalizing well to unseen data. By incorporating the known symmetries and structures of the domain into the model architecture, equivariance ensures that the model can effectively capture and utilize these patterns even when presented with novel examples. This leads to improved generalization performance, making equivariant models valuable in scenarios where extrapolation or prediction on unseen instances is crucial.

4. Efficient Processing of Graph-Structured Data

Graph-structured data possess rich relational information and symmetries. Equivariant models specifically tailored for graph data offer a natural and efficient way to model and reason about these complex relationships. By considering the symmetries of the graph, equivariant models can effectively capture the local and global patterns, enabling tasks such as node classification, link prediction, and graph generation.

Example

Traditional machine learning (ML) algorithms face challenges when predicting molecular properties due to the representation of molecules. Typically, molecules are represented as 3D Cartesian arrays with a shape of (points, 3). However, neural networks (NN) cannot directly process such arrays because each position in the array lacks individual significance. For instance, a molecule can be represented by one Cartesian array centered at (0, 0, 0) and another centered at (15, 15, 15), both representing the same molecule but with distinct numerical values. This exemplifies translational variance. Similarly, rotational variance arises when the molecule is rotated instead of translated.

In these examples, if the different arrays representing the same molecule are inputted into the NN, it would perceive them as distinct molecules, which is not the case. To address these issues of translational and rotational variance, considerable efforts have been devoted to devising alternative input representations for molecules. Let’s demonstrate with some code how to go about creating functions that obey set of equivariances. We won’t be training these models because training has no effect on equivariances.

To define the molecule, we represent it as a collection of coordinates (denoted as $\mathbf{R}_i$) and corresponding features (denoted as $\mathbf{X}_i$). The features are encoded as one-hot vectors, where [1, 0] indicates a carbon atom, and [0, 1] indicates a hydrogen atom. In this specific example, our focus is on predicting the energy associated with the molecule. It's important to note that we will not be training our models, meaning the predicted energy values will not be accurate.

Let's define a random molecule with 12 atoms as an example:

In [47]:
```
import numpy as np
np.random.seed(42) # seed for reproducibility

R_i = np.random.rand(12, 3) # 12 atoms with xyz coordinates
N = R_i.shape[0] # number of atoms
X_i = np.zeros((N, 2)) # feature vectors for the atoms with shape (N, 2)
X_i[:4, 0] = 1
X_i[4:, 1] = 1
```

An example of a model that lacks equivariances is a one-hidden layer dense neural network. In this model, we concatenate the positions and features of our data into a single input tensor, which is then passed through a dense layer. The dense layer utilizes the hyperbolic tangent (tanh) activation function and has a hidden layer dimension of 16. The output layer, which performs regression to energy, does not have an activation function. The weights of the model are always initialized randomly.

Let's define our hidden model and initialize the weights:

In [30]:
```
def hidden_model(r: np.ndarray, x: np.ndarray, w1: np.ndarray, w2: np.ndarray, b1: np.ndarray, b2: float) -> np
    r"""Computes the output of a 1-hidden layer neural network model.

    Parameters
    ----------
        r : np.ndarray
            Input array for position values.
            Shape: (num_atoms, num_positions)
        x : np.ndarray
            Input array for features.
            Shape: (num_atoms, num_features)
        w1 : np.ndarray
            Weight matrix for the first layer.
            Shape: (num_positions + num_features, hidden_size)
        w2 : np.ndarray
            Weight matrix for the second layer.
            Shape: (hidden_size, output_size)
        b1 : np.ndarray
            Bias vector for the first layer.
            Shape: (hidden_size,)
        b2 : float
            Bias value for the second layer.

    Returns
    -------
```

```
        float
            Predicted energy of the molecule
    """
    i = np.concatenate((r, x), axis=1).flatten()  # Stack inputs into one large input
    v = np.tanh(i @ w1 + b1)  # Apply activation function to first layer
    v = v @ w2 + b2  # Multiply with weights and add bias for the second layer
    return v


# Initialize weights for a network with hidden size is 16
w1 = np.random.normal(size=(N * 5, 16))  # 3(#positions) + 2(#features) = 5
b1 = np.random.normal(size=(16,))
w2 = np.random.normal(size=(16,))
b2 = np.random.normal()
```

Although our model is not trained, we are not concerned about. Since, we only want see if our model is affected by permutations, translations and rotations

In [29]:
```
import scipy.spatial.transform as transform

rotate = transform.Rotation.from_euler("x", 60, degrees=True) # Rotate around x axis by 60 degrees

permuted_R_i = np.copy(R_i)
permuted_R_i[0], permuted_R_i[1] = R_i[1], R_i[0] # Swap the rows of R_i

print("without change:", hidden_model(R_i, X_i, w1, w2, b1, b2))
print("after permutation:", hidden_model(permuted_R_i, X_i, w1, w2, b1, b2))
print("after translation:", hidden_model(R_i + np.array([3, 3, 3]), X_i, w1, w2, b1, b2))
print("after rotation:", hidden_model(rotate.apply(R_i), X_i, w1, w2, b1, b2))
```

```
without change: 9.945112980229641
after permutation: 9.461406567572851
after translation: 5.963826685170721
after rotation: 7.191211524244547
```

As expected, our model is not invariant to any permutations, translations, or rotations. Let's fix them.

Permutational Invariance

In a molecular context, the arrangement or ordering of points in an input tensor holds no significance. Therefore, it is crucial to be cautious and avoid relying on this ordering. To ensure this, we adopt a strategy of solely performing atom-wise operations within the network to obtain atomic property predictions. When predicting molecular properties, we need to cumulatively combine these atomic predictions, such as using summation, to arrive at the desired result. This approach guarantees that the model does not depend on the arbitrary ordering of atoms within the input tensor.

Let's fix permutation invariance problem exists in our hidden model.

In [35]:
```
def hidden_model_perm(r: np.ndarray, x: np.ndarray, w1: np.ndarray, w2: np.ndarray, b1: np.ndarray, b2: float)
    r"""Computes the output of a 1-hidden layer neural network model with permutation invariance.

    Parameters
    ----------
        r : np.ndarray
            Input array for position values.
            Shape: (num_atoms, num_positions)
        x : np.ndarray
            Input array for features.
            Shape: (num_atoms, num_features)
        w1 : np.ndarray
            Weight matrix for the first layer.
            Shape: (num_positions + num_features, hidden_size)
        w2 : np.ndarray
            Weight matrix for the second layer.
            Shape: (hidden_size, output_size)
        b1 : np.ndarray
            Bias vector for the first layer.
            Shape: (hidden_size,)
        b2 : float
            Bias value for the second layer.

    Returns
    -------
        float
            Predicted energy of the molecule
    """
    i = np.concatenate((r, x), axis=1)  # Stack inputs into one large input
    v = np.tanh(i @ w1 + b1)  # Apply activation function to first layer
    v = np.sum(v, axis=0) # Reduce the output by summing across the axis which gives permutational invariance
    v = v @ w2 + b2  # Multiply with weights and add bias for the second layer
    return v
```

```
# Initialize weights
w1 = np.random.normal(size=(5, 16))
b1 = np.random.normal(size=(16,))
w2 = np.random.normal(size=(16,))
b2 = np.random.normal()
```

In the original implementation, the model computes intermediate activations v for each input position separately and then concatenates them along the axis 0. By summing across axis 0 with (np.sum(v, axis=0)), the model effectively collapses all the intermediate activations into a single vector, regardless of the order of the input positions.

This reduction operation allows the model to be permutation invariant because the final output is only dependent on the aggregated information from the intermediate activations and is not affected by the specific order of the input positions. Therefore, the model produces the same output for different permutations of the input positions, ensuring permutation invariance.

Now let's see if this changes affected our model's sensitivity to permutations.

In [36]:
```
print("without change:", hidden_model_perm(R_i, X_i, w1, w2, b1, b2))
print("after permutation:", hidden_model_perm(permuted_R_i, X_i, w1, w2, b1, b2))
print("after translation:", hidden_model_perm(R_i + np.array([3, 3, 3]), X_i, w1, w2, b1, b2))
print("after rotation:", hidden_model_perm(rotate.apply(R_i), X_i, w1, w2, b1, b2))
```

```
without change: -19.370847873678944
after permutation: -19.370847873678944
after translation: -67.71502903638384
after rotation: 5.311140035302996
```

Indeed! As anticipated, our model demonstrates invariance to permutations while remaining sensitive to translations or rotations.

Translational Invariance

To address the issue of translational variance in modeling molecules, one approach is to compute the distance matrix of the molecule. This distance matrix provides a representation that is invariant to translation. However, this approach introduces a challenge as the distance features change from having three features per atom to $N \times 3$ features per atom. Consequently, we have introduced a dependency on the number of atoms in our distance features, making it easier to inadvertently break permutation invariance. To mitigate this issue, we can simply sum over the newly added axis, effectively collapsing the information into a single value. This summation ensures that the model remains invariant to permutations, restoring the desired permutation invariance property.

Let's fix translation invariance problem exists in our hidden model

In [50]:
```
def hidden_model_permute_translate(r: np.ndarray, x: np.ndarray, w1: np.ndarray, w2: np.ndarray, b1: np.ndarray
    r"""Computes the output of a 1-hidden layer neural network model with permutation and translation invarianc

    Parameters
    ----------
        r : np.ndarray
            Input array for position values.
            Shape: (num_atoms, num_positions)
        x : np.ndarray
            Input array for features.
            Shape: (num_atoms, num_features)
        w1 : np.ndarray
            Weight matrix for the first layer.
            Shape: (num_positions + num_features, hidden_size)
        w2 : np.ndarray
            Weight matrix for the second layer.
            Shape: (hidden_size, output_size)
        b1 : np.ndarray
            Bias vector for the first layer.
            Shape: (hidden_size,)
        b2 : float
            Bias value for the second layer.

    Returns
    -------
        float
            Predicted energy of the molecule
    """
    d = r - r[:, np.newaxis] # Compute pairwise distances using broadcasting

    # Stack inputs into one large input of N x N x 5
    # Concatenate doesn't broadcast, so we manually broadcast the Nx2 x matrix
    # into N x N x 2
    i = np.concatenate((d, np.broadcast_to(x, (d.shape[:-1] + x.shape[-1:]))), axis=-1)
```

```
    v = np.tanh(i @ w1 + b1)  # Apply activation function to first layer

    v = np.sum(v, axis=(0, 1)) # Reduce the output over both axes by summing
    v = v @ w2 + b2  # Multiply with weights and add bias for the second layer
    return v
```

To achieve translational invariance, the function calculates pairwise distances between the position values in the r array. This is done by subtracting r from r[:, np.newaxis], which broadcasts r along a new axis, enabling element-wise subtraction.

The pairwise distance calculation is based on the fact that subtracting the positions r from each other effectively measures the distance or difference between them. By including the pairwise distances in the input, the model can learn and capture the relationship between the distances and the features. This allows the model to be invariant to translations, meaning that shifting the positions within each set while preserving their relative distances will result in the same output.

Now let's see if this changes affected our model's sensitivity to permutations.

In [51]:
```
print("without change:", hidden_model_permute_translate(R_i, X_i, w1, w2, b1, b2))
print("after permutation:", hidden_model_permute_translate(permuted_R_i, X_i, w1, w2, b1, b2))
print("after translation:", hidden_model_permute_translate(R_i + np.array([3, 3, 3]), X_i, w1, w2, b1, b2))
print("after rotation:", hidden_model_permute_translate(rotate.apply(R_i), X_i, w1, w2, b1, b2))
```

```
without change: 193.79734623037373
after permutation: 193.79734623037385
after translation: 193.79734623037368
after rotation: 188.36773620787383
```

Yes! Our model is invariant to both permutations and translations but not to rotations.

Rotational Invariance

Atom-centered symmetry functions exhibit rotational invariance due to the invariance of the distance matrix. While this property is suitable for tasks where scalar values, such as energy, are predicted from molecules, it poses a challenge for problems that depend on directionality. In such cases, achieving rotational equivariance is desired, where the output of the network rotates in the same manner as the input. Examples of such problems include force prediction and molecular dynamics.

To address this, we can convert the pairwise vectors into pairwise distances. To simplify the process, we utilize squared distances. This conversion allows us to incorporate directional information while maintaining simplicity. By considering the squared distances, we enable the network to capture and process the relevant geometric relationships between atoms, enabling rotational equivariance and facilitating accurate predictions for direction-dependent tasks.

In [52]:
```
def hidden_model_permute_translate_rotate(r: np.ndarray, x: np.ndarray, w1: np.ndarray, w2: np.ndarray, b1: np.
    r"""Computes the output of a 1-hidden layer neural network model with permutation, translation, and rotatio

    Parameters
    ----------
        r : np.ndarray
            Input array for position values.
            Shape: (num_atoms, num_positions)
        x : np.ndarray
            Input array for features.
            Shape: (num_atoms, num_features)
        w1 : np.ndarray
            Weight matrix for the first layer.
            Shape: (num_positions, hidden_size)
        w2 : np.ndarray
            Weight matrix for the second layer.
            Shape: (hidden_size, output_size)
        b1 : np.ndarray
            Bias vector for the first layer.
            Shape: (hidden_size,)
        b2 : float
            Bias value for the second layer.

    Returns
    -------
        float
            Predicted energy of the molecule
    """
    # Compute pairwise distances using broadcasting
    d = r - r[:, np.newaxis]
    # Compute squared distances
    d2 = np.sum(d**2, axis=-1, keepdims=True)

    # Stack inputs into one large input of N x N x 3
    # Concatenate doesn't broadcast, so we manually broadcast the Nx2 x matrix
```

```
    # into N x N x 2
    i = np.concatenate((d2, np.broadcast_to(x, (d2.shape[:-1] + x.shape[-1:]))), axis=-1)
    v = np.tanh(i @ w1 + b1)  # Apply activation function to first layer

    # Reduce the output over both axes by summing
    v = np.sum(v, axis=(0, 1))

    v = v @ w2 + b2  # Multiply with weights and add bias for the second layer
    return v

# Initialize weights
w1 = np.random.normal(size=(3, 16))
b1 = np.random.normal(size=(16,))
w2 = np.random.normal(size=(16,))
b2 = np.random.normal()
```

The hidden_model_permute_trans_rotate function achieves rotational invariance through the utilization of pairwise squared distances between atoms, instead of the pairwise vectors themselves. By using squared distances, the function is able to incorporate directional information while still maintaining simplicity in the calculation.

Squared distances inherently encode geometric relationships between atoms, such as their relative positions and orientations. This information is essential for capturing the directionality of interactions and phenomena in tasks like force prediction and molecular dynamics, where rotational equivariance is desired.

The conversion from pairwise vectors to pairwise squared distances allows the model to capture and process these geometric relationships. Since squared distances only consider the magnitudes of vectors, disregarding their directions, the resulting network output remains invariant under rotations of the input.

Now let's see if this changes affected our model's sensitivity to rotations.

In [30]:
```
print("without change:", hidden_model_permute_trans_rotate(R_i, X_i, w1, w2, b1, b2))
print("after permutation:", hidden_model_permute_trans_rotate(permuted_R_i, X_i, w1, w2, b1, b2))
print("after translation:", hidden_model_permute_trans_rotate(R_i + np.array([3, 3, 3]), X_i, w1, w2, b1, b2))
print("after rotation:", hidden_model_permute_trans_rotate(rotate.apply(R_i), X_i, w1, w2, b1, b2))
```

```
without change: 585.1386319324105
after permutation: 585.1386319324106
after translation: 585.1386319324105
after rotation: 585.1386319324105
```

Yes! Now our model is invariant to both permutations, translations, and rotations.

With these new changes, our model exhibits improved representation capacity and generalization while preserving the symmetry of the molecules.

References

- Bronstein, M.M., Bruna, J., Cohen, T., & Velivckovi'c, P. (2021). Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. ArXiv, abs/2104.13478.
- White, A.D. (2022). Deep learning for molecules and materials. Living Journal of Computational Molecular Science.
- Geiger, M., & Smidt, T.E. (2022). e3nn: Euclidean Neural Networks. ArXiv, abs/2207.09453.

Congratulations! Time to join the Community!

Congratulations on completing this tutorial notebook! If you enjoyed working through the tutorial, and want to continue working with DeepChem, we encourage you to finish the rest of the tutorials in this series. You can also help the DeepChem community in the following ways:

Star DeepChem on GitHub

This helps build awareness of the DeepChem project and the tools for open source drug discovery that we're trying to build.

Join the DeepChem Gitter

The DeepChem Gitter hosts a number of scientists, developers, and enthusiasts interested in deep learning for the life sciences. Join the conversation!