Generative AI for Critical Digital Twins

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Mechanical Engineering

Wenhao Ding

B.E., Department of Electronic Engineering, Tsinghua University
M.S., Machine Learning Department, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA 15213
May, 2024

© Wenhao Ding, 2024

All Rights Reserved

Acknowledgments

The research in this thesis is supported in part by the Qualcomm Innovation Fellowship, NSF CAREER CNS-2047454, US Department of Transportation Mobility 21, US Department of Transportation Safety 21, Uber, Rolls Royce, and Manufacturing Futures Institute.

I express my heartfelt gratitude to the members of my thesis committee, Dr. Ding Zhao at Carnegie Mellon University, Dr. Bo Li at the University of Chicago, Dr. Marco Pavone at Stanford University, and Dr. Conrad Tucker at Carnegie Mellon University. All of them guided my research directions when I felt perplexed, supported my research when I faced obstacles, and provided feedback on the writing of this thesis. As the chair of the committee and my Ph.D. advisor, Dr. Zhao created an optimal research environment that I can imagine, where I have enough freedom to explore research topics with a satisfying work-life balance. My collaboration with Dr. Li spanned over three years, a period during which I not only acquired substantial technical knowledge but also learned the importance of dedication to one's work. I spent an impressive internship time in the NVIDIA group managed by Dr. Pavone, in which I had a wonderful experience with a deep understanding of research topics and close collaboration with people in the group. Dr. Tucker, with his extensive expertise in digital twins, contributed significant resources and insights that enriched the breadth of my research.

In addition to the support of academic mentors, the contributions of my peers were also instrumental in my learning process. Upon embarking on my doctoral studies, I frequently engaged in detailed conversations about reinforcement learning and deep generative models with colleagues Jiacheng Zhu, Mengdi Xu, and Zuxin Liu. These exchanges were pivotal in shaping my foundational understanding of the academic literature and interdisciplinary connections, profoundly influencing my research style and taste in the following years to this day. Specifically, my discussions with Jiacheng about optimal transport and its integration with causality led to an academic proposal that won the Qualcomm Innovation Fellowship. Collaborating with Mengdi on model-based reinforcement learning for multi-task challenges resulted in my first publication at a major conference. Furthermore, my conversations with Zuxin, who was also my roommate, on the topics of safety and robustness significantly inspired my research trajectory.

I would like to thank all the collaborators who have supported my research: Haohong Lin, Yiqi Lyv, Chejian Xu, Miao Li, Mansur Arief, Mengdi Xu, Laixi Shi, Wenshuo Wang, Yulong Cao, Chaowei Xiao, Tong Che, Yuejie Chi, Nathalie Majcherczyk, Mohit Deshpande, Xuewei Qi, Shuai Wang, Minjun Xu. I am honored to have their names associated with my work, which has significantly enhanced its quality. Their contributions have been indispensable in reaching the current stage of my research. I am fortunate to exist in a dynamic and collaborative era, surrounded by outstanding individuals dedicated to advancing human potential. I am so privileged to be one of them.

Finally, I extend my heartfelt gratitude to my parents, Baojun Ding and Su'e Liang, for their unwavering support and nurturing since my birth, and for providing all the conditions to help me achieve my dream. I also want to thank my significant other, Miao Li. Despite the geographical separation during my Ph.D. studies abroad, our bond has only strengthened. I am confident that this marks merely the beginning of our lifelong journey together.

Abstract

Training and evaluating autonomous robots within the real world present significant challenges and risks, emanating from unpredictable environments, safety concerns, ethical dilemmas, and limited human oversight. As a mitigation strategy, the use of realistic simulations, also known as *digital twins*, offers virtual duplication of the actual system or environment, thus fostering the development of trustworthy autonomy.

Digital twins enable developers to evaluate the performance of systems in various scenarios to identify potential risks or failure cases. It facilitates the accumulation and subsequent analysis of datasets, which serve to validate and calibrate the autonomous system's perception and decision-making algorithms. By comparing the behavior of the digital twin with real-world data, developers can identify discrepancies, improve accuracy, and enhance the system's safety and reliability.

Scenarios that embody dynamic and interactive components reflect the intricacies of digital twins and take precedence in significance. One main value of digital twins is helping us understand how objects interact and behave. For example, in autonomous driving, the behavior of vehicles, pedestrians, and traffic conditions are crucial components of scenarios that need to be accurately modeled. However, not all scenarios in digital twins are created equal. In the pursuit of developing trustworthy autonomy, ordinary scenarios often prove insufficient in subjecting autonomous systems to extreme conditions where safety and robustness are paramount. Although critical scenarios hold the potential to expose model vulnerabilities, their rare occurrence creates a challenge. The process of manually identifying or extrapolating such critical scenarios from normal data or expert design proves not only inefficient but also contains substantial human biases.

My doctoral research seeks to harness the potential of generative AI to explore two pivotal questions: (1) Which scenarios are critical in existing data and (2) How to generate such scenarios in digital twins? The proposal begins with the definition of critical scenarios and the corresponding optimization problem and subsequently delves into three distinct categories of scenario generation frameworks: **data-driven generative models**, **adversarial generative models**, and **knowledge-guided generative models**. Concluding this thesis is future directions that effectively combine generation resources from different perspectives and improve the data flywheel.

Contents

List of Tables viii

List of Figures xi

List of Symbols xii

1 Introduction 1
- 1.1 Digital Twins, Data, and Scenarios . . . 1
- 1.2 Critical Digital Twins . . . 3
  - 1.2.1 Components . . . 3
  - 1.2.2 Objective of Generation . . . 4
  - 1.2.3 Metrics . . . 5
- 1.3 Literature of Generative Models in Digital Twins . . . 6
  - 1.3.1 Representation of Scenario . . . 6
  - 1.3.2 Rule-based Scenario Generation . . . 7
  - 1.3.3 Learning-based Scenario Generation . . . 9
  - 1.3.4 Critical Scenario Generation . . . 11
  - 1.3.5 Dataset and Tools for Scenario Generation . . . 14
- 1.4 Challenge . . . 22
- 1.5 Thesis Structure . . . 25

2 Data-driven Generative Models 27
- 2.1 Generation via Latent Space Sampling . . . 27
  - 2.1.1 Variational Auto-encoder . . . 29
  - 2.1.2 Multi-Vehicle Trajectory Generator (MTG) . . . 30
  - 2.1.3 A Metric for Disentanglement . . . 32
  - 2.1.4 Experiment and Analysis . . . 33
- 2.2 Risk-level Adjustment with Controllable Generation . . . 39
  - 2.2.1 Preliminary and Problem Formulation . . . 42
  - 2.2.2 Conditional Multiple Trajectory Synthesizer (CMTS) . . . 42
  - 2.2.3 Experiment and Analysis . . . 45

2.3 Retrieval Augmented Generation for Controllable Traffic Scenarios 51
2.3.1 A Latent Representation of Scenario 53
2.3.2 Retrieval Augmented Scenario Generation (RealGen) 57
2.3.3 Experiment 60
2.4 Summary 68

3 Adversarial Generative Models **71**
3.1 Generation with Reinforcement Learning 72
3.1.1 Representing scenario with Probabilistic Models 74
3.1.2 Learning to Collide (L2C) 77
3.1.3 Experiment and Analysis 80
3.2 Diverse Scenario Generation with Adaptive Sampler 85
3.2.1 Flow-based Models 87
3.2.2 Multi-modal Generation (MMG) 88
3.2.3 Experiment and Analysis 92
3.3 Summary 101

4 Knowledge-based Generative Models **102**
4.1 Scenario Reasoning with Causality 103
4.1.1 Graphical Representation of Scenarios 105
4.1.2 Causal Autoregressive Flow (CausalAF) 107
4.1.3 Experiment and Analysis 114
4.2 Generative Models with Causal Discovery 119
4.2.1 Problem Formulation and Preliminary 121
4.2.2 Generalizing by Discovering (GRADER) 123
4.2.3 Analysis of Performance Guarantee 133
4.2.4 Experiment and Analysis 145
4.3 Break Spurious Correlation with Generative Models 155
4.3.1 Preliminary and Limitations of Robust RL 158
4.3.2 Robust RL against Structured Uncertainty from Causal Perspective . . . 160
4.3.3 Theoretical Analyses RSC-MDPs 165
4.3.4 An Empirical Algorithm to Solve RSC-MDPs: RSC-SAC 178
4.3.5 Experiments and Analysis 181
4.4 Summary 190

5 Conclusion **192**
5.1 Combination of Data, Adversary, and Knowledge 192
5.2 Future Directions of Critical Digital Twins 194

Bibliography **197**

List of Tables

1.1 Scenario Datasets. We list existing datasets and different aspects that we are interested in. ✓/× in the weather column means the dataset contains/doesn't contain data under various weather conditions. BEV: bird's-eye view, FPV: first-person view. D: daytime, N: nighttime. H: highway, I: intersection, RA: round-about, C: campus, U: urban, S: suburban, R: rural. 15
1.2 Comparison of Traffic Simulation 17

2.1 Analysis of Trajectories in the Time Domain 36
2.2 Results of Lines Dataset 47
2.3 Dataset Complexity [1] of Argoverse Dataset 48
2.4 Error of Trajectory Prediction (2 seconds of history and 3 seconds of prediction) . 49
2.5 Accuracy of linear probing 64
2.6 Results of realism metrics. Recon.-based and Retrieval-based means using the target scenario and retrieved scenarios as input for generation, respectively. . . . 65
2.7 Safety-critical scenario generation. 67
2.8 Results of human evaluation of controllability. 68
2.9 Downstream task evaluation. 68

3.1 Hyper-parameters in our experiments 79
3.2 Performance Comparison 99

4.1 Collision rate (↑) of generated safety-critical scenarios. Bold font means the best. 116
4.2 Collision rate (↑) of RL algorithms evaluated in different scenarios. 118
4.3 Environment configurations used in experiments 145
4.4 Success rate (%) for nine settings in three environments. **Bold** font means the best. 147
4.5 Results on *Chemistry* environment (GRADER/Score). Bold font means the best. . 149
4.6 Success rate (%) for Chemistry environment (ID). **Bold** font means the best. . . . 153
4.7 Success rate (%) for Chemistry environment (OOD). **Bold** font means the best. . 154
4.8 Testing reward on shifted environments. Bold font means the best reward. 185
4.9 Testing reward on nominal environments. Underline means the reward is over 0.9. 188
4.10 Influence of modules 188
4.11 Random purterbuation 189

List of Figures

1.1 Illustration of digital with broad applications including healthcare, autonomous driving, energy, and manufacturing. 2
1.2 Compared to normal scenarios, critical scenarios are rare in the real world. 3
1.3 Illustration of three types of generation methods: data-driven methods, adversarial generation methods, and Knowledge-based generation methods. 25

2.1 Procedure of generating multi-vehicle encounter trajectories using VAE model. . 28
2.2 Model structure of our multi-vehicle trajectory generator. 30
2.3 Comparison of two evaluation metrics on Autoencoder and VAE. 33
2.4 Results of three models with traffic rationality evaluation. 34
2.5 Disentanglement of MTG. 37
2.6 Disentanglement of VAE baseline. 38
2.7 Overview pipeline of proposed CMTS. 40
2.8 Model structure of the proposed CMTS during the training stage. 41
2.9 Three interpolation examples of safe trajectories and collision trajectories using the Argoverse dataset. 46
2.10 Two interpolation examples of MTG, ACAI, and CMTS on the lines dataset. . . . 47
2.11 Examples of MTG, ACAI, CVAE, and CMTS on the MNIST digit datasets. . . . 48
2.12 T-SNE results of interpolation samples. The pink and blue points represent the safe samples and the collision samples, respectively. 49
2.13 Trajectory prediction results for the six risky scenarios with CS-LSTM. 50
2.14 (a) Conventional methods make the model memorize the data distribution for generating. (b) In contrast, our method employs a retriever to query datasets (including external data obtained after training) and uses a generative model to generate scenarios by integrating the information from the retrieved scenarios. . . 52
2.15 **Left**: the training pipeline for the encoders and decoder aimed at learning latent embeddings of scenarios. A contrastive loss is applied to behavior embeddings to ensure invariance to absolute positions. **Middle**: the training pipeline for the combiner with frozen encoder and decoder parameters. We use K-Nearest Neighbors (KNN) to retrieve scenarios similar to a template scenario in the dataset and use the retrieved behaviors to reconstruct the template scenario. **Right**: the generation pipeline with a retriever and a generator. 54
2.16 Qualitative evaluation of similar and dissimilar scenarios calculated by our scenario embedding. The rectangles represent the initial poses of the vehicles and the lines represent the future trajectories. 63

2.17 (a) Scene ID accuracy using the behavior embedding with difference distance metrics. (b) A matrix shows the Wasserstein distance between scenario segments, where each block contains the segments that belong to the same Scene ID. 64
2.18 Examples of tag-retrieved scenarios generated by RealGen for six different tags. . 66
2.19 Examples of generating crash scenarios from RealGen. The shadow rectangles represent the initial positions of agents. 67
2.20 Examples of the scenarios we used for human evaluation of controllability. We used the map and initial positions of the original image (from nuScenes) to generate new scenarios using RealGen and LCTGen. We tested five scenario tags. . . 70

3.1 Two safety-critical scenarios generated using our method. 73
3.2 One example of converting an undirected graphic model to a directed one with human knowledge. 75
3.3 Explanation of the nodes in Fig .3.2. The scale and shift parameters in (3.8) are also shown in this figure. 75
3.4 The proposed framework (left) and the structural details (right). Our model consists of two parts: the state module and the action module, both of which are implemented with linear layers. 77
3.5 Learned distributions of scenario parameters with different routes. 81
3.6 Examples of conditional distribution $p(\theta|X = x, Y = y, S = s)$. 81
3.7 The results of the collision rate and the number of iterations required to reach stability. 82
3.8 Generated safety-critical scenarios using our method. 83
3.9 The training results have large diversity since the feasible solution is a continuous space rather than a single point. 85
3.10 An example of multimodal safety-critical driving scenario. 86
3.11 Diagram of proposed framework. The modules that have learnable parameters are shown in blue. 88
3.12 An illustration example of multi-modal estimation using the uniform sampler. . . 90
3.13 An illustration example of multi-modal estimation using the generator. 90
3.14 An illustration example of multi-modal estimation using our adaptive sampler. . . 91
3.15 A toy example to compare different adaptive samplers. 93
3.16 Samples from prior model $q(x)$ that is learned from InD dataset (Bock et al., 2020). Different colors represent different velocity angles. 93
3.17 Distributions of safety-critical scenarios represented by $x = [x^s, y^s, v_x^s, v_y^s]$. . . . 96
3.18 Relationship between risk and log-likelihood of $p(x)$. 97
3.19 Testing reward and testing collision rate in four different settings. 100

4.1 The top scenario is safety-critical because the view of vehicle A is blocked by B. 103
4.2 Diagram of the generation pipeline using *CausalAF*. 104
4.3 **(a)** The generation process of a BG starting from an empty graph. **(b)** CG and BG are in the example. **(c)** The implementation of masking during the generation. 107
4.4 Three causal traffic scenarios are used in our experiments. The corresponding causal graphs are shown in the upper right of each scenario. 115

4.5 Training objective of *CausalAF* and two variants under two sampling temperatures. 116
4.6 The training objectives in the Pedestrian scenario from different numbers of irrelevant vehicles. . . . 117
4.7 **Left:** The causal graph of the pick and place task. **Right:** Three testing settings: (1) In distribution (2) Spuriousness (3) Composition. . . . 121
4.8 The paradigm of GRADER. . . . 126
4.9 Environments used in experiments. . . . 145
4.10 **Left:** Test reward of the *Crash* environment. **Right:** Accuracy of causal graph discovery. . . . 149
4.11 The testing reward and causal discovery results of *Stack* environment. . . . 150
4.12 The testing reward and causal discovery results of *Unlock* environment. . . . 151
4.13 Influence of different causal graphs in *Unlock-S*. . . . 151
4.14 Reward of Chemistry environment under ID and OOD setting. . . . 152
4.15 Top: Discovered causal graph from GRADER. Bottom: true causal graph. . . . 153
4.16 Discovered causal graphs of three environments. (Action, State, Next state) . . . 154
4.17 TV distance between goal and state distributions. . . . 155
4.18 A model trained only with heavy traffic in the daytime learns the spurious correlation between brightness and traffic density and could fail to drive in light traffic in the daytime. . . . 157
4.19 The probabilistic graphs of our formulation (SC-MDP) and related formulations. s_t^1 means the first dimension of s_t. $s_{t'}$ is a shorthand for s_{t+1}. In SC-MDP, the orange line represents the backdoor path from state $s_{t'}^1$ to action $a_{t'}$ opened by the confounder c_t, which makes the learned policy π rely on the value of c_t. . . . 161
4.20 (a) RMDPs add homogeneous noise to states, while (b) RSC-MDPs perturb the confounder to influence states, resulting in a subset of the valid space. . . . 163
4.21 The illustration of the transition kernels of the standard MDP $\mathcal{M}$ and the proposed SC-MDP $\mathcal{M}_{\mathsf{sc}}$ at the first time step $t = 1$, i.e., P_t^0 and $\mathcal{P}_1$ respectively. . . . 169
4.22 Model architecture of the structural causal model. Encoder, Decoder, position embedding, and Causal Graph are learnable during the training stage. . . . 181
4.23 Illustration of tasks in the Carla simulator. . . . 182
4.24 Illustration of tasks in the Robosuite simulator. . . . 183
4.25 The first row shows the testing reward on the nominal environments, while the second row shows the testing reward on the shifted environments. . . . 186
4.26 Comparison between SAC-Sparse and our method. α is the regularization weight. 187
4.27 Performance-robustness tradeoff with different augmentation ratio β. . . . 189
4.28 The generated transition data from different perturbation methods. (a) Trajectories collected from the policy interact with the nominal environment. (b) Generated trajectories without any perturbation. (c) Generated trajectories with perturbation but without the causal graph. (d) Generated trajectories with perturbation and with the causal graph. . . . 191

5.1 Combine as much information as possible to build a critical digital twin. . . . 193
5.2 Reasoning in the data flywheel. . . . 195

List of Symbols

$\mathcal{G}$	Causal Graph
$\mathcal{X}$	Space of scenario
$\mathcal{S}$	Space of static object in scenario
$\mathcal{I}$	Space of initial status of dynamic object in scenario
$\mathcal{B}$	Space of behavior of dynamic object in scenario
A_e	Ego agent in the scenario
x	A traffic scenario
a_e^t	Action of ego agent at time step t
π_e	Policy of ego agent in the scenario
x^t	State of scenario at time step t
a_o^t	Action of other agent at time step t
π_o	Policy of other agents in the scenario
$\mathcal{D}$	A dataset of scenario
z	Latent variable in latent space
τ	Trajectory of an agent in the scenario
$\mathcal{L}$	Objective of optimization
$\mathcal{H}$	Entropy of a distribution
$\mathbf{PA}_i(\mathcal{G})$	Parent of node i in graph $\mathcal{G}$
$\mathbb{1}$	Indicator function
C	The set of constraint
$\mathcal{V}$	Node set of a graph
$\mathcal{E}$	Edge set of a graph
v	one node in the graph
e	one edge in the graph
$\mathcal{U}$	Exogenous variable set in a causal graph
T	Max step number of a trajectory
s^t	State of time step t in MDP
a^t	Action of time step t in MDP
V	The state value function in Bellman equation
Q	The action-state value function in Bellman equation
$\mathbb{D}_{TV}$	The total variance distance of probability measures
$\mathbb{D}_{\text{KL}}$	The Kullback–Leibler divergence of probability measures
$\mathcal{C}$	Space of confounder

Chapter 1

Introduction

What I cannot create, I do not understand

Richard Feynman

Artificial intelligence (AI) has seen substantial breakthroughs in areas such as dialogue generation (OpenAI, 2023), image generation (Rombach et al., 2022), and embodied systems (Driess et al., 2023). However, as AI progresses and research broadens to encompass physical products, notably robotic systems, it becomes entangled with challenges relating to safety, robustness, and open-ended environments with unforeseen events. Therefore, autonomous systems, although ultimately deployed in the real world, are typically developed in the preliminary stages within simulations due to the potential for costly damage and the slow operational pace of the physical world. To bridge this gap and ensure a close approximation between simulations and reality, digital twins (Wang et al., 2023) are designed to emulate the real world, using a high-quality, precision-engineered physical and graphical renderer.

1.1 Digital Twins, Data, and Scenarios

There are many definitions of digital twins with slight differences. In this proposal, we broadly describe digital twins as a virtual platform that can accurately simulate a process in real-world tasks. These tasks are indispensable for human life, such as autonomous driving, medical treatment, manufacturing, etc.

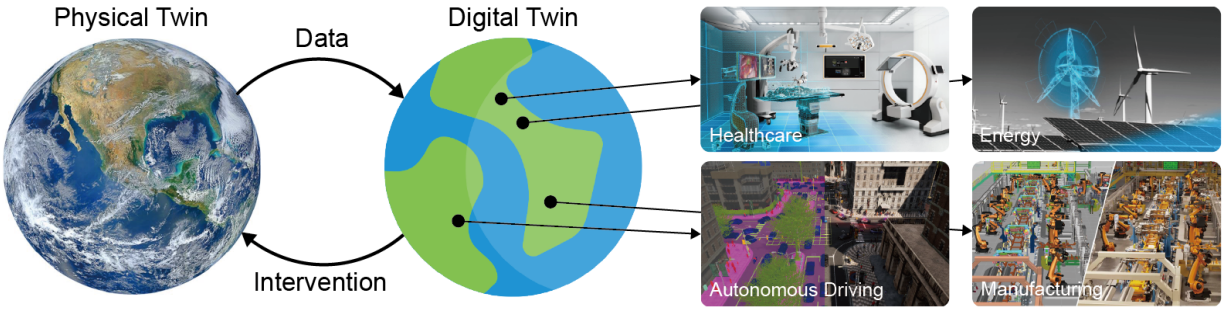


Figure 1.1: Illustration of digital with broad applications including healthcare, autonomous driving, energy, and manufacturing.

Regarding the fidelity of digital twins, initial perceptions often gravitate towards the detailed textural models of static objects, such as human faces, or the accurate inner functioning of complex objects like vehicle engines. However, the complex interaction between objects, which underpins the formation of scenarios, represents a pivotal facet of fidelity. Scenarios, with multitudinous object behaviors and inherent high dimensions of uncertainty, are complex in nature. They provide diverse training and testing grounds for autonomous systems, and any disparity between digital twin scenarios and real-world situations substantially influences system performance.

However, the task of designing scenarios that are diverse, realistic, and efficient remains a formidable challenge. The easiest approach to scenario design involves randomly sampling the position and movement patterns of objects. Although simple, this method frequently results in a surplus of invalid scenarios that contravene physical laws and basic principles, potentially creating situations that may never occur in reality. A more sophisticated alternative involves using human experts to design scenarios manually, a process that is laborious and constrained by the limitations of individual experience. To alleviate human effort, another approach involves replaying the collected data within the digital twin. However, such datasets are typically dominated by redundant scenarios, which primarily serve to test autonomous system performance under rudimentary scenarios.

So, what do critical scenarios encompass? First, these scenarios must encapsulate all real-world cases, including rare events with extremely low probability of occurrence. Second, the

categorization and parameters of the scenario should be readily controllable by users to satisfy all kinds of situations. Third, the search process for the desired scenario must be efficient. With the advent of advanced AI tools, especially deep generative models, it is now possible to design potent algorithms that can fulfill these requirements. In addition, these algorithms leverage the capabilities of digital twins to facilitate the learning process.

1.2 Critical Digital Twins

Before introducing details of scenario generation, we first provide a definition of critical digital twins, which means critical scenarios in digital twins. then we discuss the objective and metrics that are used in the generation of critical scenarios. Finally, a thorough review of the literature of related works concludes this chapter.

1.2.1 Components

To describe the environment in which the autonomy operates, we define a scenario with static and dynamic contents and also consider the behavior of dynamic objects. Formally, we have the following definition:

Definition 1 (Scenario in Digital Twin)**.** *The scenario is defined as a combination of three sets:* $x \in \mathcal{X} = \{\mathcal{S}, \mathcal{I}, \mathcal{B}\}$. $\mathcal{S}$ *represents the static environment, including the map, surrounding buildings, etc.* $\mathcal{I}$ *represents the initial condition and properties of dynamic objects (called agents).* $\mathcal{B}$ *represents the sequential behaviors of dynamic objects.*

Separating static and dynamic content improves scenario representation and generation and has shown great success in (Team et al., 2021). The geometry of the map and the static objects (e.g., traffic signs and traffic lights) create the background of the scenario, while the dynamic agents that participate in the scenario enable complex interaction between

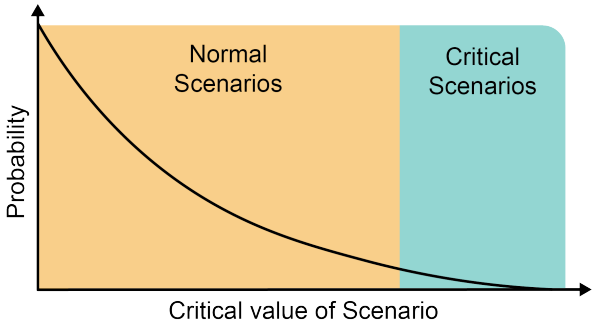


Figure 1.2: Compared to normal scenarios, critical scenarios are rare in the real world.

agents. Among all these dynamic agents, we are usually interested in a specific agent, named ego agent, that is controlled by an algorithm to be trained or evaluated:

Definition 2 (Ego agent in Scenario)**.** *The ego agent, denoted as A_e, is controlled by an algorithm provided by the user for training and evaluation.*

With the target agent A_e, we can use different metrics to describe and categorize scenarios, according to different downstream tasks. For example, in autonomous driving scenarios, we care about the safety of the ego vehicle, thus we can use the safety level of the scenario to define a safety-critical scenario:

Definition 3 (Critical Scenario)**.** *Given an evaluation metric $g(x)$ and a predefined threshold δ in the range of this metric, a critical scenario is defined as $\{x : g(x) > \delta\}$.*

We can then create critical scenarios in digital twins to make the ego agent operate under stressful conditions to evaluate the performance or find failure cases. However, designing such scenarios is not an easy task, as they have hierarchical structures and complicated components. To generate valid scenarios that satisfy the metric requirement $g(x) > \delta$, we need to use optimization methods.

1.2.2 Objective of Generation

Based on the definition of critical scenarios, we can model the generation process as an optimization problem to learn a distribution $p_\theta(x)$ with the objective:

$$\arg\max_{\theta} = \mathbb{E}_{p_\theta(x)}\left[g(x)\right], \tag{1.1}$$

where θ is the parameter of distribution $p_\theta(x)$ to be optimized.

Usually, the evaluation metric $g(x)$ involves a complex interaction between the ego agent and other components of the scenario. Therefore, we introduce the concept of action and policy to describe the dynamic behavior of all agents. For the ego agent, we define the ego action $a_e^t = \pi_e(x^t)$ that is generated by the ego policy π_e given x^t, the historical states before timestep t. Similarly, we define the action for other agents as $a_o^t = \pi_o(x^t)$. Then, we can expand the

objective of generation with the behavior of all others as:

$$\arg\max_{\theta} = \mathbb{E}_{p_\theta(x)}\left[g(x, \pi_e, \pi_o)\right]. \tag{1.2}$$

1.2.3 Metrics

The most important factor in solving the optimization problem Definition 1.2 is the metric $g(x)$. A proper risk metric makes the generated scenarios useful for discovering failure cases of the ego agent, while an improper risk metric only provides useless scenarios that either are too trivial for systems or too rare to happen in the real world.

One typical category of such a metric is safety-critical metrics that describe the level of risk or probability of unsafe events, for example, collisions, and reaching limitation of systems. In this physical world, the level of risk can be naturally described by the distance – a small distance means the risk of collision is high. This intuition can be described by Time-to-Collision (TTC) (Hayward, 1972):

$$TTC_F(t) = \frac{X_L(t) - X_F(t) - l_L}{\dot{X}_F(t) - \dot{X}_L(t)},\ \forall\ \dot{X}_F(t) > \dot{X}_L(t), \tag{1.3}$$

where X denotes the position, $\dot{X}$ denotes the derivative of X with respect to time or the speed, and l_L denotes the leading agents's length; L and F as subscripts refer to leading and following agents. Following this time-based metric, there are numerous variants of TTC such as Time Exposed TTC (TET) (Minderhoud and Bovy, 2001) and Modified TTC (MTTC) (Ozbay et al., 2008).

The other two main types of metrics are distance-based and deceleration-based metrics. The first uses the distance available to avoid a collision, for example, the Proportion of Stopping Distance (PSD) (Allen et al., 1978). The second one defines dangerous situations using the rate deceleration during an emergency, e.g., Deceleration Rate to Avoid the Crash (DRAC) (Almqvist et al., 1991). Please check (Mahmud et al., 2017) for more metrics that belong to these two types. In addition, we can also use post-hoc analysis of unsafe behavior to indicate risk, for example, the collision rate, the average distance driven out of the road, and the frequency of running stop

signs and red lights (Li et al., 2021; Ros et al., 2022).

In addition to the safety-critical metrics, we can consider robustness-critical metrics that evaluate performance under distribution shift (e.g., perturbation shift and spurious correlation), privacy-critical metrics that evaluate information leakage under attacks, and equity metrics.

1.3 Literature of Generative Models in Digital Twins

The introduction of the related work starts with the representation of scenarios that fundamentally influence the generative process. Then we discuss existing work from two optimization frameworks, including rule-based generation and learning-based generation. Finally, we will introduce critical scenario generation, which also includes work from different perspectives.

1.3.1 Representation of Scenario

Language and grammar of scenario. Similar to program language, there are also formal language and grammar to describe scenarios, which provide a unified and standard way to generate scenarios. OpenScenario from the Association for Standardization of Automation and Measuring Systems (ASAM) is widely used around the world in both academic and industrial applications. It describes complex, synchronized maneuvers that involve multiple entities, such as vehicles, pedestrians, and other traffic participants. SCENIC (Fremont et al., 2019) is another language designed for scenario generation, which contains fewer concepts than OpenScenario therefore, but includes more features of probabilistic programming languages.

Tree and graph structure of scenario. Scene trees and scene graphs are also widely used to represent static scenarios, which are good at capturing structural information. (Xiao et al., 2021a) generates point cloud sequential datasets by minimizing the gap between real-world LiDAR and simulation data. Similarly, Meta-sim (Kar et al., 2019) and Meta-Sim2 (Devaranjan et al., 2020) try to minimize the sim-to-real gap to reconstruct traffic scenarios for automatic labeling. Their main contribution is that they use a scene graph to represent the scenario, which is a hierarchical structure that makes the generation more efficient. In (Savkin et al., 2021), the scene graph is also used to generate image scenarios.

Bayesian Networks. Bayesian Networks is a probabilistic graphical model that uses nodes to represent objects and edges to represent the relation between nodes. This structured model can naturally describe the objects in the scenario. (Wheeler et al., 2015) uses Dynamic Bayesian Network (Ghahramani, 1997) to model complex traffic scenarios, and (Wheeler and Kochenderfer, 2016) uses factor graphs to model driving behaviors between multiple vehicles. In addition to using dynamic Bayesian networks or factor graphs to model the structures of traffic participants, the Gaussian process (GP) is a powerful non-parametric method to model the distribution of sequential scenarios (Huang et al., 2018b). Following this direction, (Guo et al., 2019b) combines the GP and Dirichlet process to build a model with an infinite number of clusters to discover traffic primitives (Wang and Zhao, 2018), which can be combined to create new scenarios.

1.3.2 Rule-based Scenario Generation

An intuitive way of generation is to sample from a collected dataset to reproduce the scenario from the road test log. Such sampling could be augmented by clustering and random perturbation. Moving one step forward, we can extract scenario templates from the dataset and use logic to represent them. During generation, we can uniformly sample the important parameters in the scenario templates.

Data replay. Most autonomous driving companies maintain scenario bases to store the scenarios they recognize as important (Webb et al., 2020). During this process, one crucial step is the tools that can automatically convert the scenario from data to virtual simulations (van der Made et al., 2015). Also, efficiently selecting the critical scenario from a huge number of scenarios is another problem. (Knies and Diermeyer, 2020) extracts useful scenarios according to cooperative actions for the evaluation of cooperative maneuver planning. In (Arief et al., 2018), the authors developed a method to select a testing site to accelerate the evaluation of the performance of the AVs on public streets, where the main contribution is describing the risk intensities of the traffic system in an area of interest with the non-homogeneous Poisson process model (Pham and Zhang, 2003).

Clustering. To better categorize the collected scenario, (Kruber et al., 2018) and (Kruber et al., 2019) propose to use unsupervised clustering methods to group similar scenarios, helping

to improve the efficiency of AV testing in a specific type of scenario. However, clustering the entire scenario might be inefficient and inaccurate, since the scenarios are usually complex and composed of finite building blocks. In (Wang and Zhao, 2018), the concept of *Traffic Primitive* to represent those blocks. They use the Hierarchical Dirichlet Process Hidden Markov Model (HDP-HMM) (Fox et al., 2008), a nonparametric Bayesian learning method, to unsupervised extract primitives from scenarios to better cluster similar scenarios. In (Wulfe et al., 2018), Importance Sampling (IS) is used to sample driving scenarios represented by Bayesian Networks. In (Wheeler and Kochenderfer, 2019), the authors try to find as many risky scenarios as possible and cluster them. They first uniformly sample from the clusters and then use IS to sample the specific scenarios represented by factor graphs.

Random perturbation. The main obstacle to direct conversion is that the diversity of scenarios is limited. Therefore, recent work by some leading companies has begun to use random perturbation to increase the number of scenarios. Baidu creates a physical-based LiDAR model (Fang et al., 2020) to transfer a dense point cloud to match the line-style output of the LiDAR sensor. By randomly placing the computer-assisted design (CAD) model of vehicles and pedestrians, their algorithms can generate a huge number of scenarios. Similarly, Uber builds a more precise LiDAR model (Manivasagam et al., 2020) using neural networks (NN) to mimic the reflection details of real-world sensors. In addition to the high-dimensional representation, Waymo tries to reconstruct more fatal crashes from the collected data by randomly perturbing important parameters (Scanlon et al., 2021).

Pre-defined logics. Driving scenarios are very common in daily life, thus we humans can easily design scenarios with predefined rules and specific conditions to trigger events. (Rana and Malhi, 2021) uses prior knowledge to design random risk scenarios with equations. The authors also conduct interventional experiments by training RL agents in different scenarios to get comparable results. (Menzel et al., 2018) focuses on functional and logical scenarios, which can be represented by natural language from human experts. (Bagschik et al., 2018) views ontologies as knowledge-based systems in the field of AV and proposes a generation of traffic scenes in natural language as a basis for the creation of scenarios. In (Fremont et al., 2020), the authors combine the formal specification (Lamsweerde, 2000) of scenarios and safety properties

to generate test cases from formal simulation.

Platforms that support pre-defined logics. There are also a large number of works that build the entire platform with predefined scenarios implemented. A 2D platform named SMARTS is developed in (Zhou et al., 2020) containing multiple diverse behavior models using both rule-based and learning-based models. (Li et al., 2021) proposes MetaDrive, a 3D simulator that supports different road shapes defined by users or directly imported from existing real-world datasets (e.g. Argoverse (Chang et al., 2019a)). (Ros et al., 2022) is a competition built on top of CARLA (Dosovitskiy et al., 2017) and (Contributors, 2019), which consists of a large number of pre-defined scenarios. In (drive Contributors, 2021), the authors build a rule-based scenario zoo in CARLA (Dosovitskiy et al., 2017), which shares some similar scenarios with (Contributors, 2019). To manage complex traffic scenarios with hundreds of objects, SUMMIT (Cai et al., 2020) is specifically designed for generating massive mixed traffic with an autopilot algorithm. To explore the causality between vehicles in the scenario, CausalCity (McDuff et al., 2021) is developed to evaluate causal discovery algorithms, which build an agency mechanism to define high-level behaviors. Safebench (Xu et al., 2022) is a critical scenario library built on Carla, which supports different types of critical scenario generation algorithms.

1.3.3 Learning-based Scenario Generation

Consider the scenarios following a distribution, we can use collected data to learn a density model to approximate this distribution. We divide this type of algorithm into three categories according to the density model that they use.

Deep Learning Models. Deep learning models are broadly introduced into generation in (Tan et al., 2021) and (Wen et al., 2020). SceneGen (Tan et al., 2021) inputs the current state of the AV and a high definition (HD) map to a Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) module to sequentially generate the trajectory of surrounding vehicles and pedestrians. They train their model with normal traffic data since their objective is to generate naturalistic scenarios. (Wen et al., 2020) proposes a quite complex system to generate scenarios in a simulator, which uses the Convolution Neural Network (CNN) (O'Shea and Nash, 2015) as a selector to generate agents surrounding the AV. In TrafficSim (Suo et al., 2021), both Gated

Recurrent Unit (GRU) and CNN are used to learn the behaviors of multi-agents from real-world data. This method can generate realistic multi-agent traffic scenarios. TrafficGen (Feng et al., 2023a), separates the generation of initial positions and vehicle movements, providing a more flexible traffic model. Recently, the trend of using large language models (LLMs) for generation has become popular. Both CTG++ (Zhong et al., 2023a) and LCTGen (Tan et al., 2023) use language as conditions and use LLMs as an interpreter to guide the generative model to generate the corresponding scenarios.

Deep Generative Models (DGMs). Recently, DGMs have shown great success in generating image and voice data. There are five types of modern generative models: Generative Adversarial Nets (GAN) (Goodfellow et al., 2014), Variational Auto-encoder (VAE) (Kingma and Welling, 2013), Autoregressive Models (Oord et al., 2016; Van Oord et al., 2016), Flow-based model (Chen et al., 2018; Dinh et al., 2016), Diffusion model (Ho et al., 2020; Song et al., 2020). The readers can find more details about DGMs in this survey (Harshvardhan et al., 2020). In (Feng et al., 2022), an auto-encoder structure is designed to separately generate vehicle initial positions and vehicle trajectories. With the power of VAE, (Ding et al., 2018) learns a latent space of encounter trajectories and generates unseen scenarios by sampling the latent space. However, with less understanding of the latent code, the generation is not controllable. As an improvement, the authors of (Ding et al., 2020a) propose CMTS, which combines normal and collision trajectories to generate safety-critical scenarios by performing interpolation in latent space. Regarding the usage of GAN, (Håkansson and Wall, 2021) introduces recurrent models to generate realistic scenarios of lane-change. They use real-world data in the discriminator to help improve the generator. The advantage of DGMs is that they can learn a low-dimensional latent space of high-dimensional and structured data using NNs. Therefore, we can easily generate high-dimensional sensing scenarios. SurfelGAN is proposed in (Yang et al., 2020) to directly generate point-cloud data to represent scenarios from the point of AV. (Chen et al., 2021b) can add new vehicles to collected driving videos to generate realistic video scenarios, where they also consider the motion planning of vehicles. (Ehrhardt et al., 2020) generates traffic videos with multi-object scene synthesis using a GAN framework. To make the video realistic, they integrate physical conditions into the generation. In (Li et al., 2019), a scenario simulator driven by

data is designed to generate both LiDAR and trajectory data to increase the diversity of driving scenarios.

Adaptive Stress Testing. Finally, there is a series of works called Adaptive Stress Testing (AST) that explores different ways of generating stress-testing scenarios. (Lee et al., 2015) uses Monte Carlo tree search (MCTS) to search action of the testing scenario, but this method does not target autonomous driving systems. (Koren et al., 2018) generates a scenario that controls a pedestrian to cross the road. (Koren and Kochenderfer, 2019) improves the last paper by using LSTM to generate initial conditions and actions at each step. Instead of defining heuristic reward functions, (Koren and Kochenderfer, 2020) leverage the Go-Explore framework to find failure cases. (Koren et al., 2021) extends previous work to high-fidelity simulation and changes the learning algorithm to Proximal Policy Optimization (PPO) (Schulman et al., 2017).

Imitation learning. Another way to learn the distribution of the dataset is by using the imitation learning (IL) method (Hussein et al., 2017), which takes the observation as input and directly controls agents. IL training is the same as supervised learning. After training, the IL model (Bojarski et al., 2016; Chen et al., 2019; Ho and Ermon, 2016) can reproduce the same behavior as real-world agents when the model encounters the same observation. However, when the observation is not covered by the dataset, the behavior of the model could be unreasonable and unpredictable.

1.3.4 Critical Scenario Generation

All previous methods focus on general scenario generation to cover the normal and common cases. However, generating normal data provides limited benefit as that part is already well-covered by the real-world data. Therefore, some recent literature explores the topic of critical data generation, especially from the adversarial learning perspective. Adversarial generative models attack a given system to find the worst failure case, which to some extent is consistent with critical scenario generation.

Importance sampling. A series of works on generating critical and rare scenarios in the context of Importance Sampling (IS) also appear in the literature. (Zhao et al., 2015) uses heuristic approaches to generate dangerous lane-changing scenarios. (Zhao et al., 2018, 2016) constructs

IS distribution to sample dangerous AV lane-changing and car-following scenarios. (Wang et al., 2019b) combines reachability analysis and IS to accelerate the evaluation. (Huang et al., 2017) uses piece-wise models to design a more expressive IS distribution. (O'Kelly et al., 2018) extends the simulation of rare events using IS to end-to-end driving algorithms. (Huang et al., 2018c) uses the Gaussian mixture model (GMM) for the IS distribution and further analyzes the efficiency of the GMM-based IS distribution for random forest and NN classifiers (Huang et al., 2018d). ReLU-activated NNs are considered in (Arief et al., 2020) to estimate the dangerous set and compute an IS estimator for a risk upper bound for the Gaussian case, with a more general case presented in (Arief et al., 2021a). The adaptive IS approach is used to construct adversarial environments to accelerate policy evaluation (Xu et al., 2021b). (Sinha et al., 2020) proposes Neural Bridge Sampling based on the adaptive multilevel splitting (Cérou and Guyader, 2007). (Feng et al., 2023b) inject IS into the reinforcement learning framework to density the ratio of safety-critical scenarios.

Adversarial generation of static scenarios. (Jain et al., 2019) learns the poses of vehicles and uses a differentiable render to get the first-point-view images to attack object detection algorithms. (Prakash et al., 2019) and (Khirodkar et al., 2018) extend Domain Randomization methods (Tobin et al., 2017) to adversarial generation. (Prakash et al., 2019) proposes Structured Domain Randomization, which uses a Bayesian network to actively generate vehicle poses. (Khirodkar et al., 2018) proposes Adversarial Domain Randomization, which targets the parking lot scenario. All of the above methods focus on image generation, but some works generate point cloud scenarios. (Tu et al., 2020) puts an adversarial object on the top of a vehicle and optimizes the shape of the object to make the vehicle disappear in the LiDAR detection algorithms. Similarly, (Abdelfattah et al., 2021) shares the same idea but uses both LiDAR and image information. After optimizing the shape of the object, (Cao et al., 2019) uses 3D printing to build the object in the real world. The experiment shows that the object on the road is indeed ignored by the detection algorithms. In addition to using renderers, (Zhu et al., 2021) tries to directly add new points to the existing point cloud to attack segmentation algorithms.

Adversarial generation of initial condition in dynamic scenarios. The first type is controlling the initial conditions of the scenario (e.g., initial velocity and spawn position) or providing

the entire trajectory at the beginning. The advantage is the low dimension of search space and the small computational resource required. (Ghodsi et al., 2021) generates a set of possible driving paths and identifies all possible safe driving trajectories that can be taken starting at different times. Similarly, (Wang et al., 2021a) and (Hanselmann et al., 2022) directly optimize the existing trajectories to perturb the driving paths of the surrounding vehicles. They use Bayesian Optimization (Frazier, 2018) for the optimization, and the scenario is represented with a point cloud. To generate real-world traffic scenarios, (Rempe et al., 2022) optimizes the adversarial trajectory in the latent space of a VAE model. Besides controlling the dynamic objects, some work also focuses on searching the weather parameters (e.g., sun and rain) to create different scenarios. (Ruiz et al., 2018) searches the type of weather using the REINFORCE (Williams, 1992) algorithm. (O'Kelly et al., 2018) uses Generative Adversarial Imitation Learning (Ho and Ermon, 2016) to generate weather parameters and uses the cross-entropy method for efficient scenario search.

Adversarial generation of policy in dynamic scenarios. The second type is building a policy model to sequentially control the dynamic objects, which contains the largest number of existing works. This type is usually formulated as a Reinforcement Learning (RL) problem (Sutton and Barto, 2018), where the AV belongs to the environment and the generator is the agent we can control. Intuitively, we have much more flexibility in this setting, but the complexity also increases. Since the AV and objects in scenarios interact stepwise, this problem can be formulated in an RL framework, and there are lots of works using RL methods. (Feng et al., 2021) and (Sun et al., 2021) use Deep Q-Network to generate discrete adversarial traffic scenarios. (Kuutti et al., 2020) uses Advantage Actor-Critic (A2C) (Mnih et al., 2016) to control a surrounding vehicle in car-following scenarios. (Chen et al., 2021a) uses Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al., 2015) to generate adversarial policy to control surrounding agents to generate lane-changing scenarios. (Wachi, 2019) uses Multi-agent DDPG (Lowe et al., 2017) to control two surrounding vehicles (which are called Non-player Characters) to attack the ego vehicle. This method also sets auxiliary goals for non-player characters (NPC) to avoid generating unrealistic scenarios. (Nonnengart et al., 2019) proposes CriSGen that uses constraint-based optimization and (Abeysirigoonawardena et al., 2019) uses Bayesian optimization.

Constraint optimization Another way of using explicit knowledge is resorting to the constraint optimization framework. We know that safety-critical scenarios are extremely rare in the real-world log, and random augmentation could be inefficient in generating safety-critical scenarios. One of the heuristic risk metrics is the drivable area for the autonomous vehicle. (Klischat and Althoff, 2019) and (Althoff and Lutz, 2018) minimize the drivable area by controlling the surrounding vehicles with an evolutionary method and constraint optimization methods, respectively. To explore more different types of scenarios, (Rocklage et al., 2017) generates the motion of other traffic participants with a backtracking search. To make the scenarios diverse, (Klischat et al., 2020) build a pipeline that introduces the road topology of OpenStreetMap (Bennett, 2010). Using the safety-critical scenarios from (Klischat et al., 2020), (Wang et al., 2021c) designs a comprehensive open-source toolbox to train and evaluate RL motion planners for AVs with user-customized configuration. To obtain a robust trajectory prediction model, (Zhang et al., 2022) and (Cao et al., 2022) generate adversarial trajectory by perturbing existing trajectory with feasible constraints.

1.3.5 Dataset and Tools for Scenario Generation

In this section, we introduce the tools that are useful for safety-critical scenario generation. We first discuss the scenario datasets, then turn to the traffic simulators. Finally, we review existing platforms that support the function of scenario generation.

Scenario Dataset

For modern machine learning methods, datasets are crucial and necessary. Specifically, for the scenario generation task, there are also many datasets published by companies and academic institutes. In Table 1.1, we summarize and compare existing scenario datasets in various aspects that we are especially interested in.

Fidelity. In Table 1.1, we include datasets that are collected from sensors on public roads and virtual worlds simulated by traffic simulators. Collecting real-world data is time-consuming and requires humans to operate vehicles or drones to record data in the real world in a variety of environments. However, these data are more representative of the real-world data distribution.

Table 1.1: Scenario Datasets. We list existing datasets and different aspects that we are interested in. ✓/× in the weather column means the dataset contains/doesn't contain data under various weather conditions. BEV: bird's-eye view, FPV: first-person view. D: daytime, N: nighttime. H: highway, I: intersection, RA: roundabout, C: campus, U: urban, S: suburban, R: rural.

Dataset	Real	View	Data Sensor				Annotation			Traffic Condition			
			Image	LiDAR	RADAR	Traj.	3D	2D	Lane	Weather	Time	Region	Jam
CamVid (Brostow et al., 2009)	✓	FPV	RGB	×	×	×	×	✓	✓	×	D	U	×
KITTI (Geiger et al., 2013)	✓	FPV	RGB/Stereo	✓	×	✓	✓	✓	✓	×	D	U/R/H	×
Cyclists (Li et al., 2016)	✓	FPV	RGB	×	×	×	×	✓	×	×	D	U	×
Cityscapes (Cordts et al., 2016)	✓	FPV	RGB/Stereo	×	×	✓	✓	✓	×	×	D	U	✓
SYNTHIA (Ros et al., 2016)	×	FPV	RGB	×	×	×	✓	✓	×	✓	D/N	U	✓
Campus (Robicquet et al., 2016)	✓	BEV	RGB	×	×	✓	×	✓	✓	×	D	C	×
RobotCar (Maddern et al., 2017)	✓	FPV	RGB	✓	×	✓	✓	✓	×	✓	D/N	U	×
Mapillary (Neuhold et al., 2017)	✓	FPV	RGB	×	×	×	×	✓	✓	✓	D/N	U	×
P.F.B. (Richter et al., 2017)	×	FPV	RGB	×	×	✓	✓	✓	×	✓	D/N	U	×
BDD100K (Yu et al., 2020)	✓	FPV	RGB	×	×	✓	×	✓	✓	✓	D	U/H	×
HighD (Krajewski et al., 2018)	✓	BEV	RGB	×	×	✓	×	✓	×	×	D	H	✓
Udacity (team, 2018e)	✓	FPV	RGB	×	×	×	×	✓	×	×	D	U	×
KAIST (Choi et al., 2018)	✓	FPV	RGB/Stereo	✓	×	✓	×	✓	×	×	D/N	U	×
Argoverse (Chang et al., 2019b)	✓	FPV	RGB/Stereo	✓	×	✓	✓	×	✓	✓	D/N	U	×
TRAF (Chandra et al., 2019)	✓	FPV	RGB	×	×	✓	×	✓	×	✓	D/N	U	×
ApolloScape (Huang et al., 2019)	✓	FPV	RGB/Stereo	✓	×	✓	✓	✓	✓	✓	D	U	✓
ACFR (Zyner et al., 2019)	✓	BEV	RGB	×	×	✓	×	✓	×	×	D	RA	×
H3D (Patil et al., 2019)	✓	FPV	RGB	✓	×	✓	✓	×	×	×	D	U	✓
INTERACTION (Zhan et al., 2019)	✓	BEV	RGB	×	×	✓	×	✓	✓	×	D	I/RA	×
InD (Bock et al., 2020)	✓	BEV	RGB	×	×	✓	×	✓	×	×	D	I	×
RounD (Krajewski et al., 2020)	✓	BEV	RGB	×	×	✓	×	✓	×	×	D	RA	×
nuScenes (Caesar et al., 2020)	✓	FPV	RGB	✓	✓	✓	✓	✓	×	✓	D/N	U	×
Lyft Level 5 (Houston et al., 2020)	✓	BEV	RGB	✓	✓	✓	×	✓	✓	×	D	S	×
Waymo Open (Sun et al., 2020)	✓	FPV	RGB	✓	×	✓	✓	✓	✓	✓	D/N	U/S	×
A*3D (Pham et al., 2020)	✓	FPV	RGB	✓	×	×	✓	✓	×	✓	D/N	U	✓
RobotCar Radar (Barnes et al., 2020)	✓	FPV	RGB	✓	✓	✓	×	×	×	✓	D/N	U	×
Argoverse 2 (Wilson et al., 2023)	✓	FPV	RGB/Stereo	✓	×	✓	✓	×	✓	✓	D/N	U	×
PandaSet (Xiao et al., 2021b)	✓	FPV	RGB	✓	×	✓	✓	×	×	×	D/N	U	×
ONCE (Mao et al., 2021)	✓	FPV	RGB/Stereo	✓	×	✓	✓	✓	×	✓	D/N	U	×

Models developed with these realistic data can be applied directly to the real world. Synthetic datasets, on the other hand, are simple to collect. However, they are highly dependent on the authenticity of traffic simulators, which usually do not accurately mimic and render real-world data.

Collect View. In addition to different levels of reality, we also present datasets with different views. For example, the HighD (Krajewski et al., 2018), InD (Bock et al., 2020), and RounD (Krajewski et al., 2020) datasets collect data in bird's-eye view (BEV), which is recorded from drone cameras. KITTI (Geiger et al., 2013) and Argoverse (Wilson et al., 2023) datasets collect data in first-person view (FPV), which are captured by cameras in front of a car. The BEV data are collected in a fixed region; therefore, it is more useful to analyze the behavior of objects on a fixed background. In contrast, FPV data are more suitable for training AV algorithms that take egocentric information.

Data Sensor. For each dataset, we examine whether it has the following data types: RGB image, stereo image, LiDAR data, Radar, and trajectory. All datasets included in the table have RGB images since it is a common data type in traffic scenarios. RGB images are usually collected by cameras mounted on vehicles or drones, and they can be utilized for a variety of computer vision tasks, such as object detection and image segmentation. Stereo images are captured by stereo cameras, and LiDAR data is collected by LiDAR sensors. Both provide 3D information that is particularly useful in 3D tasks such as 3D object detection. RADAR is also a common sensor that returns 3D information similar to that of LiDAR, but at a cheaper price. It can work in harsher conditions (e.g., rain and storm) because of its longer wavelength compared to LiDAR's lights. The trajectory data are either recorded by sensors such as GPS or converted from object tracking. This type of data is frequently used in trajectory prediction tasks for planning and control purposes.

Annotation Type. The availability of various types of annotation is crucial for each dataset, as it determines the tasks for which the data set can be used. We explore three forms of data annotations in Table 1.1: 3D object annotations, 2D object annotations, and lane annotations. 3D annotations can be divided into two categories: 3D bounding-box annotations and 3D point-cloud annotations. A 3D bounding box annotation describes a cube that exactly holds one specific

Table 1.2: Comparison of Traffic Simulation

Simulator	Year	Open Source	Realistic Perception	Customized Scenario	Back-end	Map Source		API Support		
						Real World	Human Design	Python	C++	ROS
TORCE Wymann et al. (2000)	2000	✓	✓	×	None	×	✓	×	✓	×
Webots Michel (2004)	2004	✓	✓	✓	ODE	✓	✓	✓	✓	✓
CarRacing Team (2016)	2016	✓	×	×	None	×	✓	✓	×	×
CARLA Dosovitskiy et al. (2017)	2017	✓	✓	✓	UE4	×	✓	✓	✓	✓
SimMobilityST Azevedo et al. (2017)	2017	✓	×	✓	None	×	✓	✓	×	×
GTA-V Richter et al. (2017)	2017	×	✓	✓	RAGE	×	×	×	×	×
highway-env Leurent (2018)	2018	✓	×	✓	None	×	✓	✓	×	×
Deepdrive team (2018b)	2018	✓	✓	✓	UE4	×	✓	✓	✓	×
esmini team (2018c)	2018	✓	✓	✓	Unity	×	✓	✓	✓	×
AutonoViSim Best et al. (2018)	2018	×	✓	✓	PhysX	×	✓	×	×	×
AirSim Shah et al. (2018)	2018	✓	✓	✓	UE4	×	✓	✓	✓	✓
SUMO Lopez et al. (2018)	2018	✓	×	✓	None	✓	✓	✓	✓	×
Apollo team (2018a)	2018	✓	×	✓	Unity	×	✓	✓	✓	×
Sim4CV Müller et al. (2018)	2018	✓	✓	✓	UE4	×	✓	✓	✓	×
SUMMIT Cai et al. (2020)	2020	✓	✓	×	UE4	✓	✓	✓	×	✓
MultiCarRacing Schwarting et al. (2020)	2020	✓	×	×	None	×	✓	✓	×	×
SMARTS Zhou et al. (2020)	2020	✓	×	✓	None	×	✓	✓	×	×
LGSVL Rong et al. (2020)	2020	✓	✓	✓	Unity	✓	✓	✓	×	✓
CausalCity McDuff et al. (2021)	2021	✓	✓	✓	UE4	×	✓	✓	×	×
MetaDrive Li et al. (2021)	2021	✓	✓	✓	Panda3D	✓	✓	✓	×	×
L2R Herman et al. (2021)	2021	✓	✓	✓	UE4	✓	✓	✓	×	×
AutoDRIVE Samak et al. (2021)	2021	✓	✓	✓	Unity	×	✓	✓	✓	✓

object. 3D point cloud annotations assign point-wise labels to each point in the point cloud, indicating the point's category. Many tasks in autonomous driving, such as 3D object detection and 3D segmentation, require 3D annotations. Similarly, 2D annotations include 2D bounding box annotations and pixel-wise 2D semantic annotations. Lane annotations describe different types of lanes in the data, as well as the boundaries of drivable regions. This lane information can be used to integrate map and traffic rule information into the AV algorithm, enabling more efficient decision-making functions.

Traffic Condition. Datasets with a limited level of diversity in conditions and situations are skewed to redundant and highly safe scenarios, which leads to the long tail problem (Cui et al., 2019; Lee et al., 2019). To evaluate the performance of AVs in various scenarios, a dataset must include data in a variety of settings. We consider mainly the following four key aspects: weather, time, region, and traffic density. The weather conditions change the entire world, as well as the style of the data collected by the sensors. For example, the nuScenes dataset (Caesar et al., 2020) includes data from sunny, rainy, and cloudy conditions. As a result, the images have different appearances depending on the weather. The time condition specifies whether the data was obtained during the day or at night. The main distinction between these two scenarios is the lighting. The region denotes the location from which the data is gathered. For example, the INTERACTION dataset (Zhan et al., 2019) is collected at intersections and roundabouts, while the KITTI dataset (Geiger et al., 2013) is collected in urban, rural, and highway settings. Finally, the density of traffic considers the number of objects in traffic scenarios. The higher density indicates a traffic jam or congestion, which requires traffic participants to pay more attention to surrounding objects and take actions more carefully.

Traffic Simulation Platforms

In Table 1.2, we summarize existing traffic simulators and compare them in different aspects that we are particularly interested in.

Open Source. Open-source simulators are easily customized, allowing users to easily design and evaluate various safety-critical scenarios, as these scenarios are typically rare and sophisticated and require a high level of customization. Therefore, whether the simulator is open source

becomes one of our primary considerations. Most of the simulation platforms we list in Table 1.2 are open source. Simulators such as AirSim (Shah et al., 2018) and SMARTS (Zhou et al., 2020), for example, release their source code to the public, making it easier for users to modify and enrich the testing environment.

Realistic Perception. The realism and fidelity in virtual simulation have a significant impact on AV algorithms. High-fidelity, photorealistic simulators provide data that is similar to the physical world, allowing for a more accurate assessment of AV performance in the real world. For example, SUMMIT (Cai et al., 2020) can simulate 3D towns with a large number of vehicles, pedestrians, and buildings. In addition, weather conditions can be further simulated, including controlling the strength of precipitation, cloudiness, fog density, etc. Simple traffic simulators, on the other hand, are incapable of supporting such detailed simulations. However, they are usually light-weighted and easy to use (especially for RL algorithms), where algorithms can be tested quickly without complicated configurations. For example, Highway-env (Leurent, 2018) is a 2D simulator that can be installed using only one command and provides preliminary experimental results in a short amount of time.

Customized Scenario. The freedom to customize scenarios is of great importance since we mainly focus on the generation and evaluation of safety-critical scenarios. Adaptation to scenarios usually involves the modification of the positions, speeds, and behaviors of vehicles. For example, CARLA (Dosovitskiy et al., 2017) offers a systematic way for users to define a scenario, through which users can specify the number of vehicles in a town and their own behaviors. The weather and surrounding environment can even be arbitrarily adjusted to create variations in the visual appearance of the scene. These functions provide more flexibility to users and allow for more comprehensive testing of the reliability of AVs.

Back-end Engine. The simulator engines have a direct impact on the fidelity of the dynamics of the simulated vehicle and the rendered 3D environment. Most simulation platforms are built on Unreal Engine 4 (UE4) [1] or Unity [2], which are popular and professional game engines. Other less popular engines, such as Panda3D (Goslin and Mine, 2004), are also considered. Some nonrealistic environments, such as Highway-env (Leurent, 2018), do not even need back-

[1] https://www.unrealengine.com
[2] https://www.unity.com

end engines due to the low rendering requirement. A simple graphical user interface (GUI) is sufficient for such a type of platform.

Map Source. Maps are critical components in AV testing systems. The maps in these traffic simulators are created by humans or based on real-world data. For example, the Learn-to-Race (L2R) (Herman et al., 2021) platform has three racetracks in its racing simulator, all of which are based on real-world racetracks. Human-designed maps can be further classified into two types: rule-based maps and procedurally generated maps. Highway-env (Leurent, 2018) incorporates 11 rule-based maps, including highways, parking lots, roundabouts, etc., all of which are written manually by humans. MetaDrive (Li et al., 2021) maintains several basic roadblocks and generates numerous maps in a procedural manner by randomly selecting one roadblock at a time.

API Support. API support for specific programming languages is crucial in large-scale automated evaluation, since it allows users to run batches of scenarios. Popular programming languages like Python and C++ are supported by most simulation platforms. Some simulators such as CARLA (Dosovitskiy et al., 2017) and LGSVL (Rong et al., 2020) also provide support for the Robot Operating System (ROS) [3], allowing users to integrate other open-source modules developed by the ROS community.

Scenario Design Platform. There are several user-friendly platforms that support scenario design, which already implements many rule-based scenarios that are normal or safety-critical. We discuss some popular platforms below.

CARLA Scenario Runner (Contributors, 2019) provides traffic scenario definitions and an execution engine for CARLA. Scenarios can be defined through a Python interface that allows users to easily describe sophisticated and synchronized maneuvers that involve multiple entities such as vehicles, pedestrians, and other traffic participants. It also supports the OpenSCENARIO (Jullien et al., 2009) standard file format for scenario descriptions, making it simple and efficient to incorporate a variety of existing traffic scenarios from the community.

SCENIC (Fremont et al., 2022) defines a language for the specification and generation of scenarios. It describes distributions over scenes and the behaviors of their agents over time. One advantage of SCENIC over other scenario languages is that it combines the concise and readable

[3] https://www.ros.org

syntax for spatio-temporal relationships with the ability to impose hard and soft constraints over the scenario.

SafeBench (Xu et al., 2022) is an open source platform focusing on systematically evaluating the safety and robustness of autonomous driving algorithms based on various testing scenarios and comprehensive evaluation metrics. The platform integrates eight types of safety-critical scenarios and incorporates four generating algorithms. Users can also design their own traffic scenarios and scenario generation algorithms following the instructions. SafeBench also provides several RL-based autonomous driving algorithms with pre-trained RL model weights. Users can easily test and improve the generated scenarios based on feedback from various autonomous driving algorithms.

DI-Drive Casezoo (drive Contributors, 2021) consists of a set of scenarios used to train and evaluate the diving policy in a simulator. Similar to the CARLA Scenario Runner (Contributors, 2019), DI-drive Casezoo has a routing scenario and multiple single scenarios that can be triggered along the route. There are 18 route scenarios and 8 types of single scenario that can be triggered depending on the route definition. Route scenario is defined in an XML file with its corresponding scenarios. The trigger locations along the route are defined in JSON files. A single scenario is defined in a Python file that describes the behaviors of traffic participants.

SUMO NETEDIT (team, 2018d) is a graphical scenario editor that can be used to create traffic networks from scratch and modify all aspects of existing networks, including basic network elements (junctions, edges, and lanes), advanced network elements (e.g., traffic lights), and additional infrastructure (e.g., bus stops). This tool is specifically designed for SUMO (Lopez et al., 2018), which generates mainly large-scale traffic conditions without high-fidelity rendering.

SMARTS Scenario Studio is a scenario design tool on the SMARTS (Zhou et al., 2020) platform that supports flexible and expressive scenario specification. Scenario definitions are written in the Domain Specific Language, which describes the traffic environment, such as traffic vehicles, routes, and agent missions. Scenario Studio also supports SUMO NETEDIT configuration files (team, 2018d). Maps edited by NETEDIT (team, 2018d) can be easily included and reused in Scenario Studio, which enriches the training and testing environments in the SMARTS platform.

CommonRoad (Wang et al., 2021c) is a simulator and an open source toolbox to train and evaluate RL-based motion planners for AVs. Scenario configurations are written as XML files. Users can read, modify, visualize, and store their own traffic scenarios using the Python API provided by CommonRoad. In addition, CommonRoad also supports more scenario specifications, such as Lanelet2 (Poggenhans et al., 2018) and OpenSCENARIO (Jullien et al., 2009).

1.4 Challenge

Generating critical scenarios could be a hard problem due to many constraints and required properties. In this section, we identify five main challenges that cover the difficulty of the generation process. Digging into the details of these challenges also helps us to discover potential directions that can improve existing algorithms. The five challenges are as follows.

- **Fidelity.** The ultimate goal of scenario generation is to develop safe AVs that can run in the real world. Therefore, it is useless to make AVs pass difficult but unrealistic scenarios. We need to ensure that generated scenarios have the chance to happen in real traffic situations.
- **Efficiency.** Critical scenarios are extremely rare in the real world. The generation needs to consider the efficiency and increase the density of the scenarios we are interested in.
- **Diversity.** Critical scenarios are also diverse. The generation algorithm should be able to discover and generate as many different critical scenarios as possible.
- **Adaptivity.** Scenarios are dynamic due to the interaction between the AVs and their surrounding objects. The scenarios we generate should be variable for different AVs rather than targeting one specific AV.
- **Controllability.** In most times, we want to reproduce or repeat specific scenarios rather than random ones. The generative model should be able to follow instructions or conditions to generate corresponding scenarios.

We will discuss more details of the above five perspectives in the following sections and show that the combination of the previous three types of generation methods could be very promising ways to solve those challenges.

Fidelity. The generation algorithm can create infinite scenarios, but not all of them can occur in the real world. In particular, under the adversarial generation framework, the searched scenarios are likely to violate basic traffic rules. The intuitive way to avoid this problem is to add constraints during the generation, but sometimes the constraints are not easy to define. Another promising direction is to combine real-world data and adversarial generation, where real-world data can be used as a priori distribution or constraints. Metrics such as Kullback–Leibler divergence (Kullback and Leibler, 1951) and Wasserstein distance (Kantorovich, 1960) can be used to minimize the gap between generated and real-world scenarios.

The fidelity of the scenario is also reflected by the high-dimensional sensing data, which usually requires powerful DGMs to generate. The state-of-the-art methods mainly focus on the generation of static images, such as faces. Recently, Neural Radiance Fields (NeRF) (Mildenhall et al., 2020) have been popular for visual scene generation. This method uses NNs to learn the ray-casting functions and then output different views of a scene. Extending this method to the generation of large-scale traffic scenarios is also an interesting direction, which has been explored in Block-NeRF (Tancik et al., 2022). Block-NeRF builds a large-scale traffic scene from pure image data.

Efficiency. Due to the black-box property of most autonomous systems, it is inefficient to generate adversarial examples without accessing the inner information of the systems. In the area of adversarial attack, methods with surrogate models (Papernot et al., 2017) or gradient estimation methods (Guo et al., 2019a) are utilized to address this problem. They either learn a differentiable surrogate model to imitate the original autonomous systems or query the system to estimate the approximate direction gradient.

It is also noticed that uniform sampling from collected data is quite inefficient because of the rareness of critical scenarios. Therefore, previous methods propose using IS, which focuses on the region of the distribution in which we are interested. However, it is difficult to extend IS methods to high-dimensional cases (Arief et al., 2021b). In addition, even for adversarial generation methods, the black-box property of victim AVs remains the biggest obstacle. Without access to internal failure information, generation algorithms cannot update their scenarios efficiently.

One potential solution to this problem is to take advantage of symbolic reasoning (Mao et al.,

2019) and causal discovery (Spirtes and Zhang, 2016) techniques. Instead of performing the optimization only in a large numerical space, using symbolic representation helps the generative model to reason about the elements that make the scenario critical. Causal discovery methods can uncover the underlying causality behind critical scenarios, finding the mechanisms that cause the risk.

Diversity. Critical scenarios are rare but also diverse. Most of the current generative methods focus on finding the best scenario that satisfies the requirements, but ignores diversity. To fully evaluate AV performance, we need a wide range of scenarios. It is easy to fall into the risk of overfitting if the testing scenarios are very similar. To increase the diversity of the scenario, there are generally two directions. One is from the optimization perspective, where sampled-based methods, such as the evolutionary method or Bayesian optimization, can be used to get feasible solutions from multiple modes. The other direction is applying regularization to the density estimation model or building multi-modal distribution (e.g., Gaussian Mixture Model) to represent the scenario.

Adaptivity. In the safety and robustness areas, adversarial attack is considered a common way to generate risky examples. The adaptivity, which means that the generated samples are also applicable to other algorithms, is a crucial factor in evaluating the generation method. For example, the pixel-level adversarial attack only works for the target victim. The attack is believed to occur at the group level (Xu et al., 2018) or the semantic level (Xiao et al., 2018) to achieve better transferability. An example in the critical scenario is to control one surrounding vehicle (SV) to hit the AV step by step. After training the SV policy, we change the target AV. The new AV shows a very different behavior and follows a route that the SV has never seen before. Most likely, this scenario is not risky for the new AV. In this example, we should let the SV learn at a higher level, where it contains the semantic meaning of risk. The SV can suddenly appear behind another vehicle, which leaves the AV with a very short time to react. Essentially, we need to build a hierarchical scenario where the high level makes a plan for the risky scenario and the low level executes that plan with control commands.

Controllability. Controllability is useful in two cases. One is that we want to repeat one specific scenario with several parameters fixed, and the other is to generate different scenarios with

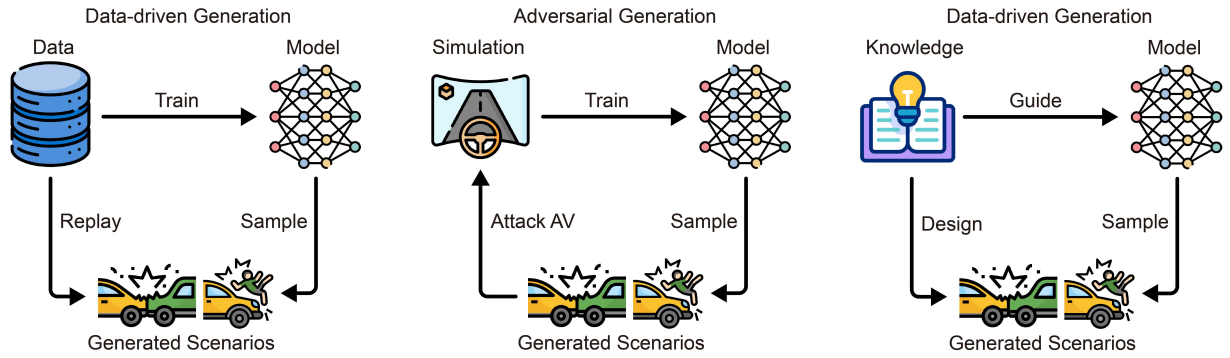


Figure 1.3: Illustration of three types of generation methods: data-driven methods, adversarial generation methods, and Knowledge-based generation methods.

similar settings. For example, we want to test the performance in a highway environment and then we want to generate vehicles that approach the AV from different directions. Conditional generative models (Mirza and Osindero, 2014; Sohn et al., 2015) are widely used to generate controllable samples, which learn a joint distribution of the condition and the data. Sometimes, the condition could be as simple as numerical values or as complex as natural languages. The challenge is that current NN-based models have poor generalization and, therefore, fail when given unseen conditions during generation. One promising direction could be to increase generalizability under such a zero-shot setting.

1.5 Thesis Structure

In the following chapters, I will introduce my work, summarized into three categories, as shown in Figure 1.3, according to the main source of information: data-driven generation, adversarial generation, and knowledge-based generation. After introducing my previous work, I will discuss future work and plans.

- **Chapter 2 (Data-driven Generative Models)** introduces two of my works for data-driven generative models, both of which leverage the Variational Auto-encoder (VAE) as the backbone. The first work investigates sampling in latent space to generate different trajectory encounters. Based on the first, the second work provides controllable generation and road-trajectory association that random sampling cannot achieve, leveraging linear interpolation

between safe data and collision data.

- **Chapter 3 (Adversarial Generative Models)** introduces two generative models of adversarial attack. The first work uses Reinforcement Learning (RL) to find critical scenarios to attack autonomous vehicles. However, only using the adversarial objective leads to unrealistic scenarios with low diversity. As an improvement, the second work incorporates a prior distribution of real-world data as regularization and introduces an adaptive sampler to find multiple critical scenarios.
- **Chapter 4 (Knowledge-based Generative Models)** introduces two methods that use prior knowledge in generative models, where knowledge is represented primarily by the causality of the critical events. The first work assumes that causality is known from human knowledge and investigates how to inject causality into generative models. The second work extends the framework to unknown causality and proposes a new generative model with automatic causal discovery.
- **Chapter 5 (Conclusion)** concludes all the methods proposed during my Ph.D. research, providing some important takeaway messages and discussion about potential future directions of using generative AI to build critical digital twins.

Chapter 2

Data-driven Generative Models

> If you torture the data long enough, it will confess to anything.
>
> *Ronald Coase*

In this section, we consider algorithms that only use information from structural datasets. For example, we control several vehicles to pass an intersection by making them follow the trajectories recorded at the same intersection in the real world. These methods are divided mainly into two parts. The first part directly samples from the dataset $x \sim \mathcal{D}$ to reproduce the real-world log, which usually suffers from the problem of rareness. The second part uses density estimation models (e.g. DGMs) $p_\theta(x)$ parameterized by θ to learn the distribution of scenarios, which enables the generation of unseen scenarios. Usually, the learning objective of these models is maximizing the log-likelihood

$$\hat{\theta} = \arg\max \sum_x \log p_\theta(x), \tag{2.1}$$

and the sampling process is conducted by $x \sim p_\theta(x)$ from a generative model.

2.1 Generation via Latent Space Sampling

Autonomous driving is being considered as a powerful tool to bring about a series of revolutionary changes in human life. However, efficient interaction with surrounding vehicles in an

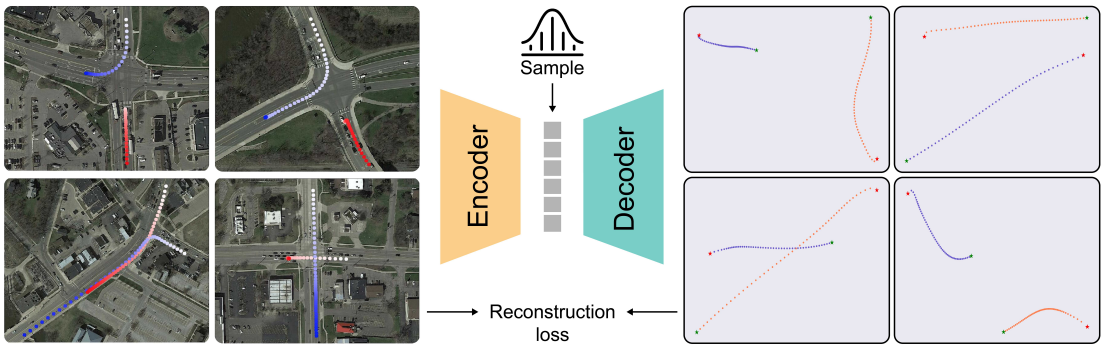


Figure 2.1: Procedure of generating multi-vehicle encounter trajectories using VAE model.

uncertain environment continues to challenge the deployment of autonomous vehicles due to the diversity of scenarios. Classifying the range of scenarios and designing appropriate associated solutions separately appears to be an alternative to overcome the challenge, but limited prior knowledge about complex driving scenarios poses an equally serious problem (Appenzeller, 2017). Some studies used deep learning technologies, such as learning controllers via end-to-end neural networks (Yang et al., 2018b) to handle large amounts of high-quality data without requiring full recovery of multi-vehicle interactions. However, deep learning methods are limited to scenarios that have never been shown in the training data set.

Most of the released databases (Wang and Zhao, 2018) do not provide sufficient information on multi-vehicle interaction scenarios due to technical limitations and the required data collection costs (Wang et al., 2017). An alternative is to generate new scenarios that are similar to the real world by modeling the limited available data, as shown in Figure 2.1. The basic underlying concept is inspired by the image style transformation (Gulrajani et al., 2017; Mirza and Osindero, 2014) and the functions of deep generative models: projecting the encounter trajectories into a latent space from which new trajectories can be generated using sampling techniques. A suitable candidate, Variational Autoencoders (VAE) (Kingma and Welling, 2013), has been developed, which characterizes the generated data more explicitly with a prior distribution (Higgins et al., 2016; Kim and Mnih, 2018). However, traditional VAE cannot fully capture the temporal features of multi-vehicle encounter trajectories because it only handles information over spatial space. As an improvement, neural networks in a recurrent frame of accounting history, such as long-short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent units

(GRU) (Cho et al., 2014) are able to tackle sequences, such as vehicle encounter trajectories, ordered by time.

On the basis of the requirement of purposely generating driving encounters and the limitation of traditional VAE, we develop a deep generative framework integrated with the GRU module to generate multi-vehicle encounter trajectories. Figure 2.2 illustrates our proposed MTG which encodes driving encounters into interpretable representations with a bidirectional GRU module (green) and generates the sequences through a multi-branch decoder (orange) separately. The reparameterization process (blue) is introduced between the encoder and the decoder (Kingma and Welling, 2013).

Model performance evaluation is challenging because of the lack of ground truth for the generated multi-vehicle trajectories. Here, we propose a new disentanglement metric for model performance evaluation in terms of interpretability and stability. Compared to the previous disentanglement metric (Kim and Mnih, 2018), our metric is not sensitive to hyperparameters and can also quantify the relevance among the latent codes.

This method contributes to the published literature by

- Introducing a deep generative model that uses latent codes to characterize the dynamic interaction of multi-vehicle trajectories.
- Generating multi-vehicle trajectories that are consistent with the real data in the spatio-temporal space.
- Proposing a new disentanglement metric that can comprehensively analyze deep generative models in terms of interpretability and stability.

2.1.1 Variational Auto-encoder

The optimization of VAE is usually converted to the Evidence Lower Bound (ELBO) as the estimation of the marginal log-likelihood is computationally intractable because of the curse of dimensionality. The optimization function of VAE and β-VAE is formulated in 2.2. When $\beta = 1$, traditional VAE (Kingma and Welling, 2013) is obtained, and when $\beta > 1$, β -VAE (Burgess

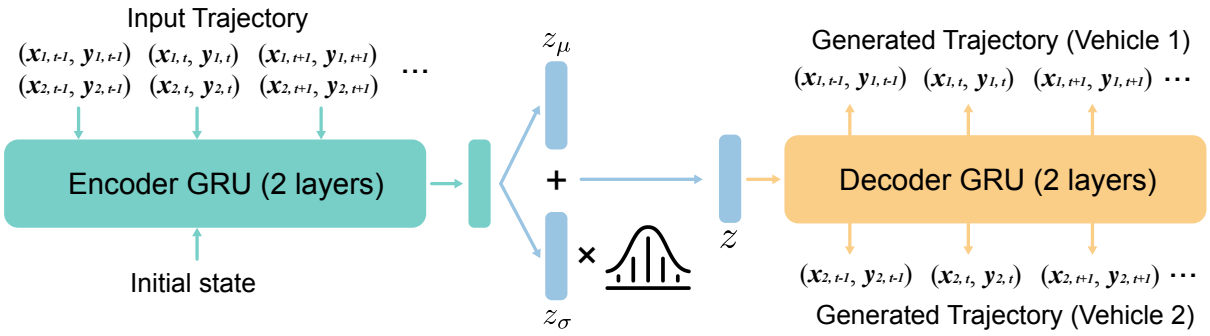


Figure 2.2: Model structure of our multi-vehicle trajectory generator.

et al., 2018; Higgins et al., 2016) is obtained.

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \beta D_{KL}(q_\phi(z|x)||p(z)), \tag{2.2}$$

where q_ϕ represents the encoder parameterized by ϕ, p_θ represents the decoder parameterized by θ, and z is a latent variable. β is used to reduce the distance between $p_\theta(z|x)$ and $q_\phi(z|x)$, thus increasing β helps to obtain more disentanglement. In 2.2, intuitively, the first term is interpreted as the reconstruction error between real and generated trajectories, and the second term is a tightness condition to guarantee $p_\theta(z|x) = q_\phi(z|x)$.

2.1.2 Multi-Vehicle Trajectory Generator (MTG)

To explore the benefits of the modified structure, we developed a single-directional GRU encoder of β-VAE. The encoder processes multiple sequences simultaneously with one GRU module, and the output (μ and σ) of the encoder is resampled through the reparameterization process (Kingma and Welling, 2013). The process is formulated as follows:

$$h_{enc} = GRU_{enc}([x_1; x_2]), \tag{2.3}$$

$$\mu = W_\mu h_{enc} + b_\mu \ , \ \sigma = \exp(\frac{W_\sigma h_{enc} + b_\sigma}{2}), \tag{2.4}$$

$$z = \mu + \sigma \times \mathcal{N}(0, I), \tag{2.5}$$

Algorithm 1: Disentanglement Metric for VAE

Input: Set of input variance Σ=$\{0.1, 0.4, 0.7, 1.0, 1.3, 1.6, 1.9, 2.2, 2.5, 2.8\}$

Initialize the set that stores output varianceΩ = []
for i *from* 1 *to* $dim(z)$ **do**
  for σ *in* Σ **do**
    initiate Θ = []
    for j *in range(1,* $dim(z)$*)* **do**
      when $i \neq j$, sample latent dimension $z_j \sim \mathcal{N}(0, \sigma^2)$
    for n *in range(1,* N*)* **do**
      sample latent dimension i $z_i^n \sim \mathcal{N}(0, \sigma^2)$
      $\hat{z} \leftarrow$ Encoder(x), $x \leftarrow$ Decoder(z)
      $\Theta \leftarrow \Theta \cup \{\hat{z}\}$
    Calculate variance and collect $\Omega \leftarrow \Omega \cup \{Var(\Theta)\}$
Return σ for each i in Ω

where x_1 and x_2 are two scenario sequences (driving trajectory), and z is the latent codes in dimension K. The decoder takes z as the initial state and outputs the sequence coordinates one after another. The two sequences are generated from the decoder at the same time by 2.6. We select Tanh as the last activation function to ensure that the output value of the decoder is in the range of [-1,1].

$$[\bar{x}_1; \bar{x}_2] = GRU_{dec}(P_{start}, z). \tag{2.6}$$

To test other prevalent deep generative frameworks, we built a baseline on top of InfoGAN with the same GRU modules as our MTG. The generator in InfoGAN shares the hidden states among multiple sequences, and the discriminator encompasses a bi-directional GRU.

Compared to baseline 1, the MTG has two improvements. First, the bi-directional counterparts replace the single-directional GRU module, which enables the encoder to extract deeper representations from the trajectories. We use this design because the traffic trajectories are still practically reasonable after being reversed in the time domain. The pipeline of the MTG encoder is formulated as follows:

$$h_{enc}^{\rightarrow} = GRU_{enc}^{\rightarrow}([x_1; x_2]) \ , \ h_{enc}^{\leftarrow} = GRU_{enc}^{\leftarrow}([x_1; x_2]), \tag{2.7}$$

where GRU represents the GRU module. Then, we obtain the hidden feature by concatenating

the features from two directions:

$$h_{enc} = [h_{enc}^{\rightarrow}; h_{enc}^{\leftarrow}]. \tag{2.8}$$

Second, we separate the decoder into multiple branches and share the hidden states among them. In this way, the hidden state retains all the information over past positions and provides guidance to generate the other sequence. We note that generating two sequences independently avoids mutual influence. The MTG decoder pipeline is formulated as

$$[x_1^t, h_1^t] = GRU_{dec,1}(x_1^{t-1}, h_2^{t-1}), \quad [x_2^t, h_2^t] = GRU_{dec,2}(x_2^{t-1}, h_1^{t-1}). \tag{2.9}$$

Then the objective function is concluded as

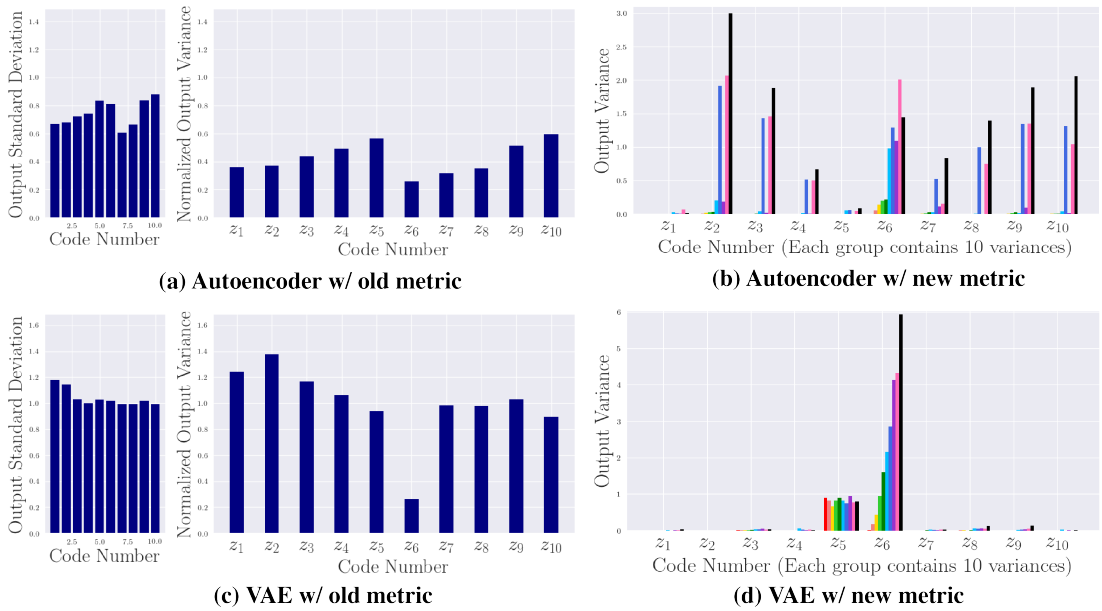
$$\mathcal{L} = \mathcal{F}(x_1, \bar{x}_1) + \mathcal{F}(x_2, \bar{x}_2) + \beta \times D_{KL}(q_\phi(z|x)||p(z)), \tag{2.10}$$

where $\mathcal{F}(\cdot, \cdot)$ is the mean square error to calculate the reconstruction error, and $\bar{x}_i x \in 1, 2$ represents the reconstructive trajectory.

2.1.3 A Metric for Disentanglement

Disentanglement is a key factor for deep generative model evaluation. Some researchers used the precision of the classifiers to represent the disentangled ability (Higgins et al., 2016; Kim and Mnih, 2018). For example, (Higgins et al., 2016) acquired the training data by calculating the difference $\|z_k^1 - z_k^2\|$ between two latent codes z_k^1 and z_k^2 for the k-th dimension. (Kim and Mnih, 2018) further considered the relationship between latent variables and latent codes. However, the extreme sensitivity of simple classifiers to hyper-parameters, can skew the evaluation result. Moreover, the metrics in (Higgins et al., 2016; Kim and Mnih, 2018) cannot be used directly to analyze the stability and dependency of the latent codes. In Section III-C, we propose a new disentanglement metric capable of comprehensively evaluating model performance without using any classifier-based approaches.

The metric in (Kim and Mnih, 2018) holds one dimension of the latent code fixed and selects other dimensions randomly, then calculates the variance of the output of the encoder under the



(a) Autoencoder w/ old metric **(b) Autoencoder w/ new metric**

(c) VAE w/ old metric **(d) VAE w/ new metric**

Figure 2.3: Comparison of two evaluation metrics on Autoencoder and VAE.

assumption that the fixed dimension should have less variance, which is easily recognized by a linear classifier. On the contrary, our metric (see **Algorithm 1**) is more stable. We divide the input data into several groups with different variances (each group has L samples) z_{k,σ_m}, in which $k \in K$ is the latent code index, and $m \in M$ is the group index of the different variances. Only one code was selected and sampled for each group, with the others fixed. We input these artificial latent codes z into the decoder to generate the trajectories and then feed them into the encoder to obtain new latent codes $\hat{z}$ again. Finally, we calculate the variances of $\hat{z}$ for each group, revealing the dependence among the latent codes and the stability of the model.

2.1.4 Experiment and Analysis

In this section, we first introduce the dataset we use and the data pre-processing. Then, we discuss the results from different perspectives.

Dataset and Preprocessing.

We used data from driving encounters collected by the University of Michigan Transportation Research Institute (UMTRI) (Wang et al., 2017) of about 3,500 vehicles equipped with onboard

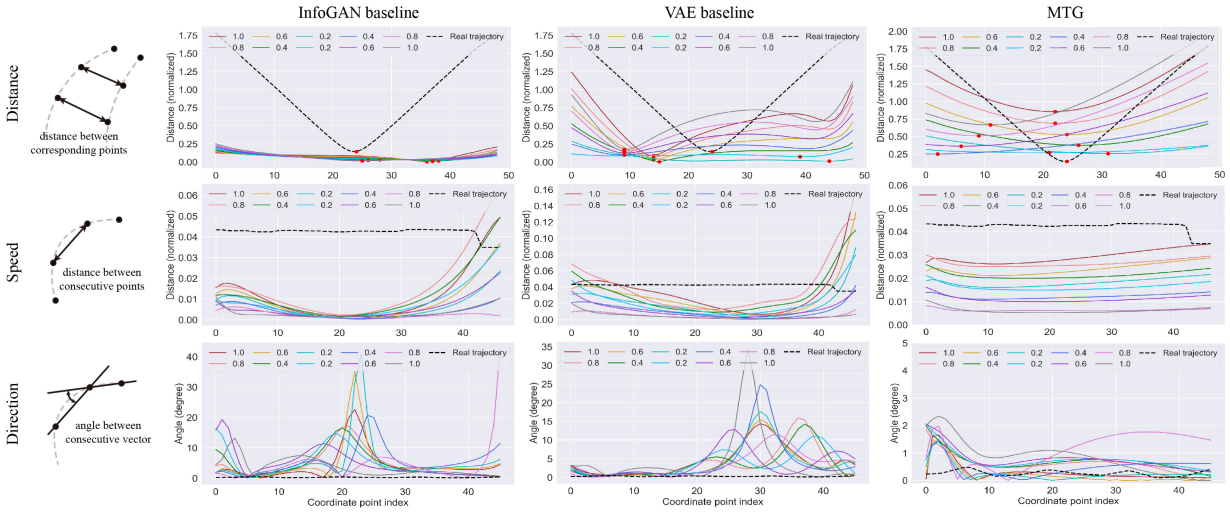


Figure 2.4: Results of three models with traffic rationality evaluation.

GPS. The GPS device recorded latitude and longitude data to represent vehicle positions. All data were collected with a sampling frequency of 10 Hz. We then linearly upscale and downscale the trajectory vectors to the size of 50. Taking into account the bias sensitivity of neural networks, we normalize the sequence value into the range of [-1, 1] by (2.11), where $\tau_i = \{(x_i, y_i)|i = 1, \cdots, 50\}$.

$$\tilde{\tau}_i = \frac{\tau_i - \text{mean}(\tau_i)}{\max(\tau_1, \tau_2)}, \quad i = 1, 2, \tag{2.11}$$

where τ_1 and τ_2 means the trajectory of the vehicle 1 and vehicle 2.

Experimental Settings and Evaluation.

In all experiments, we set the dimension of z to 10. The dimension selections could be different since we do not have prior knowledge about the dimension of the latent space. To validate the capability of each code separately, we change the value of one dimension from -1 to 1 with a step size of 0.1 and keep the other dimension fixed. This step size could be smaller for a more detailed analysis. We conducted experiments from two different aspects to compare our MTG with the two baselines.

The first aspect is to evaluate the model according to traffic rationality. As shown in the

first column of Figure 2.4, we analyze the generated trajectories in the time domain with three criteria:

- The distance between two sequences, which represents the variation of the distance between two vehicles.
- Variation in speed is expressed as the distance between two adjacent points. (i.e., a large distance represents a high speed)
- The variation of trajectory direction for smoothness evaluation, where the angle between two consecutive vectors represents the variation of moving direction.

The second aspect is to evaluate the models in the latent space with our proposed metric. For each input variance, we calculate the variance of the output trajectories and display it as a bar, as shown in Figure 2.5 and Figure 2.6. Each group consists of ten different variances distinguished by color.

Generative Trajectories Overview.

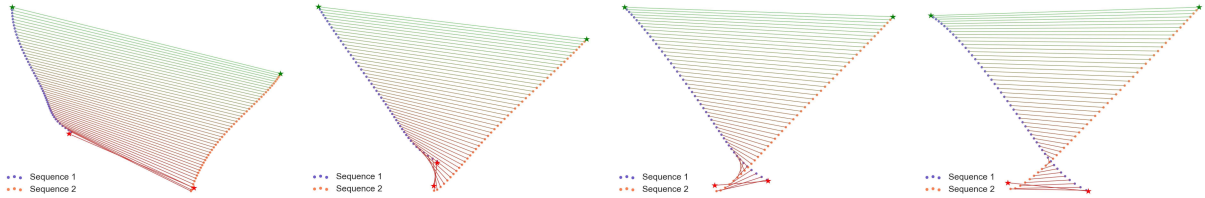
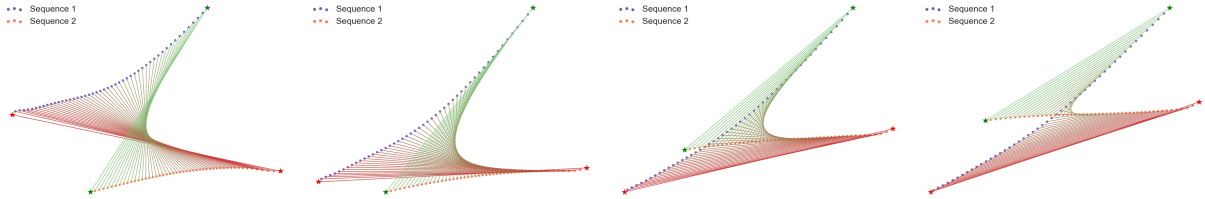
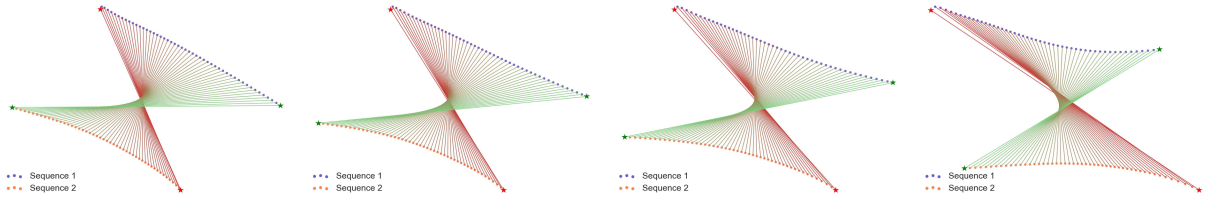
Table 2.1 shows 'zoom-in' figures for a more detailed analysis of the generated MTG trajectories. We connect the associated points in the two sequences, from the starting point (green) to the end point (red), with lines. In each row, the four figures derive from four different values of one latent code with a continuous change from left to right. These trajectories indicate that MTG is capable of controlling some properties of generated trajectories (e.g., the location where two vehicles meet and their directions) through latent codes.

Traffic Rationality Analysis.

Figure 2.4 shows some generated results based on the three criteria introduced in Section 2.1.4. Different colors represent different values of the latent code z_1, while the black dashed lines represent the real traffic data for comparison.

The corresponding distance indicates that InfoGAN outputs trajectories with a small variance even for different input values of z. This is in line with the problem of mode collapse that is common in the GAN family. The VAE baseline and MTG obtain distances closer to the real

Table 2.1: Analysis of Trajectories in the Time Domain

Generated results from MTG with 4 values of z_1 (0.1, 0.3, 0.5, 0.7)

Two vehicles change from driving in the same direction to encountering each other.

Two vehicles change from driving in the opposite direction to encountering each other.

The trajectories of two vehicles rotate in the global coordinate.

trajectory. For MTG, the distance gradually decreases and then increases with z changing from -1 to 1, and the speed of the vehicle changes along the latent code value. Comparing the last two vertical columns in Figure 2.4 indicates that MTG can generate a much smoother trajectory than VAE. In the real world, vehicles cannot turn sharply in a very short period of time due to physical constraints. Therefore, a high value of consecutive angles will reduce the validity.

Disentanglement Analysis.

Figure 2.3(a) and (b) with z_6 and others fixed explain why our metric outperforms the previous one (Kim and Mnih, 2018). We obtain Figure 2.3(a) by using the metric in (Kim and Mnih, 2018) with an auto-encoder. After being normalized by dividing the standard deviation (left plot in Figure 2.3(a)), the right part of Figure 2.3(a) shows the output variances of $\hat{z}$. Although there are few differences in all codes, z_6 still reaches the lowest value. Certainly, if all the results are

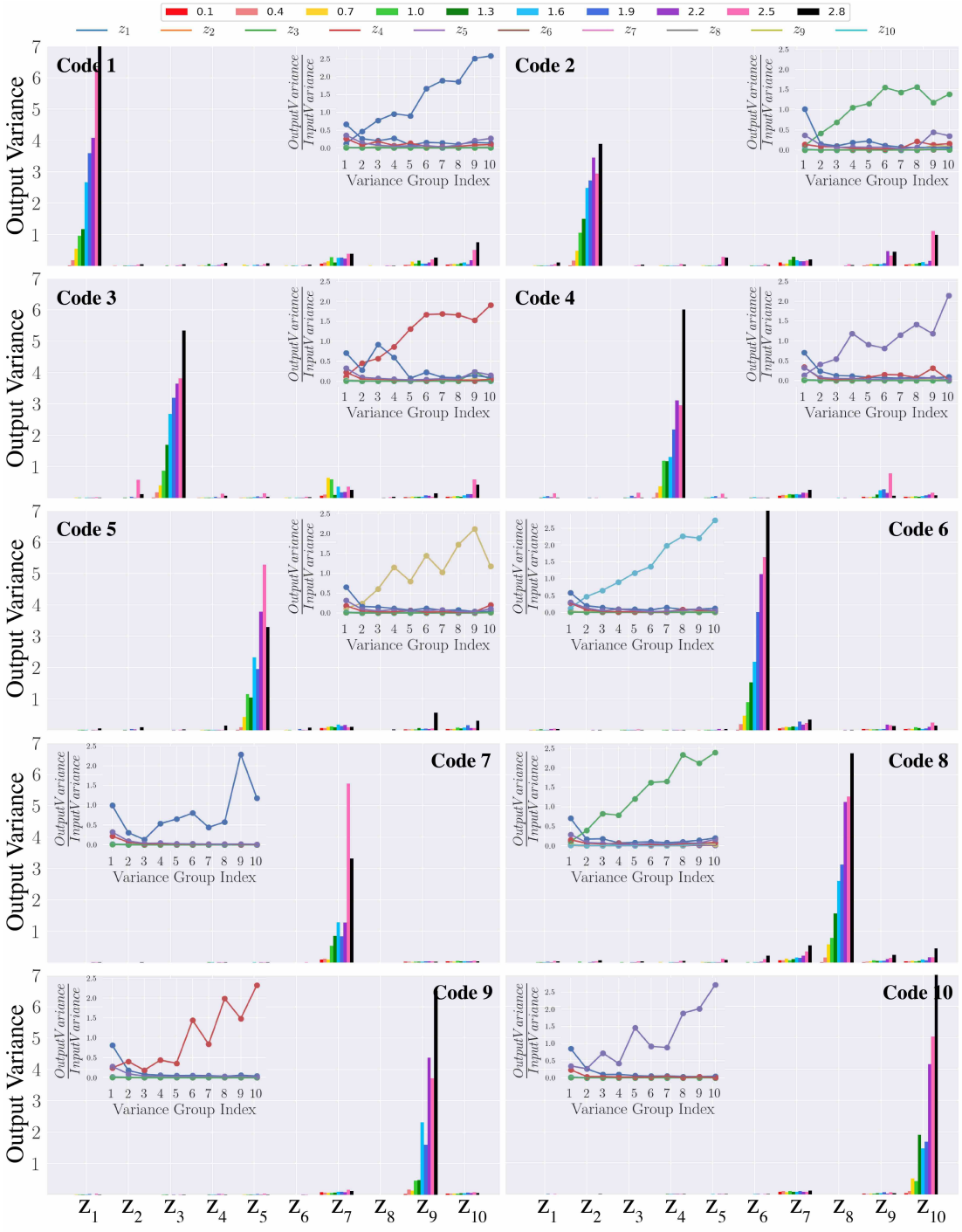


Figure 2.5: Disentanglement of MTG.

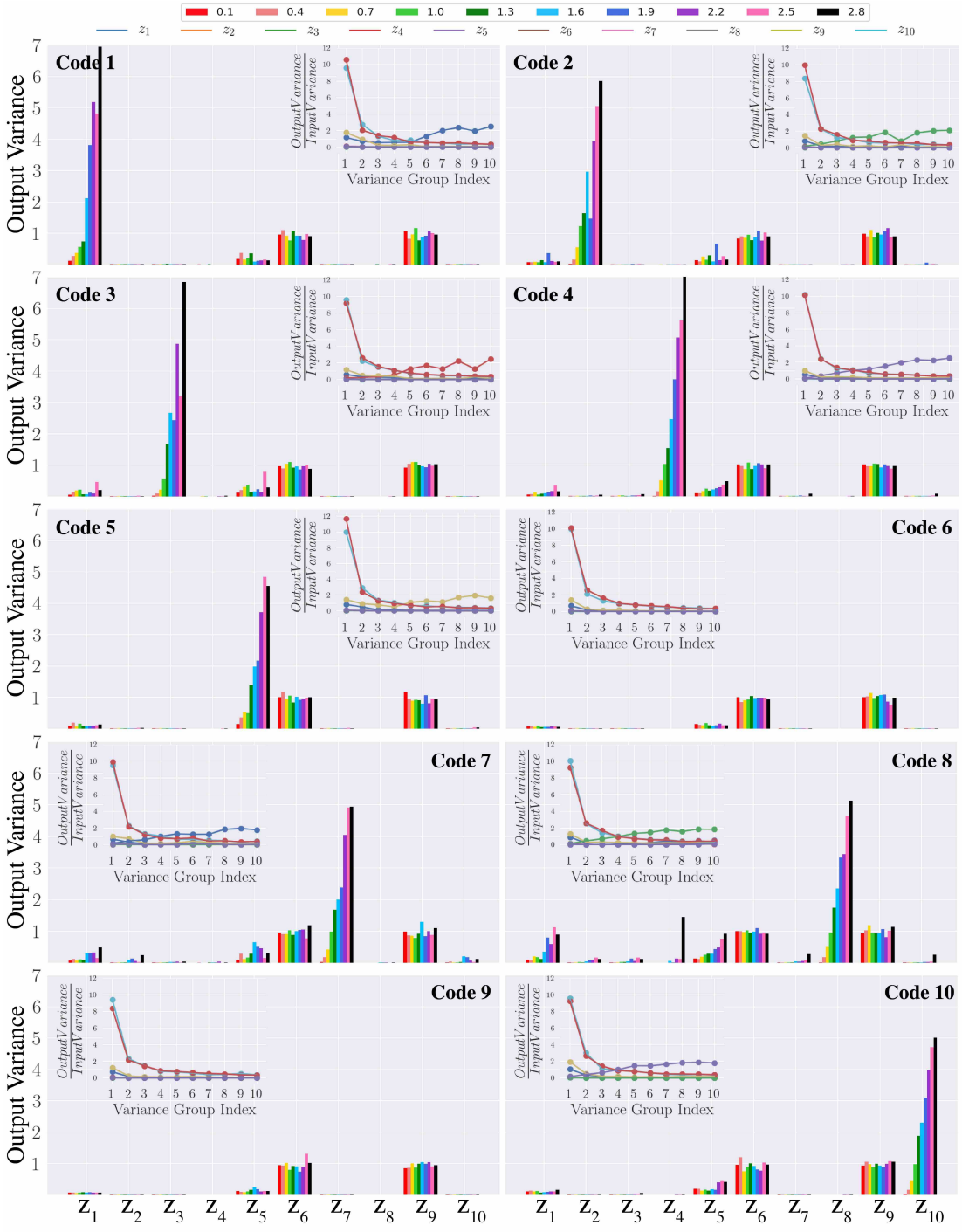


Figure 2.6: Disentanglement of VAE baseline.

close to this case, we can obtain a high accuracy of the classifier while evaluating an autoencoder without any disentangled ability; thus, the metric in (Kim and Mnih, 2018) has a problem in evaluating the disentanglement. On the contrary, our metric (Figure 2.3(b)) can identify the code that dominates and also reveals the dependency among the codes. High values of latent codes except z_6 indicate a strong dependency among all codes z_i.

We then evaluated and compared two metrics on the VAE. Although z_6 attains a much lower value (Figure 2.3(c)) because of the capability of VAE to disentangle the latent code, it is still uncertain whether the other codes are influenced by z_6. Figure 2.3(d) shows the nearly zero value of the remaining codes (except z_5), that is, the independence between the codes. In addition to proving the disentanglement, we find that z_5 is a normal distribution without information (output variance equal to 1).

Finally, we use the proposed metric to compare the VAE baseline with our MTG. Figure 2.5 shows (1) that only the sampling code obtains an increasing output variance when increasing the input variance, and (2) that the other codes are close to zero. In other words, there is almost no dependency among the latent codes in MTG, or changing one code does not influence other features. Figure 2.6 shows two normal distributions at the positions of z_6 and z_9, indicating that the VAE baseline obtains two latent codes without useful information. The plot of Code 8 in Figure 2.6 also shows that z_8 influences z_1 and z_5 because their output variances are non-zero.

The subplots inside Figure 2.5 and Figure 2.6 show the ratio of the output variance and the input variance. A more robust approach will force the value close to 1. A robust generative model requires the encoder to recognize all trajectories generated from the decoder, i.e., the variances of the output and the input should be the same. However, the values in both figures are much higher than 1, indicating that neither the VAE baseline nor MTG is robust enough.

2.2 Risk-level Adjustment with Controllable Generation

Different institutes of autonomous driving research have already released their datasets that contain millions of data (Caesar et al., 2020; Chang et al., 2019a). While it costs a lot of effort and money, these datasets provide a valuable asset for the advancement of self-driving technologies.

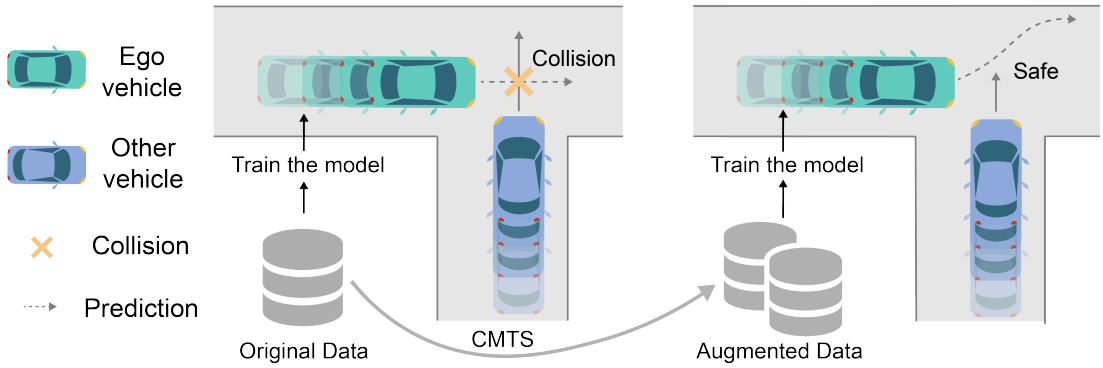


Figure 2.7: Overview pipeline of proposed CMTS.

However, most naturalistic driving data are safe; thus, performance evaluations of autonomous driving algorithms can be unreliable because the algorithms tend to overfit non-emergency scenarios and are hard to realize safety-critical scenarios, as shown in Figure 2.7.

Directly generating the safety-critical data may effectively augment the naturalistic collection process. Generative models, such as Generative Adversarial Networks (GAN) (Goodfellow et al., 2014), Variational Auto-encoder (VAE) (Kingma and Welling, 2013), and flow-based method (Kingma and Dhariwal, 2018) widely used in the image field, have become popular recently. These models fit a distribution based on large amounts of collected data and then generate new data by sampling from the distribution. We will use the released datasets to obtain the parameters that shape the distribution representing the safety-critical cases. Manifold learning (Zhu et al., 2018) shows that high-dimensional data (images, point clouds, trajectories, etc.) can be expressed in a low-dimensional space, making it possible to manipulate the original data more semantically. The use of real data and simulation provides researchers with safe and collision-driving data.

Based on these preconditions, we propose a variational Bayesian framework, which we call CMTS (shown in Figure 2.8) to synthesize safety-critical driving data from naturalistic safe trajectory data and artificial collision trajectory data. The three-step framework encodes the distributions of safe data and collision data into a low-dimensional latent space; embeds the road information in this latent space; and synthesizes the risk scenarios from the interpolated intermediate distribution. We use a style transfer method to separate the road information and the

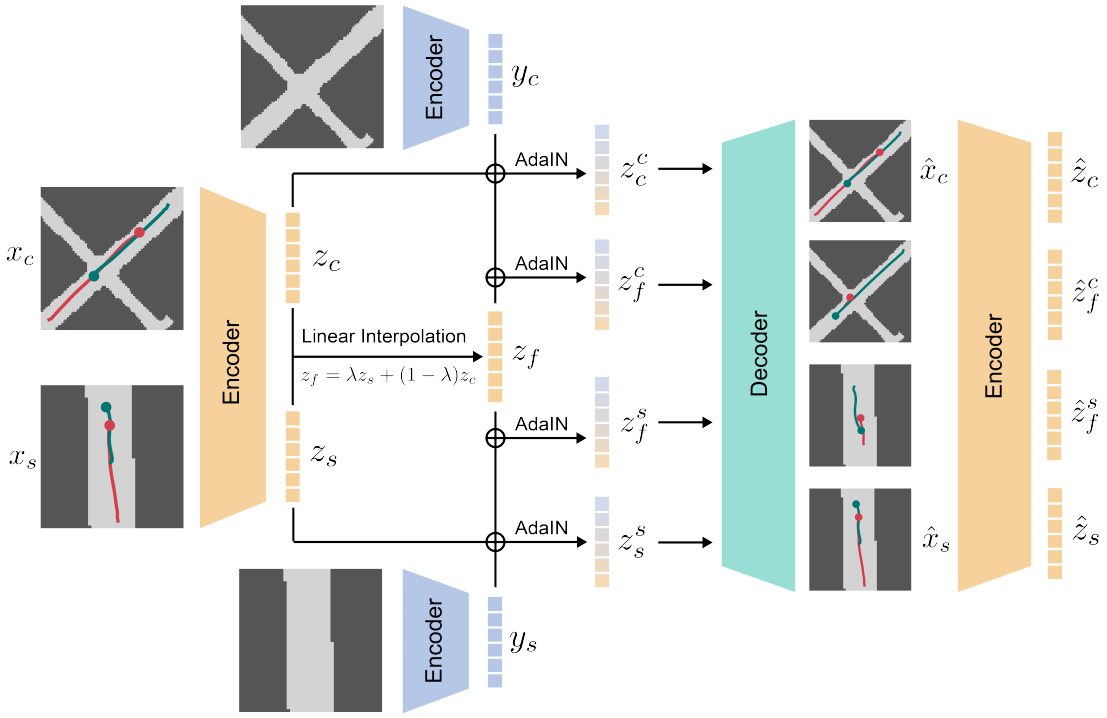


Figure 2.8: Model structure of the proposed CMTS during the training stage.

driving behavior before we obtain the representation in a low-dimensional space.

To evaluate the effectiveness of our framework, we leverage the statistical indicators to measure the complexity of the augmented dataset and the original dataset. Experimental results show that the augmented dataset improves the accuracy of trajectory prediction algorithms. In addition, the generated safety-critical scenarios help the model deal with unseen risky samples during the testing stage. The contribution of this method is threefold:

- A variational Bayesian framework is proposed to fuse information of two domains.
- Rare near-miss driving data is generated by leveraging the proposed framework using public driving datasets.
- The accuracy improvement of trajectory prediction algorithms and the enhancement of the capability to deal with risky scenarios.

2.2.1 Preliminary and Problem Formulation

VAE (Kingma and Welling, 2013; Rezende et al., 2014) is a directed graphical model with an inference process and a generative process. In the generative process, the latent variable z is generated from the prior distribution $p_\theta(z)$ characterized by θ, and the data x are generated by $p_\theta(x|z)$. The parameter of the generative part θ is obtained through optimization. A direct way to optimize the auto-encoder is to maximize the likelihood $log\, p_\theta(x)$ of all data points:

$$\begin{aligned} log\, p_\theta(x) &= E_{q_\phi(z|x)}\left[log\, p_\theta(x,z)\right] - E_{q_\phi(z|x)}\left[log\, p_\theta(z|x)\right] \\ &= E_{q_\phi(z|x)}\left[log\frac{p_\theta(x,z)\, q_\phi(z|x)}{q_\phi(z|x)\, p_\theta(z|x)}\right] \\ &= \underbrace{E_{q_\phi}\left[log\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]}_{=\mathcal{L}(x)} + \underbrace{KL\left(q_\phi(z|x)||p_\theta(z|x)\right)}_{\geq 0}, \end{aligned} \tag{2.12}$$

where KL is the Kullback-Leibler divergence (KLD), and $q_\phi(z|x)$ is the variational approximation of $p_\theta(z|x)$. Since the KL term is nonnegative, $\mathcal{L}_{\theta,\phi(x)}$ is the lower bound of the likelihood $log\, p_\theta(x)$, which is also called the evidence lower bound (ELBO).

$$\begin{aligned} \mathcal{L}(x) &= E_{q_\phi}\left[-log\, q_\phi(z|x) + log\, p_\theta(x,z)\right] \\ &= -KL(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi}\left[log\, p_\theta(x|z)\right]. \end{aligned} \tag{2.13}$$

The goal of maximization of the likelihood is equivalent to maximizing the ELBO in Equation (2.13), the second term of which is denoted as the negative reconstruction error in the auto-encoder terminology.

2.2.2 Conditional Multiple Trajectory Synthesizer (CMTS)

The proposed CMTS Figure 2.8 shows the: (1) Gated Recurrent Unit (GRU) data encoder that encodes two sequence data (one from the original dataset, another from the collision dataset) into a latent space; (2) the convolutional condition encoder that encodes the road map into a multivariate Gaussian distribution; and (3) the GRU decoder that reprojects the combination of sequence and road map back to the high-dimensional data space. The overall loss function is as

follows:

$$\mathcal{L}_{CMTS}(x|y) = \alpha(\mathcal{L}_r^s + \mathcal{L}_r^c) + \beta(\mathcal{L}_{KL}^s + \mathcal{L}_{KL}^c) + \gamma\mathcal{L}_f, \tag{2.14}$$

where y is the condition or style represented by the grid map images, α, β, and γ are the weights of different parts of the loss. $\mathcal{L}_r^s$ and $\mathcal{L}_r^c$ are the reconstruction errors of the safe and the collision datasets, and $\mathcal{L}_{KL}^s$ and $\mathcal{L}_{KL}^c$ are the KLD of the safe and the collision datasets, respectively. L_f is the consistent regularization to help improve the performance of interpolation, which will be introduced in Equation 2.19.

Merging Conditions with Style Transfer

The derivation of the conditional VAE (Sohn et al., 2015) is similar to (2.12). The only difference is that the generative and inference models are both conditioned on y. In CMTS, y denotes the information of the road restriction, which is already contained in the corresponding data point x, thus we assume that $q_\phi(z|x,y) = q_\phi(z|x)$. We further relax the constraints so that the prior distribution of latent code z is statistically independent of input variables (Kingma et al., 2014). Finally, we obtain the optimization function of conditional VAE:

$$\begin{aligned}\mathcal{L}(x|y) &= -KL(q_\phi(z|x,y)||p_\theta(z|y)) + E_{q_\phi}\left[log\, p_\theta(x|z,y)\right] \\ &= \underbrace{-KL(q_\phi(z|x)||p_\theta(z))}_{=\mathcal{L}_{KL}} + \underbrace{E_{q_\phi}\left[log\, p_\theta(x|z,y)\right]}_{=\mathcal{L}_r}.\end{aligned} \tag{2.15}$$

Note that we retain the condition y in $p_\theta(x|z,y)$ because we consider it to be the style of the data that changes in the generative part. To perform the style transfer operation, we use AdaIN (Huang and Belongie, 2017). We assume that the style information has been included in x and z, so it is reasonable to first remove the original style before assigning a new one. The formula of AdaIN is described below:

$$z_y = AdaIN(z_x, y) = \sigma(y)\frac{z_x - \mu(x)}{\sigma(x)} + \mu(y), \tag{2.16}$$

where z_x is conditioned by x and z_y is conditioned by y. Since the road map, defined by a binary matrix, is the condition, we use convolution layers to extract features and directly output the $\mu(y)$

and $\sigma(y)$ of the condition y.

Interpolation during Training.

We use linear interpolation during the training stage to obtain the composed feature. We use VAE as our basic model because its prior distribution of the latent code z is a multivariate Gaussian distribution, which is naturally disentangled. Assuming the latent code of the safe data and the collision data are z_s and z_c respectively, the linear interpolation in latent space becomes:

$$z_f = \lambda z_s + (1 - \lambda) z_c, \tag{2.17}$$

where $\lambda \in [0, 1]$ is the weight controlling the proportion of components. The interpolation distribution is formulated as:

$$p(z_f|\lambda) \sim \mathcal{N}(\lambda\mu_s + (1 - \lambda)\mu_c, \lambda^2\sigma_s^2 + (1 - \lambda)^2\sigma_c^2), \tag{2.18}$$

where λ is drawn from a uniform distribution $U[0, 1]$ for each pair of data points during the training process. In the generation stage, λ is fixed and z_f is sent to the decoder to generate the near-miss data.

Reconstruction of Fusion Samples.

Next, we measure the reconstruction error of the composed latent code z_f. For z_s and z_c, we directly calculate the difference of x and $q_\phi(x|z)$. Since there is no reference for z_f in the dataset. To solve this problem, we propose a consistent regularization from the mutual information point of view that is similar to (Yin et al., 2019). We ensure the reconstruction of z_f by building a unique mapping from x_f to z_f with the encoder $p_\theta(x|z)$. From the view of mutual information, we note there is a unique mapping from x to z if and only if the entropy $H(z|x) = 0$. However, it is intractable to access all data pairs to calculate the entropy, so we use variational inference to

obtain the upper bound of $H(z|x)$:

$$\begin{aligned} H(z|x) &\triangleq -\sum_{z} p(z|x) log\, p(z|x) \\ &= -\sum_{z} p(z|x) log\, q(z|x) - \sum_{z} p(z|x) \left[log \frac{p(z|x)}{q(z|x)} \right] \\ &= H_{p(z|x)}[log\, q(z|x)] - \underbrace{KL(p(z|x)||q(z|x))}_{\geq 0} \\ &\geq H_{\tilde{x}\sim p_\theta(x|\tilde{z}),\tilde{z}\sim q_\phi(z|x)}[log\, q_\phi(z=\tilde{z}|\tilde{x})] \triangleq \mathcal{L}_f(z,\tilde{z}). \end{aligned} \tag{2.19}$$

In the second step in (2.19), we assume $p(x)$ is a uniform distribution, which is reasonable for a dataset without any prior knowledge. The result of (2.19) uses the similarity of z and $\tilde{x}$ to represent the entropy $H(z|x)$, providing a consistent regularization term to guarantee the reconstruction of z_f. In our implementation, we use the L2-norm to calculate this $\mathcal{L}_f$.

2.2.3 Experiment and Analysis

To demonstrate the performance of CMTS, we performed experiments on the three datasets and used other data augmentation algorithms as two baselines.

Lines Dataset. Lines dataset was proposed in (Berthelot et al., 2018). The simple artificial line dataset makes it possible to interpolate smoothness easily observed by some metrics such as the *Smoothness* and *Mean Distance* in (Berthelot et al., 2018). The results of this data set are used to compare the smoothness of the interpolation.

Digit datasets. The two handwriting digit datasets, MNIST and USPS (Denker et al., 1988), each contain ten numbers but with entirely different styles. We consider the classes of the number as conditions and interpolate two different datasets in the latent space. They are used to compare the ability of models to interpolate two different domains with conditions.

Argoverse. The Argoverse Motion Forecasting dataset (Chang et al., 2019a) contains the driving trajectories of two vehicles, as well as road map information. We extracted the trajectory data from both vehicles and applied several data augmentation methods. To check the effectiveness of the generated augmented trajectory datasets, we tested three trajectory prediction

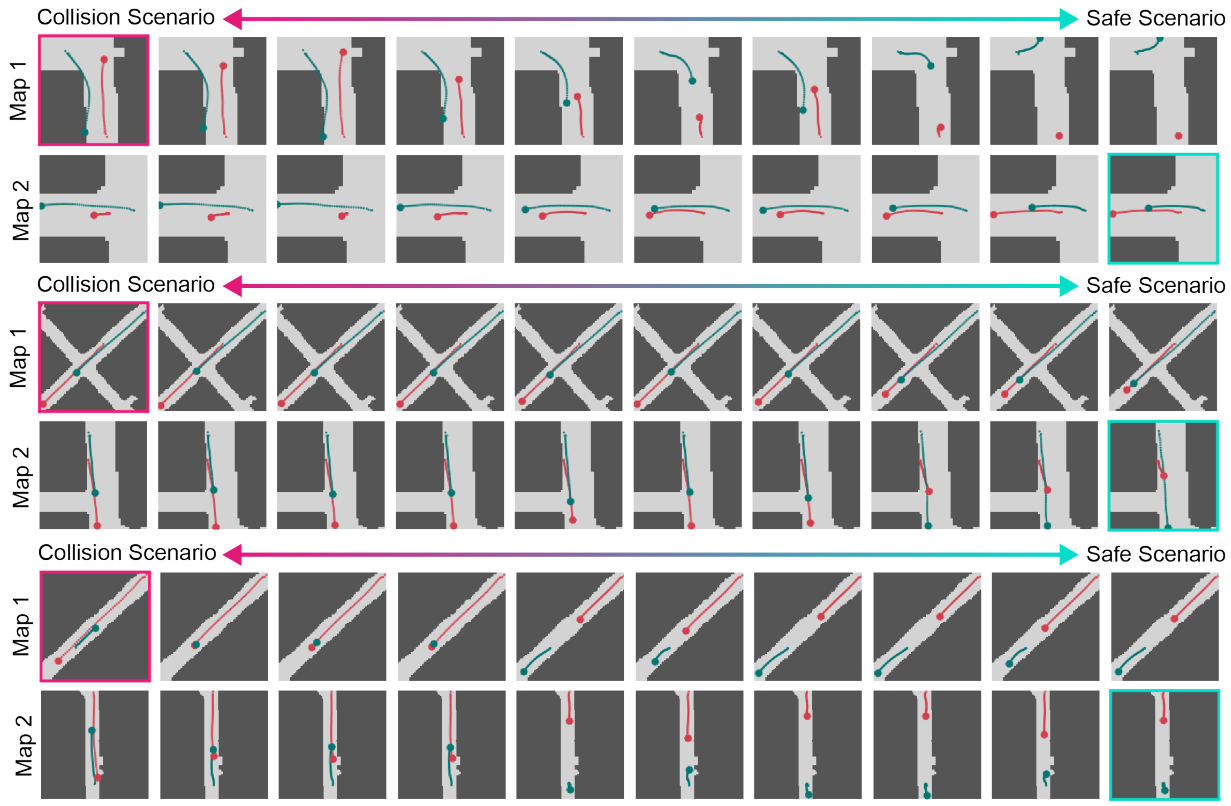


Figure 2.9: Three interpolation examples of safe trajectories and collision trajectories using the Argoverse dataset.

algorithms on both the original and augmented datasets. Furthermore, we artificially generated six of the most likely and real risky scenarios based on the Argoverse dataset.

Baselines. We select two baselines for comparison. One is the vanilla VAE structure called MTG (Ding et al., 2019), which has no AdaIN module and no fusion loss term. The second is a trajectory augmentation method proposed in (Bansal et al., 2018), which we call perturbed. This method fixes the start and end points randomly, disturbing the midpoint pose, and then fits a smooth trajectory to the perturbed point, the start and end points.

Results Analysis

Next, we provide results and an analysis of our experiments from different perspectives.

Metric 1: cluster number of datasets. To describe the complexity of a dataset, we use the number of clusters of the dataset as a metric for evaluation. Although information entropy is

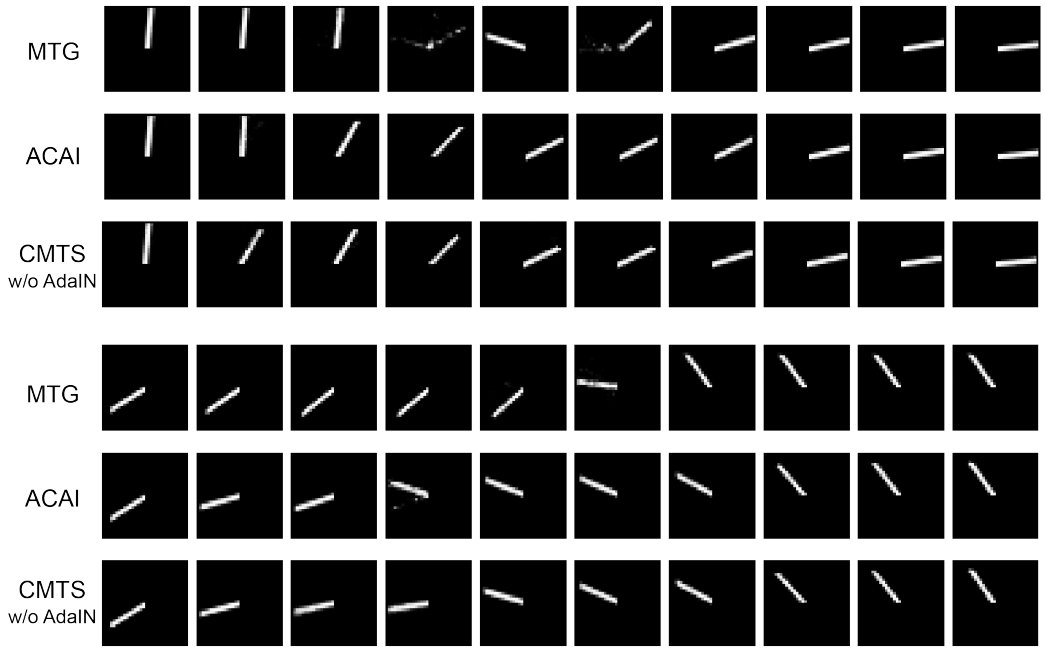


Figure 2.10: Two interpolation examples of MTG, ACAI, and CMTS on the lines dataset.

Table 2.2: Results of Lines Dataset

Metric	Mean Distance ($\times 10^{-3}$)	Smoothness
VAE (Kingma and Welling, 2013)	1.21±0.17	0.49±0.13
AAE (Makhzani et al., 2015)	3.26±0.19	0.14±0.02
VQ-VAE (Van Den Oord et al., 2017)	5.41±0.49	0.77±0.02
ACAI (Berthelot et al., 2018)	**0.24±0.01**	**0.10±0.01**
MTG	1.07±0.11	0.45±0.06
CMTS w/o AdaIN	0.32±0.02	0.15±0.01

commonly used to describe the complexity of a dataset, it requires labels that are unavailable in our case. Therefore, we use the non-parametric Bayesian method, Dirichlet Process Gaussian Mixture Models (DPGMM) (Görür and Edward Rasmussen, 2010), to cluster the dataset without a predefined cluster number K. A large K means that the dataset is complex.

Metric 2: Trajectory risk. The performance of trajectory prediction algorithms in the original dataset can be easily evaluated with mean square error (MSE), while in artificially designed near-miss trajectories, MSE cannot capture risk information. We propose using the minimal distance between two vehicles (MDV) to represent the risk of encountering and the mean distance between neighboring waypoints (MDN) to represent the smoothness.

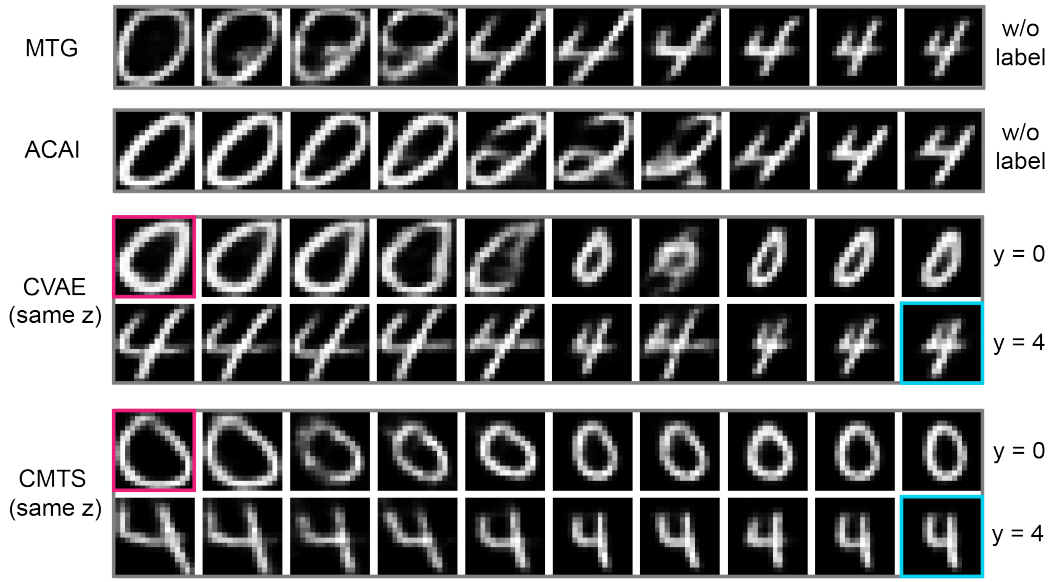


Figure 2.11: Examples of MTG, ACAI, CVAE, and CMTS on the MNIST digit datasets.

Table 2.3: Dataset Complexity [1] of Argoverse Dataset

Dataset	OD	CD	MTG	CMTS	OD+CMTS
K	64±7	76±16	53±8	71±11	**127±20**

[1] A Higher number is better. All datasets have the same number of samples.

Lines Dataset: interpolation results. The interpolation examples in Figure 2.10 show that CMTS has smoother results than the VAE baseline and the results are similar to the state-of-the-art ACAI interpolation method (Berthelot et al., 2018). Table 2.2 reports the quantitative analysis with two metrics proposed by (Berthelot et al., 2018). CMTS achieves a similar score with ACAI and outperforms other methods, which proves that CMTS is capable of smooth interpolation in a simple situation with one attribute.

Digit Datasets: conditional style transfer. Figure 2.11 shows the interpolation results for the MNIST and USPS datasets using the four methods. Since MTG and ACAI cannot use label information, we add Conditional VAE (CVAE) (Sohn et al., 2015) as a new competitor, which uses labels to control the generated results. MTG has a noisy boundary between two kinds of classes since the style and content information are entangled. This entanglement makes it difficult to achieve a smooth interpolation. In contrast, CVAE achieves better results than MTG

Table 2.4: Error of Trajectory Prediction (2 seconds of history and 3 seconds of prediction)

Method	Vanilla-LSTM				Social-LSTM				CS-LSTM			
Dataset	OD	Perturbed	MTG	CMTS	OD	Perturbed	MTG	CMTS	OD	Perturbed	MTG	CMTS
MSE	0.093	0.090	0.087	**0.083**	0.074	0.072	0.067	**0.062**	0.048	0.046	0.046	**0.042**
MDV	0.025	0.035	0.081	**0.143**	0.027	0.038	0.114	**0.125**	0.032	0.043	0.097	**0.163**
MDN	2.682	2.153	1.552	**1.181**	2.732	2.321	1.463	**1.029**	2.124	1.852	1.274	**0.961**

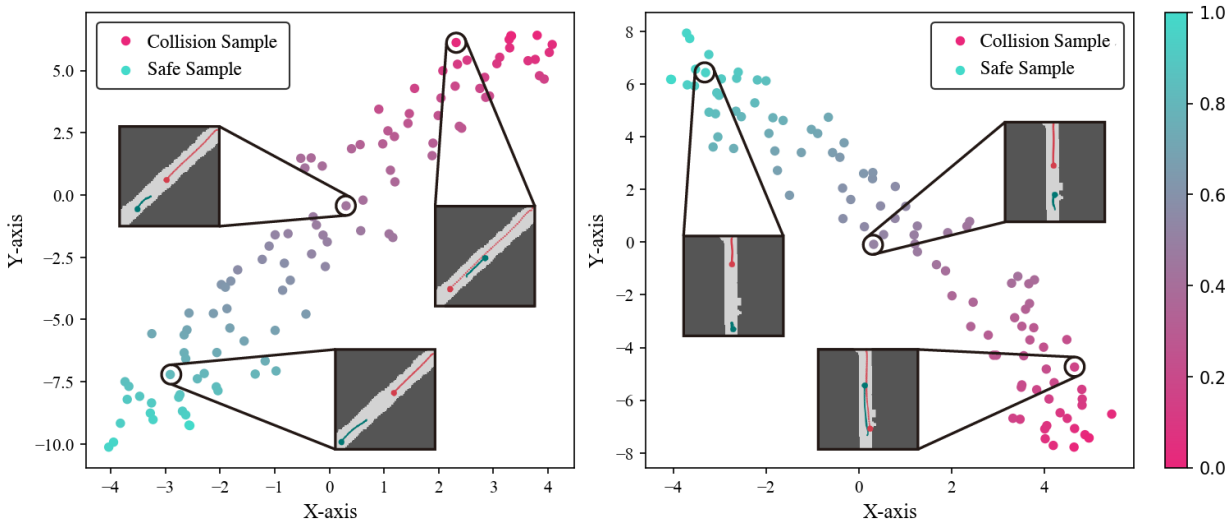


Figure 2.12: T-SNE results of interpolation samples. The pink and blue points represent the safe samples and the collision samples, respectively.

and ACAI because it separates the number class as a condition. CMTS gives better interpolation results than CVAE because the changes between two entirely different styles are much smoother. We demonstrate that CMTS has the ability to fuse information from two domains with conditions.

Argoverse: increment of data complexity. We use the DPBMM tool to evaluate the original Argoverse dataset (OD); a collision dataset (CD) obtained by translating two trajectories to a predefined collision point in OD; a dataset generated by MTG; and a dataset generated by CMTS. In the MTG and CMTS generating stage, we set the parameter λ to 0.3 to synthesize samples that are closer to the risk scenarios.

We encode each dataset into a latent space and use the low-dimensional codes to train four non-parametric Bayesian models. We set the concentrating parameter α to 1 for all models and use the output cluster number K as an indicator of the complexity of the datasets. Table. 2.3

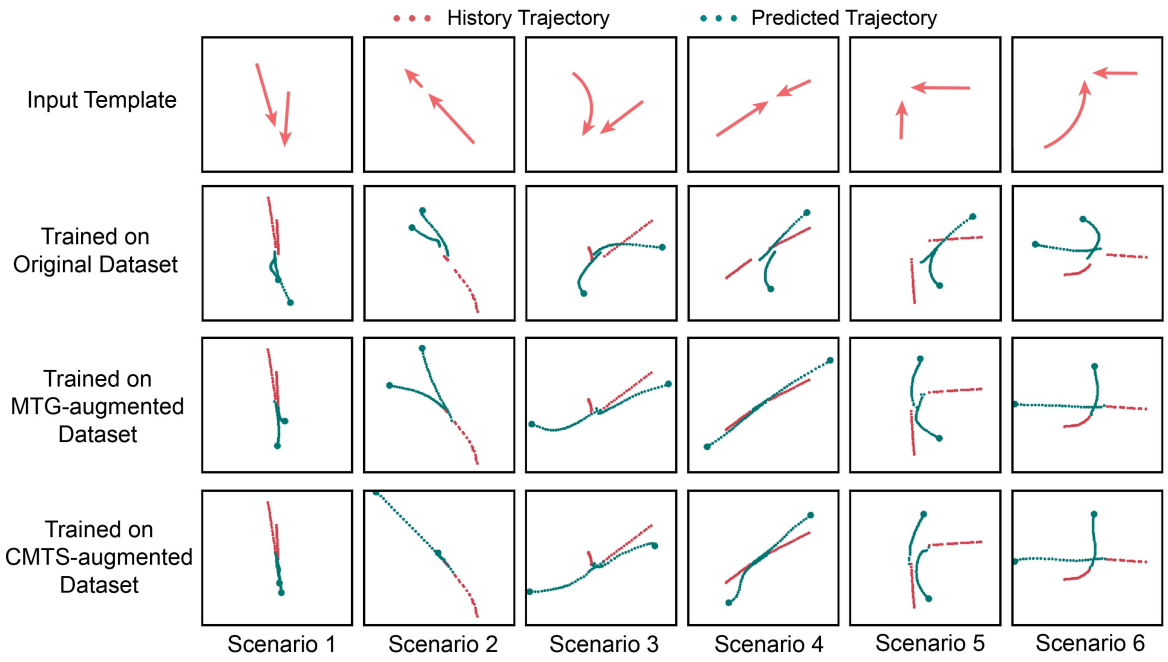


Figure 2.13: Trajectory prediction results for the six risky scenarios with CS-LSTM.

shows that the dataset generated by CMTS has larger numbers of clusters than OD and CD. The combination of the CMTS dataset and the OD has the highest number of clusters among all the datasets, which means that the CMTS dataset contains clusters unseen in OD and CD; otherwise, CMTS + OD will have similar K to OD.

To visualize the rare trajectories generated by CMTS, Figure 2.9 shows three examples, each of which has two map conditions from the original and collision datasets. In addition, we use T-SNE (Van der Maaten and Hinton, 2008) to display the linear interpolation samples in a 2-dimensional space in Figure 2.12, where the interpolation samples are roughly arranged in the 2-dimensional space.

Argoverse: improvement of trajectory prediction. Three trajectory prediction methods (Vanilla-LSTM Social-LSTM (Alahi et al., 2016) and CS-LSTM (Deo and Trivedi, 2018)) and two kinds of driving scenarios are selected to test the augmented dataset. The first kind is safe driving scenarios sampled from the original Argoverse dataset. The second kind is the risky situations that are rarely collected in the naturalistic dataset. We artificially design six risky scenarios (Figure 2.13) based on the Argoverse dataset.

In Table 2.4, we compare the performance of the prediction algorithms trained on four

datasets: original dataset (OD), perturbed dataset (augmented by random perturbation), and MTG and CMTS dataset. For all three trajectory prediction models, CMTS achieves the best result in MSE, MDV, and MDN, which shows that new scenarios help trajectory prediction algorithms improve their ability to deal with risky situations.

To evaluate performance in risky scenarios, we selected 18 CS-LSTM predicted examples as shown in Figure 2.13. Since the near-miss data do not exist in OD, the predictions from the model trained in OD are awful. The reason is that the risky scenario data comes from other domains, which has exceeded the generalization scope of the model trained only on OD. For the two models trained on MTG and CMTS, we observe that the model trained on the CMTS dataset predicts more smooth and reasonable trajectories.

2.3 Retrieval Augmented Generation for Controllable Traffic Scenarios

Simulation is indispensable in the development of autonomous vehicles (AVs), primarily due to the considerable risks associated with training and evaluating these systems in real world conditions. The biggest challenge in simulations lies in achieving realistic driving scenarios, as this realism influences the discrepancy between AV performance in simulated and actual environments. Although advances in high-quality graphical engines have significantly enhanced the perception quality of simulators, the realism of agent behavior remains constrained due to complicated interactions among naturalistic agents. To counteract this issue, data-driven simulation has emerged as a promising approach in the realm of autonomous driving, using real-world scenario datasets to accurately generate agent behaviors.

With the rapid achievement of deep generative models (Ho et al., 2020) and imitation learning algorithms (Hu et al., 2022), current data-driven simulations (Gulino et al., 2023; Li et al., 2023) can generate scenarios that closely mimic human driver behavior. However, the effectiveness of these simulations in accelerating the development of AV is limited. This limitation stems from the need for scenarios that meet specific conditions tailored for targeted training and evaluation. Achieving such controllability in simulations is challenging due to the complex nature of

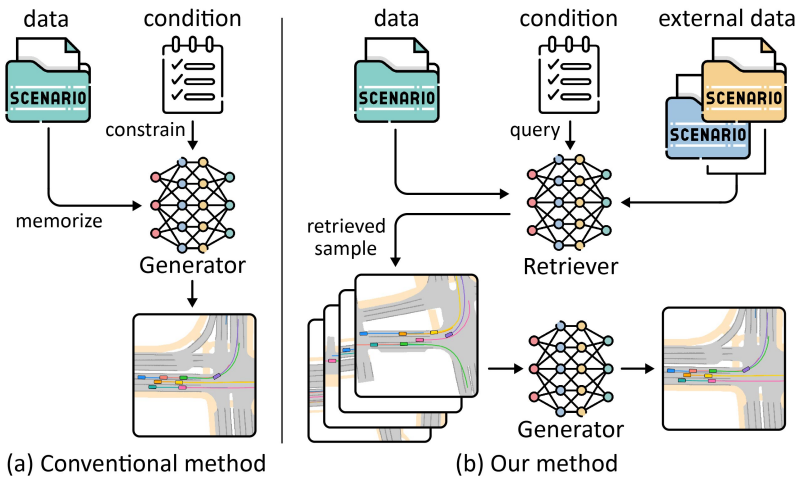


Figure 2.14: (a) Conventional methods make the model memorize the data distribution for generating. (b) In contrast, our method employs a retriever to query datasets (including external data obtained after training) and uses a generative model to generate scenarios by integrating the information from the retrieved scenarios.

driving scenarios, which involve intricate interactions, diverse road layouts, and varying traffic regulations.

In pursuit of controllability, existing work applies additional guidance, typically in the form of constraint functions (Ding et al., 2021b; Zhong et al., 2023b) or languages (Tan et al., 2023; Zhong et al., 2023a), to pretrained scenario generative models. Regularization of the generation process through these tools is straightforward and effective, yet it encounters two principal challenges. First, the training of scenario generative models typically utilizes naturalistic datasets, which might not encompass the specific scenarios desired according to the control signals. Even if such scenarios exist within the dataset, they are often omitted because of the rarity of long-tail data. The second challenge is that the representation of the guidance to the generative model may not be sufficiently expressive to accurately depict complex scenarios, such as specifying intricate interactions among multiple vehicles, using language. These limitations underscore the need for a more sophisticated and nuanced framework for the generation of controllable scenarios.

Retrieval Augmented Generation (RAG) (Chen et al., 2017a; Guu et al., 2020), which enhances the generative process by querying related information from external databases, represents great potential in the domain of large language models (Lewis et al., 2020). In contrast

to conventional models that memorize all knowledge within their parameters, as shown in Figure 2.14(a), RAG models, shown in Figure 2.14(b), learn to generate comprehensive outputs by retrieving pertinent knowledge from a database, based on the input provided. A notable aspect of RAG is the ability of the database to undergo updates even after the model has been trained, allowing continuous improvement and adaptation. This flexible framework offers possibilities for controllable scenario generation by using appropriate template scenarios as input and facilitating the generation that is not only realistic but also aligned with specific training and evaluation requirements.

In this work, we present *RealGen*, a retrieval augmented generation framework to generate traffic scenarios. This framework, as shown in Figure 2.15, begins with the training of an encoder through contrastive self-supervised learning (Oord et al., 2018) to allow the retrieval process to query similar scenarios in a latent embedding space. Using this latent representation, we subsequently train a generative model that combines retrieved scenarios to create novel scenarios. The key contributions of this paper are summarized in the following.

- We develop a novel contrastive autoencoder model to extract scenario embeddings as latent representations, which can be used for a wide range of downstream tasks.
- We propose the first retrieval augmented generation framework using the latent representation tailored for controllable driving scenario generation.
- We validate our framework through qualitative and quantitative metrics, demonstrating strong flexibility and controllability of generated scenarios.

2.3.1 A Latent Representation of Scenario

A crucial element within the retrieval system is the data retrieval selection metric, which is typically implemented through a distance function between the query sample and the candidate samples in the database. Unlike text, which can be converted into word embeddings, traffic scenarios encompass sequential behaviors and intricate interactions among entities, complicating the establishment of a similarity metric for these scenarios. Consequently, in this section, we introduce a scenario autoencoder to extract latent representations that facilitate the assessment of

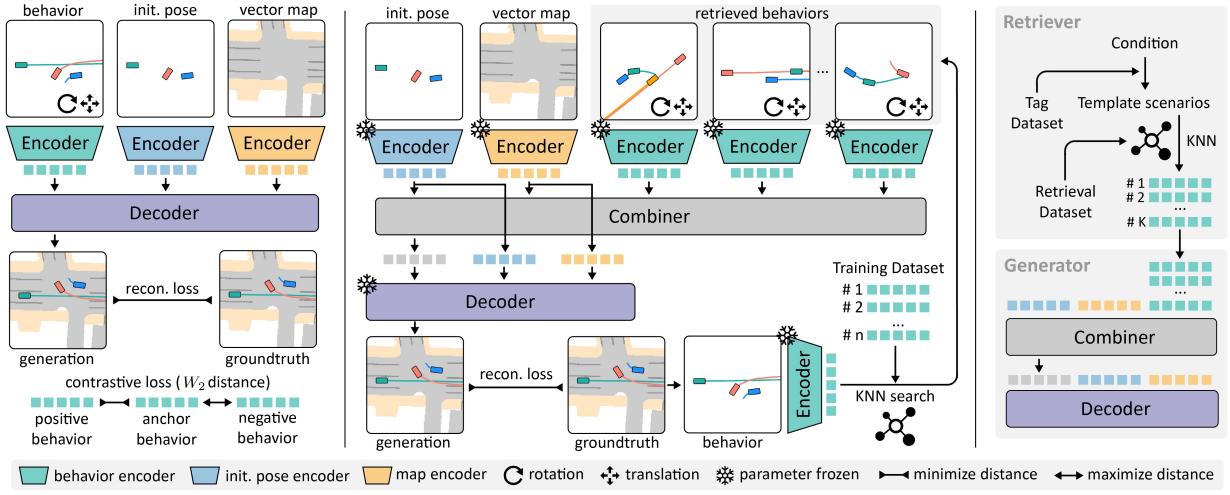


Figure 2.15: **Left**: the training pipeline for the encoders and decoder aimed at learning latent embeddings of scenarios. A contrastive loss is applied to behavior embeddings to ensure invariance to absolute positions. **Middle**: the training pipeline for the combiner with frozen encoder and decoder parameters. We use K-Nearest Neighbors (KNN) to retrieve scenarios similar to a template scenario in the dataset and use the retrieved behaviors to reconstruct the template scenario. **Right**: the generation pipeline with a retriever and a generator.

similarity between various traffic scenarios, a vital component in our RAG framework.

Scenario Definition

Each scenario is characterized by the trajectories $\tau \in \mathbb{R}^{M \times T \times 5}$, which encompasses agents M over a maximum of T time steps. The trajectory of each agent is composed of parameters $[x, y, v, c, s]$, which signify position x, position y, velocity v, cosine of the heading c, and sine of the heading s. The initial state of these entities is denoted as $\tau_0 \in \mathbb{R}^{M \times 5}$. Furthermore, the map is encapsulated by $m \in \mathbb{R}^{S \times 4}$, composed of S lane segments. The attributes of these points $[x_s, y_s, x_e, y_e]$ correspond to the starting and end positions of each segment.

Scenario Autoencoder

Autoencoders (Schmidhuber, 2015) are used to learn compressed latent representations of high-dimensional data, where an encoder projects the data into latent code and a decoder reconstructs

the code in the data space. Given that traffic scenarios encompass spatial and temporal dynamics from multiple agents, we design a hierarchical encoder structure alternating between spatial and temporal layers based on transformer architecture (Vaswani et al., 2017). As depicted in the left part of Figure 2.15, our design incorporates three encoders for behavior, map, and initial position, respectively.

First, we design a behavior encoder E_b with a spatial-temporal transformer structure that comprises a set of temporal transformer encoders (Encoder$_t$) and a set of spatial transformer encoders (Encoder$_s$). The trajectory τ is first transformed into a latent embedding z_b, where H denotes the hidden dimension, through a Multiple Linear Perception (MLP) module. Subsequently, an alternating encoding procedure extracts spatial and temporal features from z_b. To preserve temporal information, we add a sinusoidal positional embedding (PE) to the embedding before employing the temporal transformer. To retain distinct agent information and further compress the behavior feature, we calculate the mean latent embedding across the temporal dimension, resulting in the final behavior embedding $z_b \in \mathbb{R}^{M\times H}$. Besides the behavior encoder, we also process the map information represented by lane vectors with the attention mechanism (Vaswani et al., 2017). Similarly to the first step in E_b, we project m to the latent space with an MLP module. Then we apply a multi-head attention module (MHA) with layer normalization (LN) (Ba et al., 2016) to acquire the map embedding $z_m \in \mathbb{R}^{S\times H}$ with a learnable query embedding q_m.

When developing the decoder, a spatial-temporal transformer architecture was designed to decode embedding behavior z_b. Given that z_b lacks the temporal dimension, we first replicate it T times to then add a PE before inputting it into the temporal transformer encoder. Throughout the decoding process, the map embedding z_m is injected by a cross-attention mechanism: $z_b \leftarrow z_b + \text{MHA}(z_b, z_m, z_m)$. $\text{MHA}(Q, K, V)$ is the multi-head attention with Q, K, V representing query, key, and value:

$$\begin{aligned} &\text{MHA}(Q, K, V) = \text{Concatenate}(h_1, ..., h_i)W^O, \\ &h_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \tag{2.20}$$

where W^O, W_i^Q, W_i^K, and W_i^V are learnable parameters. The last component of the decoder is an MLP that projects the hidden embedding back into the data domain, resulting in the recon-

structed trajectory $\hat{\tau}$. To train the encoders and decoder, the mean square error is used as the reconstruction loss, expressed as $\mathcal{L}_r = |\hat{\tau} - \tau|_2$. Comprehensive descriptions of the encoding and decoding processes are provided in Algorithm 2.

It is important to note that, while the current design of the autoencoder is sufficient for acquiring compressed representations of scenarios, these representations are not invariant to the absolute coordinates and the order of the agents. This limitation can lead to substantial embedding distances between similar scenarios. To address these issues, two enhancements are proposed and detailed in the following section.

Invariant feature with contrastive loss

To enhance the representation of scenario similarity in the behavior embedding z_b, we employ contrastive learning to acquire invariant features. Specifically, we integrate InfoNCE (Oord et al., 2018) as an additional loss function $\mathcal{L}_c$, which optimizes the categorical cross-entropy to distinguish a positive sample from a batch of negative samples. In practice, for a query embedding z_b, a positive sample z_b^+ is generated by applying random rotation and translation to the original scenario τ, m. Meanwhile, negative samples Z_b^- are selected from the remaining samples in the same batch. In conventional InfoNCE, to bring z_b and z_b^+ closer together, the inner product is calculated between the embedding z_b and the set z_b^+, Z_b^-, using cross-entropy loss to identify the index of the positive sample.

To ensure that the ordering of the agents does not affect the outcomes, $z_b \in \mathbb{R}^{M \times H}$ should exhibit permutation invariance across the dimension M. Otherwise, simply stacking them into a one-dimensional vector could erroneously represent distinct scenarios as significantly different. To establish permutation invariance, we adopt the Wasserstein distance W_2 (Villani et al., 2009) as a more suitable metric than the cosine distance for assessing similarity. Considering z_b as a distribution representing M individual behaviors, the intuition behind this choice is that the Wasserstein distance identifies the minimal adjustments necessary for the behaviors in one scenario to mirror those in another. With such a powerful tool, we have the following contrastive loss:

$$\mathcal{L}_c = -\sum_{z_b} \log \frac{\exp\left[-W_2(z_b, z_b^+)\right]}{\sum_{z' \in \{z_b^+, Z_b^-\}} \exp\left[-W_2(z_b, z')\right]}. \tag{2.21}$$

Algorithm 2: Details of Encoder and Decoder

Behavior Encoder E_b (τ):
  $z_b \leftarrow \text{MLP}(\tau)$ // projection
  for l_e in $[1, ..., L]$ **do**
    $z_b \leftarrow \text{Encoder}_t$ (PE + z_b)
    $z_b \leftarrow \text{Encoder}_s$ (z_b)
  $z_b \leftarrow \text{mean}(z_b)$
  return behavior embedding z_b

Map Encoder E_m (m):
  Initialize a learnable query q_m
  $z_m \leftarrow \text{MLP}(m)$ // projection
  $z_m \leftarrow \text{LN}(\text{MHA}(q_m, z_m, z_m))$
  $z_m \leftarrow \text{LN}(z_m + \text{MLP}(z_m))$
  return map embedding z_m

Initial Pose Encoder E_i (τ_0):
  $z_i \leftarrow \text{MLP}(\tau_0)$ // projection
  return initial pose embedding z_i

Decoder D (z_i, z_b, z_m):
  $z_r \leftarrow z_b + \text{MHA}(z_b, z_i, z_i)$
  for l_d in $[1, ..., L]$ **do**
    $z_r \leftarrow \text{Encoder}_t$ (PE + z_r)
    $z_r \leftarrow \text{Encoder}_s$ (z_r)
    $z_r \leftarrow z_r + \text{MHA}(z_r, z_m, z_m)$
  $\hat{\tau} \leftarrow \text{MLP}(z_r)$ // projection
  return reconstructed trajectory $\hat{\tau}$

The implementation of this loss results in the behavior embedding z_b lacking absolute coordinate data, making the decoder incapable of accurately reconstructing the precise trajectory. To handle this problem, we add the initial poses τ_0 of all agents as a supplementary input for the decoder. We encode these poses into z_i with an MLP encoder Enc_i, and then integrate it into the decoder with an MHA module: $z_r \leftarrow z_b + \text{MHA}(z_b, z_i, z_i)$. Additional specifics regarding the initial pose encoder are detailed in Algorithm 2. In the training stage, the objective is to minimize a combined loss function $\mathcal{L} = \mathcal{L}_r + \lambda \mathcal{L}_c$, where λ serves as a weighting factor for contrastive loss and is uniformly set at 0.1 for all experiments.

2.3.2 Retrieval Augmented Scenario Generation (RealGen)

The auto-encoder structure introduced in Section 2.3.1 builds a one-to-one generation framework, which still needs additional modules to support the retrieval augmented generation, which is a many-to-one framework. In this section, we introduce a module called Combiner, which takes input behavior embeddings from multiple scenarios and outputs the combined embedding. To train this module, we designed a KNN-based training pipeline that forces the model to learn to combine and edit existing scenarios.

The Training of Combiner

Unlike the trajectory prediction task (Shi et al., 2022) which has ground truth as the optimization target, the objective of training the combiner is to learn the alignment of behaviors in retrieved scenarios with the specified initial pose and map. A well-trained combiner should be able to compose behaviors from these retrieved scenarios, thereby generating new scenarios that have a similarity to all the retrieved scenarios. This type of objective aligns with the concept of meta-learning, or 'learning to learn', as described in (Hospedales et al., 2021).

Within our training framework, for a given query scenario τ, m in the dataset, we initially utilize the behavior encoder to obtain z_b and then use KNN to identify K similar behavior embeddings from the database, denoted as $z_{ret} = [z_{b,1}, ..., z_{b,K}]$. Building on this, we propose a model comprising two MHA modules as the combiner:

$$\begin{aligned} z_{rag} &\leftarrow z_i + \text{MHA}(z_i, z_{ret}, z_{ret}), \\ z_{rag} &\leftarrow z_{rag} + \text{MHA}(z_{rag}, z_m, z_m), \end{aligned} \tag{2.22}$$

where $z_i \leftarrow E_i(x_0)$ and $z_m \leftarrow E_m(m)$ are the initial pose embedding and the map embedding of the query scenario. Gradients are stopped for z_i and z_m. Assuming that the K nearest scenarios adequately represent the query scenario, it should be feasible to reconstruct the behavior of the query scenario, z_b, using the retrieved scenario, z_{rag}. Consequently, we employ the following loss function as our training goal:

$$\mathcal{L}_{rag} = \|D(z_i, z_{rag}, z_m) - \tau\|_2, \tag{2.23}$$

wherein the parameters of the decoder $D\phi$ remain fixed during training. Essentially, this training method is designed to learn an "inverse" operation of the KNN, aiming to reconstruct a query scenario that closely resembles all K retrieved scenarios in the database.

The generation pipeline

The scenario generation pipeline, as delineated in Figure 2.15, is categorized into the retriever and generator components. The generation process is expedited by preprocessing all scenarios in

the database using a behavior encoder, which yields behavior embeddings that facilitate efficient similarity computation.

The retriever enhances the versatility of generation by dividing the process into two stages. In the initial stage, users can choose a set of template scenarios that represent specific conditions. These may include manually annotated scenarios with tags denoting actions like left or right turns, thereby enabling the generation of additional scenarios under similar tags or even a combination thereof. Additionally, templates can include critical and interesting scenarios collected from real-world data. Solely relying on these templates for generation could be limited, which is mitigated by incorporating a secondary phase, which employs a KNN approach to fetch high-quality scenarios from a vast and unlabeled database to augment adaptability.

Subsequently, the generator, comprising a combiner and a decoder, follows the same inference as the combiner training with the initial pose and lane map specified by the user. We obtain the RAG embedding z_{rag} with Eq. (2.22), and then infer the generated scenario through $\tau_{rag} \leftarrow D(z_i, z_{rag}, z_m)$.

Details of model architecture

In addition to the model architecture introduced in Algorithm 2, we also want to provide more details about the spatial-temporal transformer structure. For the projected embedding with shape $[B, M, T, H]$ with batch size B, we reshape it to shape $[T, B \times M, H]$ and treat T as the sequence dimension. Then we add a positional embedding to it before putting it into the temporal transformer encoder. After that, we reshape it to $[M, B \times T, H]$ and treat M as the sequence dimension. Then we directly put it into the spatial transformer encoder.

During the decoding process, we repeat the reconstructed embedding z_r with shape $[B, M, H]$ along with the temporal dimension to shape $[B, M, T, H]$. Then we follow the same process as the encoding process for spatial-temporal decoding with a positional embedding added before the temporal transformer encoder.

2.3.3 Experiment

In this section, we start with an overview of the experimental setup and details regarding the implementation of the RealGEN framework. Then we discuss the baselines, including different types of autoencoders and RealGen variants, used for comparison. In the discussion of the findings, we evaluate, respectively, the quality of scenario embedding and the generation capability of RAG.

Settings and Implementation Details

We conducted all training and evaluation with the nuScenes (Caesar et al., 2020) dataset using the trajdata (Ivanovic et al., 2023) package for data loading and processing. Each scenario spans a duration of 8 seconds, operating at a frequency of 2 Hz, and encompasses a maximum of 11 agents. We filter out agents that travel less than 3 meters in 8 seconds and select 11 agents closest to the ego vehicle. The map contains 100 lanes (each lane has 20 points) that are ordered according to the distance between the center of the lane and the center of the ego vehicle.

To train RealGen, we use Adam (Kingma and Ba, 2014) as the optimizer to update the parameters of the autoencoder and the combiner. To make the calculation of the Wasserstein distance efficient, we use the Sinkhorn distance (Cuturi, 2013), an entropy regularized approximation of the Wasserstein distance (Villani et al., 2009), with implementation in the GeomLoss (Feydy et al., 2019) package.

Baselines

In the experimental section, we evaluate the following reconstruction-based generative models as baselines for comparison. Autoencoder (**AE**) shares the same behavior encoder, map encoder, and decoder structures as RealGen, serving as the most straightforward baseline for scenario reconstruction. **Contrastive AE** mirrors the structure of the RealGen autoencoder but omits the initial pose as absolute information. **Masked AE** is a self-supervised learning baseline, which has been investigated for trajectory data as described in (Chen et al., 2023; Cheng et al., 2023; Wu et al., 2023; Yang et al., 2023). To evaluate the controllable generation capability, we select a state-of-the-art model**LCTGen** (Tan et al., 2023) as the baseline, which takes a high-level

agent representation z and a vector map m as input. We also consider **LCTGen w/o** z, a variant model proposed in the original paper to show the reconstruction results without using the agent information. Given that our method has two phases, we refer to the autoencoder component specifically as **RealGen-AE** and the complete model as **RealGen**.

Details of evaluation metrics

We provide details on the metrics we use in the experiment section.

Maximum Mean Discrepancy (MMD) is a statistical measure used to test the similarity of two distributions, denoted as P and Q, by calculating the distance between the means of these distributions in a feature space defined by a kernel function. In our implementation, we use a Gaussian kernel $k(x, y)$ (also known as a radial base function kernel). Assume we have samples $X = \{x_1, x_2, ..., x_m\}$ from distribution P and $Y = \{y_1, y_2, ..., y_n\}$ from distribution Q, we calculate the MMD statistic as:

$$\begin{aligned} &\text{MMD}^2(P, Q) = \\ &\frac{\sum_{i,j} k(x_i, x_j)}{m^2} + \frac{\sum_{i,j} k(y_i, y_j)}{n^2} + \frac{2\sum_{i,j} k(x_i, y_j)}{mn}, \end{aligned} \tag{2.24}$$

where m and n are the number of samples from P and Q respectively. A higher MMD value indicates a greater discrepancy between the two distributions.

Mean Average Displacement Error (mADE) is a metric commonly used in the field of trajectory prediction, which quantifies the accuracy of predicted trajectories by comparing them with actual trajectories. Assume the predicted trajectory is $\hat{\tau}_i = \{\hat{p}_1^i, \hat{p}_2^i, ..., \hat{p}_T^i\}$ and the ground truth trajectory is $\tau_i = \{p_1^i, p_2^i, ..., p_T^i\}$ for agent i with trajectory length T, we calculate mADE with

$$\text{mADE} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{T} \sum_{t=1}^{T} \|\hat{p}_t^i - p_t^i\| \right), \tag{2.25}$$

where N is the number of agents in the validation dataset.

Mean Final Displacement Error (mFDE) is another important metric used in trajectory prediction, similar to mADE, but with a specific focus on the prediction accuracy at the final

time step of the trajectories. With the same notation defined above, we calculate mFDE with

$$\text{mFDE} = \frac{1}{N}\sum_{i=1}^{N} \|\hat{p}_T^i - p_T^i\|. \tag{2.26}$$

Scene Collision Rate is used to calculate the overlap between two vehicles for all time steps in all trajectories. Specifically, a collision is said to occur between two vehicles if the Intersection Over Union (IOU) of their representing rectangles exceeds the threshold θ:

$$\text{Collision}(R_1, R_2) = \begin{cases} 1, \text{if IOU}(R_1, R_2) > \theta, \\ 0, \text{otherwise}, \end{cases} \tag{2.27}$$

where the threshold θ is set to 0.1 and IOU is defined as:

$$\text{IOU}(R_1, R_2) = \frac{\text{Area of Intersection } (R_1,\, R_2)}{\text{Area of Union } (R_1,\, R_2)}. \tag{2.28}$$

Therefore, the scene collision rate is defined as:

$$\text{Scene Collision Rate} = \frac{1}{N}\sum_{i=1}^{N} \text{Collision}(R_1, R_2). \tag{2.29}$$

Off-Road Rate is used to calculate whether the trajectory is in the drivable area. We first calculate the indicator function of off-road with

$$\text{Off-Road}(\tau_t) = \begin{cases} 1, \text{if } \tau_t \text{ not in drivable area} \\ 0, \text{otherwise} \end{cases}. \tag{2.30}$$

Then we calculate the rate with

$$\text{Off-Road Rate} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{T}\sum_{t=1}^{T} \text{Off-Road}(\tau_t^i)\right). \tag{2.31}$$

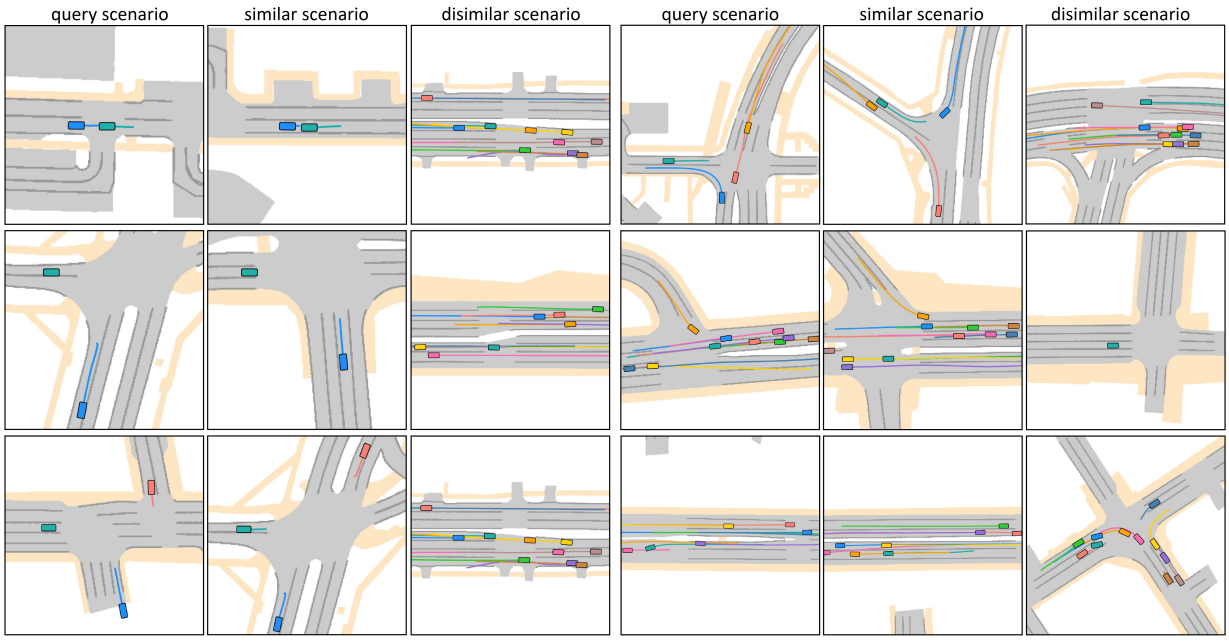


Figure 2.16: Qualitative evaluation of similar and dissimilar scenarios calculated by our scenario embedding. The rectangles represent the initial poses of the vehicles and the lines represent the future trajectories.

Evaluation of Behavior Embedding

We first evaluate the quality of the learned representation of behavior, which is critical for the following retrieval and generation processes.

Visualizing similar and dissimilar scenarios. After training the auto-encoder with contrastive loss, the distance between behavior embeddings can be used as an indicator of the similarity of the two scenarios. To validate this statement, we visualize qualitative examples of using a query to find the most similar (minimal W_2 distance) and dissimilar (maximal W_2 distance) scenarios in Figure 2.16. We observe that the most similar scenario contains the same behavior and number of vehicles as in the query scenario.

Classifying Scene ID with behavior embedding. To further provide quantitative results on how well the behavior embedding encodes behavior information, we leverage the Scene ID of the nuScenes dataset. Each scene typically lasts 20 seconds, and our scenario segments last 8 seconds, so we get multiple segments belonging to the same Scene ID. We assume that the segments having the same Scene ID have similar behaviors so that we can calculate the accuracy by

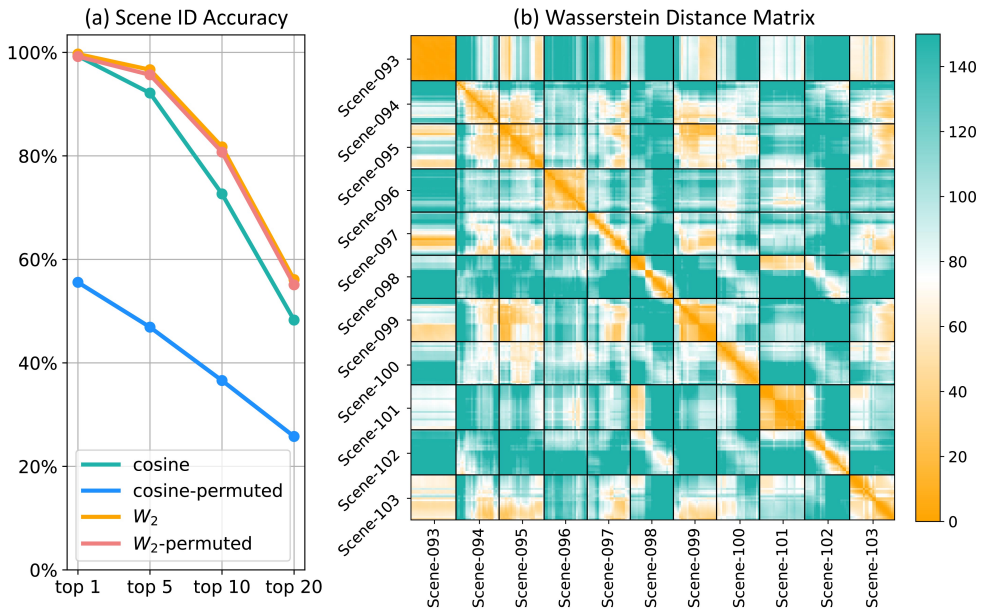


Figure 2.17: (a) Scene ID accuracy using the behavior embedding with difference distance metrics. (b) A matrix shows the Wasserstein distance between scenario segments, where each block contains the segments that belong to the same Scene ID.

Table 2.5: Accuracy of linear probing

AE	Contrastive AE	Masked AE	RealGen-AE
82.5%	67.1%	86.2%	**87.8%**

using the distance to find the closest segment. In Figure 2.17, we summarize the results in the left part, where top-k means we calculate the accuracy for the closest k segments. To validate the effectiveness of W_2-distance, we also permute the order of agents in the behavior embedding, which is denoted as *cosine-permuted* and W_2*-permuted*. We find that the cosine distance cannot deal with the permuted setting, as the order is important when stacking the embedding of agents. Our method performs well in top-1 and top-5 settings but badly in others, which could be explained by the fact that segments in one scene have very different behaviors. To validate this, we plot the distance matrix between the segments for 11 scenes in the right part of Figure 2.17. We find sub-blocks in the diagonal blocks, indicating that segments are similar in a small time interval but could be different when the time interval is large.

Linear probing of behavior embedding. Linear probing, a commonly employed method

Table 2.6: Results of realism metrics. Recon.-based and Retrieval-based means using the target scenario and retrieved scenarios as input for generation, respectively.

Category	Method	mADE	mFDE	Speed	Heading	SCR	ORR
Recon. -based	AE	**0.18**$\pm$**0.03**	0.41$\pm$0.06	**0.04**$\pm$**0.01**	0.10$\pm$0.01	**0.02**$\pm$**0.00**	**0.02**$\pm$**0.00**
	Masked AE	0.16$\pm$0.01	**0.39**$\pm$**0.01**	**0.04**$\pm$**0.01**	**0.09**$\pm$**0.01**	0.03$\pm$0.00	**0.02**$\pm$**0.00**
	Contrastive AE	0.92$\pm$0.02	1.47$\pm$0.04	0.12$\pm$0.00	0.36$\pm$0.02	0.04$\pm$0.00	0.04$\pm$0.00
	RealGen-AE	0.31$\pm$0.01	0.53$\pm$0.01	0.08$\pm$0.00	0.15$\pm$0.01	0.03$\pm$0.00	**0.02**$\pm$**0.00**
Retrieval -based	AE-KNN	14.3$\pm$0.03	16.4$\pm$0.05	0.57$\pm$0.01	0.59$\pm$0.02	0.15$\pm$0.01	0.15$\pm$0.01
	LCTGen	4.76$\pm$0.09	6.24$\pm$0.08	0.52$\pm$0.06	0.57$\pm$0.03	0.07$\pm$0.01	0.07$\pm$0.01
	LCTGen w/o z	14.2$\pm$0.07	16.7$\pm$0.09	2.04$\pm$0.04	1.42$\pm$0.00	0.16$\pm$0.02	0.13$\pm$0.04
	RealGen-AE-KNN	13.1$\pm$0.06	14.1$\pm$0.03	0.46$\pm$0.01	0.44$\pm$0.00	0.12$\pm$0.01	0.11$\pm$0.00
	RealGen	**1.54**$\pm$**0.04**	**1.21**$\pm$**0.03**	**0.21**$\pm$**0.03**	**0.21**$\pm$**0.01**	**0.05**$\pm$**0.00**	**0.04**$\pm$**0.00**

to assess representations in self-supervised learning (SSL), involves training a linear classifier using the derived embeddings (He et al., 2022). To train this classifier, we implemented heuristic rules to assign basic behavioral labels (acceleration, deceleration, stopping, keeping speed, left/right turn) to each agent's embedding. The results, along with comparisons with the baseline models, are presented in Table 2.5. These findings demonstrate that the embeddings generated by RealGen exceed all baseline models in terms of accuracy.

Evaluation of Retrieval Augmented Generation

Evaluating scene-level controllability and quality of scenario generation is an open problem due to the lack of quantitative metrics. Following most previous work (Feng et al., 2023a; Tan et al., 2023; Zhong et al., 2023b), we provide the results of the realism metrics and show the qualitative results of using RealGen for various downstream tasks.

Realism of generated scenario. To evaluate the realism of generated scenarios, we consider the following metrics and show the results in Table 2.6. We use the maximum mean discrepancy (MMD) (Gretton et al., 2012) to measure the similarity in velocity and direction between the original scenarios and the generated scenarios. We also compare the mean average displacement error (mADE) and the mean final displacement error (mFDE) for the average reconstruction performance. To evaluate scene-level realism, we calculate the scene collision rate (SCR) and the off-road rate (ORR) following the metrics defined in (Tan et al., 2023). The recon-based generation methods in Table 2.6 use the behavior of the target scenario as input. For this category, we

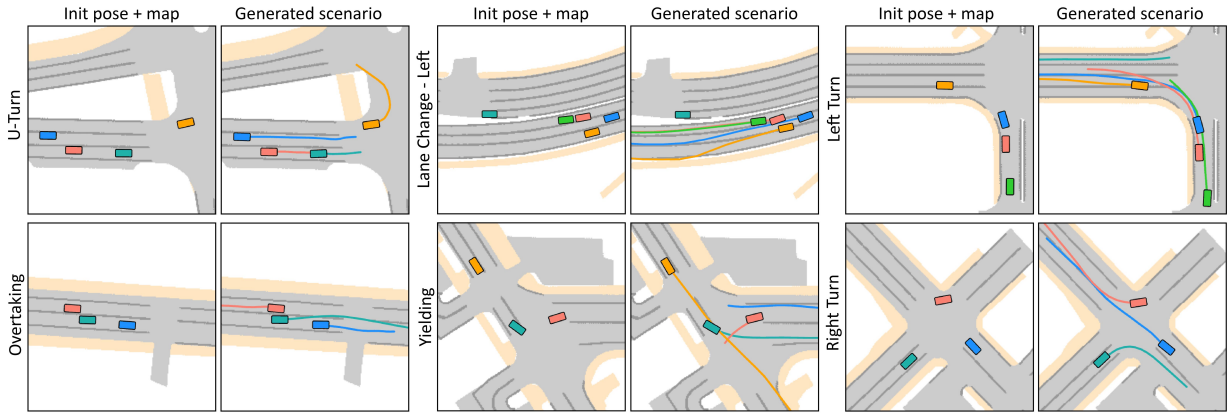


Figure 2.18: Examples of tag-retrieved scenarios generated by RealGen for six different tags.

compare RealGen-AE, only using the encoder and decoder modules, with three baseline methods. Due to the additional contrastive term, RealGen-AE achieves slightly worse performance than AE and Masked AE. However, RealGen is designed for retrieval-based generation, which uses retrieved scenarios rather than the target scenario as input. To fairly compare the generation performance, we design two baselines named AE-KNN and RealGen-AE-KNN, which use KNN to find the most similar behavior embedding to the target scenario, and use this embedding as input to the decoder for generation. According to the results, we find that RealGen achieves comparable performance as recon-based generation and outperforms baselines, which indicates the important role of the combiner in fusing the information of the retrieved scenarios.

Generating tag-retrieved scenarios. Now we explore the qualitative performance of using RealGen for tag-retrieved generation. Given a target behavior tag, we first obtain several template scenarios from a small dataset and use them to retrieve more scenarios from the training database, which will be used for generation in the combiner. Since there is no existing dataset with tags, we manually labeled six tags – U-Turn, Overtaking, Left Lane Change, Right Lane Change, Left Turn, Right Turn – for the nuScenes dataset to get template 1349 scenarios. We plot the generated scenarios for each tag in Figure 2.18, where the left part of each example shows the given initial pose and map, and the right part of each example shows the generated scenario from RealGen.

Safety-critical scenario generation. Beyond the tag mentioned above, RealGen demonstrates its capacity for in-context learning by generating critical and unseen crash scenarios,

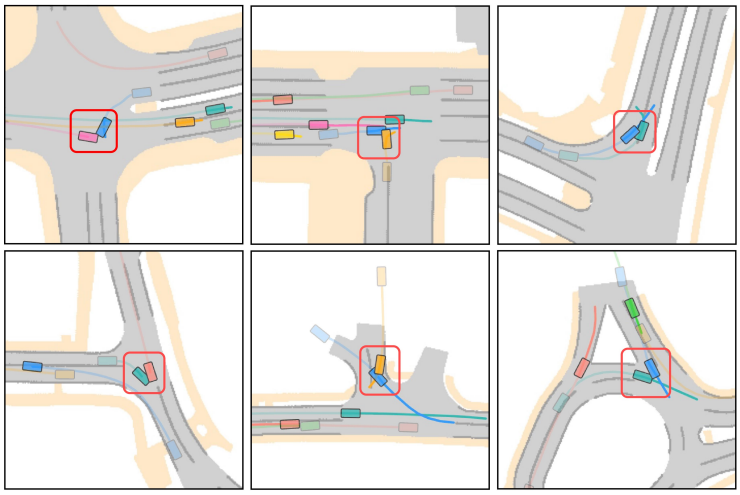


Figure 2.19: Examples of generating crash scenarios from RealGen. The shadow rectangles represent the initial positions of agents.

Table 2.7: Safety-critical scenario generation.

Method	Realgen-AE-R	RealGen-R	RealGen
Collision Rate (↑)	0.92	0.83	**0.59**

divergent from those of the training datasets. This is initiated by manually creating several crash scenario templates, guided by the scenarios recorded in the NHTSA Crash Report (NHTSA, 2023). Subsequently, we generate crash scenarios using existing initial poses and maps of the dataset. Figure 2.19 illustrates six instances in which the shadowed rectangles denote the initial positions of agents and the red box highlights the point of collision. To quantitatively assess the performance of safety-critical scenarios, we compare RealGen with two baselines. *RealGen-AE-R* random samples behavior embeddings in the Realgen-AE model, and *RealGen-R* random retrieves behavior embeddings. According to the results in Table 2.7, we find that the scenarios generated by RealGen using crash scenarios as templates achieve the highest collision rate, which means that RealGen has more efficiency.

Human evaluation of controllability. As there is no automatic way to evaluate controllability, we follow the protocol in (Tan et al., 2023) to perform A / B testing using human evaluation. For each category of scenarios, we select 10 examples from each generation algorithm. Each time, we ask a human evaluator to select one image from two images generated from our method

Table 2.8: Results of human evaluation of controllability.

Category	Left-Turn	Right-Turn	Left-Lane-Change	Right-Lane-Change	Straight
RealGen Preferred (%)	81.8	91.7	97.8	93.3	100.0
RealGen Score (0-5)	**4.27±1.05**	**4.27±0.69**	**3.96±1.94**	**4.17±1.93**	**3.94±2.27**
LCTGen Score (0-5)	2.15±1.44	2.08±1.19	2.42±1.95	2.0±1.96	2.14±2.31

and the baseline. The evaluator should also give a score (0-5) for both images. The final result is obtained by averaging the feedback from five evaluators. In Figure 2.20, we show some examples of the scenarios that we used for human evaluation. We report the preferred ratio of our method, as well as the absolute score (0-5) of both our method and the baseline. In Table 2.8, we find that RealGen-generated scenarios are highly preferred in most categories. The absolute score of RealGen is also much higher than that of LCTGen.

Table 2.9: Downstream task evaluation.

Method	Original	Random Aug.	RealGen Aug.
mADE (↓)	3.544	2.920	**2.309**
collision rate (↓)	0.049	0.037	**0.018**

Downstream task evaluation. A direct downstream task of our method is to use the generated data to augment the training dataset of trajectory prediction models. We use Autobots (Girgis et al., 2021) as a predictor and report the results trained on different datasets in Table 2.9. *Original* means using the original data in nuScenes (Caesar et al., 2020), *Random Aug.* means augmenting the original dataset with Gaussian noise, and *RealGen Aug.* means augmenting the original dataset with scenarios from RealGen. We observe that the model trained with the RealGen dataset achieves the lowest mADE and the lowest collision rate.

2.4 Summary

In this chapter, I discuss three of my previous works that use data-driven generative models to generate critical scenarios (Ding et al., 2023, 2018, 2020a). The success of such generative models is highly dependent on the enormous amount of data, which is the main reason for their impressive results in text and image generation. However, for physical-world tasks, such as

autonomous driving, robotics, and healthcare, the data scarcity problem has not been fully solved. The framework of using pure data-driven methods to generate critical data implies a chicken-and-egg paradox, since critical data are rare in the dataset.

Therefore, in the next chapter, I will introduce another framework for generating critical scenarios, which is based on adversarial optimization. With proper constraints, we can search for the parameters of scenarios to make the ego agent fail and the optimization does not require existing data of the critical scenario.

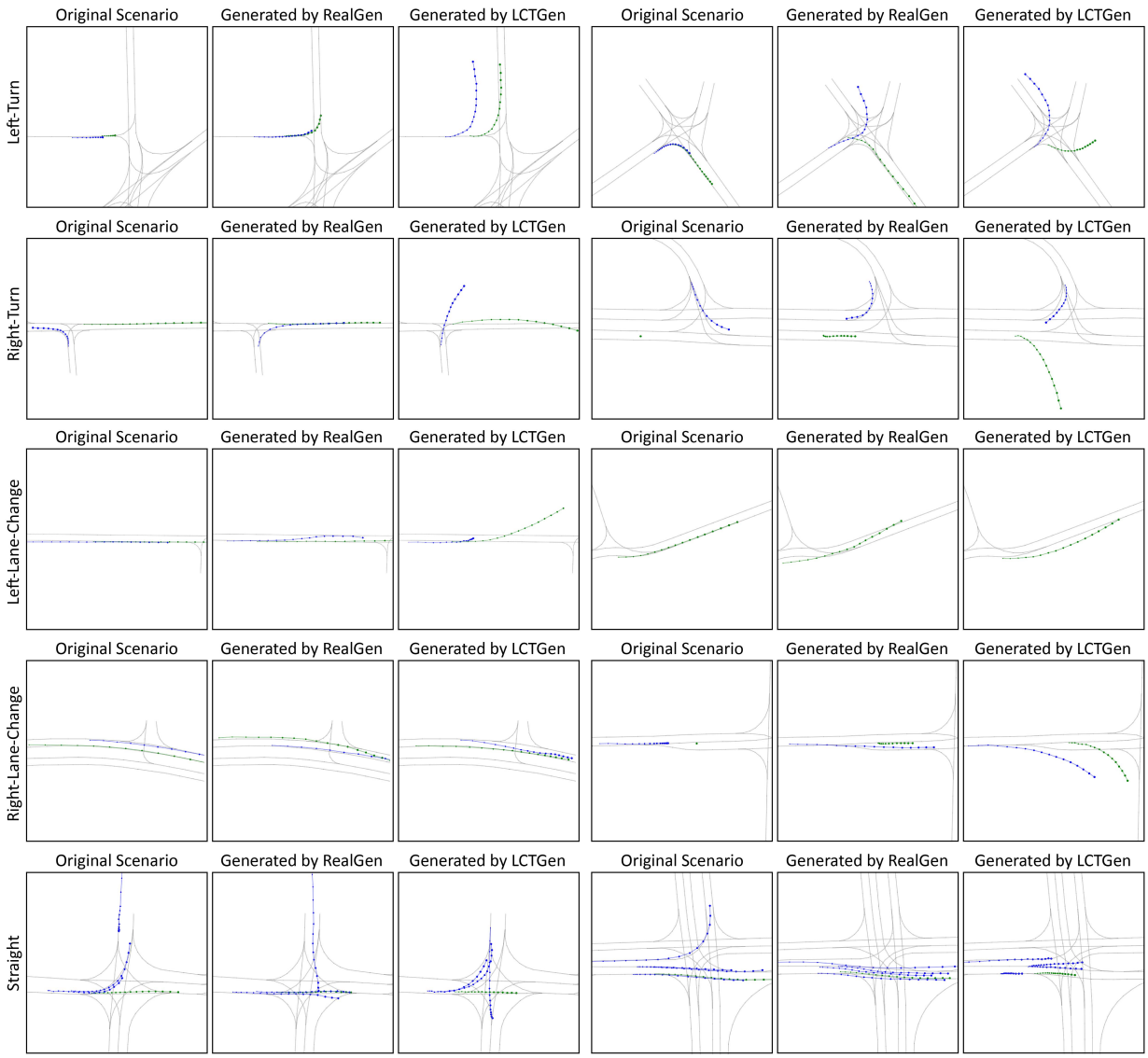


Figure 2.20: Examples of the scenarios we used for human evaluation of controllability. We used the map and initial positions of the original image (from nuScenes) to generate new scenarios using RealGen and LCTGen. We tested five scenario tags.

Chapter 3

Adversarial Generative Models

> The more you sweat in peace, the less you bleed in war.
>
> *Norman Schwarzkopf*

In this section, we consider a more efficient way for scenario generation, which actively creates risky scenarios by attacking the AV system. For example, we control a pedestrian to cross the road and intentionally make it collide with the AV. Although the AV may avoid the collision in most cases, we can still obtain safety-critical scenarios where accidents occur. This framework, named adversarial generation, consists of two components, one is the generator and the other is the victim model, that is, the AV. Then the targeted generation process can be formulated as

$$\hat{\theta} = \arg\min \mathbb{E}_{p_\theta(x)}[g(x, \pi_e)], \tag{3.1}$$

where $g(\cdot, \pi_e)$ is a quantitative function to indicate the performance of the policy π_e taken by the AV. We notice that adversarial generation will mainly focus on a specific small set of scenarios; therefore, it would be good to consider diversity by adding constraint or entropy of the distribution $\mathcal{H}$. Since we consider the influence of AV, this type is named Vehicle-in-the-loop testing in previous work (Fernández Llorca and Gómez, 2021). Since the autonomous driving system consists of multiple modules, we divide these methods according to the type of victim models. When the model is used for single frame inputs, e.g. object detection and segmentation, we only need to generate static scenarios. When the victim model requires a sequential testing case, we

generate dynamic scenarios that contain the motion of all objects.

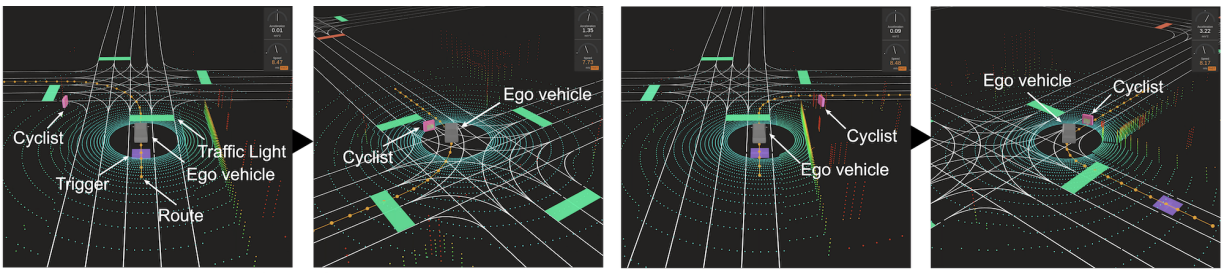
3.1 Generation with Reinforcement Learning

Today, the performance of most perception and prediction algorithms is quite sensitive to the imbalance of the training data (also known as the long tail problem (Cui et al., 2019; Lee et al., 2019)). Rare events are often difficult to collect and easily neglected in the huge data flow, which greatly challenges the real-world application of robots, especially in safety-critical domains, e.g. autonomous driving.

In the industry, companies usually resort to simulations to reproduce the safety-critical scenarios collected during their test driving. One brute-force method is to create risky scenarios by adjusting all variables in one scenario with the grid search to create similar risky scenarios, which is time- and labor-intensive. An alternative method known as worst-case evaluation (Kou et al., 2008) is proposed to search for the worst cases to evaluate controllers in the vehicle field. Although some cases excavated by worst-case evaluation may be useful, some extremely risky scenarios are almost impossible to appear in the real world. The algorithms evaluated in these scenarios may not represent the deployment in the real world. Therefore, other previous works concentrate on generating risky and reasonable scenarios (Zhao et al., 2017, 2016) with importance sampling. These works show the possibility of modeling the scenarios with probability distributions and more efficient sampling from them.

With the recent popularity of deep generative models (Goodfellow et al., 2014; Kingma and Welling, 2013), another promising way is directly generating safety-critical scenarios instead of sampling existing data. The advantage of the generative model is that more diverse scenarios, even open-world scenarios that do not exist in the collected data, could be created. Unfortunately, the most prevalent deep generative models are not designed for generating rare events. These models generate random samples that are similar to the given dataset without elaborated explicit generating processes (Ding et al., 2018), which is not in line with our goal of generating rare and risky events with extremely low probability.

Another difficulty in generating risk scenarios is finding the appropriate representation, since



(a) Left-turn Scenario (b) Right-turn Scenario

Figure 3.1: Two safety-critical scenarios generated using our method.

scenarios consisting of statistics and dynamic objects, routes, and maps are difficult to model. A low-dimensional and easy-to-sample representation would dramatically increase the efficiency of generating new scenarios.

To address such challenges, we propose to combine the task algorithm and the data generation model in this work. Two generated scenario examples are shown in Figure 3.1. First, we use a factorized graphic model to represent the traffic scenario, which is inspired by (Wheeler and Kochenderfer, 2016) and (Wang et al., 2018). The motivation is to integrate human knowledge to create scenarios by factorizing the scenarios into (independent) conditional probabilities. We refer to probabilities as the building blocks in this work since they could be shared among different scenarios. Using this graphic model allows us to generate new scenarios by sampling the distributions according to the dependence relationship. These building blocks vary the initial parameters to cover a wide range of scenarios, which was shown to be an effective method for modeling transportation (Wheeler and Kochenderfer, 2016). After representing the scenario, we consider the generative model as an agent (or a generator) and regard one specific task algorithm as the environment (or a discriminator). The parameters (e.g. target speed) and the route of the task algorithm will be used as input states for the agent. The riskier the scenario the agent generates, the higher the reward it receives. We sample the input states from a uniform distribution during model training. Therefore, given a specific task algorithm, our proposed framework adaptively generates safety-critical scenarios for different routes and input parameters, even for settings that are not seen in the training stage.

In summary, this method makes three contributions to the autonomous driving safety literature:

- We develop a framework to generate traffic scenarios by sampling from the joint distributions of autoregressive building blocks.
- We design an algorithm that uses the task module to guide the generation module for generating safety-critical scenarios.
- Autonomous driving researchers can use scenarios to evaluate the safety of driving algorithms. The method could be extended to create test scenarios for other types of robots.

3.1.1 Representing scenario with Probabilistic Models

Most traffic scenarios could be divided into several building blocks, e.g., location, orientation, and velocity of traffic participants (Kar et al., 2019). Mathematically, we use a probability graphic model to represent a scenario in which each node represents a building block.

The most intuitive way to model the scenarios is using an undirected graph like (Wheeler and Kochenderfer, 2016) when we have no information about the elements in the scenario. We use a joint probability $p(x_1, x_2, ...x_n)$ to represent one scenario, where x_i is the distribution of setting parameters, such as the orientation or speed of a cyclist. Then, we represent the scenario with an undirected graph $\mathcal{G}$:

$$p(x_1, ..., x_n) = \frac{1}{Z} \prod_{v \in V} \Psi(\boldsymbol{x}_v), \tag{3.2}$$

where V indicates all nodes in the graph $\mathcal{G}$ and $\Psi(\cdot)$ is the potential function to describe the correlation associated with the nodes of $\mathcal{G}$. Z is the partition function that ensures that the representation satisfies the probability requirement. The drawbacks of this representation are two-fold: (1) Z is usually intractable because it requires the integration of all latent variables, and (2) $\Psi(\boldsymbol{x}_c)$ is difficult to define. Therefore, we simplify this representation to a directed graph $\mathcal{H}$ by introducing human knowledge:

$$p(x_1, ..., x_n) = \prod_{i \in V} p(x_i) \prod_{j \in V} p(x_j | x_{\mathbf{PA}_i(\mathcal{G})_j}), \tag{3.3}$$

where $\mathbf{PA}_i(\mathcal{G})_j$ is the set of parents of j. The first production in (3.3) represents the independent building blocks, while the second production usually represents the blocks with an autoregressive

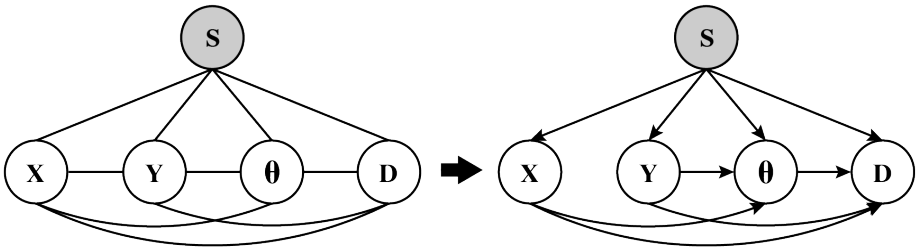


Figure 3.2: One example of converting an undirected graphic model to a directed one with human knowledge.

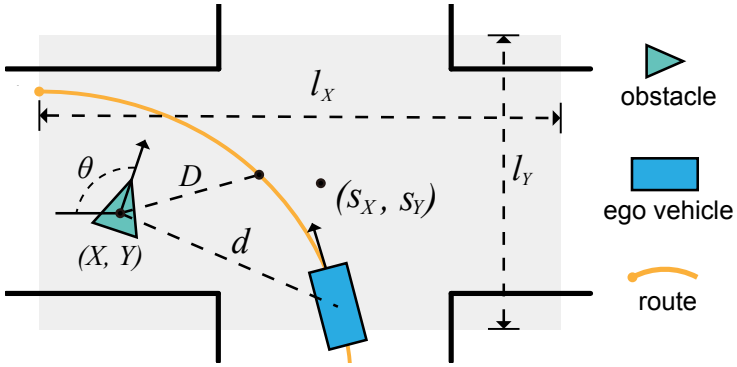


Figure 3.3: Explanation of the nodes in Fig .3.2. The scale and shift parameters in (3.8) are also shown in this figure.

structure. The autoregressive structure means that several nodes $\{v_0, ..., v_g\} \in V$ form a specific group with the following relation:

$$p(x_j|x_0, ..., x_{j-1}), \quad \forall\, j \in \{1, ..., g\}. \tag{3.4}$$

Human knowledge of traffic scenarios could help with factorizing the graph and designing the relation among all blocks.

A factorization example is shown in Figure 3.2 and an example of this graph is explained in Figure 3.3. In this scenario, we want to generate one cyclist and move it when the AV reaches the trigger distance to the cyclist. We define four nodes to represent the scenario: generate position (X, Y), orientation (θ), and trigger distance (D). (S) represents the input states, such as the route the AV will follow and its target speed. The factorized result shown on the right of Figure 3.2 is

the same as:

$$p(X, Y, \theta, D|S) = p(X|S)P(Y|S)p(\theta|S, X, Y)p(D|S, X, Y, \theta). \tag{3.5}$$

The reason for the simplification is that the orientation of the cyclist only depends on the position of the spawn point, given that we want to make the cyclist collide with the AV. For the trigger distance D, it is a conditional probability that depends on the value of the position and the orientation of the spawn point. We only adopt one reasonable representation according to prior knowledge, although there could be other reasonable ways.

In this method, we focus more on critical scenarios that have few participants, e.g., one AV and one dynamic obstacle. However, complex scenarios in the real world can involve hundreds of vehicles and pedestrians, which is challenging to represent with this method. In those cases, we may need to divide the nodes into groups and model the scenario with groups rather than nodes. This is still an unsolved problem that needs to be studied in the future.

In our implementation, we use the Gaussian distribution $\mathcal{N}(\mu, \sigma)$ to model continuous blocks and the multinomial distribution to model the discrete ones. Neural networks (NNs) are used for conditional probability inference. The reparameterization trick (Kingma and Welling, 2013) is used to perform the sampling with backpropagation:

$$\mu_k, \sigma_k \leftarrow \mathcal{M}_k(S, a_{k-1}), \tag{3.6}$$

$$a_k = \mu_k + \sigma_k \times \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1), \tag{3.7}$$

where a_k is the action sampled from the k-th node. $\mathcal{M}_k$ is the model that represents the conditional distribution of the k-th action. Then a_k needs to be rescaled and shifted to represent the parameter of the real-world scenario:

$$b_k = a_k \times l_k + s_k, \tag{3.8}$$

where l_k and s_k are the range and average value of the k-th action, respectively. We note that b_k will be truncated if its value is beyond the range boundary.

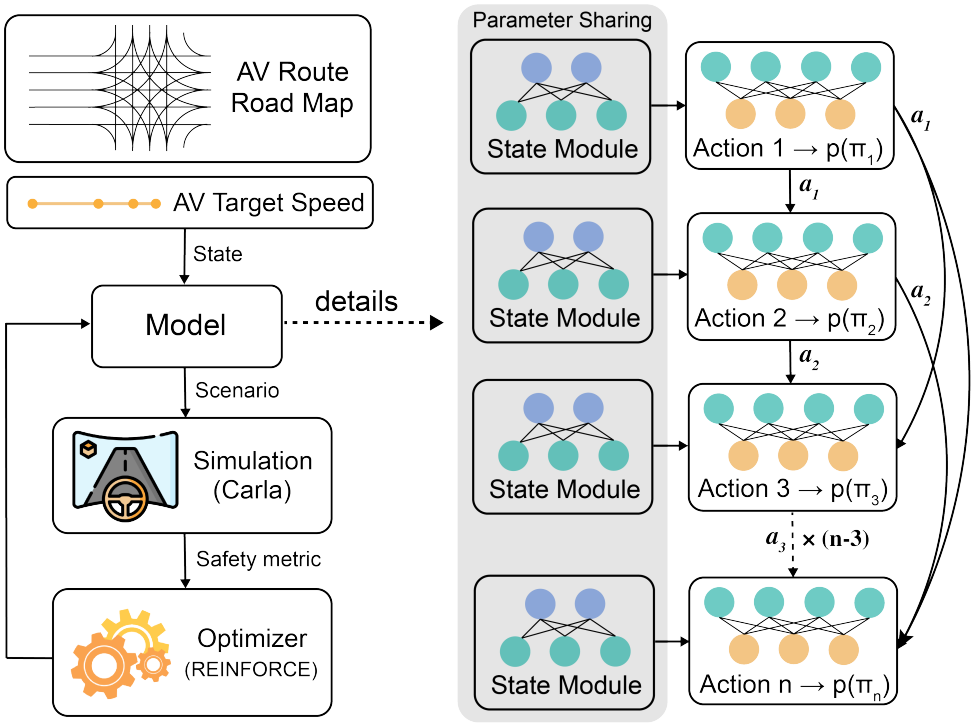


Figure 3.4: The proposed framework (left) and the structural details (right). Our model consists of two parts: the state module and the action module, both of which are implemented with linear layers.

3.1.2 Learning to Collide (L2C)

We have shown our proposed framework in Figure 3.4. We first introduce how we represent the scenario generation process (right part of Figure 3.4), then describe the pipeline of our model and the training procedures (left part of Figure 3.4).

Scenario generation framework

With the language of reinforcement learning, we regard the aforementioned scenario generation model as an agent, and the full-stack AV algorithm to be evaluated as an environment. Our goal is to obtain a safety-critical scenario generation model.

The state of the environment has two parts. The first part contains information on route $\boldsymbol{\eta}$, which is the reference trajectory for the task algorithm, and the road map with lane information. The second part consists of several parameters of the task algorithm $\boldsymbol{\xi}$, which influence the

Algorithm 3: Training Procedure of Proposed Framework

Initiate max epoch number E
Initiate rewards $\boldsymbol{r_b}$, $\boldsymbol{r_p}$
Build the model $\mathcal{M}$ with (3.3) and initiate parameters ϕ
for $e \leftarrow$ *1 to* E **do**
  for $i \leftarrow 1$ *to* N **do**
    Sample route from the distribution $\boldsymbol{\xi}^{(i)}, \boldsymbol{\eta}^{(i)} \sim S$
    Get scenario parameters $\boldsymbol{a}^{(i)} \leftarrow \mathcal{M}\ (\boldsymbol{\xi}^{(i)}, \boldsymbol{\eta}^{(i)}; \phi)$
    $\rho_{AV}, \rho_o \leftarrow$ Simulator $(\boldsymbol{\xi}^{(i)}, \boldsymbol{\eta}^{(i)}, \boldsymbol{a}^{(i)})$
    $\mathcal{R}^{(i)} = -r_d(\rho_{AV}, \rho_o) + r_b - r_p$
  $\nabla_\phi \mathcal{J}(\phi) = \frac{1}{N} \sum_i^N \nabla_\phi \log(\pi_\phi) \mathcal{R}^{(i)} - \lambda \mathcal{H}(\pi_\phi)$
  $\phi = \phi - \alpha \nabla_\phi \mathcal{J}(\phi)$

decision-making ability of the AV. One simple example is the target speed, which our generative model needs to consider to adjust the position of the obstacles.

The reward function consists of three parts:

$$\mathcal{R} = -r_d(\rho_{AV}, \rho_o) + r_b - r_p(\rho_o, \gamma), \tag{3.9}$$

where ρ_{AV} and ρ_o represent the positions of the AV and the obstacle (the cyclist in our experiments), respectively. The first term is the risk metric r_d, in which we use the distance between the obstacle we generated and the AV to represent:

$$r_d(\rho_{AV}, \rho_o) = \|\rho_{AV} - \rho_o\|_2. \tag{3.10}$$

In the simulation, the distance between two rigid objects is always larger than 0, which means the distance cannot be used to represent the collisions. Therefore, we will provide the agent an extra bonus r_b if a collision happens:

$$r_b = \begin{cases} R_b, \text{collision} = True, \\ 0, \text{collision} = False. \end{cases} \tag{3.11}$$

Finally, we use a penalty $\boldsymbol{r}_p$ to avoid a special case in which obstacles are spawned too close to the route, which is not a reasonable scenario. We use a threshold γ to determine whether this

Table 3.1: Hyper-parameters in our experiments

Hyper-parameter	Description	Value
E	max epoch number	100
α	learning rate	0.008
N	batch size	16
λ	weight for policy entropy	0.001
R_b	reward of collision bonus	10
R_p	penalty of route occupy	20
γ	threshold in $\boldsymbol{r}_p$	3
s_X	average value of X	0
s_Y	average value of Y	0
s_θ	average value of θ	180
s_D	average value of D	20
l_X	scale of action X	100
l_Y	scale of action Y	18
l_θ	scale of action θ	360
l_D	scale of action D	40
h_s	# of variable in state module	64
h_a	# of variable in action module	32

penalty is implemented:

$$r_p = \begin{cases} R_p, \|\eta_i - \rho_o\|_2 < \gamma \ \ \forall i \\ 0, \text{otherwise} \end{cases}. \tag{3.12}$$

where η_t is the i-th route waypoint.

Optimization process

We follow the policy gradient method REINFORCE (Williams, 1992) to solve our optimization problem. The gradient for updating model parameter ϕ is:

$$\nabla_\phi \mathcal{J}(\phi) = E_{a\sim\pi_\phi}\left[\nabla_\phi \log(\pi_\phi)\right]\mathcal{R}(a) \approx \frac{1}{N}\sum_i^N \nabla_\phi \log(\pi_\phi(a_i))\mathcal{R}(a_i), \tag{3.13}$$

where $\mathcal{J}(\phi) = E_{a\sim\pi_\phi}[\mathcal{R}]$ is the objective function, and a is the action sampled from the policy distribution π_ϕ. To encourage the diversity of the policy, we add an entropy term to the objective

function as proposed in (Haarnoja et al., 2018):

$$\mathcal{H}(\pi_\phi) = -\int \pi_\phi(x) \log \pi_\phi(x) dx. \tag{3.14}$$

When we use an autoregressive Gaussian distribution to model the policy π_ϕ, we are still able to calculate the joint probability with the chain rule:

$$\log P(\boldsymbol{\pi}) = \log P(\pi_0) + \sum_{i=1}^{n} \log P(\pi_i | \pi_0, ..., \pi_{i-1}), \tag{3.15}$$

where each term is calculated according to the density function of Gaussian distribution. The entropy is also easy to calculate, since the joint distribution is still a Gaussian distribution. The entire algorithm is shown in **Algorithm** 3.

3.1.3 Experiment and Analysis

Experimental settings

We implemented our algorithm with Pytorch (Paszke et al., 2019) and used Adam (Kingma and Ba, 2014) as the optimizer. Details about the hyperparameter of our experiments are listed in Table 3.1. We use *Carla* (Dosovitskiy et al., 2017) as our simulation platform, and we use Carla ScenarioRunner as the backbone to generate traffic scenarios. We modify the *Scenario04* and use it as our testbed to evaluate our framework. The description of the setting of this scenario is discussed in Section 3.1.1.

The AV algorithm to be evaluated is a simple trajectory that follows a model with a PID controller. we use the visualization and debugging tool CarlaViz, which is an open source tool developed by our group. We use a single fully connected layer to build our state and action modules, and the number of hidden variables is summarized in Table 3.1.

Verification experiment

We conduct a verification experiment to show that our proposed framework can generate risky scenarios. We trained our model on 10 different routes and randomly sampled target speeds from

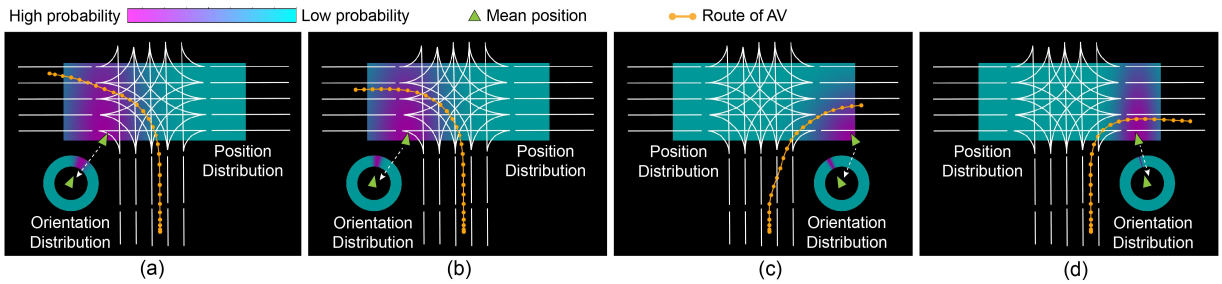


Figure 3.5: Learned distributions of scenario parameters with different routes.

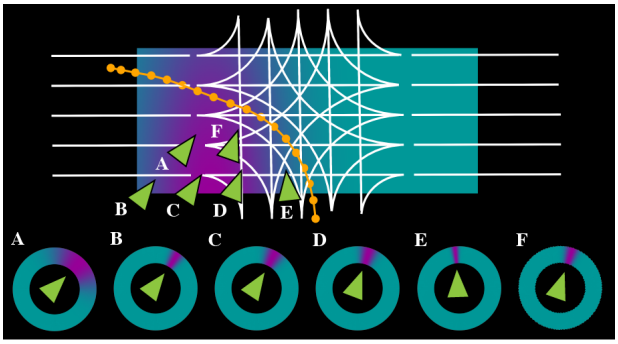


Figure 3.6: Examples of conditional distribution $p(\theta|X = x, Y = y, S = s)$.

$20km/h$ to $50km/h$. Then we test this model on 4 different routes and the results are displayed in Figure 3.5. It is shown that our model outputs different policies when different routes and speeds are fed in. For left-turn scenarios, the distributions of the positions fall to the left of the intersection; and for right-turn scenarios, the distributions are on the right. We note that the high probability regions are exactly the riskiest ones a human driver will encounter in reality, i.e., the driver tends to ignore these blind spots when passing through these intersections.

To verify the contribution of our autoregressive structure, we sample different initial positions (x and y) from $p(X|S)$ and $p(Y|S)$ and use them as conditions to obtain the orientation output $p(\theta|X = x, Y = y, S = s)$. The results are shown in Figure 3.6. As expected, when different x and y are used as conditions, the orientation is quite different, which proves that our model can learn the dependencies between different policies.

Comparison experiment

In this section, we construct four baselines for comparison:

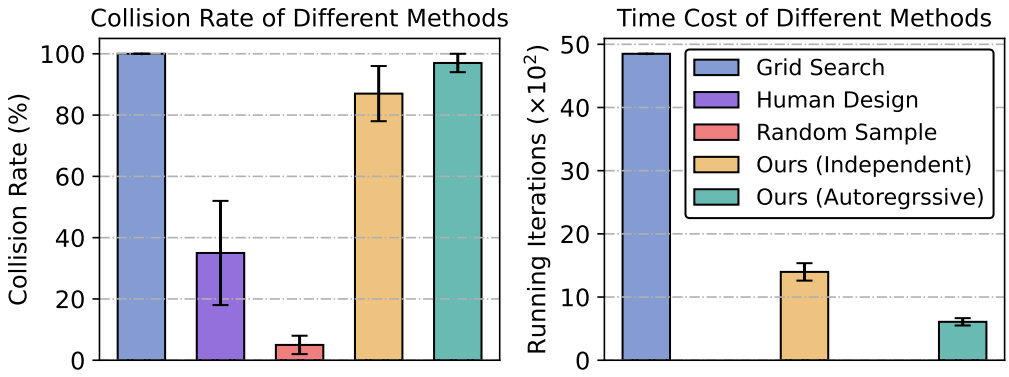


Figure 3.7: The results of the collision rate and the number of iterations required to reach stability.

Grid Search. This is the easiest way to search for risk scenarios. Since we need to consider the combinations of all policies, the search space could exponentially grow as the dimension of scenario representation increases. To reduce the searching expense, we discretize the policies by steps [4, 3, 20, 10] for [X, Y, θ, D] and search all combinations to solutions.

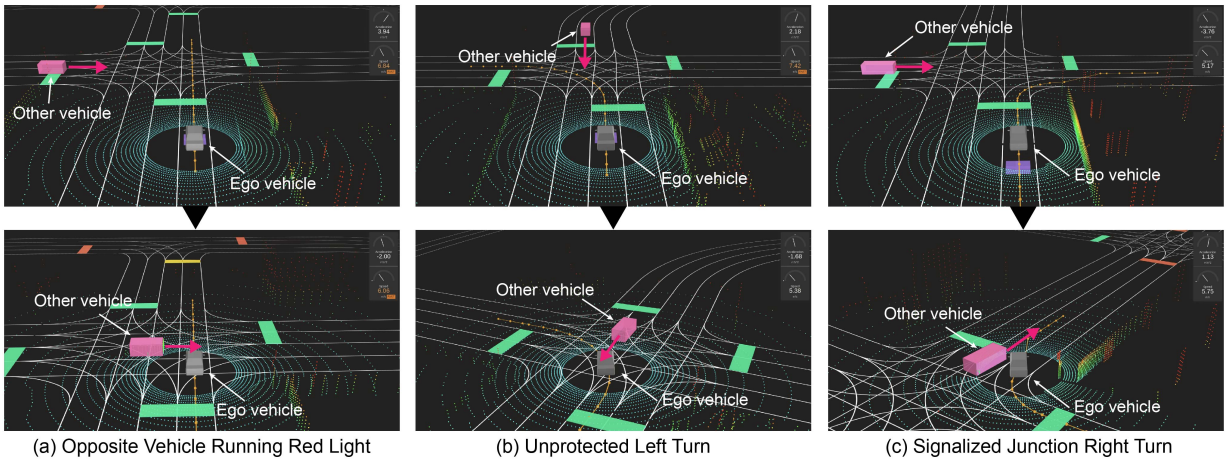
Human Design. The Carla Scenario Runner Library is used for artificially creating scenarios in Carla Autonomous Driving Challenge. This competition aims to test and evaluate AV algorithms in risky scenarios. We use the same parameters and procedures of the testbed of this competition as our baseline.

Random Sampling. In this baseline, we sample all policies from the uniform distribution. This method is designed to simulate the methods that are not combined with task algorithms, in which case only random scenarios are generated.

Independent Policy. This method is almost the same as our proposed one. The only difference is that we treat all policies as independent blocks, i.e., each policy is modeled by a Gaussian distribution only conditioned on the state S.

We used two metrics for comparison: the collision rate after the model achieved a stable policy and the number of iterations required for the model to reach stability. We conducted 30 experiments on each method with different routes and target speeds. The experimental results are shown in Figure 3.7.

The results of the collision rate show that the human design method has very poor adaptability (large variance) because the artificially designed parameters are only useful when the AV



(a) Opposite Vehicle Running Red Light (b) Unprotected Left Turn (c) Signalized Junction Right Turn

Figure 3.8: Generated safety-critical scenarios using our method.

algorithm satisfies some conditions, e.g. the target speed belongs to a specific region. For the random sampling method, the low rate is in line with our expectations, since there is no guidance to help generate risky scenarios. The independent method has a slightly lower collision rate than our method because the independence among the actions makes the generating process very inefficient. Decoupling the policies may cause the loss of some constraints.

The results of the time cost show that the *Independent Policy* method requires more time to reach the stable state than our method because the variance of each policy could influence other policies, making the training process unstable. Even if we have selected large steps to discretize the space (leading to very few collision scenarios), the grid search method still requires too much time.

In Figure 3.7, all results have a variance with the exception of the grid search because, for different routes or target speeds, the search process must be carried out from scratch, which is a huge drawback of the grid search method. All of the other methods are adaptive to generate different risky scenarios according to the input states. The lower the variance, the more adaptive the method is.

Results of Other Scenarios

We tested three other scenarios to verify the effectiveness of our method. We chose these scenarios because they all cause millions of losses every year according to the report on pre-crash sce-

narios from the National Highway Traffic Safety Administration (NHTSA) (Najm et al., 2007).

Opposite Vehicle Running Red Light. The ego vehicle passes through an intersection along a straight route, while another vehicle takes priority over the ego vehicle running a red traffic light. The action space has three dimensions: position X, position Y, and speed V. The orientation of the other vehicle is fixed in this scenario.

Unprotected Left Turn. The ego vehicle turns left, while another vehicle approaches from the opposite direction. To avoid a collision, the ego vehicle should wait for the other vehicle or accelerate to cross the intersection. The action space has the same three dimensions as the last scenario.

Signalized Junction Right Turn. The ego vehicle turns right, while another vehicle approaches from the left. The action space has the same three dimensions as the last scenario.

The risk scenarios generated with our framework are shown in Figure 3.8. As shown in the figure, in all three new scenarios, our method also finds the risky distributions of all building blocks.

Exploration of Solution Space

We explore the solution space of the risky scenario parameters. Six different stable solutions during our experiments are shown in Figure 3.9. All of them are initialized with different values. Although we model the solution with Gaussian distributions, the solution space may not be a convex set, even not only have one mode. For example, both sides of the route should be feasible solutions, but using the REINFORCE algorithm with Gaussian policies only models a subspace of the entire solution space.

In our future work, one possible improvement is separating the action space and the scenario representation space. While the action space is still modeled by the Gaussian distribution, the scenarios are modeled with implicit and complex distributions. Tools such as VAE (Kingma and Welling, 2013) and flow-based models (Kingma and Dhariwal, 2018) could be used to build a mapping from a simple Gaussian distribution to a complex distribution. Then a joint optimization of the mapping model and the generative model should be done to achieve our goal.

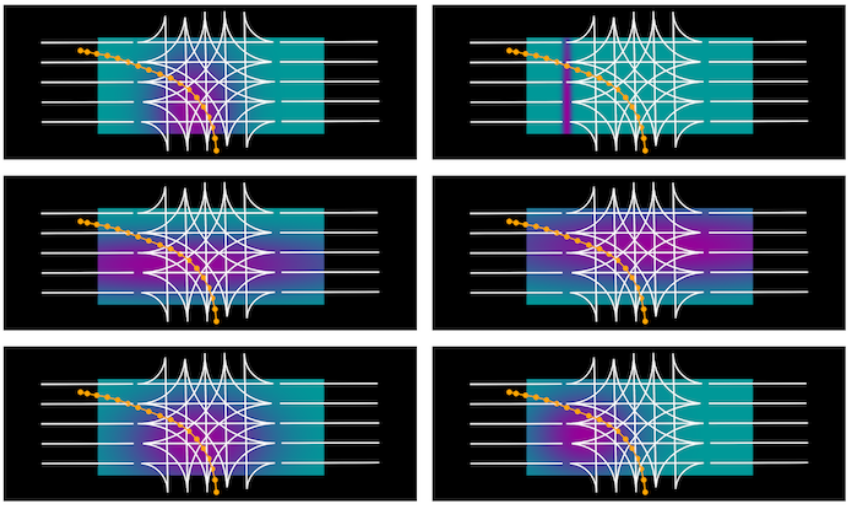


Figure 3.9: The training results have large diversity since the feasible solution is a continuous space rather than a single point.

3.2 Diverse Scenario Generation with Adaptive Sampler

Robustness and safety are crucial factors in determining whether a decision-making algorithm can be deployed in the real world (Fryman and Matthias, 2012). However, most of the data collected from simulations or in the wild are skewed to redundant and highly safe scenarios, which leads to the long tail problem (Cui et al., 2019). Furthermore, a self-driving vehicle has to drive hundreds of millions of miles to collect safety-critical data (Kalra and Paddock, 2016), resulting in expensive development and evaluation phases. Meanwhile, a large number of safe Reinforcement Learning (RL) algorithms (Garcıa and Fernández, 2015) have recently been proposed, yet the evaluation of these algorithms uses mostly uniform sampling scenarios, which have been proven to be insufficient due to the poor coverage of rare risky events.

The adversarial attack (Chen et al., 2017b; Nazemi and Fieguth, 2019) is widely used to obtain specific examples when assessing the robustness of the model. This method only addresses extreme conditions, and thus it does not provide comprehensive performance evaluations of the system. Researchers (Fawzi et al., 2018; Shafahi et al., 2018) point out that there will always be loopholes in a neural network (NN) that can be attacked, hence testing at different stress levels is considered to provide more information about the robustness of the system. On the other hand,

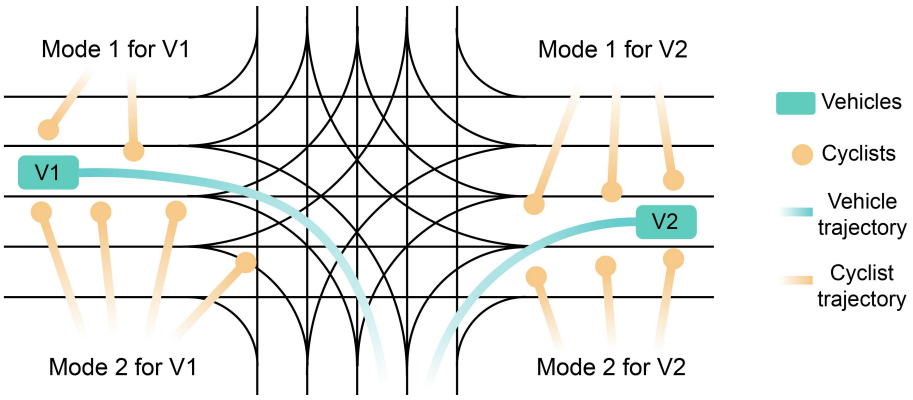


Figure 3.10: An example of multimodal safety-critical driving scenario.

although the perturbation is limited during the attack, there is no guarantee that the obtained samples will occur in the real world. It is a waste of resources to request robots to pass the tests that are unlikely to happen in practice.

Real-world scenarios are complicated with a huge number of parameters, and risk scenarios do not always occur within certain modalities (Yan et al., 2019). The multimodal distribution is a more realistic representation, for example, accidents could occur in different locations for a self-driving car, as shown in Figure 3.10. Although previous work (Ding et al., 2020b; Kuutti et al., 2020) tried to search for risk scenarios in the RL framework, they only used the single-mode Gaussian distribution policy, leading to overfitting and unstable training problems. Covering diverse testing cases provides a more accurate comparison of algorithms. Even if a robot overfits one specific risk modality, it will fail to handle other potential risk scenarios. To our knowledge, few people have explored multimodal estimation of safety-critical data.

In this work, we use a flow-based generative model to estimate the multimodal distribution of safety-critical scenarios. We use the weighted maximum likelihood estimation (WMLE) (Wang, 2001) as the objective function, where the weight is related to the risk metric so that the log-likelihood of the sample will be approximately proportional to the risk level. We treat the algorithm that we want to evaluate as a black box and then get the risk value through the interaction with the simulation environment. To increase the generalization of generated scenarios, our generator also has a conditional input, so that the generated samples will be adaptively changed according to the characteristics of the task.

We model the whole training process as an on-policy optimization framework that shares the same spirit as the Cross-Entropy Method (CEM) (Rubinstein and Kroese, 2004), and we propose an adaptive sampler to improve sample efficiency. In this framework, we can dynamically adjust the region of interest of the sampler according to the feedback of the generator. The adaptive process is guided by the gradient estimated from the Natural Evolution Strategy (NES) method (Wierstra et al., 2014). During the training stage, the sampler focuses on the unexplored and risky areas and finally completes the multimodal modeling. We also consider the distribution of real-world data when designing the risk metric to ensure that the generated data have a high probability of occurrence in the real world.

We carried out extensive experiments on the decision-making task of autonomous driving in an intersection environment. A safety-critical generator is trained, analyzed, and compared with traditional methods. Our generator outperforms others in terms of efficiency and multimodal coverage capability. We also evaluate the robustness of several RL algorithms and claim that our generator is more informative than the uniform sampling methods. In summary, the contribution is three-fold:

- We propose a multimodal flow-based generative model that can generate adaptive safety-critical data to efficiently evaluate decision-making algorithms.
- We design an adaptive sampling method based on black-box gradient estimation to improve the sample efficiency of multimodal density estimation.
- We evaluated a variety of RL algorithms with our generated scenarios and provided empirical conclusions that can help design and develop safe autonomous agents.

3.2.1 Flow-based Models

The variable $x \in \mathcal{X}$ represents parameters that build a scenario, for instance, the initial position of a pedestrian or the weather conditions in the self-driving context. The variable $y \in \mathcal{Y}$ represents the properties of the task, such as the target position or the target velocity. $\pi(a|x, y)$ is the task algorithm that takes the scenario x and the task condition y as input and outputs an action $\boldsymbol{a} \in \mathcal{A}$. With a risk metric $r : \mathcal{X} \times \mathcal{A} \to \mathcal{R}$, each scenario corresponds to a value that indicates safety

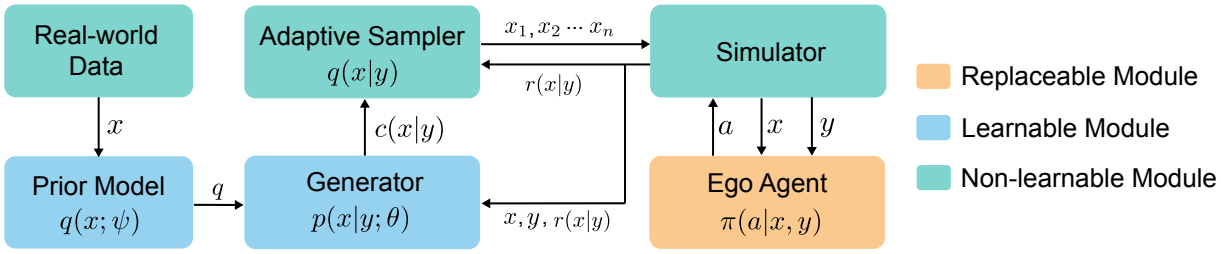


Figure 3.11: Diagram of proposed framework. The modules that have learnable parameters are shown in blue.

under π.

Instead of exploring extreme scenarios that produce low $r(x|y)$ as in an adversarial attack field, we aim to estimate a multimodal distribution $p(x|y)$ where the log-likelihood is proportional to the value of the risk measure. Then we can efficiently generate scenarios that have different risk levels by sampling from $p(x|y)$. The condition y follows the distribution $p(y)$. A sampler $\pi(x|y)$ is used to collect training data. We also assume that real-world data of similar scenarios is accessible and has a distribution $q(x)$. The pipeline of our proposed evaluation method is shown in Figure 3.11. We explain the details of three learnable modules in the following sections.

3.2.2 Multi-modal Generation (MMG)

Pre-trained Prior Model

Firstly, we consider the probability that each scenario happens in the real world to make the result practical. For that, we pre-train a generative model $q(x;\psi)$ to approximate the distribution of the real data $\mathcal{D}$. The objective of the training is to maximize the following log-likelihood:

$$\hat{\psi} = \arg\max_{\psi} \mathbb{E}_{x\sim\mathcal{D}} \log q(x;\psi). \tag{3.16}$$

We select RealNVP (Dinh et al., 2016), a flow-based model, to implement $q(x;\psi)$ for exact likelihood inference. In a flow-based model, a simple distribution $q(z)$ is transformed into a complex distribution $p(x)$ by the change of the variable theorem. Suppose that we choose $z \sim$

$\mathcal{N}(0,1)$ to be the simple distribution. Then we have the following equation to calculate the exact likelihood of x:

$$q(x) = q(z)\left|\frac{\partial z}{\partial x}\right| = p(f(x))\left|\frac{\partial f(x)}{\partial x}\right|, \tag{3.17}$$

where we have an invertible mapping $f : \mathcal{X} \to \mathcal{Z}$. For more details on the flow-based model, refer to (Dinh et al., 2016).

After training, scenarios can be generated by $x = f^{-1}(z)$ where z is sampled from $\mathcal{N}(\mathbf{0}, \sigma)$. A smaller σ will make the samples more concentrated, therefore, the generated scenarios will have a higher likelihood. Since our model is trained with WMLE, the high-likelihood samples are also corresponding to high risk.

Multu-modal Generative Model

We formulate the safety-critical data generation as a density estimation problem. The traditional way to estimate the multimodal distribution $p(x|y)$ using a deep generative model is to maximize the likelihood of data. To integrate the risk information, we solve the problem using WMLE (Wang, 2001). For one data point x_i, we have:

$$L(x_i|y_i;\theta) = p(x_i|y_i;\theta)^{w(x_i)}, \tag{3.18}$$

$$\log L(x_i|y_i;\theta) = w(x_i|y)\log p(x_i|y_i;\theta), \tag{3.19}$$

where $w(x_i)$ is the weight and $p(x_i|\boldsymbol{y}_i;\theta)$ is our generator with learnable parameter θ, corresponding to the i-th data point. Assuming that we have a sampling distribution $\pi(x|y)$ of x, then our objective is:

$$\hat{\theta} = \arg\max_{\theta} \mathbb{E}_{x\sim\pi(x|y),y\sim p(y)} \log L(x|y;\theta). \tag{3.20}$$

The definition of $w(x|y)$ is relevant to both $r(x|y)$ and $q(x;\psi)$:

$$w(x_i) = r(x_i|y) + \beta q(x_i;\psi), \tag{3.21}$$

where β is a hyperparameter to balance $r(x)$ and $q(x;\psi)$.

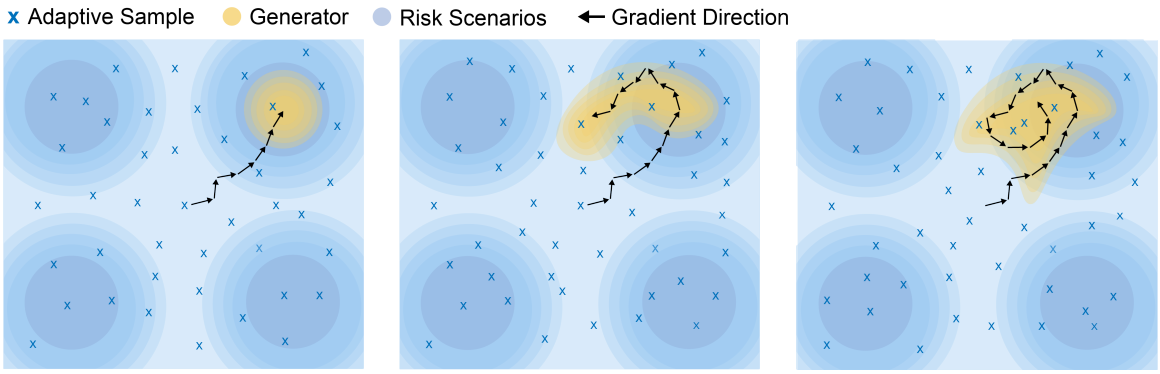


Figure 3.12: An illustration example of multi-modal estimation using the uniform sampler.

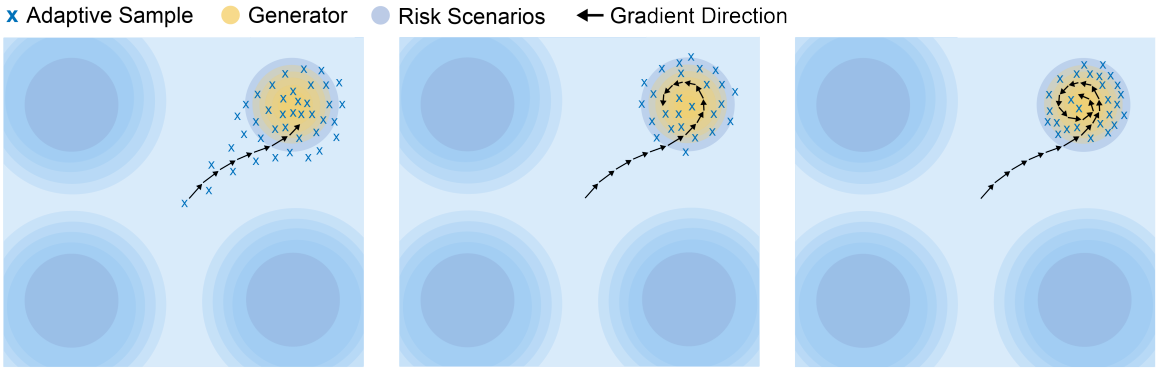


Figure 3.13: An illustration example of multi-modal estimation using the generator.

We implement $p(x|y;\theta)$ using a modified flow-based model that has a conditional input (Winkler et al., 2019). Suppose $y \in \mathcal{Y}$ is the conditional input, then the mapping function should be $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ and (3.17) will be rewritten as:

$$p(x|y) = p(f(x|y)) \left| \frac{\partial f(x|y)}{\partial x} \right|. \tag{3.22}$$

Adaptive Sampler

The uniform distribution is a trivial choice for $\pi(x|y)$ in (3.20) to search the solution space. However, uniform sampling is inefficient in a high-dimensional space, especially when risky scenarios are rare. Therefore, we propose an adaptive sampler that takes advantage of gradient information to gradually cover all modes of risky scenarios. Suppose that we have a metric $c(x|y)$ that indicates the exploration value: the higher $c(x|y)$, the more worth exploring x. We then use NES, a black-box optimization method, to estimate the gradient of $c(x|y)$. The sampler $\pi(x|y)$

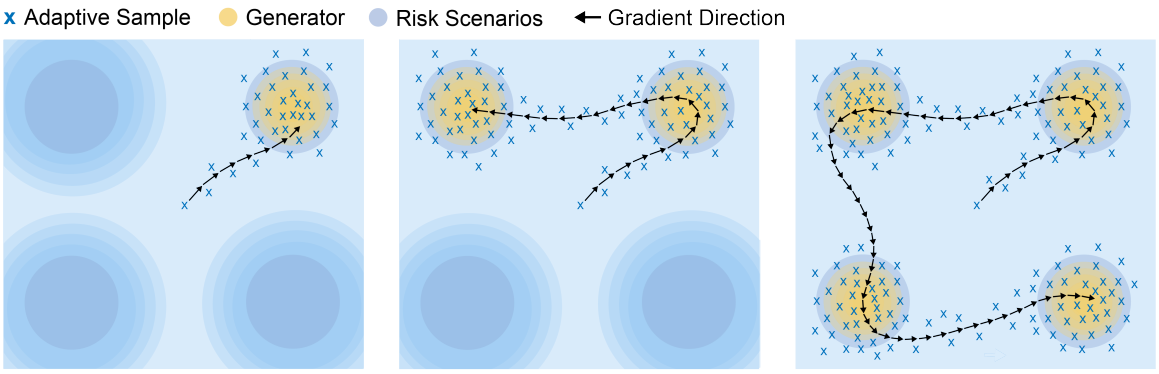


Figure 3.14: An illustration example of multi-modal estimation using our adaptive sampler.

then follows the generating rule (we omit y in gradient derivation):

$$x^{t+1} \leftarrow x^t + \alpha \nabla_x c(x^t), \tag{3.23}$$

where α is the step size. The gradient $\nabla_x c(x^t)$ in (3.23) can be estimated by:

$$\nabla_x c(x^t) = \nabla_x \mathbb{E}_{x \sim \mathcal{N}(x^t, \sigma^2 I)} \left[c(x) \right] = \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} \left[\epsilon \cdot c(x^t + \sigma \epsilon) \right]. \tag{3.24}$$

In practice, we will approximate the above expectation with the Monte Carlo method:

$$\nabla_x c(x^t) = \frac{1}{\sigma} \sum_{i=1}^{M} \epsilon_i \cdot c(x^t + \sigma \epsilon_i), \quad \epsilon_i \sim \mathcal{N}(0, I). \tag{3.25}$$

The design of $c(\cdot)$ strongly influences the performance of the adaptive sampler. Inspired by some curiosity-driven literature (Pathak et al., 2017), where uncertainty, Bayesian surprise, and prediction error are used to guide exploration, we choose a metric that involves the generative model $p(x|y)$:

$$c(x|y) = r(x|y) - \gamma \cdot p(x|y; \theta), \tag{3.26}$$

where γ is a hyperparameter that balance $r(x)$ and $p(x|y; \theta)$. Intuitively, when one mode (some similar risky scenarios) is well learned by $p(x|y; \theta)$, the metric $c(x)$ will decrease and force the sampler to explore other modes. Finally, the multimodal distribution will be captured by $p(x|y; \theta)$. An example of the sampling process is shown in Figure 3.14 with a Gaussian Mixture Model (GMM) distribution. For intuitive understanding and comparison, we also show the sam-

pling process of using a uniform sampler and the learned generator as the sampler in Figure 3.12 and Figure 3.13, respectively.

3.2.3 Experiment and Analysis

In this section, we first demonstrate the advantage of our proposed adaptive sampler with a toy example. After that, we show the generated safety-critical scenarios with different settings in an intersection environment. Finally, we evaluate the robustness of several popular RL algorithms using our generated scenarios and provide conclusions about their robustness.

Efficiency of Adaptive Sampler

We first conduct experiments to evaluate the efficiency of the proposed adaptive sampler.

Environment settings. As discussed in Section 3.2.2, we shall expect that the estimated gradient of $c(x|y)$ improves the efficiency of the sampling procedure. To assess this, we compare our method with two baselines in a GMM example. In the first baseline, we use $c(x) = r(x)$, a straightforward and common choice in the adversarial attack literature. In the second baseline, we use the variance of the posterior of Gaussian Processes (GP) (MacKay et al., 1998) to model the uncertainty of search space and combine this uncertainty with $r(x)$.

Explanation: The comparison results is displayed in Figure 3.15. Both two baselines are facing the mode collapse problem to varying degrees, while our method effectively covers all modes. The reasoning is as follows. The first baseline uses only limited information about the multimodal landscape and thus is easily trapped into one modality. The second baseline, which gradually decreases the importance of the explored points, can cover all modes even other unimportant points. However, the rapidly descending uncertainty and the lack of adaptivity to the generator $p(x)$ lead to suboptimal results and unbalanced data collection. Our proposed method uses the feedback of the generator that gives the sampler both the capability of uncertainty exploration and balanced data collection, hence attaining all the modalities.

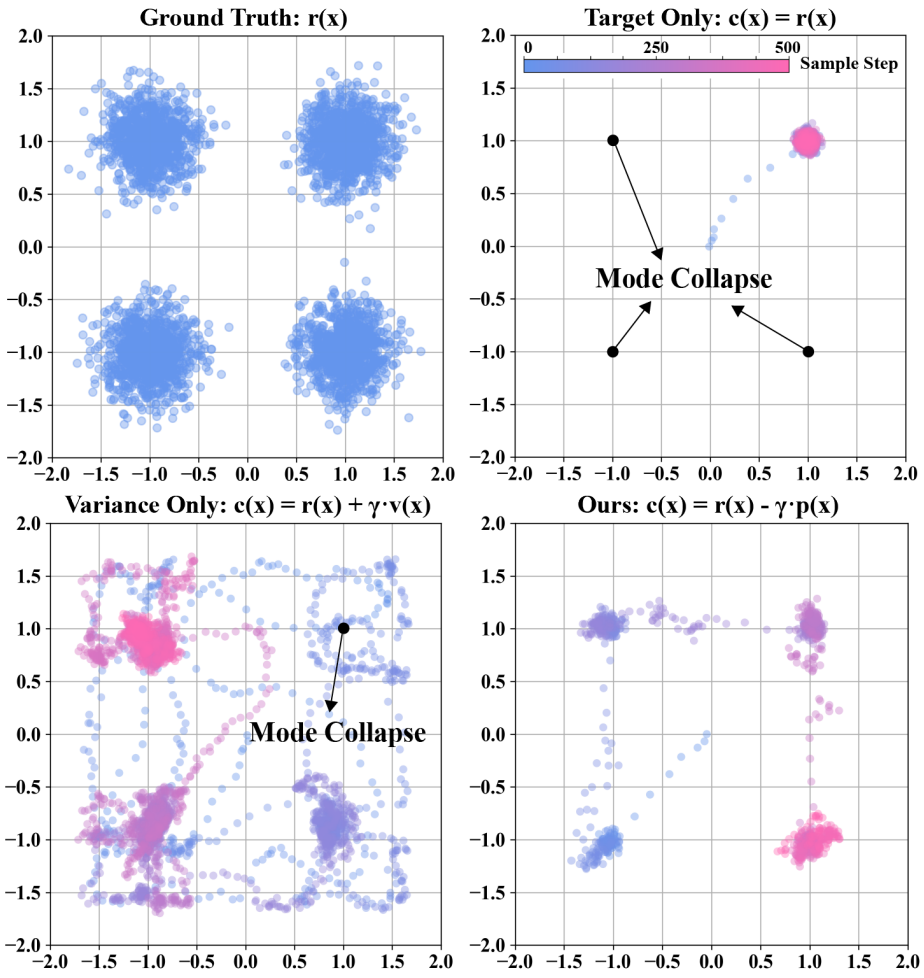


Figure 3.15: A toy example to compare different adaptive samplers.

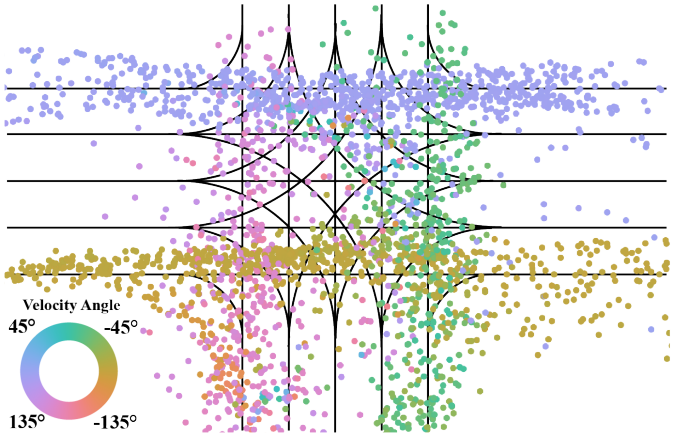


Figure 3.16: Samples from prior model $q(x)$ that is learned from InD dataset (Bock et al., 2020). Different colors represent different velocity angles.

Safety-critical Scenario Generation

In the following, we provide an evaluation and analysis of the results of the generation of safety-critical scenarios.

Environment settings. An intersection environment is used to conduct our experiment on the Carla simulator (Dosovitskiy et al., 2017). We represent the scenario as a 4-dimensional vector $x = [x^s, y^s, v_x^s, v_y^s]$, which represents the initial position and initial velocity of a cyclist. The cyclist is spawned in the environment and travels at a constant speed. Then we place an ego vehicle controlled by an intelligent driver model. The reference route of the ego vehicle is represented by the condition y. The minimal distance between the cyclist and the ego vehicle is used to calculate $r(x)$:

$$r(x) = \exp\{-\|p_v - p_c\|_2\}, \tag{3.27}$$

where p_v and p_c represent the position of the vehicle and the cyclist respectively. A lower distance corresponds to a higher $r(x)$. This scenario is defined as a precrash scenario in (Najm et al., 2007) and also adopted in the Carla Autonomous Driving Challenge. This setting allows us to test the collision avoidance capability of decision-making algorithms H. Note that more intelligent algorithms can replace the current agent during the evaluation stage.

Real-world data distribution. There are numerous datasets collected in the intersection traffic environment. We train our prior model $q(x)$ with trajectories from the InD dataset (Bock et al., 2020). A well-trained prior model can be used to infer the likelihood of a given sample and generate new samples as well. We display the position and velocity direction of some generated samples in Figure 3.16. These samples roughly describe the distribution of a cyclist in an intersection.

Generated scenarios display. We train a generator $p(x|y)$ to generate safety-critical scenarios given the route condition y. In Figure 3.17, we compare the samples from two generators: one does not use prior (middle row) and the other uses $q(x)$ as the prior model (bottom row), where the same color map is used as in Figure 3.16. The top row of Figure 3.17 shows the collected scenarios by our adaptive sampler. Each of these samples is corresponding to a risk value that is not shown in the figure. In Figure 3.17, it is shown that without real-world data

prior, the generator learns the distribution of all risky scenarios collected by the adaptive sampler (first row). After incorporating the prior model (samples are shown in Figure 3.16), the generator concentrates more on the samples that are more likely to happen in the real world. This results in the removal of samples that have opposite directions to the real data.

Baseline and metric settings. We select seven algorithms as our baseline. The details of each algorithm are discussed below:

- Grid Search: We set the search step for all parameters to 100. Since x has four dimensions, all search iterations should be 10^8.
- Human Design: We use the rules defined in Carla Autonomous Driving Challenge, which trigger the movement of the cyclist according to the location of the ego vehicle.
- Uniform Sampling: The scenario parameter x is uniformly sampled from the entire space. This method is widely used in the evaluation of safety decision-making algorithms. For instance, obstacles are generated randomly to test the performance of collision avoidance in the Safety-Gym environment (Ray et al., 2019).
- L2C (Ding et al., 2020b): This method uses the REINFORCE framework with a single Gaussian distribution policy. This kind of policy can only represent a single modality.
- L2C+GMM: The policy distribution of (Ding et al., 2020b) is replaced by a GMM. The purpose is to explore the multimodal capability of the REINFORCE algorithm.
- Ours-Uniform: We replace the adaptive sampler in our method with a uniform sampler.
- Ours-HMC: We replace the adaptive sampler in our method with an HMC sampler to explore the efficiency of the gradient-based MCMC method.

We use query time and collision rate as our metrics. The query time means the number of queries to the simulation during the training stage. Methods without training have 0 query times. A rough value is recorded when the distribution of the samples is stable. The second metric is the collision rate, which is calculated after the training stage. We sample 1000 scenarios for 10 different routes and get a collision rate for each route. We then calculate the mean and variance on the ten routes.

Comparison with baseline methods. The results are shown in Table. 3.2. The grid search is

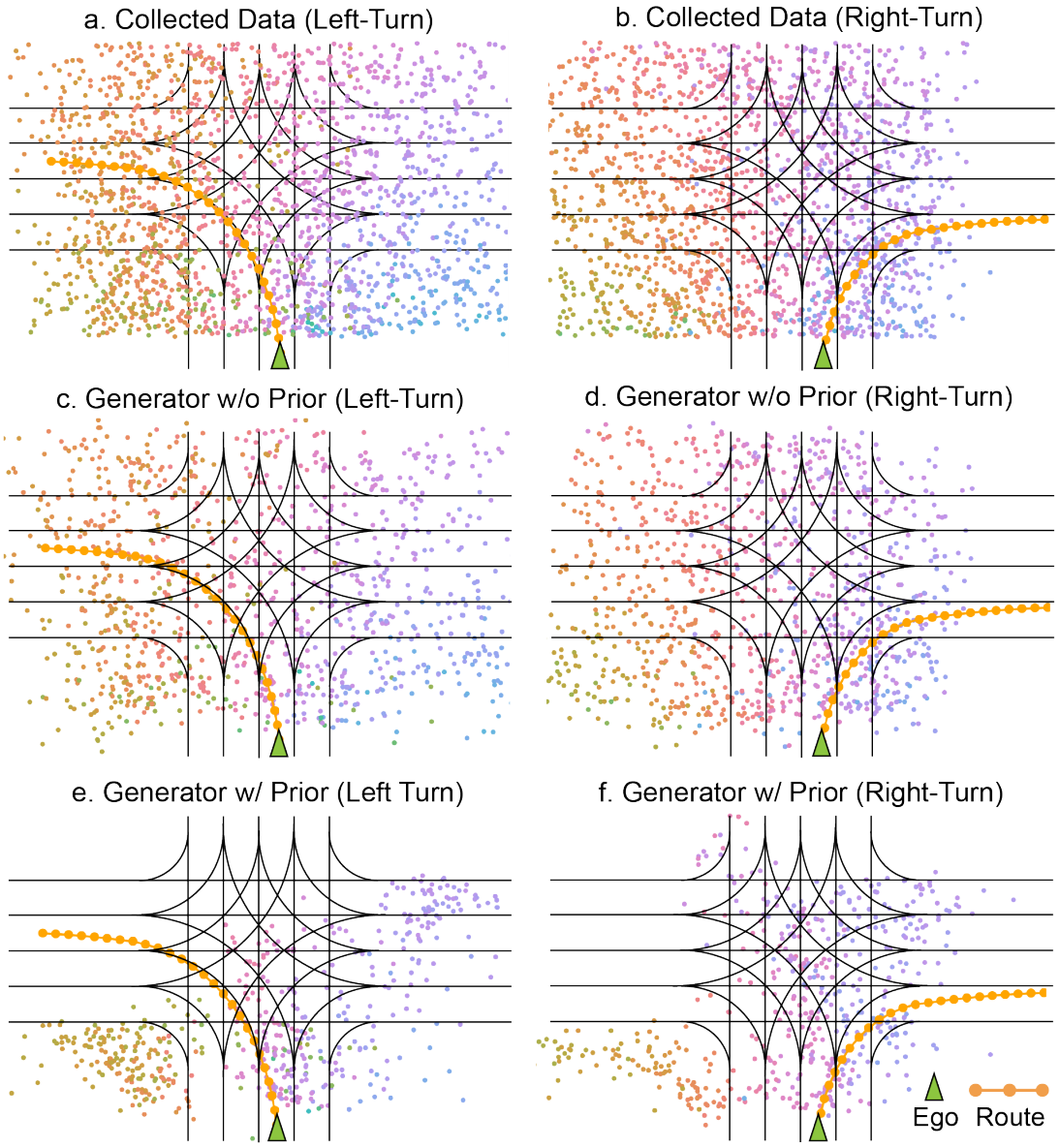


Figure 3.17: Distributions of safety-critical scenarios represented by $x = [x^s, y^s, v_x^s, v_y^s]$.

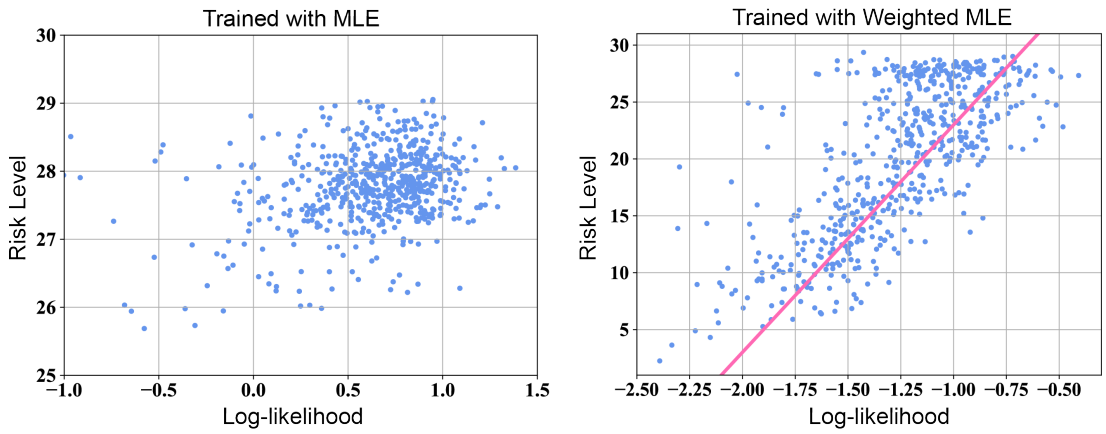


Figure 3.18: Relationship between risk and log-likelihood of $p(x)$.

the most trivial way comparable to our method of finding multimodal risk scenarios. However, the query time grows exponentially as the dimension of x and the step size increase. In simulation, human design is a possible way to reproduce the risk scenarios that occur in the real world, but these scenarios are fixed and do not adapt to changes in the task parameters y, leading to a low collision rate. Our experiment also found that the uniform method achieves a collision rate less than 10%. In dense reward situations, uniform sampling could be a good choice, while in most real-world cases, the rare event risk scenarios make this method quite inefficient. REINFORCE-Single searches the risk scenario in the RL framework (Ding et al., 2020b). Although this method converges faster than ours, it cannot handle multimodal cases with a single Gaussian distribution policy. The REINFORCE-GMM method extends the original version with a multimodal policy module. However, it has similar results as REINFORCE-Single. The reason is that the on-policy sampling method in REINFORCE is easy to trap into a single modality, even if the policy itself is multimodal. The final weight in GMM is highly imbalanced and only one component dominates. The ablation study reveals that our adaptive sampler (Ours-Adaptive) is more efficient than the uniform version (Ours-Uniform). The MCMC version (Ours-HMC) requires less query time than our adaptive sampler, while its samples only concentrate on one modality.

Relationship between risk level and log-likelihood. Since our generator is trained with WMLE, we make use of all collected samples rather than only the risky ones as in (Norden et al., 2019). We compare two generators that are trained with MLE and WMLE and plot the results in Figure 3.18. The generator trained with MLE by only using the risk data concentrates

on the high-risk area, while our WMLE generator has a linear relationship between the risk and log-likelihood. Therefore, our generator can not only generate risky scenarios but also generate scenarios with different risk levels by considering the likelihood of samples.

Evaluation of RL algorithms

To prove that our generated scenarios help improve algorithm evaluation, we implemented six popular RL agents (DQN (Mnih et al., 2013), A2C (Mnih et al., 2016), PPO (Schulman et al., 2017), DDPG (Lillicrap et al., 2015), SAC (Haarnoja et al., 2018), Model-based RL (Kaiser et al., 2019)) as $H(\mu|x, y)$ on the navigation task in the aforementioned environment. The agent's target is to arrive at a goal point $[x^g, y^g]$ and avoid reaching the non-driving area. At the same time, we place a cyclist at the intersection to create a traffic scenario. Finally, the state of the agent is

$$s = [x^g, y^g, x^a, y^a, v_x^a, v_y^a, x^s, y^s, v_x^s, v_y^s], \tag{3.28}$$

where $[x^a, y^a, v_x^a, v_y^a]$ and $[x^s, y^s, v_x^s, v_y^s]$ represent the position and velocity of the agent and the cyclist, respectively. Agents should also avoid colliding with the cyclist; otherwise they will receive a penalty. The reward consists of three parts:

$$R(x) = r_g + r_s \times \mathbb{1}_s(x) + r_c \times \mathbb{1}_c(x), \tag{3.29}$$

where r_g is calculated by the reduction of distance between the agent and the goal, r_s and r_c indicate the penalty of non-driving area violation and cyclist collision. $\mathbb{1}_s(x)$ and $\mathbb{1}_c(x)$ are two indicator functions that equal to 1 when the two events occur. The episode ends when the agent collides with the cyclist or when the agent reaches the target. We implement DQN and A2C on a discrete action space with a controller that follows a pre-defined route. Their action space only influences the acceleration. The other agents have a continuous action space that controls the throttle and steering.

We have two environments for training and testing: 1) Uniform Risk Scenarios (URS): the initial state x of the cyclist is uniformly sampled; 2) Generated Risk Scenarios (GRS): the initial state x is sampled from our generated $p(x|y; \theta)$ with $\sigma = 0.2$. We train and test six RL algorithms

Table 3.2: Performance Comparison

Methods	Queries (↓)	Collision Rate (↑)
Grid Search	1×10^8	$\mathbf{100\%}$
Human Design	-	$35\% \pm 21\%$
Uniform Sampling	-	$9\% \pm 1\%$
L2C-Single (Ding et al., 2020b)	$\mathbf{1 \times 10^3}$	$97\% \pm 2\%$
L2C-GMM (Ding et al., 2020b)	$\mathbf{1 \times 10^3}$	$98\% \pm 1\%$
Ours-Uniform	1×10^5	$\mathbf{100\%}$
Ours-HMC	$\mathbf{1 \times 10^3}$	$\mathbf{100\%}$
Ours-Adaptive	3×10^3	$\mathbf{100\%}$

with different environments and Figure 3.19 displays the testing reward and the testing collision rate. Based on the comparison between different settings, we draw three main conclusions.

More informative evaluation (Column 1 vs. Column 2): Agents tested on URS have similar final rewards and collision rates. These nearly indistinguishable results make it difficult to compare the robustness of different algorithms. In contrast, the results on GRS show a great discrepancy, which helps us to obtain clearer conclusions.

Generalization of agents (Column 1 vs. Column 3): We train the agents on URS and GRS but test them both on URS. We notice that the performance of both settings is similar, which means that all agents do not sacrifice their generalization to URS.

Robustness of different agents (Column 2 vs. Column 4): The expected results should be that all agents have an improvement in column 4. However, we notice that different algorithms still show different robustness because of their mechanisms. We roughly divide the six algorithms into three categories and explain them, respectively. 1) *Much improved:* MBRL is robust to the risk scenario, even if it is not trained on GRS. The reason is that the target of MBRL is to learn a dynamics model and plan with it. Even if it is trained on normal scenarios, it learns how to predict the trajectory of the cyclist. Then, when tested in risky scenarios, MBRL can easily avoid collisions. Training in GRS will not improve it. 2) *Improved a little*: DDPG, DQN, and SAC slightly improve performance. From the bottom figure of column 4, we notice that the collision rates first increase but quickly decrease, which means they learn how to deal with most of the risk scenarios. However, their rewards are lower than MBRL because they cannot handle all risky scenarios. The explanation for the little improvement is that these are off-policy methods with

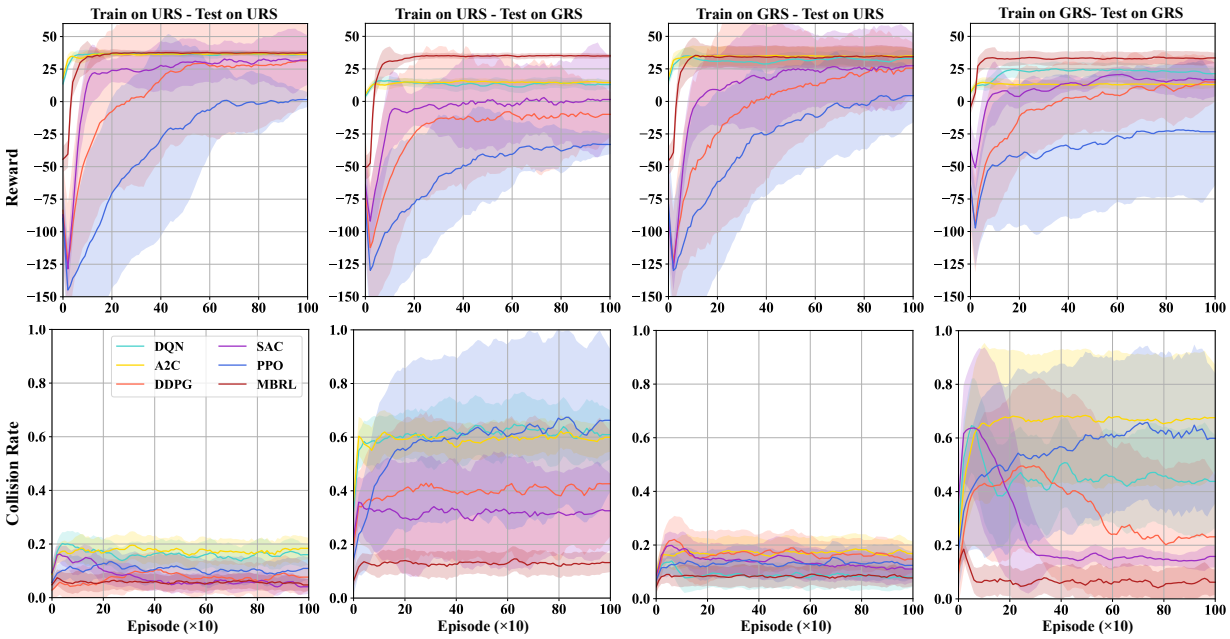


Figure 3.19: Testing reward and testing collision rate in four different settings.

memory buffers. The average of stored risky scenarios from the buffer makes the training stable and, therefore makes the agents successfully handle the scenarios they have met. However, they still fail in some unseen risky scenarios. 3) *Not Improved:* PPO and A2C are on-policy methods, which learn the policy according to current samples. However, the risky scenarios cause the instability of training, because our generator finds different modes (types) of risky scenarios. In contrast, a normal scenario will not cause such a problem because the state of the cyclist does not have much influence. In column 4, the collision rates of PPO and A2C gradually increase and never decrease, which means that they cannot handle risky scenarios.

Note that the above empirical conclusions might only be valid in this environment. A further comparison of these RL algorithms should be carefully designed in multiple other settings. Nevertheless, our generator is indeed proven to be more insightful than the uniform sampler. Beyond RL algorithms, our proposed generating framework can also be used to efficiently evaluate other decision-making methods developed to deal with more risky scenarios.

3.3 Summary

In this chapter, two works that use an adversarial generation framework to design critical scenarios (Ding et al., 2021a, 2020b). The first work directly leverages a reinforcement learning method to find the solution, which may result in unrealistic scenarios and a loss of diversity. As a remedy, the second work uses the advantage of the data-driven generation method as a prior regularization and introduces an adaptive sampler to find as many solutions as possible.

Although generating critical scenarios based on optimization is an applicable direction, we merely understand the underlying reason that causes the critical event in the scenario. Therefore, in the next chapter, we will investigate more on using human knowledge to help generate critical scenarios. Specifically, we consider causality (Peters et al., 2017) as the representation of knowledge, which describes the cause and effect between objects in the scenario.

Chapter 4

Knowledge-based Generative Models

> All truths are easy to understand once they are discovered; the point is to discover them.
>
> *Galileo Galilei*

In previous categories, we discussed methods that purely use data or interact with the AV to generate scenarios. However, the scenarios constructed in the physical world need to satisfy the traffic rules and the physical laws. The samples in the density we estimate or the adversarial examples we generate could easily violate these constraints. Additionally, domain knowledge also improves the efficiency of generation. After traffic accidents occur, we humans analyze the scenario and find the reasons for the accidents. Finding the underlying causality, for example, the view of the sensor is blocked, is important to efficiently generate safety-critical scenarios.

Therefore, we now consider methods that incorporate external domain knowledge into the generation process. We will first explore rule-based methods that artificially design the structure and parameters of scenarios. Then we turn to the learning-based methods that use explicit knowledge to guide generation. Assume we can obtain certain domain knowledge constraints $c \in C$ from experts, then we can augment the learning with

$$\hat{\theta} = \arg\max \mathbb{E}_{x \sim p_\theta(x|c)} \left[g(x, \pi_e) \right], \tag{4.1}$$

where the scenarios are sampled from a conditional distribution $p_\theta(x|c)$. In addition, we can also

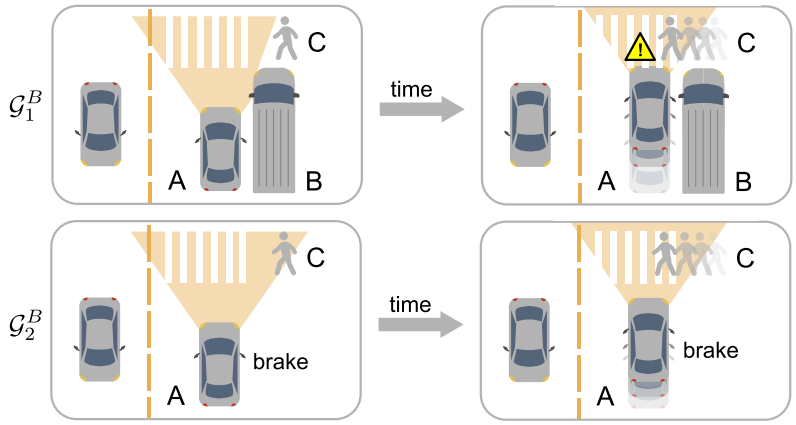


Figure 4.1: The top scenario is safety-critical because the view of vehicle A is blocked by B.

use C to manipulate existing scenarios:

$$\tilde{x} = \arg\max g(x, \pi_e) \;\; s.t.\; C(x) \geq 0, \tag{4.2}$$

where $C(x) \geq 0$ means the constraint is satisfied.

4.1 Scenario Reasoning with Causality

Generating safety-critical scenarios with Deep Generative Models (DGMs), which estimate the distribution of data samples with neural networks, is viewed as a promising way in recent works (Ding et al., 2022b). The existing literature either searches in the latent space to build scenarios (Ding et al., 2021a,b) or directly uses optimization to find adversarial examples (Ding et al., 2020b; Zhang et al., 2022). However, such a generation task is still challenging, since we are required to simultaneously consider fidelity to avoid conjectural scenarios that will never happen in the real world, as well as the safety-critical level, which is indeed rare compared with normal scenarios. In addition, generating reasonable threats to vehicles' safety can be inefficient if the model purely relies on unstructured observational data, as the safety-critical scenarios are rare and follow fundamental physical principles. Inspired by the fact that humans are good at abstracting the causation beneath the observations with prior knowledge, we explore a new direction toward causal generative models for this generation task.

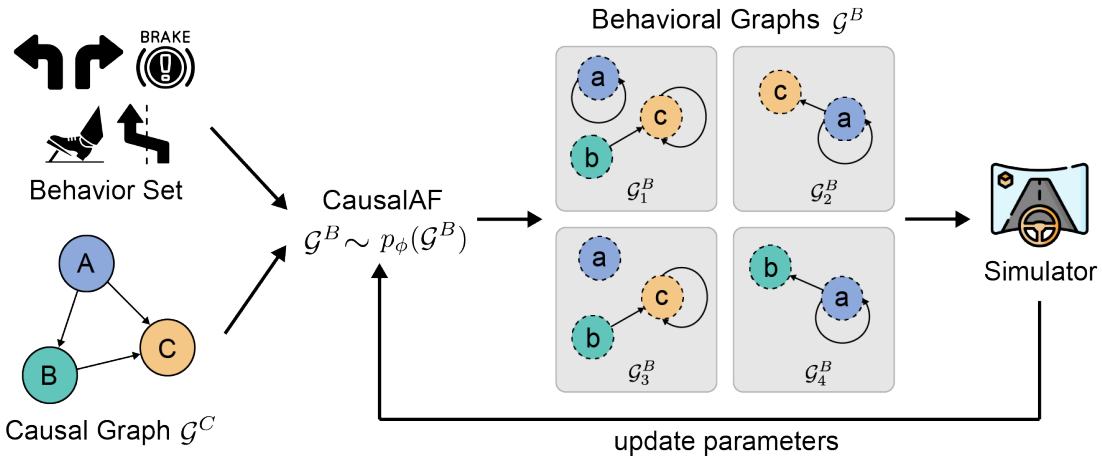


Figure 4.2: Diagram of the generation pipeline using *CausalAF*.

To have a look at causality in traffic scenarios, we show an example in Figure 4.1. When a vehicle B is parked in the middle between the autonomous vehicle A and pedestrian C, the view of A is blocked, making A have little time to brake and thus have a potential collision with C. As human drivers, we believe that B should be the cause of the accident. This scenario may take AVs millions of hours to collect (Feng et al., 2021). Even if we use traditional generative models to generate this scenario, the model tends to memorize the location of all objects without learning the reasons. As a remedy, we can incorporate causality into generative models for the efficient generation of such safety-critical scenarios.

In this work, we propose a structured generative model with causal priors. We model the causality as a directed acyclic graph (DAG) named Causal Graph (CG) (Pearl, 2009). To facilitate CG in the traffic scenario, we propose another Behavioral Graph (BG) to represent the interaction between objects in scenarios. The graphical representation of both graphs makes it possible to use the BG to unearth the causality given by the CG. Based on BG, we propose the first generative model that integrates causality into the graph generation task and names it *CausalAF* (Ding et al., 2021c). Specifically, we propose two types of causal masks: Causal Order Masks (COM) that modify the node order for node generation and Causal Visibility Masks (CVM) that remove irrelevant information for edge generation. We show the diagram of *CausalAF* generation in Figure 4.2 and summarize our main contributions as follows:

- We propose a causal generative model *CausalAF* that integrates causal graphs with two novel

mask operators for safety-critical scenario generation.

- We show that *CausalAF* dramatically improves efficiency and performance in three standard traffic settings compared to purely data-driven baselines.
- We show that training in generated safety-critical scenarios improves the robustness of 4 reinforcement learning-based driving algorithms.

4.1.1 Graphical Representation of Scenarios

We start by proposing a novel representation of traffic scenarios using a graph structure. Then, we propose to generate such a graphical representation with an autoregressive generative model.

Behavioral Graph

Traffic scenarios mainly consist of interactions between static and dynamic objects, which can be naturally described by a graph structure. Therefore, we define the Behavioral Graph $\mathcal{G}^B$ to represent driving scenarios with the following definition.

Definition 4 (Behavioral Graph, BG)**.** *Suppose a scenario has maximum m objects with n types. A Behavioral Graph $\mathcal{G}^B = (\mathcal{V}^B, \mathcal{E}^B)$ is a directed graph with node matrix $\mathcal{V}^B \in \mathbb{R}^{m \times n}$ representing the types of objects and edge matrix $\mathcal{E}^B \in \mathbb{R}^{m \times m \times (h_1 + h_2)}$ representing the interaction between objects, where h_1 is the number of edge types and h_2 is the dimension of edge attributes.*

According to this definition, $\mathcal{G}^B$ works as a planner that controls the behavior of objects in the scenario based on the types of nodes $\mathcal{V}^B$ and edges $\mathcal{E}^B$. For example, two nodes v_1 and v_2 represent two vehicles and the edge of v_1 to v_2 represents the relative velocity between v_1 and v_2. Specifically, a self-loop edge (i, i) represents that an object takes one action irrelevant to other objects (e.g., a car goes straight or turns left with no impact on other road users), while other edges (i, j) mean that the object i takes one action related to object j (e.g., a car i moves towards a pedestrian j). The edge attributes represent the properties of the actions. For instance, the attribute $[x, y, v_x, v_y]$ of one edge has the following meaning: x and y are positions and v_x and v_y are velocities.

Behavioral Graph Generation with Autoregressive Flow

Generally, there are two ways to generate graphs: one is to simultaneously generate all nodes and edges, and the other is to iteratively generate nodes and add edges between nodes. Considering the directed nature of $\mathcal{G}^B$, we utilize the Autoregressive Flow model (AF) (Huang et al., 2018a), which is a type of sequentially DGMs, to generate nodes and edges of $\mathcal{G}^B$ step by step. It uses an invertible and differentiable transformation $\mathcal{F}_\phi$ parameterized by ϕ to convert the graph $\mathcal{G}^B$ to a latent variable $\boldsymbol{z}$ that follows a base distribution $p(\boldsymbol{z})$ (for example, normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$). According to the change of variables theorem, we can obtain $p_\phi(\mathcal{G}^B) = p(\mathcal{F}_\phi(\mathcal{G}^B)) \left|\det \frac{\partial \mathcal{F}_\phi(\mathcal{G}^B)}{\partial \mathcal{G}^B}\right|$.To increase the representing capability, $\mathcal{F}_\phi$ contains multiple functions f_i for $i \in \{0, \ldots, K\}$. The entire transformation is represented as $\mathcal{G}^B = \boldsymbol{z}_K = f_K^{-1} \circ \cdots \circ f_0^{-1} \triangleq \mathcal{F}_\phi^{-1}(\boldsymbol{z}_0)$ by repeatedly substituting the variable for the new variable z_i, where $\circ$ means the composition of the function. Eventually, we obtain the likelihood

$$\log p_\phi(\mathcal{G}^B) = p(\boldsymbol{z}_0) - \sum_{i=1}^{K} \log \left|\det \frac{df_i^{-1}}{d\boldsymbol{z}_{i-1}}\right|, \tag{4.3}$$

which will be used to learn the parameter ϕ based on empirical distribution of $\mathcal{G}^B$. After training, we can sample from $p_\phi(\mathcal{G}^B)$ by using the reverse function $\mathcal{F}_\phi^{-1}$. Let $V_{[i]}^B \in \mathbb{R}^n$ and $E_{[i,j]}^B \in \mathbb{R}^{h_1+h_2}$ represent the node i and the edge (i, j) of $\mathcal{G}^B$, then we can generate them with the sampling procedure:

$$\mathcal{V}_{[i]}^B \sim \mathcal{N}\left(\mu_i^v, (\sigma_i^v)^2\right) = \mu_i^v + \sigma_i^v \odot \epsilon \text{ and } \mathcal{E}_{[i,j]}^B \sim \mathcal{N}\left(\mu_{ij}^e, (\sigma_{ij}^e)^2\right) = \mu_{ij}^e + \sigma_{ij}^e \odot \epsilon, \tag{4.4}$$

where $\odot$ denotes the element-wise product and ϵ follows a Normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Variables μ_i^v, σ_i^v, μ_{ij}^e, and σ_{ij}^e are obtained from $\mathcal{F}_\phi$ in an autoregressive manner:

$$\mu_i^v, \sigma_i^v = \mathcal{F}_\phi\Big(\mathcal{V}_{[0:i-1]}^B, \mathcal{E}_{[0:i-1,0:m]}^B\Big) \text{ and } \mu_{ij}^e, \sigma_{ij}^e = \mathcal{F}_\phi\Big(\mathcal{V}_{[0:i]}^B, \mathcal{E}_{[0:i,0:j-1]}^B\Big), \tag{4.5}$$

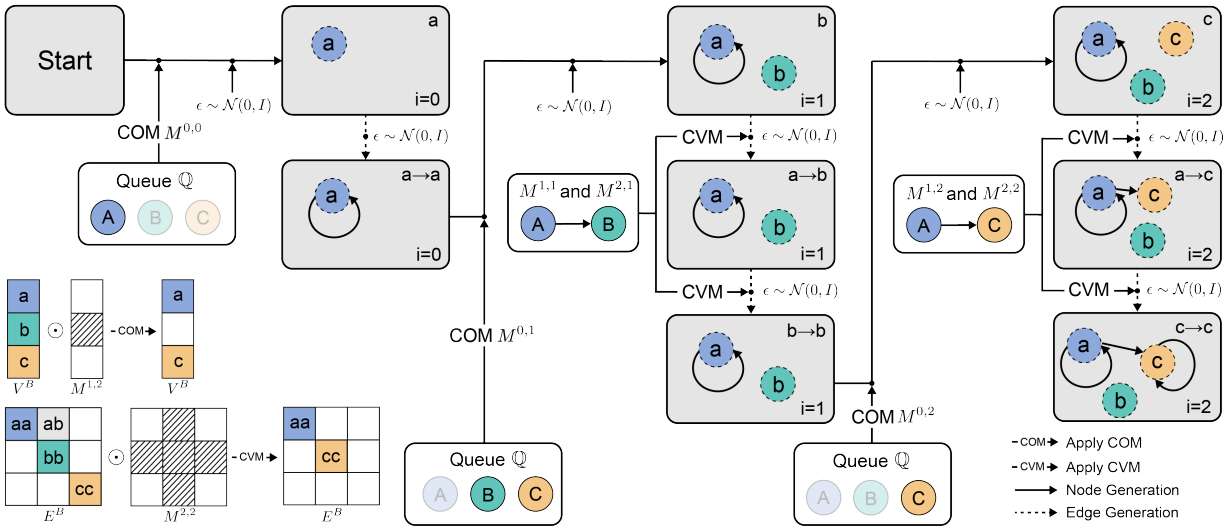


Figure 4.3: **(a)** The generation process of a BG starting from an empty graph. **(b)** CG and BG are in the example. **(c)** The implementation of masking during the generation.

where $[0 : i]$ represents the elements from index 0 to index i. After sampling, we obtain the node and edge type by converting $\mathcal{V}^B$ and part of $\mathcal{E}_B$ from continuous values to one-hot vectors:

$$\mathcal{V}^B_{[i]} \leftarrow \text{onehot}\Big[\arg\max(\mathcal{V}^B_{[i]})\Big], \quad \mathcal{E}^B_{[i,j,0:h_1]} \leftarrow \text{onehot}\Big[\arg\max(\mathcal{E}^B_{[i,j,0:h_1]})\Big] \quad \forall i, j \in [m]. \tag{4.6}$$

Intuitively, the generation of one node depends on all previously generated nodes and edges. One node only has edges that point to the nodes that are generated before it. To illustrate this autoregressive generation process, we provide an example with three nodes in Figure 4.3(a).

4.1.2 Causal Autoregressive Flow (CausalAF)

In this section, we discuss how to integrate causality into the autoregressive generating process of the Behavioral Graph $\mathcal{G}^B$. In general, we transfer prior knowledge from a causal graph to $\mathcal{G}^B$ by increasing the structural similarity. However, calculating such similarity is not easy because of the discrete nature of graphs. To solve this problem, we propose *CausalAF* with two causal masks, i.e., Causal Order Masks (COM) and Causal Visible Masks (CVM), which make the generated $\mathcal{G}^B$ follow the causal information.

Algorithm 4: Training process of CausalAF

Input: Causal Graph $\mathcal{G}^C$, Goal $\mathcal{L}_g$, Learning rate α, Maximum node number m

while ϕ *not converged* **do**
  // Sample a BG $\mathcal{G}^B \sim p_\phi(\mathcal{G}^B|\mathcal{G}^C)$
  for $i < m$ **do**
    Sample node matrix $\mathcal{V}^B_{[i]}$ by (4.4)
    Get node type $\mathcal{V}^B_{[i]}$ by (4.6)
    Apply Causal Order Masks (COM) $M^{0,i}$ to $\mathcal{V}^B_{[i]}$
    Apply Causal Visible Masks (CVM) $M^{1,i}$, $M^{2,i}$ to $\mathcal{V}^B_{[i]}$, $\mathcal{E}^B_{[i,j]}$
    for $j \leq i$ **do**
      Sample edge matrix $\mathcal{E}^B_{[i,j]}$ by (4.4)
      Get edge type $\mathcal{E}^B_{[i,j]}$ by (4.6)
  Collect one scenario $\mathcal{G}^B = \{\mathcal{V}^B, \mathcal{E}^B\}$
  // Learn model parameters
  Calculate the likelihood $p_\phi(\mathcal{G}^B|\mathcal{G}^C)$
  Execute $\tau = \mathcal{F}(\mathcal{G}^B)$ and get $\mathcal{L}_g(\tau)$
  Use (4.9) to update $\phi \leftarrow \phi - \alpha\nabla_\phi\mathcal{L}_g(\tau)$

Causal Generative Models

Definition 5 (Structural Causal Models (Peters et al., 2017), SCM)**.** *A structural causal model (SCM)* $\mathfrak{C} := (\mathcal{S}, \mathcal{U})$ *consists of a collection* $\boldsymbol{S}$ *of* m *functions,* $v_j := f_j(\boldsymbol{PA}_j, u_j)$, $\forall j \in [m]$*, where* $\boldsymbol{PA}_j \subset \{v_1, \ldots, v_m\}\backslash\{v_j\}$ *are called parents of* v_j*; and a joint distribution* $\mathcal{U} = \{u_1, \ldots, u_m\}$ *on the noise variables, which are required to be jointly independent.*

Definition 6 (Causal Graphs (Peters et al., 2017), CG)**.** *The causal graph* $\mathcal{G}^C$ *of an SCM is obtained by creating a node for each* v_j *and drawing directed edges from each parent in* $\boldsymbol{PA}_j(\mathcal{G}^C)$ *to* v_j*. The representation of* $\mathcal{G}^C = (\mathcal{V}^C, \mathcal{E}^C)$ *consists of the node vector* $\mathcal{V}^C \in \{0,1\}^m$ *and the adjacency matrix* $\mathcal{E}^C \in \{0,1\}^{m\times m\times h_1}$*. Each edge* (i, j) *represents a causal relation from node* i *to node* j*.*

We formally describe causality based on the above definitions of SCM and CG. In fact, the generative model $p_\phi(\mathcal{G}^B)$ mentioned in Section 4.1.1 shares a very similar definition with SCM except that $\mathcal{G}^B$ does not follow the order of causality. This inspires us that we can convert $p_\phi(\mathcal{G}^B)$ to an SCM by incorporating the causal graph $\mathcal{G}^C$ into the generation process. In this method, we assume that the causal graph $\mathcal{G}^C$ can be summarized by expert knowledge. Therefore, we

incorporate a given $\mathcal{G}^C$ into $p_\phi(\mathcal{G}^B|\mathcal{G}^C)$ by regularizing the generative process with two novel masks as shown in Figure 4.3.

Causal Graph Integration

We introduce two masks to integrate causality into the autoregressive generation model.

Causal Order Masks (COM). The order is vital during the generation of $\mathcal{G}^B$ since we must ensure that the cause is generated before the effect. To achieve this, we maintain a priority queue $\mathbb{Q}$ to store valid child types according to the causal relation in $\mathcal{G}^C$. $\mathbb{Q}$ is initialized with $\mathbb{Q} = \{i|\ \mathbf{PA}_i(\mathcal{G}^C) = \emptyset\ , \forall i \in [m]\}$, which contains all nodes that do not have parent nodes. Then, in each step of node generation, we update $\mathbb{Q}$ by removing the generated node i and adding the child nodes of i. Since one node may have multiple parents, it is valid only if all of its parents have been generated. We use $\mathbb{Q}$ to create a k-hot mask $M^{0,i} \in \mathbb{R}^n$, where the element is set to 1 if it is a valid type. Then we apply COM to the node matrix by $\mathcal{V}^B_{[i]} \leftarrow M^{0,i} \odot \mathcal{V}^B_{[i]}$, where $\mathcal{V}^B_{[i]}$ is the node vector obtained from $\mathcal{F}_\phi$ for node i. Intuitively, this mask sets the probability of the invalid node types to 0 to make sure the generated node always follows the correct order.

Causal Visible Masks (CVM). Ensure that the correct causal order is still insufficient to represent the causality. Thus, we further propose another type of mask called CVM, which removes the non-causal connections, i.e., non-parent nodes to the current node in $\mathcal{G}^C$, when generating edges. Specifically, we generate two binary masks $M^{1,i} \in \mathbb{R}^{m\times n}$ and $M^{2,i} \in \mathbb{R}^{m\times m\times(h_1+h_2)}$ with

$$M^{1,i}_{[j,:]} = 0 \text{ and } M^{2,i}_{[j,i,:]} = 0\ , \forall j \notin \mathbf{PA}_i(\mathcal{G}^C). \tag{4.7}$$

Then, we apply them to update the node matrix and edge matrix by $\mathcal{V}^B \leftarrow M^{1,i} \odot \mathcal{V}^B$ and $\mathcal{E}^B \leftarrow M^{2,i} \odot \mathcal{E}^B$. We illustrate an example of this process in Figure 4.3(c). Assume that we are generating edges for node c. We need to remove node b since $\mathcal{G}^C$ tells us that B does not have edges to node C. After applying M^v and M^e, we move the features of node c to the previous position of b. This permutation operation is important, since the autoregressive model is not permutation invariant.

Optimization of Safety-critical Generation

After introducing the generative process of *CausalAF*, we now turn to the optimization procedure. The objective is to generate scenarios $\tau = \mathcal{F}(\mathcal{G}^B)$ with an executor $\mathcal{F}$ to satisfy a given goal, which is formulated as an objective function $\mathcal{L}_g$. We define $\mathcal{L}_g(\tau) = \mathbb{1}(D(\tau) < \epsilon)$, where $D(\tau)$ represents the minimal distance between the autonomous vehicle and other objects, and ϵ is a small threshold. Therefore, the optimization is to solve the problem:

$$\max_{\phi} \mathbb{E}_{\mathcal{G}^B \sim p_\phi(\mathcal{G}^B|\mathcal{G}^C)}[\mathcal{L}_g(\mathcal{F}(\mathcal{G}^B))]. \tag{4.8}$$

Usually, $\mathcal{L}_g$ contains non-differentiable operators (e.g., complicated simulation and rendering), thus we have to utilize black-box optimization methods to solve the problem. We consider a policy gradient algorithm named REINFORCE (Williams, 1992), which obtains the estimation of the gradient from samples by

$$\nabla_\phi \mathbb{E}_{\mathcal{G}^B \sim p_\phi(\mathcal{G}^B|\mathcal{G}^C)}[\mathcal{L}_g(\mathcal{F}(\mathcal{G}^B))] = \mathbb{E}[\nabla_\phi \log p(\mathcal{G}^B|\mathcal{G}^C)\mathcal{L}_g(\mathcal{F}(\mathcal{G}^B))]. \tag{4.9}$$

Overall, the entire training algorithm is summarized in **Algorithm** 4. In addition, we can prove that the *CausalAF* guarantees monotonicity of likelihood in Theorem 1 at convergence.

Theorem 1 (Monotonicity of Likelihood)**.** *Given the true causal graph $\mathcal{G}^{C*} = (\mathcal{V}^C, \mathcal{E}^{C*})$ and the Structural Hamming Distance (SHD) (Acid and de Campos, 2003), for CG $\mathcal{G}_1^C = (\mathcal{V}^C, \mathcal{E}_1^C)$ and $\mathcal{G}_2^C = (\mathcal{V}^C, \mathcal{E}_2^C)$, if $SHD(\mathcal{G}_1^C, \mathcal{G}^{C*}) < SHD(\mathcal{G}_2^C, \mathcal{G}^{C*})$, and edge $\exists\, e$, s.t. $\mathcal{E}_1^C \cup \{e\} = \mathcal{E}_2^C$, CausalAF converges with the monotonicity of likelihood, that is, $p_\phi(D(\tau) < \epsilon \mid \mathcal{G}_2^C) < p_\phi(D(\tau) < \epsilon \mid \mathcal{G}_1^C) < p_\phi(D(\tau) < \epsilon \mid \mathcal{G}^{C*})$.*

Before providing the proof of this Theorem, we need to define several important concepts.

Definition 7 (Structural Hamming Distance (SHD))**.** *For any two DAGs $\mathcal{G}_1^C, \mathcal{G}_2^C$ with identical vertices set $\mathcal{V}$, we define the following function SHD: $\mathcal{G} \times \mathcal{G} \to \mathbb{R}$,*

$$\begin{aligned} \textit{SHD}(\mathcal{G}_1^C, \mathcal{G}_2^C) &= \#\{(i,j) \in \mathcal{V}^2 \mid \mathcal{G}_1^C \textit{ and } \mathcal{G}_2^C \textit{ have different edges } e_{ij}\} \\ &\triangleq \sum_{j \in \mathcal{V}} |\boldsymbol{PA}_j(\mathcal{G}_1^C) - \boldsymbol{PA}_j(\mathcal{G}_2^C)|, \end{aligned} \tag{4.10}$$

where $|\boldsymbol{PA}_j(\mathcal{G}_1^C) - \boldsymbol{PA}_j(\mathcal{G}_2^C)|$ is the number of the absolute difference in parental nodes for node j between causal graph $\mathcal{G}_1^C$ and $\mathcal{G}_2^C$.

Definition 8 (Nodes in Behavior Graph)**.** *Let $x_j = \left[v_j, \{e_{ij}\}_{i \in \{\boldsymbol{PA}_j(\mathcal{G}^C) \cup j\}}\right]$, where v_i is the node type of the j-th node, and $e_{\cdot j}$ are the arrows that point in the j-th node. All these components form the node v_j in the behavior graph.*

Definition 9 (Respect the graph)**.** *For any given behavior graph $\mathcal{G}^B$ with a specific causal graph $\mathcal{G}^C$, the transition model respects the graph if the distribution $p_\phi(\mathcal{G}^B|\mathcal{G}^C)$ can be factorized as:*

$$p(\mathcal{G}^B|\mathcal{G}^C) = \prod_{j \in [m]} p(v_j|\boldsymbol{PA}_j(\mathcal{G}^C)), \tag{4.11}$$

where m is the number of factorized nodes, and $\boldsymbol{PA}_j(\cdot)$ is for v_j's parents based on the causal graph.

Proposition 1 (CausalAF respects the graph)**.**

$$\begin{aligned}
p_\phi(\mathcal{G}^B|\mathcal{G}^C) &= \prod_{j \in [m]} \Big[\underbrace{p_\phi(v_j|\boldsymbol{PA}_j(\mathcal{G}^C))}_{COM} \underbrace{p_\phi(e_{jj}|v_j, \boldsymbol{PA}_j(\mathcal{G}^C)) \prod_{i \in \boldsymbol{PA}_j(\mathcal{G}^C)} p_\phi(e_{ij}|v_j, \boldsymbol{PA}_j(\mathcal{G}^C))}_{CVM} \Big] \\
&= \prod_{j \in [m]} \Big[p_\phi(v_j, e_{jj}|\boldsymbol{PA}_j(\mathcal{G}^C)) \prod_{i \in \boldsymbol{PA}_j(\mathcal{G}^C)} p_\phi(e_{ij}|v_j, \boldsymbol{PA}_j(\mathcal{G}^C)) \Big] \\
&= \prod_{j \in [m]} p_\phi(v_j, \{e_{ij}\}_{i \in \{\boldsymbol{PA}_j(\mathcal{G}^C) \cup j\}}|\boldsymbol{PA}_j(\mathcal{G}^C)) \\
&= \prod_{j \in [m]} p_\phi(\mathcal{G}_j|\boldsymbol{PA}_j(\mathcal{G}^C)).
\end{aligned} \tag{4.12}$$

The CausalAF node generation process combines two phases: firstly, we use COM to determine the generation order of the node, which prevents the generation of child nodes before their parent nodes. This COM can also be interpreted as a node ordering with topological sorting, therefore, CausalAF should always respect the term $p(v_j|\boldsymbol{PA}_j(\mathcal{G}^C)), \forall j$ in Equation (4.12).

On the other hand, CVM is used to guarantee that the output of the autoregressive flow model uses proper structural information (i.e. the parents of the current node) to generate the self-loop edge as well as edges between new nodes and their parents accordingly, the CVM trick thus

guarantees that CausalAF respects the term

$$p(e_{jj}|v_j, \boldsymbol{PA}_j(\mathcal{G}^C)) \prod_{i \in \boldsymbol{PA}_j(\mathcal{G}^C)} p(e_{ij}|v_j, \boldsymbol{PA}_j(\mathcal{G}^C)), \forall j \tag{4.13}$$

in Equation (4.12).

Assumption 1 (Local Optimality)**.** *Let $\mathcal{G}^{C^*}$ be the ground truth of the causal graph, for any nodes v_j with its parental set $\boldsymbol{PA}_j(\mathcal{G}_1^C) \neq \boldsymbol{PA}_j(\mathcal{G}^{C^*})$. At convergence, CausalAF will have*

$$\max_{\phi} p_\phi(v_j|\boldsymbol{PA}_j(\mathcal{G}^{C^*})) > \max_{\phi} p_\phi(v_j|\boldsymbol{PA}_j(\mathcal{G}_1^C)). \tag{4.14}$$

Assumption 2 (Local Monotonicity of Behavior Graph)**.** *For a single node v_j, its local monotonicity of likelihood means for any conditional set, we have*

$$\boldsymbol{PA}_j(\mathcal{G}_1^C), \boldsymbol{PA}_j(\mathcal{G}_2^C) \neq \boldsymbol{PA}_j(\mathcal{G}^C). \tag{4.15}$$

if $|\boldsymbol{PA}_j(\mathcal{G}_1^C) - \boldsymbol{PA}_j(\mathcal{G}^C)| < |\boldsymbol{PA}_j(\mathcal{G}_2^C) - \boldsymbol{PA}_j(\mathcal{G}^C)|$, and $\exists\, v$, s.t. $\boldsymbol{PA}_j(\mathcal{G}_2^C) \cup v = \boldsymbol{PA}_j(\mathcal{G}_1^C)$. Then we have:

$$\max_{\phi} p_\phi(\mathcal{G}_j|\boldsymbol{PA}_j(\mathcal{G}_1^C)) > \max_{\phi} p_\phi(\mathcal{G}_j|\boldsymbol{PA}_j(\mathcal{G}_2^C)). \tag{4.16}$$

With these assumptions and definitions, we now provide a proof of the monotonicity of likelihood in CausalAF.

Monotonicity of Likelihood. Given that $\mathcal{G}^B \sim p_\phi(\mathcal{G}^B|\mathcal{G}^C), \tau = \mathcal{F}(\mathcal{G}^B)$, by using the change of variable theorem, we have

$$\tau \sim p_\phi(\mathcal{F}^{-1}(\tau)|\mathcal{G}^C)|\det \frac{\partial \mathcal{F}^{-1}(\tau)}{\partial \tau}| \triangleq \hat{p}_\phi(\tau|\mathcal{G}^C). \tag{4.17}$$

Then, the optimization process of CausalAF can be rewritten as below:

$$\begin{aligned}
&\max_{\phi} \mathbb{E}_{\mathcal{G}^B \sim p_\phi(\mathcal{G}^B|\mathcal{G}^C)}[\mathbb{1}(D(\mathcal{F}(\mathcal{G}^B)]) < \epsilon)] \\
&\quad = \max_{\phi} \mathbb{E}_{\hat{p}_\phi(\tau|\mathcal{G}^C)}[\mathbb{1}(D(\tau) < \epsilon)] \\
&\quad = \max_{\phi} \hat{p}_\phi(D(\tau) < \epsilon|\mathcal{G}^C) = \max_{\phi} \hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}^C),
\end{aligned} \tag{4.18}$$

where $\mathcal{A} = \{\mathcal{G}^B | D(\mathcal{F}(\mathcal{G}^B)) < \epsilon\}$. Since the CausalAF respects the graph, as shown in Proposition 1, for true CG $\mathcal{G}^{C^*}$ and another CG $\mathcal{G}_1^C \neq \mathcal{G}^{C^*}$. By applying the local monotonicity in the previous assumptions, when CausalAF converges, we will have

$$\begin{aligned}
\hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}_1^C) &= \prod_j \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_1^C)) \\
&= \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)=\mathbf{PA}_j(\mathcal{G}^{C^*})}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_1^C)) \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)\neq\mathbf{PA}_j(\mathcal{G}^{C^*})}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_1^C)) \\
&< \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)=\mathbf{PA}_j(\mathcal{G}^{C^*})}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}^{C^*})) \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)=\mathbf{PA}_j(\mathcal{G}^{C^*})}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}^{C^*})) \\
&= \prod_j \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}^{C^*})) = \hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}^{C^*}).
\end{aligned} \tag{4.19}$$

Then we assume we have another Causal Graph $\mathcal{G}_2^C \neq \mathcal{G}_1^C$ if $SHD(\mathcal{G}_1^C, \mathcal{G}^{C^*}) < SHD(\mathcal{G}_2^C, \mathcal{G}^{C^*})$ and $\exists\, e$, s.t. $\mathcal{E}_1^C \cup \{e\} = \mathcal{E}_2^C$,

$$\begin{aligned}
\hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}_2^C) &= \prod_j \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_2^C)) \\
&= \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)=\mathbf{PA}_j(\mathcal{G}_2^C)}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_2^C)) \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)\neq\mathbf{PA}_j(\mathcal{G}_2^C)}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_2^C)) \\
&< \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)=\mathbf{PA}_j(\mathcal{G}_2^C)}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_1^C)) \prod_{\substack{\forall j,s.t. \\ \mathbf{PA}_j(\mathcal{G}_1^C)=\mathbf{PA}_j(\mathcal{G}_1^C)}} \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_1^C)) \\
&= \prod_j \hat{p}_\phi(v_j \in \mathcal{A}_j|\mathbf{PA}_j(\mathcal{G}_1^C)) = \hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}_1^C).
\end{aligned} \tag{4.20}$$

Based on the derivation above, we conclude that $\hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}_2^C) < \hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}_1^C) <$

$\hat{p}_\phi(\mathcal{G}^B \in \mathcal{A}|\mathcal{G}^{C*})$, which indicates that at convergence, the likelihood of collision samples converge with monotonicity guarantees:

$$p_\phi(D(\tau) < \epsilon \mid \mathcal{G}_2^C) < p_\phi(D(\tau) < \epsilon \mid \mathcal{G}_1^C) < p_\phi(D(\tau) < \epsilon \mid \mathcal{G}^{C*}). \tag{4.21}$$

□

Scenario Sampling and Execution

Thanks to the autoregressive generation of CausalAF, we can conduct generation conditioned on arbitrary numbers or types of nodes. Instead of generating from scratch, we can start from an existing $\mathcal{G}_c^B$ for the generation with $\mathcal{G}^B \sim p_\phi(\cdot|\mathcal{G}_c^B, \mathcal{G}^C)$. Conditional generation can be used for interactive scenarios, e.g., using the autonomous vehicle's information or the data of partial scenarios in the real world as conditions to generate diverse and realistic scenarios. After sampling the scenarios, the physical properties (for example, position and velocity) defined in the generated $\mathcal{G}^B$ are executed in the simulator $\mathcal{F}$ to create sequential scenarios τ. After execution, the simulator outputs the objective function $L_g(\tau)$ as a result.

4.1.3 Experiment and Analysis

We evaluate *CausalAF* using three main pre-crash traffic scenarios defined by the US Department of Transportation (Najm et al., 2013) and the Euro New Car Assessment Program (Van Ratingen et al., 2016). Our empirical results show that it may not be trivial for generative models to learn the underlying causality even if such causality seems understandable to humans. In particular, we conducted a series of experiments to answer the following main questions. Q1: How does CausalAF perform compared to other scenario generate methods? Q2: How does causality help the generation process? Q3: How can we use the generated safety-critical scenarios? In this section, we will first introduce the designed environment and baseline methods. Then we will answer the above questions by carefully investigating the experiment results.

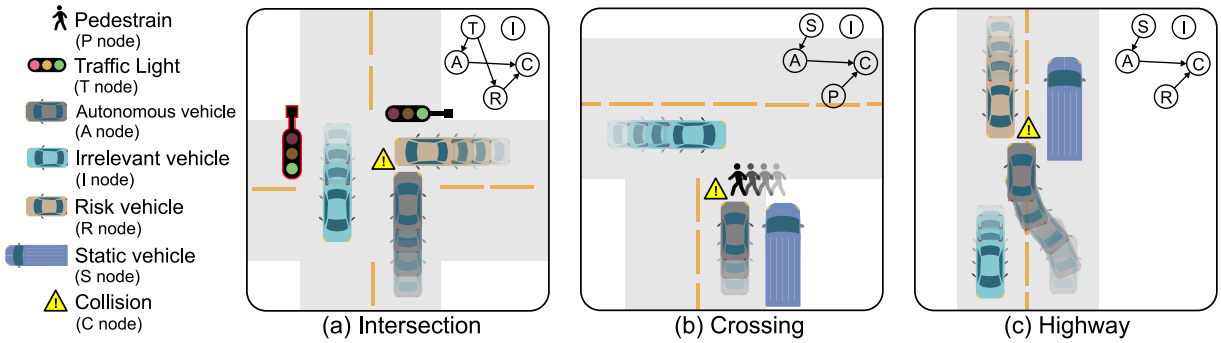


Figure 4.4: Three causal traffic scenarios are used in our experiments. The corresponding causal graphs are shown in the upper right of each scenario.

Simulations and Baselines

We first introduce the scenarios we used in our experiment and the baselines we consider as a comparison.

Scenario. We consider three safety-critical traffic scenarios (shown in Figure 4.4) that have a clear causation. The causal graph $\mathcal{G}^C$ for each scenario is displayed in the upper right of the scenario.

- **Intersection**. One potential safety-critical event could happen when the traffic light T turns from green to yellow to give the road right to an autonomous vehicle A. Here, A and R are influenced by T. R runs the red light, colliding with A perpendicularly, therefore, causing the collision C together with A. I does not influence other objects.
- **Crossing**. A pedestrian P and an autonomous vehicle A cross the road in vertical directions. There also exists a static vehicle S parked on the side of the road. Then a potentially risky scenario could happen when S blocks the vision of A. In this scenario, S is the parent of A, and P and A cause the collision C. I does not influence other objects.
- **Highway**. An autonomous vehicle A takes a lane-changing behavior due to a static car S parked in front of it. Meanwhile, a vehicle R drives in the opposite lane. Since S blocks the vision of A, A is likely to collide with R. In this scenario, S is the parent of A, and R and A cause the collision C. I does not influence other objects.

Simulator. We implement the above scenarios in a 2D simulator, where all agents have radar

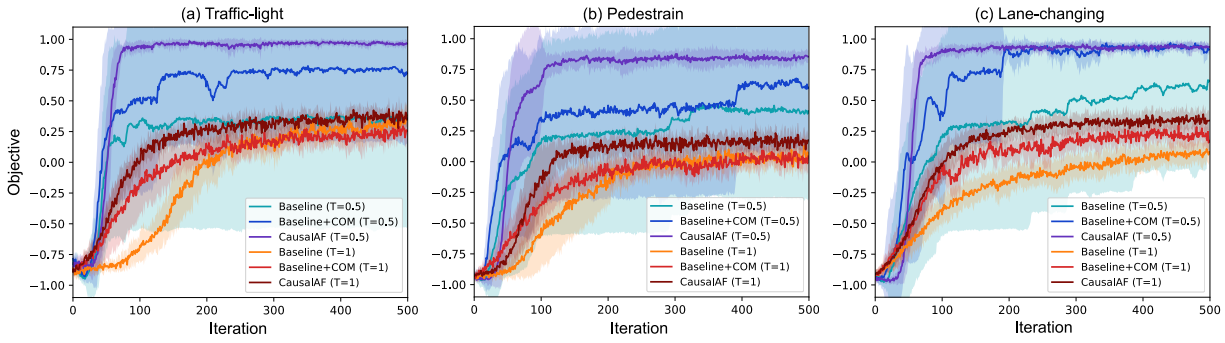


Figure 4.5: Training objective of *CausalAF* and two variants under two sampling temperatures.

Table 4.1: Collision rate (↑) of generated safety-critical scenarios. Bold font means the best.

Method	L2C	MMG	SAC	STRIVE	Baseline	Baseline+COM	CausalAF
Intersection	0.63±0.28	0.31±0.54	0.47±0.61	0.64±0.12	0.29±0.84	0.69±0.52	**0.98±0.01**
Crossing	0.69±0.41	0.43±0.56	0.38±0.49	0.55±0.10	0.35±0.65	0.57±0.48	**0.83±0.13**
Highway	0.85±0.10	0.56±0.36	0.58±0.41	0.67±0.16	0.53±0.69	0.88±0.04	**0.91±0.06**

sensors and are controlled by a simple vehicle dynamic. During the running, the autonomous vehicle is controlled by a rule-based policy, which will decelerate if it detects any obstacles in front of it within a certain range Thus, the safety-critical scenario will not happen unless the radar of one agent is blocked and the distance is smaller than the braking distance, avoiding the creation of unrealistic scenarios. The action space contains the acceleration and steering of all objects, and the state space contains the position and heading of all objects and the status of the traffic lights if applicable.

Baselines. We consider 7 algorithms as baselines, including 5 scenario generation methods and 2 variants of *CausalAF*. Learning to collide (L2C) (Ding et al., 2020b) uses a Bayesian network to describe the relationship between objects and uses RL to search the scenario parameters. Multi-modal Generation (MMG) (Ding et al., 2021a) uses an adaptive sampler to increase sample diversity. STRIVE (Rempe et al., 2022) learns traffic prior from datasets and uses adversarial optimization to generate risk scenarios. SAC is a standard RL algorithm that uses the objective as a reward function. To further investigate the contribution of COM and CVM, we design two variants that share the same network structure as *CausalAF*. Baseline does not use COM or CVM, and Baseline+COM only uses COM.

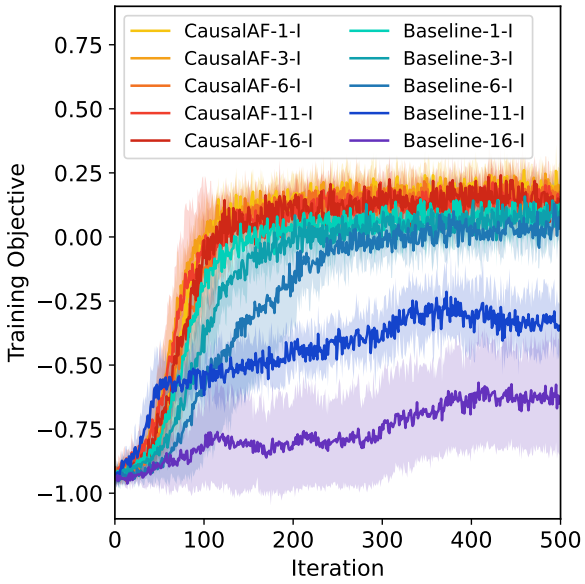


Figure 4.6: The training objectives in the Pedestrian scenario from different numbers of irrelevant vehicles.

Results Discussion

Next, we show results and analysis to answer the following research questions.

How does *CausalAF* perform on safety-critical scenario generation? (Q1) We train all generation methods in 3 environments and report the final objective values in Table 4.1. We observe that *CausalAF* achieves the best performance among all methods. L2C performs better than MMG and SAC because it also considers the structure of the scenario. We also notice that both Baseline and Baseline+COM have performance drops compared to *CausalAF*, indicating that the COM and CVM modules contribute to the autoregressive generating process. Baseline+COM performs better than Baseline, which validates our hypothesis that COM is not powerful enough to represent causality. To investigate the training procedure, we plot the training objectives in Figure 4.5 with two different sampling temperatures T, which controls the sampling variance in $\epsilon \sim \mathcal{N}(0, T)$. A high temperature provides strong exploration but causes slow convergence. However, we find that using a small temperature leads to unstable training with high variance due to poor exploration capability.

How does causality help the generation process? (Q2) The baseline design represents the

model that uses the full graph. Therefore, the results in Table 4.1 also demonstrate that the causal graph is more helpful than the full graph. To investigate why the causal graph helps learning, we conduct an ablation study on the number of irrelevant nodes (I node), which do not have edges in the causal graph. In Figure 4.6, we can see that the addition of more irrelevant vehicles increases the gap between *CausalAF* and Baseline – the performance of the baseline gradually drops as the number of I nodes increases but *CausalAF* has consistent performance. The reason is that *CausalAF* is able to diminish the impact of irrelevant information with COM and CVM.

How can we use the generated scenarios? (Q3) Finally, we explore how to use the generated safety-critical scenarios. We train 4 RL agents ({SAC, PPO, DDPG, MBRL}-Norm) under normal scenarios (uniformly sample the parameters of objects in the scenario), then we evaluate them under scenarios generated by four different methods: Normal, L2C, MMG, and Ours (*CausalAF*) to test performance in safety-critical scenarios. We also train another 4 agents under scenarios generated by our method ({SAC, PPO, DDPG, MBRL}-Ours) and evaluate under four different scenarios. We report the collision rate in Table 4.2. We find that scenarios generated by our CausalAF cause more collisions to the RL agents, which also shows that training in normal scenarios is not enough for safety. After training in scenarios generated by CausalAF, the agents achieve a lower collision rate in all scenarios, indicating the usefulness of training on safety-critical scenarios.

Table 4.2: Collision rate (↑) of RL algorithms evaluated in different scenarios.

Method	Intersection				Crossing				Highway			
	Norm	L2C	MMG	Ours	Norm	L2C	MMG	Ours	Norm	L2C	MMG	Ours
SAC-Norm	0.05	0.57	0.64	**0.91**	0.04	0.54	0.67	**0.92**	0.03	0.79	0.75	**0.95**
SAC-Ours	0.01	0.03	0.04	**0.08**	0.00	0.04	0.06	**0.11**	0.02	0.01	0.04	**0.09**
PPO-Norm	0.07	0.44	0.48	**0.86**	0.03	0.53	0.61	**0.80**	0.02	0.62	0.64	**0.92**
PPO-Ours	0.00	0.04	0.01	**0.12**	0.02	0.03	0.03	**0.08**	0.01	0.02	0.03	**0.13**
DDPG-Norm	0.12	0.76	0.62	**0.89**	0.07	0.71	0.76	**0.85**	0.04	0.72	0.61	**0.95**
DDPG-Ours	0.01	0.02	0.05	**0.13**	0.02	0.01	0.04	**0.12**	0.03	0.03	0.03	**0.16**
MBRL-Norm	0.04	0.78	0.74	**0.98**	0.05	0.68	0.85	**0.97**	0.05	0.79	0.87	**0.98**
MBRL-Ours	0.00	0.01	0.01	**0.07**	0.00	0.01	0.02	**0.09**	0.00	0.03	0.01	**0.10**

4.2 Generative Models with Causal Discovery

Generalizability, which enables an algorithm to handle unseen tasks, is fruitful but challenging in multifarious decision-making domains. Recent literature (Cranmer et al., 2006; Li et al., 2020; Xu et al., 2021a) reveals the critical role of reasoning in improving the generalization of reinforcement learning (RL). However, most off-the-shelf RL algorithms (Sutton and Barto, 2018) have not considered reasoning an indispensable accessory, thus usually suffering from data inefficiency and performance degradation due to the mismatch between training and testing settings. To achieve generalization in the testing stage, some effort was put into incorporating domain knowledge to learn structured information, including subtask decomposition (Lu et al., 2021) and program generation (Han and Zhou, 2020; Landajuela et al., 2021; Sun et al., 2019; Yang et al., 2021; Zhao et al., 2021), which guide the model to solve complicated tasks in an explainable way. However, such symbolism-dominant methods heavily depend on the reusability of subtasks and predefined grammars, which may not always be accessible in decision-making tasks.

Inspired by the close link between reasoning and the cause-and-effect relationship, causality has recently been incorporated to compactly represent the structured knowledge mentioned above in RL training (Gershman, 2017). Based on the form of causal knowledge, we divide related work into two categories, that is, *implicit* and *explicit* causation. With a *implicit* causal representation, researchers ignore the detailed causal structure. For instance, (Zhang et al., 2020a) extracts invariant features as one node that influences the reward function, while the other node consists of task-irrelevant features (Bica et al., 2021; Sodhani et al., 2022; Sontakke et al., 2021; Tomar et al., 2021). This neat structure has good scalability but requires access to multiple environments that share the same invariant feature (Bica et al., 2021; Han et al., 2021; Zhang et al., 2020a). In contrast, one can turn to the *explicit* side by estimating detailed causal structures (Gasse et al., 2021; Seitzer et al., 2021; Volodin et al., 2020; Wang et al., 2021d), which uses directed graphical models to capture causality in the environment. A pre-request for this estimation is the object-level or event-level abstraction of the observation, which is available in most tasks and also becoming a frequently studied problem (Abel, 2022; Abel et al., 2018; Shanahan and Mitchell, 2022). However, existing *explicit* causal reasoning RL models either require the true causal

graph (Nair et al., 2019) or rely on heuristic design without theoretical guarantees (Wang et al., 2021d).

In this work, we propose ***GeneRA**lizing by **D**iscov**ER**ing (GRADER)* (Ding et al., 2022a), a causal reasoning method that augments the RL algorithm with data efficiency, interpretability, and generalizability. We focus mainly on Goal-Conditioned RL (GCRL) (Liu et al., 2022), where different goal distributions during training and testing reflect the generalization. We formulate the GCRL as a probabilistic inference problem (Levine, 2018) with a learnable causal graph as latent variable. This novel formulation naturally explains the learning objective with three components – transition model learning, planning, and causal graph discovery – leading to an optimization framework that alternates between causal discovery and policy learning to gain generalizability. Under some mild conditions, we prove the unique identifiability of the causal graph and the theoretical performance guarantee of the proposed framework.

To demonstrate the effectiveness of the proposed method, we conduct comprehensive experiments in environments that require strong reasoning capability. Specifically, we design two types of generalization settings, that is, spuriousness and composition, and provide an example to illustrate these settings in Figure 4.7. The evaluation results confirm the advantages of our method in two aspects. First, the proposed data-efficient discovery method provides an explainable causal graph, but requires much less data than previous methods, increasing data efficiency and interpretability during task solving. Second, simultaneously discovering the causal graph during policy learning dramatically increases the success rate in solving tasks. In summary, the contribution of this work is threefold:

- We use the causal graph as a latent variable to reformulate the GCRL problem and then derive an iterative training framework from solving this problem.
- We prove that our method uniquely identifies true causal graphs, and the performance of iterative optimization is guaranteed with a lower bound given the converged transition dynamics.
- We design nine tasks in three environments that require strong reasoning capability and show the effectiveness of the proposed method against strong baselines on these tasks.

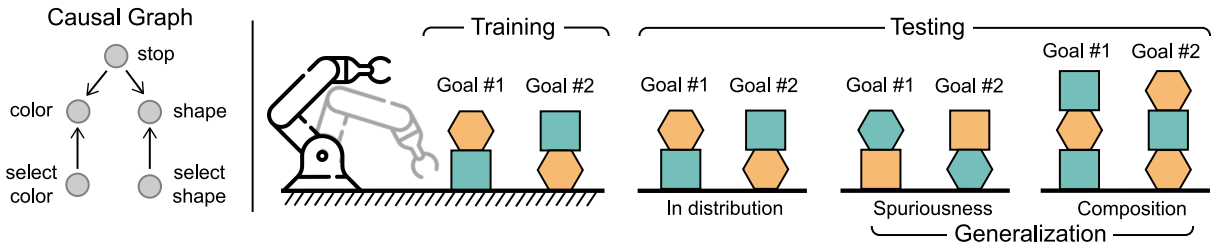


Figure 4.7: **Left:** The causal graph of the pick and place task. **Right:** Three testing settings: (1) In distribution (2) Spuriousness (3) Composition.

4.2.1 Problem Formulation and Preliminary

We start by discussing the setting we consider in this work and the assumptions required in causal reasoning. We then briefly introduce the necessary concepts related to causality and causal discovery.

Factorized Goal-conditioned RL

We assume that the environment follows the goal-conditioned Markov Decision Process (MDP) setting with full observation. This setting is represented by a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, G)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the probabilistic transition model, $G \subset \mathcal{S}$ is the goal space which is a set of assignment of values to states, and $r(s, g) = \mathbb{1}(s = g) \in \mathcal{R}$ is the sparse deterministic reward function that returns 1 only if the state s matches the goal g. In this method, we focus on the goal-conditioned generalization problem, where the goal for the training and testing stages will be sampled from different distributions $p_{\text{train}}(g)$ and $p_{\text{test}}(g)$. We refer to a goal $g \in G$ as a task and use these two terms interchangeably. To accomplish the causal discovery methods, we make a further assumption similar to (Boutilier et al., 2000; Seitzer et al., 2021) for the state and action space:

Assumption 3 (Space Factorization)**.** *The state space* $\mathcal{S} = \{\mathcal{S}_1 \times \cdots \times \mathcal{S}_M\}$ *and action space* $\mathcal{A} = \{\mathcal{A}_1 \times \cdots \times \mathcal{A}_N\}$ *can be factorized to disjoint components* $\{\mathcal{S}_i\}_{i=1}^M$ *and* $\{\mathcal{A}_i\}_{i=1}^N$.

Components representing one event or object's property usually have explicit semantic meanings for better interpretability. This assumption can be satisfied by abstraction of state and action, which has been widely investigated in (Abel, 2022; Abel et al., 2018; Shanahan and Mitchell,

2022). Such factorization also helps to deal with high-dimensional states since it could be intractable to treat each dimension as one random variable (Wang et al., 2021d).

Causal Reasoning with Graphical Models

Reasoning with causality is based on specific causal structures, which are commonly represented as directed acyclic graphs (DAGs) (Peters et al., 2017) over variables. Consider random variables $\boldsymbol{X} = (X_1, \ldots, X_d)$ with index set $\mathcal{V} := \{1, \ldots, d\}$. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of nodes $\mathcal{V}$ and edges $\mathcal{E} \subseteq \mathcal{V}^2$ with (i, j) for any $i, j \in \mathcal{V}$. A node i is called a parent of j if $e_{ij} \in \mathcal{E}$ and $e_{ji} \notin \mathcal{E}$. The set of parents of j is denoted by $\mathbf{PA}_j(\mathcal{G})$. We formally discuss the graph representation of causality with two definitions.

Definition 10 (Structural Causal Models (Peters et al., 2017))**.** *A structural causal model (SCM)* $\mathfrak{C} := (\boldsymbol{S}, \mathcal{U})$ *consists of a collection* $\boldsymbol{S}$ *of* d *functions*

$$v_j := f_j(\boldsymbol{PA}_j(\mathcal{G}), u_j),\ j \in [d], \tag{4.22}$$

where $\boldsymbol{PA}_j \subset \{v_1, \ldots, v_d\} \backslash \{v_j\}$ *are called parents of* X_j*; and a joint distribution* $\mathcal{U} = \{u_1, \ldots, u_d\}$ *over the noise variables, which are required to be jointly independent.*

Definition 11 (Causal Graph (Peters et al., 2017), CG)**.** *The causal graph* $\mathcal{G}$ *of an SCM is obtained by creating one node for each* v_j *and drawing directed edges from each parent in* $\boldsymbol{PA}_j^{\mathcal{G}}$ *to* v_j*.*

We note that CG describes the structure of causality and SCM further considers the specific causation of the parents of v_j to v_j via f_j as well as exogenous noises u_j. To uncover the causal structure from the data distribution, we assume that the CG satisfies the *Markov Property* and *Faithfulness* (Peters et al., 2017), which make the independences consistent between the joint distribution $P(v_1, \ldots, v_n)$ and the graph $\mathcal{G}$. We also follow the *Causal Sufficiency* assumption (Spirtes et al., 2000) that supposes that we have measured all common causes of the measured variables.

Existing work (Pitis et al., 2020; Seitzer et al., 2021) believes that two objects have causality only if they are close enough, while there is no edge between them if the distance is large. Instead

of using such a local view of causality, we assume that the causal graph is consistent across all time steps, which also handles the local causal influence. The specific influence indicated by the edges is estimated by the function $f_j(\mathbf{PA}_j, u_j)$.

4.2.2 Generalizing by Discovering (GRADER)

With proper definitions and assumptions, we now look at the proposed method. We first derive the framework of GRADER by formulating the GCRL problem as a latent variable model, which provides a variational lower bound to optimize. Then, we divide this objective function into three parts and iteratively update them with a performance guarantee.

GCRL as Latent Variable Models

The general objective of RL is to maximize the expected reward function with respect to a learnable policy model π. In particular, in the goal-conditioned setting, such an objective is represented as $\max_\pi \mathbb{E}_{\tau\sim\pi, g\sim p(g)}[\sum_{t=0}^{T} r(s^t, g)]$, where $p(g)$ is the distribution of the goal and $\tau := \{s^0, a^0, \ldots, s^T\}$ is the action-state trajectory with time step T. We denote the end of the trajectory as the goal state $g = s^T$.

Inspired by viewing the RL problem as a probabilistic inference (Abdolmaleki et al., 2018; Levine, 2018), we replace the objective from *"How to find actions to achieve the goal?"* to *"What are the actions if we achieve the goal?"*, leading to a likelihood maximization problem for $p(\tau|s^*)$ with $s^* := \mathbb{1}(g = s^T)$. Different from previous work (Marino and Yue, 2019) that recasts actions as latent variables and infers actions that result in "observed" high reward, we

decompose $p(\tau|s^*)$ with $\mathcal{G}$ as the latent variable to get the evidence lower bound (ELBO)

$$\begin{aligned}
\log p(\tau|s^*) &= \log \int p(\tau|\mathcal{G}, s^*) p(\mathcal{G}|s^*) d\mathcal{G} \\
&= \log \int q(\mathcal{G}|\tau) \frac{p(\tau|\mathcal{G}, s^*) p(\mathcal{G}|s^*)}{q(\mathcal{G}|\tau)} d\mathcal{G} \\
&\geq \int q(\mathcal{G}|\tau) \log \frac{p(\tau|\mathcal{G}, s^*) p(\mathcal{G}|s^*)}{q(\mathcal{G}|\tau)} d\mathcal{G} \\
&= \int q(\mathcal{G}|\tau) \left(\log p(\tau|\mathcal{G}, s^*) + \log \frac{p(\mathcal{G}|s^*)}{q(\mathcal{G}|\tau)} \right) d\mathcal{G} \\
&= \int q(\mathcal{G}|\tau) \log p(\tau|\mathcal{G}, s^*) d\mathcal{G} + \int q(\mathcal{G}|\tau) \log \frac{p(\mathcal{G}|s^*)}{q(\mathcal{G}|\tau)} d\mathcal{G} \\
&= \mathbb{E}_{q(\mathcal{G}|\tau)}[\log p(\tau|\mathcal{G}, s^*)] - \mathbb{D}_{\mathrm{KL}}[q(\mathcal{G}|\tau)||p(\mathcal{G})],
\end{aligned} \tag{4.23}$$

where the prior $p(\mathcal{G})$ and variational posterior $q(\mathcal{G}|\tau)$ represent distributions over graph structures, i.e., the probability of the existence of edges in graphs. The third line is obtained by Jensen's inequality and the last line is because the prior of the causal graph $\mathcal{G}$ is independent of the achieved goal s^*. $\mathbb{D}_{\mathrm{KL}}$ denotes the Kullback–Leibler (KL) divergence between two graphs, which will be explained in Section 4.2.2. Recall that the space of the goal G is a subset of the state; we extend the meaning of g and assume that all trajectories achieve the goal in the final state (Andrychowicz et al., 2017), i.e. $g = s^T$. This extension makes it possible to further decompose the first term of (4.23). According to the decomposition of state-action trajectory,

$$p(\tau) = p(s^0) \sum_{t=0}^{T-1} p(s^{t+1}|s^t, a^t) p(a^t|s^t). \tag{4.24}$$

we can get the following result.

$$\begin{aligned}
\log p(\tau|\mathcal{G}, s^*) &= \log(s^0, a^0, s^1, a^1, \cdots, a^{T-1}, s^T|\mathcal{G}, s^*) \\
&= \log p(s^0|\mathcal{G}, s^*) + \sum_{t=0}^{T-1} \log p(s^{t+1}|s^t, a^t, \mathcal{G}, s^*) + \sum_{t=0}^{T-1} \log p(a^t|s^t, \mathcal{G}, s^*) \\
&= \log p(s^0) + \sum_{t=0}^{T-1} \log p(s^{t+1}|s^t, a^t, \mathcal{G}) + \sum_{t=0}^{T-1} \log p(a^t|s^t, \mathcal{G}, s^*).
\end{aligned} \tag{4.25}$$

Here, we use the fact that $\mathcal{G}$ is effective in both the transition model $p(s^{t+1}|a^t, s^t, \mathcal{G})$ and the policy model $\log p(a^t|s^t, s^*, \mathcal{G})$, g only influences the policy model, and the initial state s_0 depends neither on $\mathcal{G}$ nor g. We also assume that the initial state $\log p(s_0)$ is irrelevant to the model parameters. Thus, the first term of (4.25) can be ignored. The policy term π, selecting action a^t according to the current state s^t and the goal g, is implemented with the planning method and is further discussed in Section 4.2.2. Finally, we maximize the likelihood $p(\tau|s^*)$ with the ELBO reformulated as the objective.

$$\mathcal{J}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathcal{G}|\tau)} \sum_{t=0}^{T-1} \left[\log p_\theta(s^{t+1}|s^t, a^t, \mathcal{G}) + \log \pi_\theta(a^t|s^t, s^*, \mathcal{G})\right] - \mathbb{D}_{\mathrm{KL}}[q_\phi(\mathcal{G}|\tau)||p(\mathcal{G})], \tag{4.26}$$

where θ is the shared parameter of transition model $p_\theta(s^{t+1}|a^t, s^t, \mathcal{G})$ and policy $\pi_\theta(a^t|s^t, s^*, \mathcal{G})$, and ϕ is the parameter of causal graph $q_\phi(\mathcal{G}|\tau)$. To efficiently solve this optimization problem, we iteratively update the parameter ϕ (causal discovery, Section 4.2.2) and the parameter θ (model and policy learning, Section 4.2.2), as shown in Figure 4.8. Intuitively, these processes can be, respectively, viewed as the discovery of graph and the update of f_i, which share tight connections as discussed in Section 4.2.1.

Model and Policy Learning

Let us start with a simple case where we already obtain a $\mathcal{G}$ and use it to guide the learning of the parameter θ by $\max_\theta \mathcal{J}(\theta, \phi)$. Since the KL divergence of $\mathcal{J}(\theta, \phi)$ does not involve θ, we only need to deal with the first expectation term, that is, the likelihood of transition model and policy. For the transition $p_\theta(s^{t+1}|a^t, s^t, \mathcal{G})$, we connect it to the causal structure by further defining a particular type of CG and denote it as $\mathcal{G}$ in the rest of this section.

Definition 12 (Transition Causal Graph)**.** *We define a bipartite graph $\mathcal{G}$, whose vertices are divided into two disjoint sets $\mathcal{U} = \{\mathcal{A}^t, \mathcal{S}^t\}$ and $\mathcal{V} = \{\mathcal{S}^{t+1}\}$. $\mathcal{A}^t$ represents action nodes at step t, $\mathcal{S}^t$ state nodes at step t, and $\mathcal{S}^{t+1}$ the state nodes at step $t+1$. All edges start from set $\mathcal{U}$ and end in set $\mathcal{V}$.*

Model learning. This definition builds the causal graph between two consecutive time steps, indicating that the values of states in step $t+1$ depend on the values in step t. It also implies

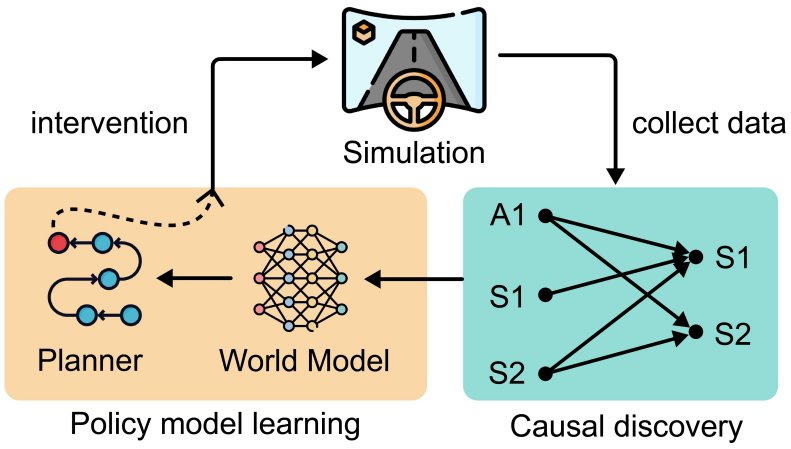


Figure 4.8: The paradigm of GRADER.

that the interventions (Peters et al., 2017) in the nodes in $\mathcal{U}$ are obtained directly since they do not have parent nodes. We denote the marginal distribution of $\mathcal{S}$ as $p_{\mathcal{I}_\pi^s}$, which is collected by the RL policy π. Combined with the definition 10 of SCM, we find that $p_\theta(s^{t+1}|a^t, s^t, \mathcal{G})$ essentially approximates a collection of functions f_j following the structure $\mathcal{G}$, which take as input the values of the parents of the state node s_j and outputs the value s_j. Thus, we propose to model the transition corresponding to $\mathcal{G}$ with a collection of neural networks $f_\theta(\mathcal{G}) := \{f_{\theta_j}\}_{j=1}^M$ to obtain

$$s_j^{t+1} = f_{\theta_j}([\mathbf{PA}_j^{\mathcal{G}}]^t, U_j), \tag{4.27}$$

where $[\mathbf{PA}_j^{\mathcal{G}}]^t$ represents the values of all parents of node s_j^t at time step t and U_j follows Gaussian noise $U_j \sim \mathcal{N}(0, \mathbf{I})$. In practice, we use Gated Recurrent Unit (Chung et al., 2014) as f_j because it supports varying numbers of input nodes. We take s_j^t as the initial hidden embedding, and the rest of the parents $[\mathbf{PA}_j^{\mathcal{G}} \backslash s_j]^t$ as the input sequence to f_j. The entire model is optimized by stochastic gradient descent with the log-likelihood $\log p_\theta(s^{t+1}|a^t, s^t, \mathcal{G})$ as objective.

Policy learning with planning. Then we turn to the policy term $\pi_\theta(a^t|s^t, s^*, \mathcal{G})$ in $\mathcal{J}(\theta, \phi)$. We optimize it with planning methods that take advantage of the estimated transition model. Specifically, the policy aims to optimize an action-state value function.

$$Q(s^t, a^t) = \mathbb{E}\left[\sum_{t'=0}^{H} \gamma^{t'} r(s^{t'+t}, a^{t'+t})|s^t, a^t\right], \tag{4.28}$$

which can be obtained by unrolling the transition model with a horizon of H steps and discount factor γ. In practice, we use model predictive control (MPC) (Camacho and Alba, 2013) with random shooting (Richards, 2005), which selects the first action on the fixed horizon trajectory that has the highest action-state value $Q(s^t, a^t)$, i.e. $\hat{\pi}(s^t) = \arg\max_{a^t \in \mathcal{A}} Q_\theta^{\mathcal{G}}(s^t, a^t)$. The formulation we have derived so far is highly correlated with the model-based RL framework (Wang et al., 2019a). However, the main difference is that we obtain it with variational inference by regarding the causal graph as a latent variable.

Data-Efficient Causal Discovery

In this step, we relax the assumption of knowing $\mathcal{G}$ and aim to estimate the posterior distribution $q(\mathcal{G}|\tau)$ to optimize ELBO (4.26) w.r.t. parameter ϕ. In most score-based methods (Chickering, 2002), likelihood is used to assess the correctness of the causal graph, that is, a better causal graph leads to a higher likelihood. Since the first term of (4.26) represents the likelihood of the transition model, we convert the problem of $\max_\phi \mathcal{J}(\theta, \phi)$ to causal discovery that finds the true causal graph based on the collected data samples. As for the second term of (4.26), the following proposition shows that the KL divergence between $q_\phi(\mathcal{G}|\tau)$ and $p(\mathcal{G})$ can be approximated by sparsity regularization.

Proposition 2 (KL Divergence as Sparsity Regularization)**.** *With entry-wise independent Bernoulli prior $p(\mathcal{G})$ and point mass variational distribution $q(\mathcal{G}|\tau)$ of DAGs, $\mathbb{D}_{KL}[q_\phi \| p]$ is equivalent to an ℓ_1 sparsity regularization for the discovered causal graph.*

Before proofing this proposition, we first make several mild assumptions.

Assumption 4 (Markov property)**.** *Given a DAG $\mathcal{G}$ and a joint distribution $P_{\boldsymbol{X}}$, this distribution is said to satisfy*

- *(i) the global Markov property with respect to the DAG $\mathcal{G}$ if*

$$\boldsymbol{A} \perp\!\!\!\perp_{\mathcal{G}} \boldsymbol{B} | \boldsymbol{C} \Rightarrow \boldsymbol{A} \perp\!\!\!\perp \boldsymbol{B} | \boldsymbol{C} \tag{4.29}$$

  for all disjoint vertex sets $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$. The symbol $independent_{\mathcal{G}}$ denotes d-separation.

- *(ii) the local Markov property with respect to the DAG $\mathcal{G}$ if each variable is independent of its*

non-descendants (without its parents) given its parents, and

- *(iii) the Markov factorization property with respect to the DAG $\mathcal{G}$ if*

$$p(\boldsymbol{x}) = p(x_1, \ldots, x_d) = \prod_{j=1}^{d} p(x_j | \boldsymbol{PA}_j(\mathcal{G})), \tag{4.30}$$

where we assume that $P_{\boldsymbol{X}}$ has a density p.

Assumption 5 (Faithfulness)**.** *Consider a distribution $P_{\boldsymbol{X}}$ and a DAG $\mathcal{G}$, $P_{\boldsymbol{X}}$ is faithful to the DAG $\mathcal{G}$ if we know*

$$\boldsymbol{A} \perp\!\!\!\perp \boldsymbol{B} | \boldsymbol{C} \Rightarrow \boldsymbol{A} \perp\!\!\!\perp_{\mathcal{G}} \boldsymbol{B} | \boldsymbol{C} \tag{4.31}$$

for all disjoint vertex sets $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.

Similar to the assumption in Factorized MDP, we construct the prior distribution $p(\mathcal{G})$ as an independent Bernoulli Distribution in the transition causal graph.

$$p(\mathcal{G}) = \prod_{i \in [M+N], j \in [M]} p(\mathcal{G}_{ij}) = \prod_{i \in [M+N], j \in [M]} p_{ij}, \tag{4.32}$$

where $\mathcal{G}_{ij}$ represents the edge from i-th node in source node set $\mathcal{U} = \{\mathcal{A} \cup \mathcal{S}\}$ to the j-th node in target node set $\mathcal{V} = \{\mathcal{S}'\}$ in the bipartite transition causal graph.

On the other hand, for the variational posterior $q(\mathcal{G}|\tau)$, for the discovered transition causal graph, it needs to satisfy two constraints: (i) We want $q(\mathcal{G}|\tau)$ to be a DAG, denoted $\mathcal{Q}_{DAG}$, and more specifically a bipartite graph. We denote such a subset of DAG as $\mathcal{Q}_{Bi}$, (ii) We want $q(\mathcal{G}|\tau)$ to be as sparse as possible.

Common score-based causal discovery works use two regularization terms, 'DAGNess' and ℓ_1 regularization to constrain the causal graph discovered in the constraint set, while in our work, we explicitly constrain the posterior variational distribution $q(\mathcal{G}|\tau) \in \mathcal{Q}_{Bi} \subset \mathcal{Q}_{DAG}$. We then show in the following section that by defining a certain independent Bernoulli prior $p(\mathcal{G})$, the KL divergence between the variational posterior $q(\mathcal{G}|\tau)$ and $p(\mathcal{G})$ can be equivalent to a sparsity regularization.

According to our constraint-based causal reasoning modules, $\mathcal{Q}_{Bi}$ consists of $M(M+N)$ independent binary classifiers (that form a DAG) parameterized by our kernel-based independent

testing modules ϕ, i.e.

$$q_\phi(\mathcal{G}|\tau) = \prod_{i\in[M+N],j\in[M]} q_\phi(\mathcal{G}_{ij}|\tau) \triangleq \prod_{i\in[M+N],j\in[M]} q_{ij}. \tag{4.33}$$

Proof of Proposition 2. Let the prior $p_{ij} = \epsilon_\mathcal{G} \in (0, \frac{1}{2}], \forall i \in [M+N], j \in [M]$, based on the definition above, the KL divergence term in (4.26) can be expanded as follows:

$$\begin{aligned}
\mathbb{D}_{\text{KL}}(q_\phi(\mathcal{G}|\tau)\|p(\mathcal{G})) &= \sum_{q\in\mathcal{Q}_{Bi}} \left(\prod_{i,j} q_{ij}\right) \log \frac{\prod_{i,j} q_{ij}}{\prod_{i,j} p_{ij}} = \sum_{i,j} \left[q_{ij} \log \frac{q_{ij}}{p_{ij}} + (1-q_{ij}) \log \frac{1-q_{ij}}{1-p_{ij}}\right] \\
&= \sum_{i,j} \left[q_{ij} \log \frac{q_{ij}}{\epsilon_\mathcal{G}} + (1-q_{ij}) \log \frac{1-q_{ij}}{1-\epsilon_\mathcal{G}}\right] \\
&= \sum_{i,j} \left[q_{ij} \log q_{ij} + (1-q_{ij}) \log(1-q_{ij}) - q_{ij} \log \epsilon_\mathcal{G} - (1-q_{ij}) \log(1-\epsilon_\mathcal{G})\right].
\end{aligned} \tag{4.34}$$

Since $q_{ij} \in \{0, 1\}$, $\lim_{q_{ij}\to 0} q_{ij} \log q_{ij} = \lim_{q_{ij}\to 1} (1-q_{ij}) \log(1-q_{ij}) = 0$,

$$\begin{aligned}
\mathbb{D}_{\text{KL}}(q_\phi(\mathcal{G}|\tau)\|p(\mathcal{G})) &= \sum_{i,j} \left[-q_{ij} \log(\epsilon_\mathcal{G}) - (1-q_{ij}) \log(1-\epsilon_\mathcal{G})\right] \\
&= \sum_{i,j} \left[-\mathbb{I}(q_{ij}=1) \log \epsilon_\mathcal{G} - \mathbb{I}(q_{ij}=0) \log(1-\epsilon_\mathcal{G})\right] \\
&= \sum_{i,j} \left[-(1-\mathbb{I}(q_{ij}=1)) \log(1-\epsilon_\mathcal{G}) - \mathbb{I}(q_{ij}=1) \log \epsilon_\mathcal{G}\right] \\
&= \log \frac{1-\epsilon_\mathcal{G}}{\epsilon_\mathcal{G}} \sum_{i,j} \mathbb{I}(q_{ij}=1)) - \sum_{i,j} \log(1-\epsilon_\mathcal{G}) \\
&= \log \left(\frac{1-\epsilon_\mathcal{G}}{\epsilon_\mathcal{G}}\right) |q_\phi(\mathcal{G}|\tau)|_1 + \text{const} \\
&\triangleq \mu |q_\phi(\mathcal{G}|\tau)|_1 + \text{const}.
\end{aligned} \tag{4.35}$$

Therefore, the KL divergence term is equivalent to a ℓ_1 sparsity regularizer in score-based causal discovery (Brouillard et al., 2020). The strength of this regularizer $\mu = \log\left(\frac{1-\epsilon_\mathcal{G}}{\epsilon_\mathcal{G}}\right) \in [0, \infty)$. The larger $\epsilon_\mathcal{G}$ in the prior Bernoulli distribution indicates the smaller strength of this sparsity regularizer (e.g., when $\epsilon_\mathcal{G} = \frac{1}{2}, \mu = 0$). In the implementation of data-efficient causal discovery, we adjust the classifier parameter to set the strength of this sparsity constraint. □

We restrict the posterior $q_\phi(\mathcal{G}|\tau)$ to the point mass distribution and use a threshold η to control the sparsity. We perform the discovery process from the classification perspective by proposing binary classifiers $q_\phi(e_{ij}|\tau,\eta)$ to determine the existence of an edge e_{ij}. This classifier $q_\phi(e_{ij}|\tau,\eta)$ is implemented by statistic *Independent Test* (Chalupka et al., 2018) and η is the threshold for the p-value of the hypothesis. A larger η corresponds to harder sparsity constraints, leading to a sparse $\mathcal{G}$ since two nodes are more likely to be considered independent. According to the definition 12, we only need to perform the classification to the edges connecting the nodes between $\mathcal{U}$ and $\mathcal{V}$. If two nodes are dependent, we add an edge directed from the node in $\mathcal{U}$ to the node in $\mathcal{V}$. This definition also ensures that we always have $q(\mathcal{G}|\tau) \in \mathcal{Q}_{DAG}$, where $\mathcal{Q}_{DAG}$ is the class of DAG. With this procedure, we identify a unique CG $\mathcal{G}^*$ under optimality:

Proposition 3 (Identifiability)**.** *Given an oracle independent test, with an optimal interventional data distribution $p^*_{\mathcal{I}^s_\pi}$, causal discovery provides ϕ^* that correctly tells the independence between two nodes, then the causal graph is uniquely identifiable, with $e^*_{ij} = q_{\phi^*}(e_{ij}|\tau), \forall i \in [M+N], j \in [M]$.*

We construct our causal model based on the factorized MDP in the assumption 3. According to the definition, the causal graph is a directed bipartite graph, with s^t, a^t on the source side and s^{t+1} on the target side. For the theoretical analysis part in Section 4.2.2, we denote s^{t+1} as s', s_t as s, a_t as a and $\boldsymbol{x} = \{\mathcal{A} \cup \mathcal{S} \cup \mathcal{S}'\}$, $\boldsymbol{x} \in \mathbb{R}^{2M+N}$ for simplicity.

Definition 13 (Interventional Family $\mathcal{I}$)**.** *For any DAG $\mathcal{G}$, we define the interventional family $\mathcal{I} = (I_1, I_2, \cdots, I_K)$. Here, $I_1 := \emptyset$ corresponds to the pure observational setting. The joint distribution for the interventional family can be rewritten as:*

$$p^{(k)}(x_1, \cdots, x_{[2M+N]}) = \prod_{j \notin I_k} p^{(1)}_j(x_j|\boldsymbol{PA}^{\mathcal{G}}(x_j)) \prod_{j \in I_k} p^{(k)}_j(x_j|\boldsymbol{PA}^{\mathcal{G}}(x_j)). \tag{4.36}$$

Definition 14. *For a specific DAG $\mathcal{G}$, we define $\mathcal{M}(\mathcal{G})$ to be the set of strictly positive densities $p : \mathbb{R}^{2|\mathcal{S}|+|\mathcal{A}|} \to \mathbb{R}$ which satisfies:*

$$p(x_1, \cdots, x_{[2M+N]}) = \prod_{j \in [2M+N]} p_j(x_j|\boldsymbol{PA}^{\mathcal{G}}(x_j)), \tag{4.37}$$

where $\int_{\mathcal{X}_j} f_j(x_j|\boldsymbol{PA}^{\mathcal{G}}(x_j))dx_j = 1$ *for all* $\boldsymbol{PA}^{\mathcal{G}}(x_j) \in \mathcal{X}_j$ *and all* $j \in [2M+N]$.

Definition 15. *For a specific DAG* $\mathcal{G}$ *and an interventional family* $\mathcal{I}$*, we define*

$$\mathcal{M}_{\mathcal{I}}(\mathcal{G}) := \{[p^{(k)}]_{k\in[K]} \mid \forall k \in [K], p^{(k)} \in \mathcal{M}(\mathcal{G}), \forall j \notin I_k, p_j^{(k)}(x|\boldsymbol{PA}^{\mathcal{G}}(x)) = p_j^{(1)}(x|\boldsymbol{PA}^{\mathcal{G}}(x))\}. \tag{4.38}$$

This set of functions is consistent with the condition of strictly positive densities in (4.37) as well as factorization of the interventional distribution in (4.36).

Definition 16 ($\mathcal{I}$-Markov Equivalence Class, $\mathcal{I}$-MEC)**.** *Two DAGs* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are* $\mathcal{I}$*-Markov equivalent iff* $\mathcal{M}_{\mathcal{I}}(\mathcal{G}_1) = \mathcal{M}_{\mathcal{I}}(\mathcal{G}_2)$*. We denote by* $\mathcal{I} - MEC(\mathcal{G}_1)$ *the set of all DAGs which are* $\mathcal{I}$*-Markov equivalent to* $\mathcal{G}_1$*, this is the* $\mathcal{I}$*-Markov equivalence class of* $\mathcal{G}_1$*.*

Lemma 1 (Sufficient and Necessary Conditions for $\mathcal{I}$-MEC (Yang et al., 2018a))**.** *Suppose the interventional family* $\mathcal{I}$ *is such that* $\mathcal{I}_1 := \emptyset$*. Two DAGs* $\mathcal{G}_1$ *and* $\mathcal{G}_2$ *are* $\mathcal{I}$*-Markov equivalent i.f.f. their I-DAGs* $\mathcal{G}_{\mathcal{I}_1}$ *and* $\mathcal{G}_{\mathcal{I}_2}$ *share the same skeleton and v-structures.*

Proof of Proposition 3. In the bipartite graph $(\mathcal{U}, \mathcal{V}, E)$, for the discovered graph $\hat{\mathcal{G}}$ that is, in the $\mathcal{I}$-Markov equivalence class of the ground truth causal graph, $\hat{\mathcal{G}}$ is unique.

Based on the lemma 1, all possible $\hat{\mathcal{G}}$ that are $\mathcal{I}$-Markov equivalent will share an identical skeleton with G^*, so we consider only graphs obtained by reversing edges in $\hat{\mathcal{G}}$.

Due to the bipartite nature of the transition causal graph defined in Definition 12, for all v-structured colliders $c \in \mathcal{C}$, we know that $c \in \mathcal{S}'$, therefore, reversing any edge of $\hat{\mathcal{G}}$ will harm the immorality of $\hat{\mathcal{G}}$, and the new graph will no longer be an $\mathcal{I}$-MEC to $\mathcal{G}^*$ Therefore, $\hat{\mathcal{G}}$ is the only graph in the $\mathcal{I}$-MEC of $\mathcal{G}^*$, i.e. $\hat{\mathcal{G}} = \mathcal{G}^*$. □

In practice, we use the χ^2-test for discrete variables and the Fast Conditional Independent Test (Chalupka et al., 2018) for continuous variables. The sample size needed to obtain the Oracle test has been widely investigated (Canonne et al., 2018). However, testing with finite data is not a trivial problem, as stated in (Shah and Peters, 2020), especially when the data is sampled from a Goal-conditioned MDP. Usually, the random policy is not enough to satisfy the oracle assumption because some nodes cannot be fully explored when the task is complicated and has a long horizon. To make this assumption empirically possible, it is necessary to simultaneously optimize $\pi_\theta(a^t|s^t, s^*, \mathcal{G})$ to access more samples close to completion of the task, which

is further analyzed in Section 4.2.3. We find that the empirical results support this argument in Section 4.2.4

Details about Conditional Independence Test

In the previous section, we describe the discovery of a causal graph with edge inference $e_{ij} \leftarrow q_\phi(\cdot|\mathcal{B}, \eta)$ implemented by a conditional independent test. We ignore the details in that algorithm about the test process in the main context and thus provide more details in this section.

For discrete variables, we use Pearson's chi-square test, which is a statistical test applied to categorical data sets to evaluate how likely it is that any observed difference between sets arose by chance. In our experiment, we use the implementation provided by Scipy.

We first define the null hypothesis, which is true when two random variables are statistically independent. These two variables have samples stored in a contingency table O, which has c columns and r rows. Then, the "theoretical frequency" for a cell is:

$$E_{ij} = N_{p_{i\cdot}p_{\cdot j}}, \;\; p_{i\cdot} = \sum_{j=1}^{c} \frac{O_{i,j}}{N}, \;\; p_{i\cdot} = \sum_{i=1}^{r} \frac{O_{i,j}}{N}, \tag{4.39}$$

where N is the total sample size in the table, $O_{i,j}$ is the sample size of cell (i, j). Then, we can calculate the value of the test statistic:

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}. \tag{4.40}$$

Now, we can obtain a p-value (falls in $[0, 1]$) that indicates the significance of this statistic follows the χ^2 distribution from chi-square probability. We compare this p-value with a threshold η and reject the null hypothesis if the p-value is smaller than η. Therefore, the larger we set η, the more likely we find that the two variables are dependent. This testing process is summarized in Algorithm 5.

If the two variables are continuous, we can no longer use the above statistical test. We turn to a more advanced test method proposed in (Chalupka et al., 2018). The general idea is that if $P(X|Y, Z) = P(X, Y)$, Z is not useful as a feature to predict X. To achieve this, the authors

Algorithm 5: Independence Test for Discrete Variables.

Input: A contingency table O with samples for two variables X and Y.

Define the null hypothesis: X and Y are independent.

Calculate $p_{i\cdot} = \sum_{j=1}^{c} \frac{O_{i,j}}{N}$ and $p_{i\cdot} = \sum_{i=1}^{r} \frac{O_{i,j}}{N}$

Calculate expected frequencies $E_{ij} = N_{p_{i\cdot}p_{\cdot j}}$

Calculate the chi-square statistic $\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c} \frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}$

Obtain the p-value p from the chi-square probability

if $p < \eta$ **then**

    Reject the Null hypothesis, i.e., X and Y are dependent.

Algorithm 6: GRADER Training

Input: Trajectory buffer $\mathcal{B}_\tau$, Causal graph $\mathcal{G}$, Transition model f_θ, discovery threshold η

while θ *not converged* **do**

    // Policy from planning

    Sample a goal from the training distribution $g \sim p_{\text{train}}(g)$

    while $t < T$ **do**

        Get action from planner $a^t \leftarrow$ Planner(f_θ, s^t, g)

        Get the next state from the environment s^{t+1}, $r^t \leftarrow$ Env(a^t, g)

        Append new transition to buffer $\mathcal{B}_\tau \leftarrow \mathcal{B}_\tau \cup \{a^t, s^t, s^{t+1}\}$

    // Estimate causal graph

    for $i \leq M + N$ **do**

        for $j \leq M$ **do**

            Infer edge $e_{ij} \leftarrow q_\phi(\cdot|\mathcal{B}, \eta)$

    // Learn transition model

    Update $f_\theta(\mathcal{G})$ via (4.27) with $\mathcal{B}$

propose to use decision tree regression to predict Y using both X and Z, and also using Z only.

4.2.3 Analysis of Performance Guarantee

The entire GRADER pipeline is summarized in Algorithm 6. To analyze the performance of optimization of (4.26), we first list important lemmas that connect the previous steps and then show that iteration of these steps in *GRADER* leads to a virtuous cycle.

By the following lemmas, we show the following performance guarantees step by step. Lemma 2 shows model learning is monotonically better at convergence given a better causal graph from causal discovery. Then the learned transition model helps to improve the lower bound of the value function during planning according to the Lemma 4. Lemma 5 reveals the

connection between policy learning and interventional data distribution, which in turn improves the quality of causal discovery, as shown in Lemma 6.

Model Learning with Causal Graph

Lemma 2 (Monotonicity of Transition Likelihood)**.** *Assume $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^*)$ be the true CG, for two CG $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}, \mathcal{E}_2)$, if SHD$(\mathcal{G}_1, \mathcal{G}^*) <$ SHD$(\mathcal{G}_2, \mathcal{G}^*)$, $\exists\, e,\ s.t.\ \mathcal{E}_1 \cup \{e\} = \mathcal{E}_2$, when transition model θ converges, the following inequality holds for the transition model in (4.26):*

$$\log p_\theta(s^{t+1}|a^t, s^t, \mathcal{G}^*) \geq \log p_\theta(s^{t+1}|a^t, s^t, \mathcal{G}_1) \geq \log p_\theta(s^{t+1}|a^t, s^t, \mathcal{G}_2), \tag{4.41}$$

where SHD is the Structural Hamming Distance defined in Definition 17.

The optimization in the model learning step can be described below:

$$\arg\max_\theta [\sum_t \log p_\theta(s^{t+1}|s^t, a^t, \mathcal{G})], \tag{4.42}$$

where $\tau = [s^1, a^1, \cdots, s^T]$ is the trajectory in data buffer, and $\mathcal{G}$ is the given causal graph.

In the following, we show some necessary definitions and propositions to prove the lemma 2.

Definition 17 (Structural Hamming Distance (SHD))**.** *For any two DAGs $\mathcal{G}_1, \mathcal{G}_2$ with identical vertices set $\mathcal{V}$, we define the following function SHD: $\mathcal{G}_1 \times \mathcal{G}_2 \to \mathbb{R}$,*

$$\textit{SHD}(\mathcal{G}_1, \mathcal{G}_2) = \#\{(i,j) \in V^2 \mid \mathcal{G} \textit{ and } \mathcal{H} \textit{ have different edges } e_{ij}\}. \tag{4.43}$$

Definition 18 (Respect the graph)**.** *For any given transition model with specific causal graph $\mathcal{G}$, the transition model respects the graph if the distribution $p(s_{t+1}|a_t, s_t, \mathcal{G})$ can be factorized as:*

$$p(s'|s, a, \mathcal{G}) = \prod_{i \in [M]} p(s'_i | \boldsymbol{PA}(s'_i), \mathcal{G}), \tag{4.44}$$

where M is the total number of factorized states, $\boldsymbol{PA}(\cdot)$ represents the parents in the causal graph.

Proposition 4 (GRU model respects the graph)**.** *As the parameterized transition model* $p_\theta(s'|s, a, \mathcal{G})$ *reaches steady state, it respects the graph.*

Proof of Proposition 4. The GRU modules with parameter $\theta = [W, U]$ can be rewritten as a message passing process, where $AGG.(\cdot)$ is the iterative aggregation function.

$$\begin{aligned} \text{Node Encoder} : h_j^{(0)} &= f_{\text{encoder}}(x_j), \\ \text{Aggregation} : h_j^{(\ell)} &= AGG_{i\in\mathcal{N}(j)}(f_\theta(x_j^{(\ell-1)}, h_i^{(\ell-1)})), \\ \text{Node Decoder} : \boldsymbol{x}_i^{(\ell)} &= f_{\text{decoder}}(h_j^{(\ell-1)}). \end{aligned} \tag{4.45}$$

As an iterated process of message passing, where the input causal graph controls the information flow between different entities, this GRU model can be rewritten as a fixed point iteration (Gilboa et al., 2019):

$$\boldsymbol{x}_i^{(\ell)} = F_\theta(\mathbf{PA}(\boldsymbol{x}_i)^{(\ell-1)}, \boldsymbol{x}_i^{(\ell-1)}). \tag{4.46}$$

With proper initialization and some sufficient conditions provided by (Gilboa et al., 2019), F has a unique equilibrium point, where

$$\boldsymbol{x}_i^{\infty} = F_\theta(\mathbf{PA}(\boldsymbol{x}_i)^{\infty}, \boldsymbol{x}_i^{\infty}). \tag{4.47}$$

In our bipartite graph, when GRU reaches the equilibrium point, we can get a structural causal model:

$$s_i' = F_\theta(\mathbf{PA}(s_i'), s_i), \quad \text{where } s_i' \in \mathcal{S}', \mathbf{PA}(s_i') \in \mathcal{S} \cup \mathcal{A}. \tag{4.48}$$

Based on the SCM derivation 4.48, we can then factorize the transition model as:

$$p_\theta(s'|s, a, \mathcal{G}) = \prod_{i\in[M]} p_\theta(s_i'|\mathbf{PA}(s_i'), \mathcal{G}). \tag{4.49}$$

We denote the ground truth causal graph as $G^* = (\mathcal{V}, \mathcal{E}^*)$, and $\mathbf{PA}^*(s_i')$ as the true parents of s_i' in G^*. □

Definition 19 (Causal optimality at equlibrium point)**.** *For any* $G' \neq G^*$ *with at least one pair*

of flawed parental relationship $\boldsymbol{PA}'(s_i') \neq \boldsymbol{PA}^*(s_i')$*, the following inequality holds:*

$$p_\theta(s_i'|\boldsymbol{PA}'(s_i'), \mathcal{G}) \leq p_\theta(s_i'|\boldsymbol{PA}^*(s_i'), \mathcal{G}). \tag{4.50}$$

Lemma 3 (Local monotonicity)**.** *Given one state variable* s_i *and its parental relationship* $\boldsymbol{PA}^1(s_i), \boldsymbol{PA}^2(s_i)$*, if* $\#(\boldsymbol{PA}^1(s_i) \cup \boldsymbol{PA}^*(s_i)) \geq \#(\boldsymbol{PA}^2(s_i) \cup \boldsymbol{PA}^*(s_i))$*, then in steady state, the SCM derived in 4.48 will miss part of the message provided from the true parents; therefore* $p_\theta(s_i'|\boldsymbol{PA}^1(s_i'), \mathcal{G}_1) \geq p_\theta(s_i'|\boldsymbol{PA}^2(s_i'), \mathcal{G}_2)$

Proof of Lemma 2. Based on the factorization defined in 4.49, we denote the parental relationship in $\mathcal{G}_1$ as $\mathbf{PA}^1(\cdot)$,

$$p_\theta(s'|a, s, \mathcal{G}^*) = \prod_{i\in[M]} p_\theta(s_i'|\mathbf{PA}^*(s_i'), \mathcal{G}) \geq \prod_{i\in[M]} p_\theta(s_i'|\mathbf{PA}^1(s_i'), \mathcal{G}) = p_\theta(s'|a, s, \mathcal{G}_1). \tag{4.51}$$

For $\mathcal{G}_1$ and $\mathcal{G}_2$, suppose the only different edges e has a target node s_j', with $\text{SHD}(\mathcal{G}_1, \mathcal{G}^*) < \text{SHD}(\mathcal{G}_2, \mathcal{G}^*)$, based on Lemma 3:

$$\begin{aligned}
p_\theta(s'|a, s, \mathcal{G}_1) &= p_\theta(s_j'|\mathbf{PA}^1(s_j'), \mathcal{G}_1) \prod_{i\in[M]\setminus j} p_\theta(s_i'|\mathbf{PA}^1(s_i'), \mathcal{G}_1) \\
&\geq p_\theta(s_j'|\mathbf{PA}^2(s_j'), \mathcal{G}_2) \prod_{i\in[M]\setminus j} p_\theta(s_i'|\mathbf{PA}^1(s_i'), \mathcal{G}_2) \\
&= p_\theta(s_j'|\mathbf{PA}^2(s_j'), \mathcal{G}_2) \prod_{i\in[M]\setminus j} p_\theta(s_i'|\mathbf{PA}^2(s_i'), \mathcal{G}_2) \\
&= p_\theta(s'|a, s, \mathcal{G}_2).
\end{aligned} \tag{4.52}$$

Based on the inequality derived in 4.51 and 4.52,

$$\log p_\theta(s'|s, a, \mathcal{G}^*) \geq \log p_\theta(s'|s, a, \mathcal{G}_1) \geq \log p_\theta(s'|s, a, \mathcal{G}_2). \tag{4.53}$$

The monotonicity of likelihood in Lemma 2 is proved. □

Transition Model and Value Function

Lemma 4 (Bounded Value Function in Policy Learning)**.** *Given a planning horizon $H \to \infty$, if we already have an approximate transition model $\mathbb{D}_{TV}(\hat{p}(s'|s,a), p(s'|s,a)) \leq \epsilon_m$, the approximate policy $\hat{\pi}$ achieves a near-optimal value function:*

$$\|V^{\pi^*}(s) - V^{\hat{\pi}}(s)\|_\infty \leq \frac{\gamma}{(1-\gamma)^2}\epsilon_m. \tag{4.54}$$

We could define a more general form of the goal-conditioned reward based on distance: $r(s,g) = 1 - d(s',g)$. Where $\mathbb{D}$ is a (normalized) distance measure between two vectors in the state space, s.t. $\forall s, g \in \mathcal{S}$, $0 \leq d(s,g) \leq 1$. For example, if we choose $d(s,g) = \mathbb{1}(s \neq g)$, the derived reward under this distance measure will go back to the reward function defined in Section 4.1.1. By defining a (normalized) ℓ_p distance between s' and g:

$$d(s,g) = \frac{\|s-g\|_p}{\max_{s_1,s_2\in\mathcal{S}} \|s_1 - s_2\|_p}, \tag{4.55}$$

we can also shape a continuous form of goal-conditioned step reward $r(s,g)$ between 0 and 1. Notice that all the Euclidean-based distances are all valid metrics with symmetry, non-negativity, the identity of indiscernibles, and the triangle inequality. With such a definition, the estimated value function will fit in with:

$$V(s) \leq \frac{1 - \mathcal{W}(p_{\mathcal{I}_\pi^s}, p_g)}{1-\gamma}, \tag{4.56}$$

where $\mathcal{W}$ is some Wasserstein distance between the marginal state distribution and the goal distribution. Therefore, optimizing the value of Q is equivalent to minimizing an upper bound for some types of statistical distance between the goal distribution and the target distribution.

For the term related to policy in (4.26), we define the goal-conditioned policy distribution as:

$$\pi(a^t|s^t, g) \propto \exp(Q(s^t, a^t)). \tag{4.57}$$

As a result, $\arg\max_\pi \sum_{t=0}^{T-1} \log \pi_\theta(a^t|s^t, s^*, \mathcal{G}) = \arg\max_\pi \sum_{t=0}^{T-1} Q(s^t, a^t)$ However, the real

$Q(s^t, a^t)$ is intractable, so we alternatively optimize the $\hat{Q}(s^t, a^t)$ at each time step. Next, we start to derive a bound between $\hat{Q}(s, \hat{\pi}(s))$ and $Q(s, \pi^*(s))$.

Proof of Lemma 4. For simplicity, we denote the learned transition function $\hat{p}(s'|s, a) = p_\theta(s'|s, a, \mathcal{G})$, which is ϵ_m-approximate dynamics, $\mathbb{D}_{\text{TV}}(\hat{p}, p) = \|\hat{p}(s'|s, a) - p(s'|s, a)\|_\infty \leq \epsilon_m$. First, we show by value iteration that the estimated value function $\hat{V}(s)$ will converge to $V(s)$: Assume that there exists $K > 0$, st. $\forall k > K$, $\|\hat{p}(s'|, s, a) - p(s, |s, a)\|_\infty \leq \epsilon_m$

$$\hat{V}^{(k+1)}(s) = r(s, \pi(s)) + \gamma \sum_{s'} p(s'|s, \pi(s))\hat{V}^{(k)}(s). \tag{4.58}$$

Given the result of the Bellman Contraction,

$$\|\hat{V}^{(k+1)}(s) - V^{\pi^*}(s)\|_\infty \leq \gamma \|\hat{V}^{(k)}(s) - V^{\pi^*}(s)\|_\infty. \tag{4.59}$$

$$\lim_{k \to \infty} \|\hat{V}^{(k+1)}(s) - V^{\pi^*}(s)\|_\infty = 0. \tag{4.60}$$

Based on the definition of greedy policy in planning: $\hat{\pi}(s) = \arg\max_{a \in \mathcal{A}} \hat{Q}(s, a)$, we can derive the inequality:

$$r(s, \hat{\pi}(s)) + \gamma \sum_{s'} \hat{p}(s'|s, \hat{\pi}(s))\hat{V}(s') \geq r(s, \pi^*(s)) + \gamma \sum_{s'} \hat{p}(s'|s, \pi^*(s))\hat{V}(s'). \tag{4.61}$$

$$r(s, \pi^*(s)) - r(s, \hat{\pi}(s)) \leq \gamma \left[\sum_{s'} \hat{p}(s'|s, \hat{\pi}(s))\hat{V}(s') - \sum_{s'} \hat{p}(s'|s, \pi^*(s))\hat{V}(s') \right]. \tag{4.62}$$

We can get the following results with $\|\hat{V}(s) - V^{\pi^*}(s)\|_\infty \to 0$:

$$r(s, \pi^*(s)) - r(s, \hat{\pi}(s)) \leq \gamma \left[\sum_{s'} \hat{p}(s'|s, \hat{\pi}(s))V^{\pi^*}(s') - \sum_{s'} \hat{p}(s'|s, \pi^*(s))V^{\pi^*}(s') \right]. \tag{4.63}$$

Let s be the state with the largest error of the value.

$$
\begin{aligned}
& V^{\pi^*}(s) - V^{\hat{\pi}}(s) \\
=& r(s, \pi^*(s)) - r(s, \hat{\pi}(s)) \\
& + \gamma \Big[\sum_{s'} p(s'|s, \pi^*(s)) V^{\pi^*}(s') - \sum_{s'} p(s'|s, \hat{\pi}(s)) V^{\hat{\pi}}(s') \Big] \\
\leq & \gamma \sum_{s'} \Big[\hat{p}(s'|s, \hat{\pi}(s)) V^{\pi^*}(s') - \hat{p}(s'|s, \pi^*(s)) V^{\pi^*}(s') \Big] \\
& + \gamma \sum_{s'} \Big[p(s'|s, \pi^*(s)) V^{\pi^*}(s') - p(s'|s, \hat{\pi}(s)) V^{\hat{\pi}}(s') \Big] \\
=& \gamma \sum_{s'} \Big[p(s'|s, \pi^*(s)) - \hat{p}(s'|s, \pi^*(s)) \Big] V^{\pi^*}(s') - \gamma \sum_{s'} \Big[p(s'|s, \hat{\pi}(s)) - \hat{p}(s'|s, \hat{\pi}(s)) \Big] V^{\pi^*}(s') \\
& + \gamma \sum_{s'} p(s'|s, \hat{\pi}(s)) \Big[V^{\pi^*}(s) - V^{\hat{\pi}}(s) \Big].
\end{aligned}
\tag{4.64}
$$

Since $r(s, g) \in [0, 1]$, the value function $V(s) \in [0, \frac{1}{1-\gamma}]$, also by $\|\hat{p}(s'|s, \hat{\pi}(s)) - p(s'|s, \hat{\pi}(s))\| \leq \epsilon$, we have

$$
\begin{aligned}
V^{\pi^*}(s) - V^{\hat{\pi}}(s) &\leq \gamma \epsilon_m (V_{\max} - V_{\min}) + \gamma \sum_{s'} p(s'|s, \hat{\pi}(s)) \Big[V^{\pi^*}(s) - V^{\hat{\pi}}(s) \Big] \\
&= \frac{\gamma \epsilon_m}{1 - \gamma} + \gamma \sum_{s'} p(s'|s, \hat{\pi}(s)) \left[V^{\pi^*}(s') - V^{\hat{\pi}}(s') \right].
\end{aligned}
\tag{4.65}
$$

We already analyzed the state s with the largest value error, and it is sufficient to show:

$$
\begin{aligned}
\|V^{\pi^*}(s) - V^{\hat{\pi}}(s)\|_\infty &\leq \frac{\gamma \epsilon_m}{1 - \gamma} + \gamma \sum_{s'} p(s'|s, \hat{\pi}) \|V^{\hat{\pi}}(s) - V^{\pi^*}(s)\|_\infty \\
&= \frac{\gamma \epsilon_m}{1 - \gamma} + \gamma \|V^{\pi^*}(s) - V^{\hat{\pi}}(s)\|_\infty.
\end{aligned}
\tag{4.66}
$$

By combining $\|V^{\pi^*}(s) - V^{\hat{\pi}}(s)\|_\infty$ on both sides, we have

$$
\|V^{\pi^*}(s) - V^{\hat{\pi}}(s)\|_\infty \leq \frac{\gamma}{(1 - \gamma)^2} \epsilon_m. \tag{4.67}
$$

□

Policy Learning Improves Interventional Data Distribution

Lemma 5. *With a step reward $r(s,a) = \mathbb{1}(s=g)$, we show that the value function determines an upper bound for TV divergence between the interventional distribution with its optimal:*

$$\mathbb{D}_{TV}(p_{\mathcal{I}_\pi^s}, p_g) \leq 1 - (1-\gamma)V^\pi(s). \tag{4.68}$$

where $p_{\mathcal{I}_\pi^s}$ is the marginal state distribution in interventional data, and p_g is the goal distribution. A better policy with larger $V^\pi(s)$ enforces the distribution of interventional data toward the goal.

The optimization in the planning part is:

$$\max_\pi \sum_{t=0}^{T-1} \log \pi_\theta(a^t|s^t, \mathcal{G}, s^*) = \max_{[a_0,\cdots,a^{T-1}]} \sum_{t=0}^{T-1} \log \hat{Q}(s^t, a^t). \tag{4.69}$$

Ideally, given access to real dynamics $p(s'|s,a)$ and goal distribution $p(g)$. We first define the expected goal-conditioned state-action reward $r(s,a,g) = \mathbb{E}_{s'\sim p(\cdot|s,a)} r(s',g)$, and the expected state-action reward $r(s,a) = \mathbb{E}_{g\sim p_g(\cdot)} r(s,a,g)$. In practice, due to the inaccuracy of the transition model, we can only query the following reward estimate for a certain state-action pair: $r(s,a,g) = \mathbb{E}_{s'\sim p_\theta(\cdot|s,a,\mathcal{G})} r(s',g), r(s,a) = \mathbb{E}_{g\sim p_g(\cdot)} r(s,a,g)$.

Then we consider the distribution of the goal $g \sim p_g(\cdot)$, which is supported in the state space $\mathcal{S}$. Based on Algorithm 6, our interventional data is collected by the MPC that maximizes the expected discounted cumulative reward of the learned dynamics. Thus, we could denote the interventional distribution of state (depending on the current policy π) in the data buffer as $p_{\mathcal{I}_\pi^s}$, $s \sim p_{\mathcal{I}_\pi^s}(\cdot)$ which is also supported in the state space $\mathcal{S}$.

Proof of Lemma 5. Assume our planning algorithm has an infinite planning horizon, with the optimal transition dynamics and optimal policy, the action-value function Q^* can be expressed as:

$$Q^*(s^t, a^t) \overset{\text{def}}{=} \mathbb{E}_{s\sim p_{\mathcal{I}_{\pi^*}^s}(\cdot), a\sim\pi^*(s)} \left[\sum_{t'=t}^{\infty} \gamma^{t'-t} r(s^{t'}, a^{t'}) \mid s^t, a^t \right]. \tag{4.70}$$

The estimation of action value function $\hat{Q}(s^t, a^t) = Q^{\hat{\pi}}_{\theta,\mathcal{G}}(s^t, a^t)$ can be written as:

$$\begin{aligned}\hat{Q}(s^t, a^t) &\stackrel{\text{def}}{=} \mathbb{E}_{s\sim p_{\mathcal{I}^s_{\hat{\pi}}}(\cdot), a\sim\hat{\pi}(s)}\left[\sum_{t'=t}^{\infty}\gamma^{t'-t} r(s^{t'}, a^{t'}) \mid s^t, a^t\right] \\ &= \mathbb{E}_{a\sim\hat{\pi}(s)}\left[\sum_{t'=t}^{\infty}\gamma^{t'-t}\mathbb{E}_{g\sim p_g(\cdot), s\sim p_{\mathcal{I}^s_{\hat{\pi}}}(\cdot)}(1-\mathbb{1}(s'=g)) \mid s^t, a^t\right].\end{aligned} \tag{4.71}$$

The policy by MPC in algorithm 6 can be deducted by: $\hat{\pi}(s^t) = \arg\max_{a^t\in\mathcal{A}} \hat{Q}(s^t, a^t)$, let $s^0 = s$, and we could derive value function under the MPC policy as follows:

$$\begin{aligned}V(s) &= \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{p_g}\mathbb{E}_{p_{\mathcal{I}^s_{\hat{\pi}}}}\mathbb{1}(s=g) = \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{p_g}\mathbb{E}_{p_{\mathcal{I}^s_{\hat{\pi}}}}[1-\mathbb{1}(s\neq g)] \\ &= \sum_{t=0}^{\infty}\gamma^t - \sum_{t=0}^{\infty}\gamma^t \mathbb{E}_{p_g}\mathbb{E}_{p_{\mathcal{I}^s_{\hat{\pi}}}}\mathbb{1}(s\neq g) \\ &\leq \frac{1-\mathbb{D}_{TV}(p_{\mathcal{I}^s_{\hat{\pi}}}, p_g)}{1-\gamma}.\end{aligned} \tag{4.72}$$

where $\mathbb{D}_{\text{TV}}(p_{\mathcal{I}^s_{\hat{\pi}}}, p_g)$ is the total variation distance between the marginal state distribution $p_{\mathcal{I}^s_{\hat{\pi}}}$ in the data buffer, as well as the goal distribution p_g, both of which share the same support. □

Interventional Data Benefits Causal Discovery

Lemma 6. *For $\epsilon_g = \min_{p_g>0} p_g$, $\mathbb{D}_{TV}(p_{\mathcal{I}^s_{\hat{\pi}}}, p_g) < \epsilon_g$, the error of our causal discovery is upper-bounded with $\mathbb{E}_{\hat{\mathcal{G}}}[SHD(\hat{\mathcal{G}}, \mathcal{G}^*)] \leq |\mathcal{S}| - 1$.*

Before we put the formal proof, we first list several assumptions that are quite common in the causal discovery literature (Addanki et al., 2020).

Assumption 6 (Oracle Conditional Independence Test)**.** *The conditional independent test could tell the independence between any two variables in the causal graph.*

Proof of Lemma 6. Given the Oracle conditional independence test in 6, if the state distribution covers all the support of the goal distribution, with abundant actions from the action space, we can cover all the connections between the current state and the next states.

When $\mathbb{D}_{\text{TV}}(p_{\mathcal{I}^s_{\hat{\pi}}}, p_g) < \epsilon_g$, it is sufficient to derive that $p_{\mathcal{I}^s_{\hat{\pi}}}(s) > 0, \forall s \in \text{Supp}_g$, where Supp_g

is the support set of the goal distribution p_g.

We then discuss the three possible circumstances under such a condition:

- Case 1: Both the source state and target state distributions in the buffer are fully supported in Supp_g. In this case, given our assumption 6 and abundant samples, our causal discovery $q_\phi(\mathcal{G}|\tau)$ will correctly classify all the edges of the transition of the causal graph.
- Case 2: Only the target state distribution is supported on Supp_g, while the source side is away from the goal node. In this case, independent tests may not be able to distinguish the (in)dependence relationship between goal nodes and other state nodes, $\text{SHD}(\hat{\mathcal{G}}, \mathcal{G}^*) \leq |\mathcal{S}| - 1$.
- Case 3: Only the source state is supported on Supp_g, while the target side is away from the goal node. This case corresponds to the case where some initial states hit the goal, while the learned transition model and policy fail to guide the future states to the goal. The causal discovery model ϕ may falsely classify the edges of all the source states (except the source goal state) toward the target goal states. Thus, $\text{SHD}(\hat{\mathcal{G}}, \mathcal{G}^*) \leq |\mathcal{S}| - 1$.

In conclusion, for all the three cases that satisfy $\mathbb{D}_{\text{TV}}(p_{\mathcal{I}_\pi^s}, p_g) \leq \epsilon_g$, we have

$$\mathbb{E}_{\hat{\mathcal{G}} \sim q_\phi(\cdot|\tau)} \Big[\text{SHD}(\hat{\mathcal{G}}, \mathcal{G}^*) \Big] \leq \max_{\hat{\mathcal{G}} \sim q_\phi(\cdot|\tau)} \Big[\text{SHD}(\hat{\mathcal{G}}, \mathcal{G}^*) \Big] \leq |\mathcal{S}| - 1. \tag{4.73}$$

□

After the close-loop analysis of our model, we are now able to analyze the overall performance of the proposed framework. Under the construction of $p_\theta(s^{t+1}|a^t, s^t, \mathcal{G})$ with NN parameterized functions, the following theorem shows that the learning process will guarantee the accurate estimation of the true ELBO under iterative optimization between model learning, planning, and causal discovery.

Theorem 2. *With T-step approximate transition dynamics* $\mathbb{D}_{TV}\Big(\hat{p}(s'|s,a), p(s'|s,a)\Big) \leq \epsilon_m$*, if the goal distribution satisfies* $\epsilon_g > \frac{\gamma}{1-\gamma}\epsilon_m$*, and the distribution prior CG is entry-wise independent*

Bernoulli($\epsilon_{\mathcal{G}}$)*, GRADER guarantees to achieve an approximate ELBO* $\hat{\mathcal{J}}$ *to the true ELBO* $\mathcal{J}^*$*:*

$$\|\mathcal{J}^*(\theta,\phi) - \hat{\mathcal{J}}(\hat{\theta},\hat{\phi})\|_\infty \leq \left[1 + \frac{\gamma}{(1-\gamma)^2}\right]\epsilon_m T + \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right)(|\mathcal{S}|-1), \tag{4.74}$$

An intuitive understanding of the performance guarantee is that a better transition model indicates a better approximation of the objective $\mathcal{J}$. Based on all the derivations from the previous sections, we finally give the proof of the overall performance of the iterative optimization in *GRADER*.

Proof of Theorem 2. Let $d_{max} = \max_{s_1,s_2\in\mathcal{S}} \|s_1 - s_2\|^2, d_\theta = \|\hat{s}'(\theta) - s'\|^2$, then the log likelihood term becomes

$$p_\theta(s'|s,a) \propto \exp(d_{max} - d_\theta). \tag{4.75}$$

In the model learning part, since we take the log space, we have $\log p_\theta(s'|s,a) = (d_{max} - d_\theta) - C$. We neglect the constant term C when deriving the bound. Without loss of generality, we set $p_\theta(s'|s,a) = \exp(d_{max} - d_\theta), \log p_\theta(s'|s,a) = d_{max} - d_\theta$ in (4.26). As $d_{max} - d_\theta \geq 0$, we have the Lipchitz $L \leq 1$ of log function,

$$\left\|\log\hat{p}(s'|s,a) - \log p(s'|s,a)\right\|_\infty \leq L\left\|\hat{p}(s'|s,a) - p(s'|s,a)\right\|_\infty \leq \left\|\hat{p}(s'|s,a) - p(s'|s,a)\right\|_\infty \leq \epsilon_m. \tag{4.76}$$

Based on Lemma 4, we have the policy learning term

$$\begin{aligned}\left\|\log\hat{\pi}(a|s,g) - \log\pi^*(a|s,g)]\right\|_\infty &= \left\|\hat{Q}(s,\hat{\pi}(s)) - Q(s,\pi^*(s))\right\|_\infty \\ &= \left\|Q(s,\hat{\pi}(s)) - Q(s,\pi^*(s))\right\|_\infty \\ &= \left\|V^{\hat{\pi}}(s) - V^{\pi^*}(s)\right\|_\infty \leq \frac{\gamma}{(1-\gamma)^2}\epsilon_m.\end{aligned} \tag{4.77}$$

For the KL divergence term, if the goal distribution satisfies $\epsilon_g > \frac{\gamma}{1-\gamma}\epsilon_m$, the following conditions hold:

$$V^{\hat{\pi}}(s) > V^{\pi^*}(s) - \frac{\gamma}{(1-\gamma)^2}\epsilon_m. \tag{4.78}$$

According to Lemma 5 and the condition that $V(s) \in [0, \frac{1}{1-\gamma}]$,

$$\mathbb{D}_{\text{TV}}(p_{\mathcal{I}_\pi^s}, p_g) \leq 1 - (1-\gamma)V^{\hat{\pi}}(s) < 1 - (1-\gamma)V^{\pi^*}(s) + \frac{\gamma}{1-\gamma}\epsilon_m = (1-\gamma^{t^*-1}) + \epsilon_g \overset{t^*=1}{=} \epsilon_g, \tag{4.79}$$

where t^* is the shortest time step to reach the goal. We assume $t^* = 1$ for the optimal policy in the theoretical design part, while in practice the bound loosens for larger t^* or smaller γ.

Since $\mathbb{D}_{\text{TV}}(p_{\mathcal{I}_\pi^s}, p_g) \leq \epsilon_g$, according to Lemma 6, we have

$$\begin{aligned}
&\Big\|\mathbb{D}_{\text{KL}}(q_\phi||p) - \mathbb{D}_{\text{KL}}(q_\phi^*||p)\Big\|_\infty \\
&= \Big\| \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \|q_\phi(\mathcal{G}|\tau)\|_1 - \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \|q_\phi^*(\mathcal{G}|\tau)\|_1 \Big\|_\infty \\
&= \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \Big\| \|q_\phi(\mathcal{G}|\tau)\|_1 - \|q_\phi^*(\mathcal{G}|\tau)\|_1 \Big\|_\infty \\
&\leq \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \Big\| q_\phi(\mathcal{G}|\tau) - q_\phi^*(\mathcal{G}|\tau) \Big\|_\infty \\
&= \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \max_{\mathcal{G}} \Big[\text{SHD}(\mathcal{G}, \mathcal{G}^*)\Big] \\
&\leq \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \Big(|\mathcal{S}| - 1\Big).
\end{aligned} \tag{4.80}$$

Finally, we can derive the overall performance guarantee as follows:

$$\begin{aligned}
&\|\mathcal{J}^*(\theta, \phi) - \hat{\mathcal{J}}(\hat{\theta}, \hat{\phi})\|_\infty \\
&= \Big\| \sum_{t=0}^{T-1} \Big\{ \Big[\log \hat{p}(s^{t+1}|s^t, a^t) - \log p(s^{t+1}|s^t, a^t) \Big] \\
&\quad + \Big[\log \hat{\pi}(a^t|s^t, s^*) - \log \pi^*(a^t|s^t, s^*) \Big] \Big\} + \Big[\mathbb{D}_{\text{KL}}(\hat{q}_\phi||p) - \mathbb{D}_{\text{KL}}(q_\phi^*||p) \Big] \Big\|_\infty \\
&\leq \sum_{t=0}^{T-1} \Big(\epsilon_m + \frac{\gamma}{(1-\gamma)^2}\epsilon_m \Big) + \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \Big(|\mathcal{S}| - 1\Big) \\
&= \Big[1 + \frac{\gamma}{(1-\gamma)^2}\Big]\epsilon_m T + \log\left(\frac{1-\epsilon_{\mathcal{G}}}{\epsilon_{\mathcal{G}}}\right) \Big(|\mathcal{S}| - 1\Big).
\end{aligned} \tag{4.81}$$

□

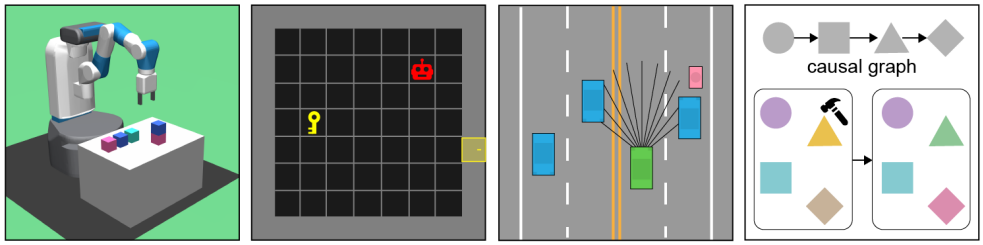


Figure 4.9: Environments used in experiments.

Table 4.3: Environment configurations used in experiments

Parameters	Stack	Unlock	Crash	Chemistry
Max step size	5	15	30	10
State dimension	50	110	22	100
Action dimension	12	8	8	100
Action type	Discrete	Discrete	Continuous	Discrete

4.2.4 Experiment and Analysis

In this section, we first discuss the setting of our designed environments, as well as the baselines used in the experiments. Then, we provide numerical results and detailed discussions to answer the following important research questions: **Q1.** Compared to existing strong baselines, how does GRADER gain performance improvement under both in-distribution and generalization settings? **Q2.** Compared to a random offline policy, how does a well-trained policy improve the results of causal discovery? **Q3.** Compared to score-based causal discovery, does the proposed data-efficient causal discovery pipeline guarantee the identification of the true causal graph as stated in Section 4.2.2? **Q4.** Considering the correctness of causal graphs, how does the imperfect causal graph influence the task-solving performance of GCRL agents?

Environments and Baselines

Since most commonly used RL benchmarks do not explicitly require causal reasoning for generalization, we design three new environments in addition to the *Chemistry* (Ke et al., 2021) environment, which are shown in Figure 4.9. These environments use the true state as observation to disentangle the reasoning task from visual understanding. For each environment, we

design three settings – in-distribution (*I*), spuriousness (*S*), and composition (*C*) – corresponding to different goal distributions for generalization. We use $p_{\text{train}}(g)$ and $p_{\text{test}}(g)$ to represent the goal distribution during training and testing, respectively. I uses the same $p_{\text{train}}(g)$ and $p_{\text{test}}(g)$, S introduces spurious correlations in $p_{\text{train}}(g)$ but remove them in $p_{\text{test}}(g)$, and C contains more sub-goals in $p_{\text{test}}(g)$ than in $p_{\text{train}}(g)$. The details of these settings are briefly summarized in the following.

- *Stack*: We design this manipulation task inspired by the CausalWorld (Ahmed et al., 2020), where the agent must stack objects to match specific shapes and colors. In *Stack-S*, we let the same shape have the same color in $p_{\text{train}}(g)$ but randomly sample the color and shape in $p_{\text{test}}(g)$. In *Stack-C*, the maximum number of objects is two in $p_{\text{train}}(g)$ but five in $p_{\text{test}}(g)$.
- *Unlock*: We designed this indoor housekeeping task for the agent to collect a key to open the doors. This environment is built on the Minigrid (Chevalier-Boisvert et al., 2018). In *Stack-S*, the door and the key are always in the same row in $p_{\text{train}}(g)$ but uniformly sampled in $p_{\text{test}}(g)$. In *Unlock-C*, there are one door in $p_{\text{train}}(g)$ but two doors in $p_{\text{test}}(g)$.
- *Crash*: The occurrence of accidents is usually based on causality, for example, an autonomous vehicle (AV) collides with a jaywalker because its view is blocked by another car (Tavares et al., 2021). We design such a crash scenario based on highway-env (Leurent, 2018), where the goals are to create crashes between a pedestrian and different AVs. In *Stack-S*, the initial distance between the AV and the pedestrian is constant in $p_{\text{train}}(g)$ but irrelevant in $p_{\text{test}}(g)$. In *Stack-C*, there is one pedestrian in $p_{\text{train}}(g)$ but two in $p_{\text{test}}(g)$.
- *Chemistry* (Ke et al., 2021): There are 10 nodes with different colors. An underlying causal graph controls the color-changing mechanism of all nodes. In one step, the agent changes the color of one node. The goal is to match the given colors of all nodes. In the spuriousness setting, we let all nodes have the same target color. There is no composition setting in this environment.

We use the following methods as our baselines to fairly demonstrate the advantages of GRADER. **SAC:** (Haarnoja et al., 2018) Soft Actor-Critic is a well-known model-free RL method that uses entropy to increase the diversity of action. **ICIN:** (Nair et al., 2019) It uses DAgger (Ross et al., 2011) to learn the goal-conditioned policy with the causal graph estimated

Table 4.4: Success rate (%) for nine settings in three environments. **Bold** font means the best.

Method	Stack-I	Stack-S	Stack-C	Unlock-I	Unlock-S	Unlock-C	Crash-I	Crash-S	Crash-C
SAC	34.7±16.1	22.1±14.0	31.7±5.1	0.1±0.5	0.0±0.2	0.4±1.7	22.5±17.6	18.6±8.7	6.7±3.8
ICIN	71.8±6.9	71.0±7.4	58.6±8.3	31.7±9.6	32.7±8.6	31.5±8.5	27.9±6.1	15.8±17.2	7.8±8.8
PETS	**97.2±6.9**	77.7±13.5	73.7±10.3	59.5±7.2	20.6±5.9	28.3±10.0	52.3±11.5	44.6±12.5	37.1±5.1
TICSA	85.9±8.4	88.8±10.1	76.2±8.3	58.5±12.3	33.6±14.3	29.8±8.3	68.9±5.9	56.8±8.6	15.0±8.2
ICIL	93.7±5.9	81.2±14.4	62.8±13.0	**67.1±11.6**	15.9±4.7	53.6±15.3	55.3±20.9	21.7±17.7	14.3±7.3
GNN	45.7±9.1	39.0±10.4	41.7±8.6	3.4±2.3	3.4±2.4	4.5±3.0	4.2±4.0	5.1±5.1	3.8±2.8
Score	92.7±7.4	90.5±7.5	73.9±8.5	44.9±28.1	23.1±7.6	36.2±30.1	42.3±17.5	53.4±18.7	8.4±6.1
Full	92.9±6.3	86.0±9.5	75.7±10.3	63.8±9.2	18.3±7.4	53.7±14.3	69.8±14.0	52.6±12.8	42.0±17.2
Offline	96.8±5.8	95.4±6.1	81.4±7.8	13.8±8.1	13.9±7.5	11.7±6.9	13.1±16.2	30.2±16.5	14.9±12.4
GRADER	95.6±5.4	**97.6±6.0**	**93.7±8.4**	64.2±9.1	**61.4±4.4**	**82.1±9.2**	**91.5±4.4**	**84.3±10.0**	**84.7±7.3**

from the expert policy. We assume that it can access the true causal graph for supervised learning. **PETS:** (Chua et al., 2018) We consider the ensemble transition model with random shoot planning as a baseline, which achieves generalization with the uncertainty-aware design. **TICSA:** (Wang et al., 2021d) This is a causal-augmented MBRL method that simultaneously optimizes a soft adjacent matrix (representing causality) and a transition model. **ICIL:** (Bica et al., 2021) This method proposes an invariant feature learning structure that captures the implicit causality of multiple tasks. We only use it for transition model learning since the original method is designed for imitation learning. **GNN:** (Schlichtkrull et al., 2018) Since graph neural networks are good at learning structural information, we implement a GNN-based baseline using a Relational Graph Convolutional Network.

Dtails of Environment Design

More details about the design of the environments are summarized in the following. The basic parameters of all environments can be found in Figure 4.3.

Stack: Manipulation is important for household and factory assembly. Sometimes, the color of the object is not relevant to the task but may leak information by sharing spuriousness with the task. Also, the goal could be composed of several previously seen goals, such as repeating similar actions. Totally, we have 5 different shapes and 5 different colors and the goals are some combinations of the colors and shapes. At each step, the agent can stack an object with a chosen color and shape or stop stacking. The state is the colors and shapes of all current objects. The agent receives a positive reward when the goal is achieved and a punishment if it stacks a new

object.

Unlock: Collecting specific objects to meet the required conditions is useful for mobile robots. In this environment, there is a causality between the key and the door. The action contains six operations, including four-direction movements (Move), pick key (Pick Key), and open door (Open Door). The state is the position of the agent, the position of the key, and the status of the door. In the first generalization setting, we intentionally create a spurious correlation between the position of the key and the door. If the agent figures out that the key can open the door regardless of its position, it will ignore the spurious correlation. In the second generalization setting, we increase the number of doors from one to two. This setting contains two sub-tasks and can be used to test the compositional generalization.

Crash: The causality in this environment is mainly between the pedestrian (Ped), the ego vehicle (Ego), and another vehicle (Car 1) (Ding et al., 2022b). The collision between Ped and Ego only occurs when the view of Ego is blocked by Car 1. To make this happen, we designed a rule-based AV that will brake if it detects any obstacles within a certain distance. Therefore, if the pedestrian directly hits the AV, the AV will stop and the crash will not occur. To make this task difficult, we also placed two other vehicles (Car 2 and Car 3) on the scene, but they will not interrupt the crash scenario. The agent can control the acceleration and steering of Ped, Car 1, Car 2, and Car 3. The state is the position and velocity of all objects plus the status of whether a collision occurs. To create a spurious correlation, we fix the initial distance between Ego and Ped to a constant, since this creates a shortcut for the feature extractor. However, remembering this distance is not enough since we change the initial distance in the testing stage.

Chemistry: Please refer to (Ke et al., 2021) for more details.

Results Discussion

Next, we show the results to answer the following four research questions.

Overall Performance (Q1). We compare the test reward of all methods under nine tasks and summarize the results in Table 4.4 to demonstrate overall performance. Generally, our method outperforms the baselines in all tasks except *Stack-I* because this task is too simple for all methods. We note that the gap between our method and baselines in *S* and *C* settings is more signifi-

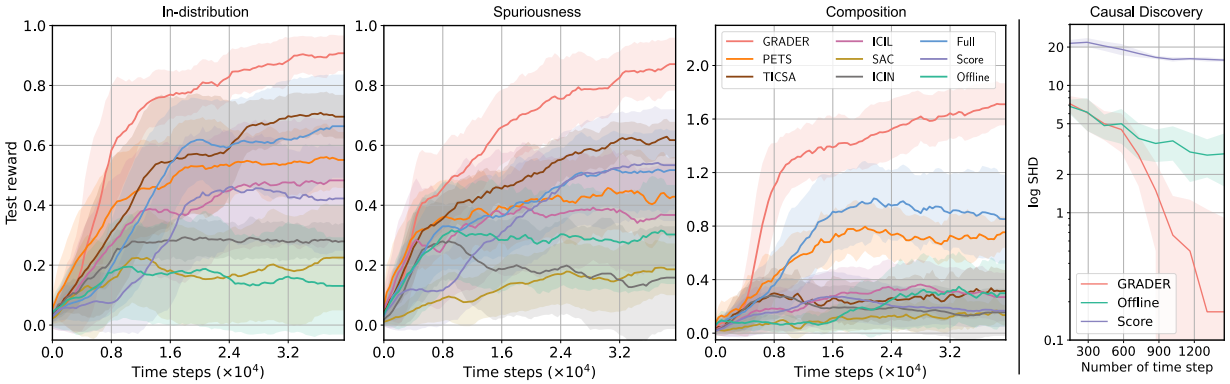


Figure 4.10: **Left:** Test reward of the *Crash* environment. **Right:** Accuracy of causal graph discovery.

Table 4.5: Results on *Chemistry* environment (GRADER/Score). Bold font means the best.

Metric	Collider	Chain	Jungle	Full
SHD (↓)	**3.70±1.79**/15.4±7.03	**2.80±1.83**/14.0±1.18	**7.00±2.19**/13.8±0.40	**2.40±1.20**/11.0±5.31
Accuracy (↑)	**0.99±0.00**/0.87±0.06	**0.99±0.00**/0.88±0.01	**0.98±0.00**/0.89±0.00	**0.99±0.00**/0.91±0.09
Precision (↑)	**0.90±0.05**/0.73±0.10	**0.94±0.04**/0.79±0.03	0.86±0.04/**0.88±0.01**	1.00±0.00/1.00±0.00
Recall (↑)	**0.99±0.02**/0.83±0.07	**0.96±0.03**/0.73±0.00	**0.96±0.02**/0.73±0.00	**0.96±0.02**/0.83±0.17
F-score (↑)	**0.94±0.03**/0.77±0.06	**0.95±0.03**/0.76±0.02	**0.91±0.03**/0.80±0.00	**0.98±0.01**/0.90±0.10

cant than in the *I* setting, showing that our method still works well in the nontrivial generalization task. As a model-free method, SAC fails in all three tasks of *Unlock* and *Crash* environments since they have very sparse rewards. Without learning the causal structure of the environment, PETS even cannot fully solve *Unlock-S*, *Unlock-C*, and all *Crash* tasks. Both TICSA and ICIL learn the causality underlying the task so that they are relatively better than SAC and PETS. However, they are still worse than GRADER in two generalization settings due to the unstable and inefficient causal reasoning mechanism. We also find that even if ICIN is given the true causal graph, the policy learning part cannot efficiently leverage the causality, leading to worse performance in generalization settings.

To further analyze the learning trend, we plot the curves of all the methods in *Crash* in Figure 4.10. Our method quickly learns to solve tasks at the beginning of training, demonstrating high data efficiency. GRADER also outperforms other methods with large gaps in the later training phase. The training figures for the other two environments can be found in 4.2.4.

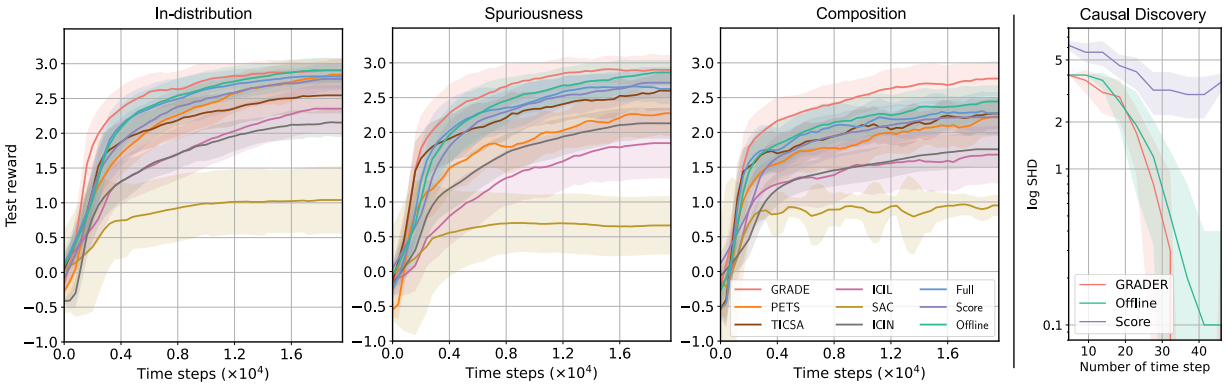


Figure 4.11: The testing reward and causal discovery results of *Stack* environment.

Importance of Policy Learning (Q2). As we mentioned in Section 4.2.2, we empirically compare GRADER and **Offline** (Zhu et al., 2022), which uses data collected from offline random policy, and plot results in the right part of Figure 4.10. We use SHD (Tsamardinos et al., 2006) to compute the distance between the estimated causal graph and the true causal graph. The true causal graph for each environment can be found in 4.2.4. When we only use samples collected offline by random policy, we cannot obtain values of nodes in $\mathcal{G}$ that require long-horizon reasoning, for example, the door can be opened only if the agent is close to the door and has the key. As a consequence, the causal graph obtained by the Offline method harms the performance, as shown in Figure 4.10. Instead, GRADER gradually explores more regions and quickly obtains the true causal graph when we iteratively discover the causal graph and update the policy.

Advantage of Data-efficient Causal Discovery (Q3). To show the advantage of the proposed constraints-based methods, we design a model named **Score** that optimizes a soft adjacent matrix using the score-based method (Brouillard et al., 2020), which has recently been combined with NN for differentiable training, for example, in TICSA. According to the precision of the discovery shown in the right part of Figure 4.10, we find that score-based discovery is inefficient. Based on the performance of the Score model summarized in Table 4.4, we also conclude that it is not as good as our constraint-based method and has a large variance due to unstable learning of the causal graph.

Influence of Causal Graph (Q4) To illustrate the importance of the causal graph, we implement another variant of GRADER named **Full**, which uses a fixed full graph that connects

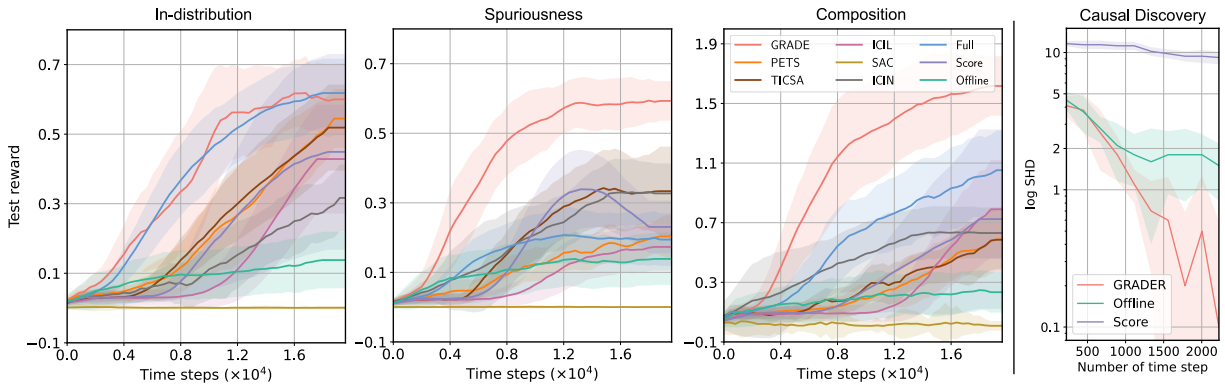


Figure 4.12: The testing reward and causal discovery results of *Unlock* environment.

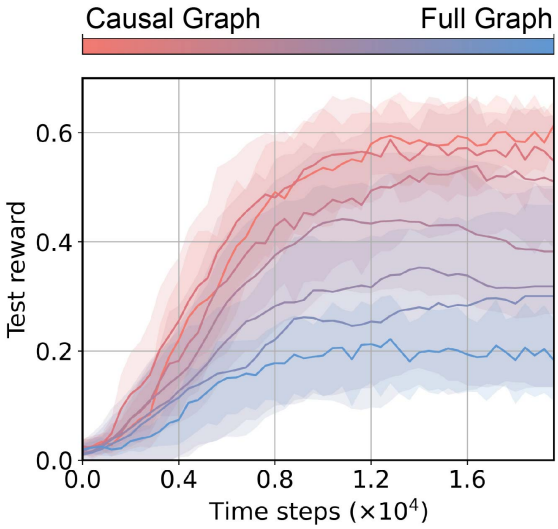


Figure 4.13: Influence of different causal graphs in *Unlock-S*.

all nodes between the sets $\mathcal{U}$ and $\mathcal{V}$. According to the performance shown in Table 4.4 and Figure 4.10, we find that the full graph achieves worse results than GRADER due to the redundant and spurious correlation. Intuitively, unrelated information causes additional noises in the learning procedure, and spurious correlation creates a shortcut that makes the model extract wrong features, leading to worse results in spuriousness generalization as shown in Table 4.4.

We then investigate how the correctness of the causal graph influences the performance. We used fixed graphs interpolating from the best causal graph to the full graph to train a GRADER model in *Unlock-S* and summarize the results in Figure 4.13. The more correct the graph, the higher reward the agent obtains, which supports our statements in Section 4.2.3 that the causal

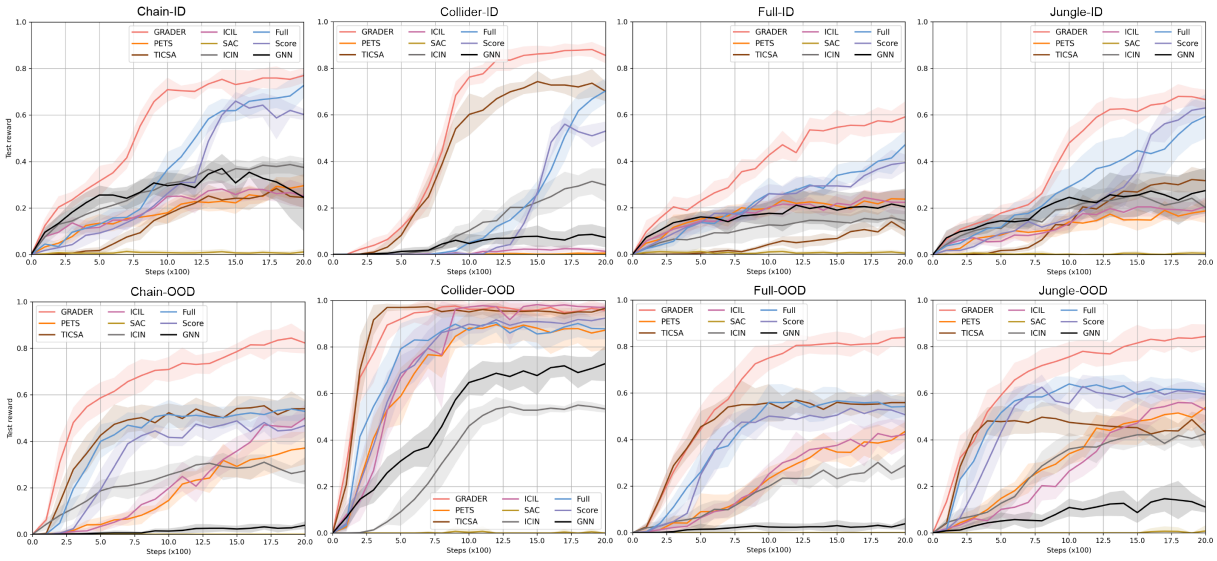


Figure 4.14: Reward of Chemistry environment under ID and OOD setting.

graph is important for the reasoning tasks – a better causal graph helps the model to have better task-solving performance.

Overall Performance on Other Environments

The overall performance results corresponding to Table 4.4 for *Stack* and *Unlock* environments are shown in Figure 4.11 and Figure 4.12.

In all *Stack* experiments, we find that the advantage of GRADER over other methods is small. The reason is that this task is simple and the true causal graph only contains 7 nodes as shown in Figure 4.2.4. Due to the simple causal graph, even the offline random policy can obtain the true causal graph, so there is almost no difference between the discovery efficiency between GRADER and Offline as shown in the right part of Figure 4.11.

In the *Unlock-I* experiment, there is no gap between GRADER and Full, which means that the causal graph may not have many contributions to solving this task. However, there are large gaps in *Unlock-S* and *Unlock-C* settings since indicating that the causal graph helps the model to obtain better generalizable performance. As for the Offline method, since the causal graph is wrongly discovered, the performance is bad in all three settings.

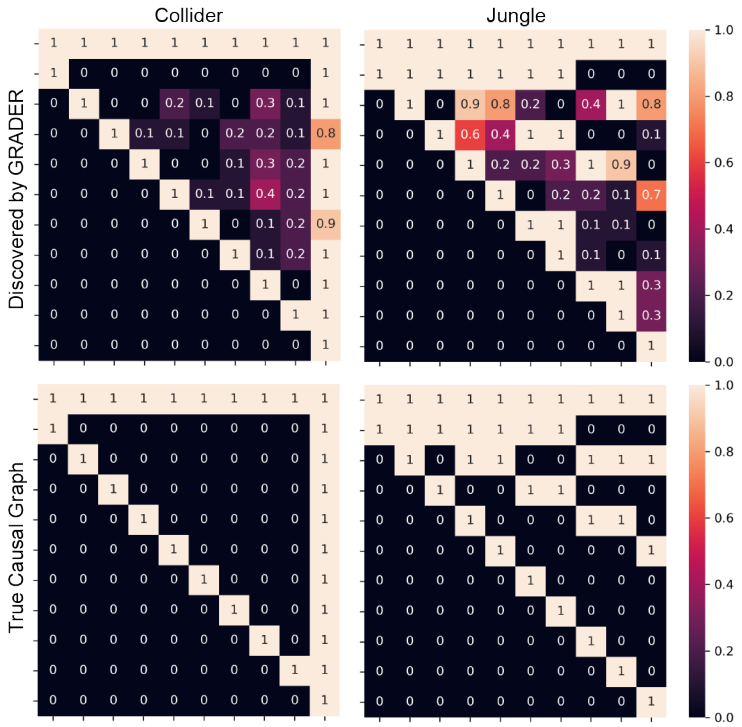


Figure 4.15: Top: Discovered causal graph from GRADER. Bottom: true causal graph.

Table 4.6: Success rate (%) for Chemistry environment (ID). **Bold** font means the best.

Env	SAC	ICIN	PETS	TICSA	ICIL	GNN	GRADER	Score	Full
Collider	0.0±0.0	29.8±7.2	0.6±0.8	70.1±4.2	1.3±1.3	7.3±4.3	**85.5±3.4**	53.0±4.1	70.3±5.3
Chain	1.1±1.3	37.5±4.0	29.6±4.9	24.5±4.3	25.3±5.1	24.6±14.5	**77.0±3.2**	60.2±2.4	72.7±5.3
Jungle	0.6±0.8	20.2±1.5	18.8±5.2	31.8±4.5	20.6±3.9	27.5±9.8	**69.6±4.3**	63.0±2.3	59.4±9.5
Full	0.5±0.8	4.5±4.0	23.7±4.3	10.4±3.0	22.3±5.1	20.4±7.8	**59.1±6.6**	39.4±3.5	47.1±6.1

Analysis of Discovered Causal Graph

In the *Chemistry* experiment, the downstream task is to change the color of the nodes to match the given colors within maximum steps ($T = 10$). A reward $r = 1$ is received if all colors are matched. The results are reported with 200 episodes. We use the planning horizon $H = 5$. We provide the results of the RL downstream tasks in Table 4.6 (ID setting) and Table 4.7 (OOD setting). The test reward is shown in Figure 4.14. The graphs in the ID setting have 10 nodes, while those in the OOD setting have 5 nodes. In the ID setting, we randomly sample the target colors in the goal. In the OOD setting, we set the target colors of all nodes to the same color during training to create spurious correlations, then randomly set the target colors during testing.

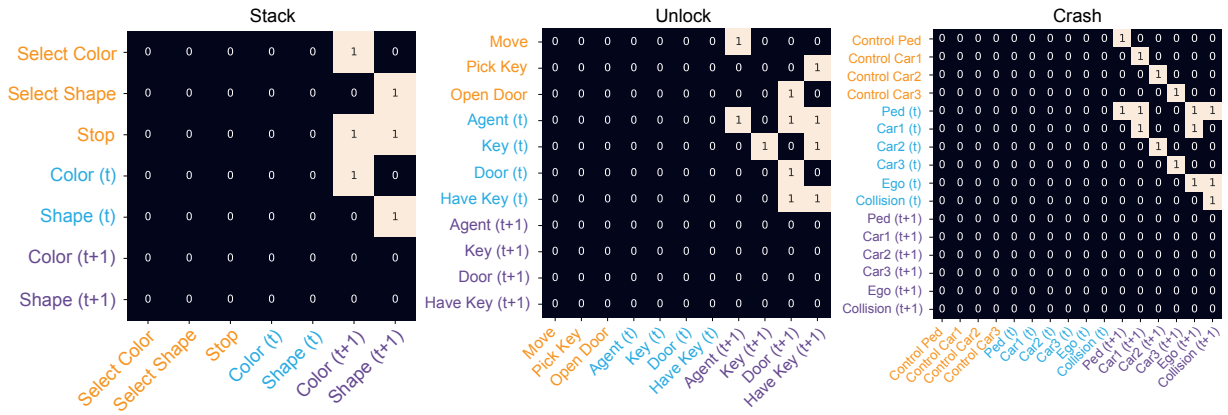


Figure 4.16: Discovered causal graphs of three environments. (Action, State, Next state)

Table 4.7: Success rate (%) for Chemistry environment (OOD). **Bold** font means the best.

Env	SAC	ICIN	PETS	TICSA	ICIL	GNN	GRADER	Score	Full
Collider	0.0±0.0	53.3±1.6	87.2±8.5	96.6±1.4	97.0±2.0	72.8±7.5	**95.8±2.6**	92.4±3.5	87.8±4.4
Chain	0.0±0.0	27.3±5.9	37.1±7.0	54.0±3.8	50.0±5.8	3.9±1.6	**82.3±4.5**	46.8±5.0	52.9±4.3
Jungle	0.8±2.4	42.6±4.9	53.9±5.5	43.1±7.5	52.9±7.0	11.1±2.4	**84.4±5.1**	59.5±2.7	60.8±3.5
Full	0.0±0.0	28.9±5.0	43.5±4.1	55.9±4.5	42.2±5.9	3.8±2.5	**83.9±4.4**	50.7±6.0	54.2±4.1

Since the environments we designed have clear and explicit causality, we can get the true causal graph with human analysis. We plot the true causal graphs corresponding to the three environments in Figure 4.16, where the semantic meanings of all nodes are explained in 4.2.4. We observe that the causal graphs are sparse with very few edges, indicating that non-causal methods that use the full graph may import redundant or even wrong information.

We also conduct a further analysis of the discovery performance on the *Chemistry* environment (Ke et al., 2021), which is a standard benchmark for evaluating causal discovery methods. In this environment, the colors of nodes are controlled by the causal graph; therefore, finding the true causal graph makes it much easier to achieve the goal that requires matching all target colors. The agent can discover the graph by doing interventions by interacting with the environment. We consider four types of causal graphs (*Collider*, *Chain*, *Jungle*, *Full*) with 10 nodes in this experiment.

The discovery performance is shown in Table 4.5 with five metrics that indicate the classification error. We can see that GRADER outperforms the Score method in all 4 types of graphs. In Figure 4.15, we show the graphs discovered in GRADER (averaged over 10 seeds) and the

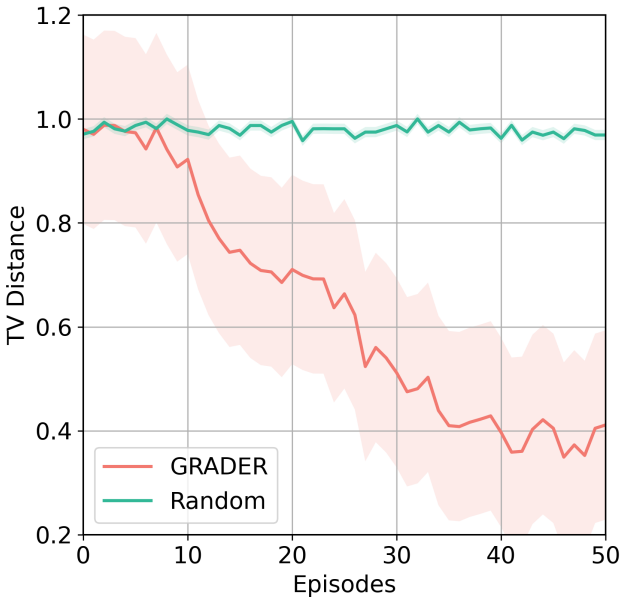


Figure 4.17: TV distance between goal and state distributions.

true causal graphs of *Collider* and *Jungle* settings.

Distance between Goal and State Distribution

In Figure 4.17, we empirically show the upper bound proved in (4.72), which describes the TV distance between the goal distribution and the state distribution collected from the GRADER policy. We used 10 trails and plotted the mean and standard derivation of the distance. We observe that the distance becomes smaller as the policy improves in GRADER. This supports our statement that the planning module helps to collect better data samples, which will be used in the causal discovery module. We also plot the distance with a random policy, which is always large, since the goal is not easy to achieve by random actions.

4.3 Break Spurious Correlation with Generative Models

Reinforcement learning (RL), aiming to learn a policy to maximize cumulative reward through interactions, has been successfully applied to a wide range of tasks such as language generation (OpenAI, 2023), game play (Silver et al., 2016), autonomous driving (Bojarski et al., 2016),

etc. Although standard RL has achieved remarkable success in simulated environments, a growing trend in RL is to address another critical concern – **robustness** – with the hope that the learned policy still performs well when the deployed (test) environment deviates from the nominal one used for training (Ding et al., 2022a). Robustness is highly desirable, since the performance of the learned policy could significantly deteriorate due to uncertainty and variations of the test environment induced by random perturbation, rare events, or even malicious attacks (Mahmood et al., 2018; Zhang et al., 2021).

Despite the various types of uncertainty that have been investigated in RL, this work focuses on the uncertainty of the environment with semantic meanings resulting from some unobserved underlying variables. This type of environment uncertainty, denoted as **structured uncertainty**, is motivated by numerous applications in the real world but still receives little attention in sequential decision-making tasks (De Haan et al., 2019). To specify the phenomenon of structured uncertainty, let us consider a concrete example (illustrated in Figure 4.18) in a driving scenario, where a shift between training and test environments caused by an unobserved confounder can potentially lead to a serious safety issue. Specifically, the observations *brightness* and *traffic density* do not have cause and effect on each other, but are controlled by a confounder (i.e. *sun* and *human activity*) that is usually unobserved [1] to the agent. During training, the agent could memorize the **spurious state correlation** between *brightness* and *traffic density*, i.e., the traffic is heavy during the day but light at night. However, such a correlation could be problematic during testing when the value of the confounder deviates from the training one, e.g. traffic becomes heavy at night due to special events (*human activity* changes), as shown at the bottom of Figure 4.18. Consequently, the policy dominated by the spurious correlation in training fails in out-of-distribution samples (observations of heavy traffic at night) in the test scenarios.

The failure of the driving example in Figure 4.18 is attributed to the widespread and harmful spurious correlation, namely, the learned policy is not robust to the structured uncertainty of the test environment caused by the unobserved confounder. However, ensuring robustness to structured uncertainty is challenging since the targeted uncertain region – the structured uncertainty set of the environment – is carved by the unknown causal effect of the unobserved

[1] sometimes they are observed but ignored given so many variables to be considered in neural networks.

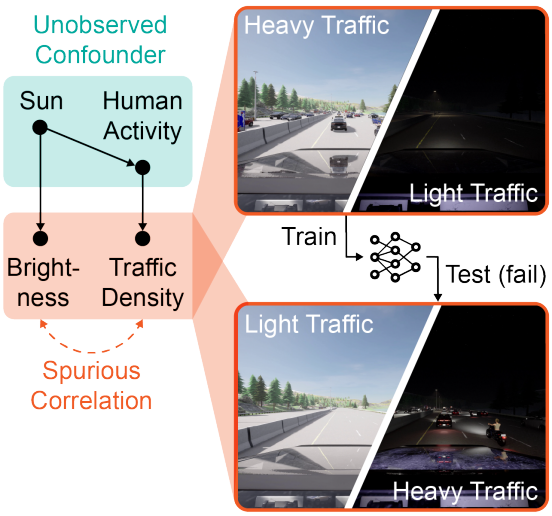


Figure 4.18: A model trained only with heavy traffic in the daytime learns the spurious correlation between brightness and traffic density and could fail to drive in light traffic in the daytime.

confounder and thus hard to characterize. In contrast, previous work on robustness in RL (Moos et al., 2022) usually considers a homogeneous and structure-agnostic uncertainty set around the state (Han et al., 2022; Zhang et al., 2021, 2020c), action (Tan et al., 2020; Tessler et al., 2019), or the training environment (Iyengar, 2005; Shi and Chi, 2022; Yang et al., 2022) measured by some heuristic functions (Moos et al., 2022; Shi and Chi, 2022; Zhang et al., 2020c) to account for unstructured random noise or small perturbations. Consequently, these prior works could not cope with the structured uncertainty since their uncertainty set is different from and cannot tightly cover the desired structured uncertainty set, which could be heterogeneous and allow for potentially large deviations between the training and test environments.

In this work, to address the structured uncertainty, we first propose a general RL formulation called State-confounded Markov decision processes (SC-MDPs), which model the possible causal effect of the unobserved confounder in an RL task from a causal perspective. SC-MDPs better explain the reason for semantic shifts in the state space than traditional MDPs. Then, we formulate the problem of seeking robustness to structured uncertainty as solving Robust SC-MDPs (RSC-MDPs), which optimizes the worst performance when the distribution of the unobserved confounder lies in some uncertainty set. The key contributions of this work are summarized below.

- We propose a new type of robustness concerning structured uncertainty to address spurious correlation in RL and provide a formal mathematical formulation called RSC-MDPs, which are well-motivated by ubiquitous real-world applications.
- We theoretically justify the advantage of the proposed RSC-MDP framework against structured uncertainty over the prior formulation in robust RL without semantic information.
- We implement an empirical algorithm to find the optimal policy of RSC-MDPs and show that it outperforms the baselines on eight real-world tasks in manipulation and self-driving.

4.3.1 Preliminary and Limitations of Robust RL

In the following, we first introduce the preliminary formulation of standard RL and then discuss a natural type of robustness that is widely considered in the RL literature and is most related to this work – robust RL.

Standard Markov decision processes (MDPs). An episodic finite-horizon standard MDP is represented by $\mathcal{M} = \big\{\mathcal{S}, \mathcal{A}, T, r, P\big\}$, where $\mathcal{S} \subseteq \mathbb{R}^n$ and $\mathcal{A} \subseteq \mathbb{R}^{d_A}$ are the state and action spaces, respectively, with n/d_A being the dimension of state/action. Here, T is the length of the horizon; $P = \{P_t\}_{1\le t\le T}$, where $P_t : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ denotes the probability transition kernel at the time step t, for all $1 \le t \le T$; and $r = \{r_t\}_{1\le t\le T}$ denotes the reward function, where $r_t : \mathcal{S} \times \mathcal{A} \to [0, 1]$ represents the deterministic immediate reward function. A policy (action selection rule) is indicated by $\pi = \{\pi_t\}_{1\le t\le T}$, that is, the policy at the time step t is $\pi_t : \mathcal{S} \to \Delta(\mathcal{A})$ based on the current state s_t as $\pi_t(\cdot \,|\, s_t)$. To represent the long-term cumulative reward, the value function $V_t^{\pi,P} : \mathcal{S} \to \mathbb{R}$ and Q-value function $Q_t^{\pi,P} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ associated with policy π at step t are defined as $V_t^{\pi,P}(s) = \mathbb{E}_{\pi,P}\big[\sum_{k=t}^{T} r_k(s_k, a_k) \mid s_k = s\big]$ and $Q_t^{\pi,P}(s, a) = \mathbb{E}_{\pi,P}\big[\sum_{k=t}^{T} r_k(s_k, a_k) \mid s_t = s, a_t = a\big]$, where the expectation is taken over the sample trajectory $\{(s_t, a_t)\}_{1\le t\le T}$ generated following $a_t \sim \pi_t(\cdot \,|\, s_t)$ and $s_{t+1} \sim P_t(\cdot \,|\, s_t, a_t)$.

Robust Markov decision processes (RMDPs). As a robust variant of standard MDPs motivated by distributionally robust optimization, RMDP is a natural formulation to promote robustness to the uncertainty of the transition probability kernel (Iyengar, 2005; Shi and Chi, 2022), represented as $\mathcal{M}_{\mathsf{rob}} = \big\{\mathcal{S}, \mathcal{A}, T, r, \mathcal{U}^{\sigma}(P^0)\big\}$. Here, we reuse the definitions of $\mathcal{S}, \mathcal{A}, T, r$ in standard MDPs and denote $\mathcal{U}^{\sigma}(P^0)$ as an uncertainty set of probability transition kernels centered around

a nominal transition kernel $P^0 = \{P_t^0\}_{1\le t\le T}$ measured by some 'distance' function ρ with radius σ. In particular, the uncertainty set obeying the (s,a)-rectangularity (Wiesemann et al., 2013) can be defined over all (s,a) state-action pairs at each time step t as

$$\mathcal{U}^\sigma(P^0) \coloneqq \otimes\, \mathcal{U}^\sigma(P^0_{t,s,a}), \qquad \mathcal{U}^\sigma(P^0_{t,s,a}) \coloneqq \left\{P_{t,s,a} \in \Delta(\mathcal{S}) : \rho\left(P_{t,s,a}, P^0_{t,s,a}\right) \le \sigma\right\}, \quad (4.82)$$

where $\otimes$ denotes the Cartesian product. Here, $P_{t,s,a} \coloneqq P_t(\cdot \,|\, s,a) \in \Delta(\mathcal{S})$ and $P^0_{t,s,a} \coloneqq P^0_t(\cdot \,|\, s,a) \in \Delta(\mathcal{S})$ denote the transition kernel P_t or P_t^0 at each state-action pair (s,a), respectively. Consequently, the next state s_{t+1} follows $s_{t+1} \sim P_t(\cdot \,|\, s_t, a_t)$ for any $P_t \in \mathcal{U}^\sigma(P^0_{t,s_t,a_t})$, namely, s_{t+1} can be generated from any transition kernel belonging to the uncertainty set $\mathcal{U}^\sigma(P^0_{t,s_t,a_t})$ rather than a fixed one in standard MDPs. As a result, for any policy π, the corresponding *robust value function and* robust Q function are defined as

$$V_t^{\pi,\sigma}(s) \coloneqq \inf_{P\in\mathcal{U}^\sigma(P^0)} V_t^{\pi,P}(s), \qquad Q_t^{\pi,\sigma}(s,a) \coloneqq \inf_{P\in\mathcal{U}^\sigma(P^0)} Q_t^{\pi,P}(s,a), \quad (4.83)$$

which characterize the cumulative reward in the worst case when the transition kernel is within the uncertainty set $\mathcal{U}^\sigma(P^0)$. Using samples generated from the nominal transition kernel P^0, the goal of RMDPs is to find an optimal robust policy that maximizes $V_1^{\pi,\sigma}$ when $t = 1$, i.e., performs optimally in the worst case when the transition kernel of the test environment lies in a prescribed uncertainty set $\mathcal{U}^\sigma(P^0)$.

Lack of semantic information in RMDPs. In spite of the rich literature on robustness in RL, prior works usually hedge against the uncertainty induced by unstructured random noise or small perturbations, specified as a small and homogeneous uncertainty set around the nominal one. For example, in RMDPs, people usually prescribe the uncertainty set of the transition kernel using a simple heuristic function ρ with a relatively small σ. However, the unknown uncertainty in the real world could have a complicated and semantic structure that cannot be well covered by a homogeneous ball regardless of the choice of the uncertainty radius σ, leading to over-conservative policy (when σ is large) or insufficient robustness (when σ is small). Together, we obtain the natural motivation for this work: *How to formulate such a structured uncertainty and ensure robustness against it?*

4.3.2 Robust RL against Structured Uncertainty from Causal Perspective

To describe structured uncertainty, we choose to study MDPs from a causal perspective with a basic concept called a structural causal model (SCM). Armed with the concept, we formulate State-confounded MDPs – a broader set of MDPs in the face of the unobserved confounder in the state space. Next, we provide the main formulation considered in this work, robust state-confounded MDPs, which promote robustness to structured uncertainty.

Structural causal model. We denote a structural causal model (SCM) (Pearl, 2009) by a tuple $\{X, Y, F, P^x\}$, where X is the set of exogenous (unobserved) variables, Y is the set of endogenous (observed) variables, and P^x is the distribution of all the exogenous variables. Here, F is the set of structural functions that capture the causal relations between X and Y such that for each variable $y_i \in Y$, $f_i \in F$ is defined as $y_i \leftarrow f_i\big(\mathsf{PA}(y_i), x_i\big)$, where $x_i \subseteq X$ and $\mathsf{PA}(y_i) \subseteq Y \setminus y_i$ denote the parents of node y_i. We say that a pair of variables y_i and y_j are confounded by a variable C (confounder) if they are both caused by C, that is, $C \in \mathsf{PA}(y_i)$ and $C \in \mathsf{PA}(y_j)$. When two variables y_i and y_j do not have direct causality, they are still correlated if they are confounded, in which case this correlation is called a spurious correlation.

State-confounded MDPs (SC-MDPs)

We now present state-confounded MDPs (SC-MDPs), whose probabilistic graph is illustrated in Figure 4.19(a) with a comparison to standard MDPs in Figure 4.19(b). In addition to the components of standard MDPs $\mathcal{M} = \big\{\mathcal{S}, \mathcal{A}, T, r\big\}$, we introduce a set of unobserved confounders $C_{\mathsf{s}} = \{c_t\}_{1 \le t \le T}$, where $c_t \in \mathcal{C}$ denotes the confounder generated from some unknown but fixed distribution P_t^c at the time step t, that is, $c_t \sim P_t^c \in \Delta(\mathcal{C})$.

To characterize the causal effect of the confounder C_{s} on the state dynamic, we resort to an SCM, where C_{s} is the set of exogenous (unobserved) confounders, and endogenous variables include all dimensions of states $\{s_t^i\}_{1 \le i \le n, 1 \le t \le T}$, and actions $\{a_t\}_{1 \le t \le T}$. Specifically, the structural function F is considered as $\{\mathcal{P}_t^i\}_{1 \le i \le n, 1 \le t \le T}$ – the transition from the current state s_t, action a_t and the confounder c_t to each dimension of the next state s_{t+1}^i for all time steps, i.e. $s_{t+1}^i \sim \mathcal{P}_t^i(\cdot \mid s_t, a_t, c_t)$. In particular, the specified SCM does not confound the reward, that is, $r_t(s_t, a_t)$ does not depend on the confounder c_t.

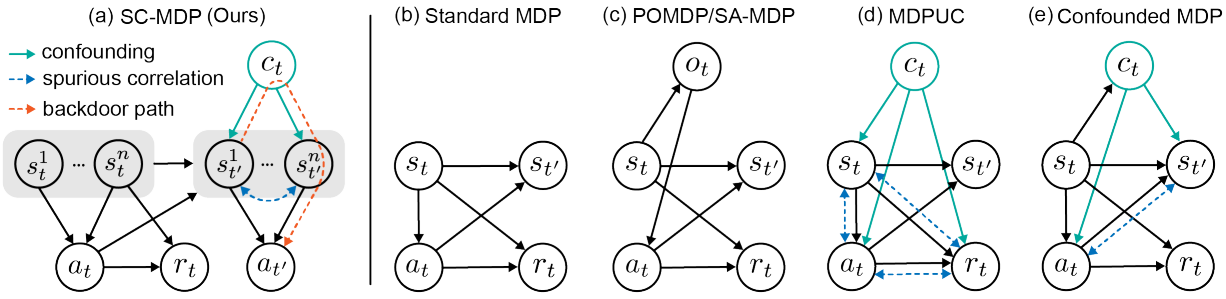


Figure 4.19: The probabilistic graphs of our formulation (SC-MDP) and related formulations. s_t^1 means the first dimension of s_t. $s_{t'}$ is a shorthand for s_{t+1}. In SC-MDP, the orange line represents the backdoor path from state $s_{t'}^1$ to action $a_{t'}$ opened by the confounder c_t, which makes the learned policy π rely on the value of c_t.

Armed with the above SCM, denoting $P^c := \{P_t^c\}$, we can introduce state-confounded MDPs (SC-MDPs) represented by $\mathcal{M}_{\text{sc}} = \{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, P^c\}$ (Figure 4.19(a)). A policy is indicated as $\pi = \{\pi_t\}$, where each π_t results in an intervention (possibly stochastic) that sets $a_t \sim \pi_t(\cdot \mid s_t)$ at the time step t regardless of the value of the confounder.

State-confounded value function and optimal policy. Given s_t, the causal effect of a_t on the next state s_{t+1} plays an important role in characterizing value function/Q-function. To ensure the identifiability of the causal effect, the confounder c_t are assumed to obey the backdoor criterion (Pearl, 2009; Peters et al., 2017), leading to the following *state-confounded value function* (SC-value function) and *state-confounded Q-function* (SC-Q function) (Wang et al., 2021b):

$$\widetilde{V}_t^{\pi,P^c}(s) = \mathbb{E}_{\pi,P^c}\left[\sum_{k=t}^{T} r_k(s_k,a_k) \mid s_t = s; c_k \sim P_k^c, s_{k+1}^i \sim \mathcal{P}_k^i(\cdot \mid s_k, a_k, c_k)\right],$$
$$\widetilde{Q}_t^{\pi,P^c}(s,a) = \mathbb{E}_{\pi,P^c}\left[\sum_{k=t}^{T} r_k(s_k,a_k) \mid s_t = s, a_t = a; c_k \sim P_k^c, s_{k+1}^i \sim \mathcal{P}_k^i(\cdot \mid s_k, a_k, c_k)\right]. \tag{4.84}$$

Remark 1. Note that the proposed SC-MDPs serve as a general formulation for a broad family of RL problems that include standard MDPs as a special case. Specifically, any standard MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, T, r\}$ can be equivalently represented by at least one SC-MDP $\mathcal{M}_{\text{sc}} = \{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, P^c\}$ as long as $\mathbb{E}_{c_t \sim P_t^c}\left[\mathcal{P}_t^i(\cdot \mid s_t, a_t, c_t)\right] = \left[P(\cdot \mid s_t, a_t)\right]_i$ for all $1 \le i \le n, 1 \le t \le T$.

Robust state-confounded MDPs (RSC-MDPs)

In this work, we consider robust state-confounded MDPs (RSC-MDPs) – a variant of SC-MDPs promoting robustness to the uncertainty of the unobserved confounder distribution P^c, denoted by $\mathcal{M}_{\text{sc-rob}} = \left\{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, \mathcal{U}^\sigma(P^c)\right\}$. Here, the perturbed distribution of the unobserved confounder is assumed in an uncertainty set $\mathcal{U}^\sigma(P^c)$ centered around the nominal distribution P^c with radius σ measured by some 'distance' function $\rho : \Delta(\mathcal{C}) \times \Delta(\mathcal{C}) \to \mathbb{R}^+$, i.e.,

$$\mathcal{U}^\sigma(P^c) \coloneqq \otimes\, \mathcal{U}^\sigma(P_t^c), \qquad \mathcal{U}^\sigma(P_t^c) \coloneqq \left\{P \in \Delta(\mathcal{C}) : \rho\left(P, P_t^c\right) \leq \sigma\right\}. \tag{4.85}$$

Consequently, the corresponding *robust SC-value function* and *robust SC-Q function* are defined as

$$\widetilde{V}_t^{\pi,\sigma}(s) \coloneqq \inf_{P \in \mathcal{U}^\sigma(P^c)} \widetilde{V}_t^{\pi,P}(s), \quad \widetilde{Q}_t^{\pi,\sigma}(s,a) \coloneqq \inf_{P \in \mathcal{U}^\sigma(P^c)} \widetilde{Q}_t^{\pi,P}(s,a), \tag{4.86}$$

representing the worst-case cumulative rewards when the confounder distribution lies in the uncertainty set $\mathcal{U}^\sigma(P^c)$.

Then a natural question is: Does there exist an optimal policy that maximizes the robust SC-value function $\widetilde{V}_t^{\pi,\sigma}$ for any RSC-MDP so that we can target learning? To answer this, we introduce the following theorem, which ensures the existence of an optimal policy for all RSC-MDPs. The proof can be found in 4.3.3.

Theorem 3 (Existence of an optimal policy)**.** *Let Π be the set of all non-stationary and stochastic policies. Consider any RSC-MDP, there exists at least one optimal policy $\pi^{\mathsf{sc},\star} = \{\pi_t^{\mathsf{sc},\star}\}_{1 \leq t \leq T}$ such that for all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and $1 \leq t \leq T$, one has*

$$\widetilde{V}_t^{\pi^{\mathsf{sc},\star},\sigma}(s) = \widetilde{V}_t^{\star,\sigma}(s) \coloneqq \sup_{\pi \in \Pi} \widetilde{V}_t^{\pi,\sigma}(s) \quad \textit{and} \quad \widetilde{Q}_t^{\pi^{\mathsf{sc},\star},\sigma}(s,a) = \widetilde{Q}_t^{\star,\sigma}(s,a) \coloneqq \sup_{\pi \in \Pi} \widetilde{Q}_t^{\pi,\sigma}(s,a).$$

In addition, RSC-MDPs also possess benign properties similar to RMDPs such that for any policy π and the robust optimal policy $\pi^{\mathsf{sc},\star}$, the corresponding *robust SC Bellman consistency equation* and *robust SC Bellman optimality equation* are also satisfied, as specified in 4.3.3.

Goal. Based on all the definitions and analysis above, this work aims to find an optimal policy

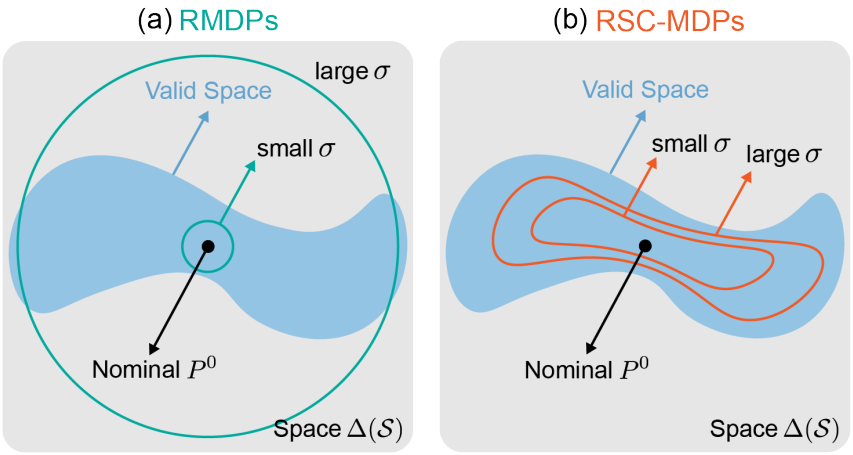


Figure 4.20: (a) RMDPs add homogeneous noise to states, while (b) RSC-MDPs perturb the confounder to influence states, resulting in a subset of the valid space.

for RSC-MDPs that maximizes the robust SC-value function in (4.86), yielding optimal performance in the worst case where the unobserved confounder distribution falls into an uncertainty set $\mathcal{U}^\sigma(P^c)$.

Advantages of RSC-MDPs over traditional RMDPs

The most relevant robust RL formulation for ours is RMDPs, which has been introduced in Section 4.3.1. Here, we provide a thorough comparison between RMDPs and our RSC-MDPs with theoretical justifications, and leave the comparisons and connections to other related formulations in Figure 4.19.

To begin, at each time step t, RMDPs explicitly introduce uncertainty into the transition probability kernels, while our RSC-MDPs add uncertainty to the transition kernels in a latent (and hence more structured) manner by perturbing the unobserved confounder that partly determines the transition kernels. As an example, imagine the true uncertainty set encountered in the real world illustrated by the blue region in Figure 4.20, which could have a complicated structure. Since the uncertainty set in RMDPs is homogeneous (illustrated by the green circles), one often faces the dilemma of being either too conservative (when σ is large) or too reckless (when σ is small). In contrast, the proposed RSC-MDPs – shown in Figure 4.20(b) – take advantage of the structured uncertainty set (illustrated by the orange region) enabled by the underlying SCM,

which can potentially lead to much better estimation of the true uncertainty set. Specifically, the varying unobserved confounder induces diverse perturbation to different portions of the state through the structural causal function, enabling heterogeneous and structural uncertainty sets over the state space.

Theoretical guarantees of RSC-MDPs: advantages of structured uncertainty. To theoretically understand the advantages of the proposed robust formulation of RSC-MDPs compared to previous work, especially RMDPs, the following theorem verifies that RSC-MDPs enable additional robustness against semantic attack in addition to small model perturbation or noise considered in RMDPs. The proof is postponed to Appendix 4.3.3.

Theorem 4. *Consider any $T \geq 2$. Consider some standard MDPs $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P^0, T, r\}$, equivalently represented as an SC-MDP $\mathcal{M}_{\mathsf{sc}} = \{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, P^c\}$ with $\mathcal{C} \coloneqq \{0, 1\}$, and total variation as the 'distance' function ρ to measure the uncertainty set (the admissible uncertainty level obeys $\sigma \in [0, 1]$). For the corresponding RMDP $\mathcal{M}_{\mathsf{rob}}$ with uncertainty set $\mathcal{U}^{\sigma_1}(P^0)$, and the proposed RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}} = \{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, \mathcal{U}^{\sigma_2}(P^c)\}$, the optimal robust policy $\pi^{\star,\sigma_1}_{\mathsf{RMDP}}$ associated with $\mathcal{M}_{\mathsf{rob}}$ and $\pi^{\star,\sigma_2}_{\mathsf{RSC}}$ associated with $\mathcal{M}_{\mathsf{sc\text{-}rob}}$ obey: given $\sigma_2 \in \left(\frac{1}{2}, 1\right]$, there exist RSC-MDPs with some initial state distribution ϕ such that*

$$\widetilde{V}_1^{\pi^{\star,\sigma_2}_{\mathsf{RSC}},\sigma_2}(\phi) - \widetilde{V}_1^{\pi^{\star,\sigma_1}_{\mathsf{RMDP}},\sigma_2}(\phi) \geq \frac{T}{8}, \quad \forall \sigma_1 \in [0, 1]. \tag{4.87}$$

In words, Theorem 4 reveals a fact about the proposed RSC-MDPs: *RSC-MDPs could succeed in intense semantic attacks while RMDPs fail*. As shown by (4.87), when fierce semantic shifts appear between training and test scenarios – perturbing the unobserved confounder in a large uncertainty set $\mathcal{U}^{\sigma_2}(P^c)$, solving RSC-MDPs with $\pi^{\star,\sigma_2}_{\mathsf{RSC}}$ succeeds in testing while $\pi^{\star,\sigma_1}_{\mathsf{RMDP}}$ trained by solving RMDPs can fail catastrophically. The proof is achieved by constructing hard instances of RSC-MDPs that RMDPs could not cope with due to inherent limitations. Furthermore, this advantage of RSC-MDPs is consistent with and verified by the empirical performance evaluation in Section 4.3.5 **R1**.

4.3.3 Theoretical Analyses RSC-MDPs

Proof of Theorem 3

In this section, we verify the existence of an optimal policy of the proposed RSC-MDPs, involving additional components — confounder C_{s} and the infimum optimization problems with comparisons with standard MDPs (Agarwal et al., 2019).

To begin with, we recall that the goal is to find a policy $\widetilde{\pi} = \{\widetilde{\pi}_t\}_{1 \leq t \leq T} \in \Pi$ such that for all $(s, a) \in \mathcal{S} \times \mathcal{A}$:

$$\widetilde{V}_t^{\widetilde{\pi},\sigma}(s) = \widetilde{V}_t^{\star,\sigma}(s) \coloneqq \sup_{\pi \in \Pi} \widetilde{V}_t^{\pi,\sigma}(s), \tag{4.88}$$

$$\widetilde{Q}_t^{\widetilde{\pi},\sigma}(s,a) = \widetilde{Q}_t^{\star,\sigma}(s,a) \coloneqq \sup_{\pi \in \Pi} \widetilde{Q}_t^{\pi,\sigma}(s,a), \tag{4.89}$$

which we called an optimal policy. To do this, we start from the first claim in equation 4.89.

Step 1: Introducing additional notation. Before proceeding, we let $\{S_t, A_t, R_t, C_t\}$ denote the random variables — state, action, reward, and confounder, at time step t for all $1 \leq t \leq T$. Then invoking the Markov properties, we know that, conditioned on the current state s_t, the future state, action, and reward are all independent of the previous $s_1, a_1, r_1, c_1, \cdots, s_{t-1}, a_{t-1}, r_{t-1}, c_{t-1}$. In addition, we represent $P_t \in \Delta(\mathcal{C})$ as some distribution of confounder at time step t, for all $1 \leq t \leq T$. For convenience, we introduce the following notation that is defined in the time step $t \leq k \leq T$:

$$\forall 1 \leq t \leq T: \quad P_{+t} \coloneqq \otimes_{t \leq k \leq T} P_k, \tag{4.90}$$

$$\mathcal{U}^{\sigma}(P_{+t}^c) \coloneqq \otimes_{t \leq k \leq T} \mathcal{U}^{\sigma}(P_k^c), \tag{4.91}$$

which represent some collections of variables from time step t to the end of the episode. Furthermore, recall that the transition kernel from time step t to $t+1$ is denoted as $s_{t+1}^i \sim \mathcal{P}_t^i(\cdot \mid s_t, a_t, c_t)$ for $i \in 1, 2 \cdots, n$. With slight abuse of notation, we denote $s_{t+1} \sim \mathcal{P}_t(\cdot \mid s_t, a_t, c_t)$ and abbreviate $\mathbb{E}_{s_{t+1} \sim \mathcal{P}_t(\cdot \mid s_t, a_t, c_t)}[\cdot]$ as $\mathbb{E}_{s_{t+1}}[\cdot]$ whenever it is clear.

Step 2: Establishing recursive relationship. Recall that the nominal distribution of the con-

founder is $c_t \in P_t^c$ at the time step t. We choose $\widetilde{\pi} = \{\widetilde{\pi}_t\}$ which obeys: for all $1 \leq t \leq T$,

$$\widetilde{\pi}_t(s) \coloneqq \underset{\pi_t \in \Delta(\mathcal{A})}{\arg\max} \left\{ \mathbb{E}_{\pi_t}[r_t(s, a_t)] + \inf_{P_t \in \mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t} \left[\mathbb{E}_{c_t \sim P_t} \left[\mathbb{E}_{s_{t+1}} \left[\widetilde{V}_{t+1}^{\star,\sigma}(s_{t+1}) \right] \right] \right] \right\}. \tag{4.92}$$

Armed with these definitions and notations, for any $(t, s) \in \{1, 2, \cdots, T\} \times \mathcal{S}$, one has

$$\begin{aligned}
&\widetilde{V}_t^{\star,\sigma}(s) \\
&\overset{\text{(i)}}{=} \sup_{\pi \in \Pi} \inf_{P \in \mathcal{U}^\sigma(P^c)} \widetilde{V}_t^{\pi,P}(s) \overset{\text{(ii)}}{=} \sup_{\pi \in \Pi} \inf_{P_{+t} \in \mathcal{U}^\sigma\left(P_{+t}^c\right)} \mathbb{E}_{\pi, P_{+t}} \left[\sum_{k=t}^{T} r_k(s_k, a_k) \right] \\
&\overset{\text{(iii)}}{=} \sup_{\pi \in \Pi} \inf_{P_{+t} \in \mathcal{U}^\sigma\left(P_{+t}^c\right)} \mathbb{E}_{\pi_t} \Bigg[r_t(s, a_t) \\
&\quad + \mathbb{E}_{c_t \sim P_t} \left[\mathbb{E}_{s_{t+1}} \left[\sum_{k=t+1}^{T} r_k(s_k, a_k) \,|\, \pi, P_{+(t+1)}, (S_t, A_t, R_t, C_t, S_{t+1}) = (s, a_t, r_t, c_t, s_{t+1}) \right] \right] \Bigg] \\
&= \sup_{\pi \in \Pi} \mathbb{E}_{\pi_t}[r_t(s, a_t)] + \inf_{P_{+t} \in \mathcal{U}^\sigma\left(P_{+t}^c\right)} \mathbb{E}_{\pi_t} \Bigg[\mathbb{E}_{c_t \sim P_t} \Bigg[\\
&\quad \mathbb{E}_{s_{t+1}} \left[\sum_{k=t+1}^{T} r_k(s_k, a_k) \,|\, \pi, P_{+(t+1)}, (S_t, A_t, R_t, C_t, S_{t+1}) = (s, a_t, r_t, c_t, s_{t+1}) \right] \Bigg] \Bigg],
\end{aligned}$$

where (i) holds by the definitions in equation 4.86, (ii) is due to equation 4.84 and that $\widetilde{V}_t^{\pi,P}(s)$ only depends on P_{+t} by the Markov property, (iii) follows from expressing the term of interest by moving one step ahead and $\mathbb{E}_{\pi_t}$ is taken with respect to $a_t \sim \pi_t(\cdot \,|\, S_t = s)$.

To continue, we observe that the $\widetilde{V}_t^{\star,\sigma}(s)$ can be further controlled as follows:

$$
\begin{aligned}
&\widetilde{V}_t^{\star,\sigma}(s) \\
&= \sup_{\pi\in\Pi} \mathbb{E}_{\pi_t}[r_t(s,a_t)] + \inf_{P_{+t}\in\mathcal{U}^\sigma\left(P^c_{+t}\right)} \mathbb{E}_{\pi_t}\Bigg[\mathbb{E}_{c_t\sim P_t}\Bigg[\\
&\quad \mathbb{E}_{s_{t+1}}\left[\sum_{k=t+1}^{T} r_k(s_k,a_k) \,|\, \pi, P_{+(t+1)}, (S_t,A_t,R_t,C_t,S_{t+1}) = (s,a_t,r_t,c_t,s_{t+1})\right]\Bigg]\Bigg] \\
&\overset{\text{(i)}}{=} \sup_{\pi\in\Pi} \mathbb{E}_{\pi_t}[r_t(s,a_t)] + \inf_{P_t\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t}\Bigg[\mathbb{E}_{c_t\sim P_t}\Bigg[\mathbb{E}_{s_{t+1}}\Bigg[\\
&\quad \inf_{P_{+(t+1)}\in\mathcal{U}^\sigma\left(P^c_{+(t+1)}\right)} \sum_{k=t+1}^{T} r_k(s_k,a_k) \,|\, \pi, P_{+(t+1)}, (S_t,A_t,R_t,C_t,S_{t+1}) = (s,a_t,r_t,c_t,s_{t+1})\Bigg]\Bigg]\Bigg] \\
&\leq \sup_{\pi\in\Pi} \mathbb{E}_{\pi_t}[r_t(s,a_t)] + \inf_{P_t\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t}\Bigg[\mathbb{E}_{c_t\sim P_t}\mathbb{E}_{s_{t+1}}\Bigg[\sup_{\pi'\in\Pi} \inf_{P_{+(t+1)}\in\mathcal{U}^\sigma(P^c_{+(t+1)})} \\
&\quad \sum_{k=t+1}^{T} r_k(s_k,a_k) \,|\, \pi', P_{+(t+1)}, (S_t,A_t,R_t,C_t,S_{t+1}) = (s,a_t,r_t,c_t,s_{t+1})\Bigg]\Bigg] \\
&\overset{\text{(ii)}}{=} \sup_{\pi\in\Pi} \mathbb{E}_{\pi_t}[r_t(s,a_t)] \\
&\quad + \inf_{P_t\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t}\Bigg[\mathbb{E}_{c_t\sim P_t}\Bigg[\mathbb{E}_{s_{t+1}}\Bigg[\sup_{\pi'\in\Pi} \inf_{P_{+(t+1)}\in\mathcal{U}^\sigma(P^c_{+(t+1)})} \mathbb{E}_{\pi', P_{+(t+1)}}\left[\sum_{k=t+1}^{T} r_k(s_k,a_k)\right]\Bigg]\Bigg]\Bigg] \\
&= \sup_{\pi\in\Pi}\left\{\mathbb{E}_{\pi_t}[r_t(s,a_t)] + \inf_{P_t\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t}\left[\mathbb{E}_{c_t\sim P_t}\left[\mathbb{E}_{s_{t+1}}\left[\widetilde{V}_{t+1}^{\star,\sigma}(s_{t+1})\right]\right]\right]\right\} \\
&= \sup_{\pi_t\in\Delta(\mathcal{A})}\left\{\mathbb{E}_{\pi_t}[r_t(s,a_t)] + \inf_{P_t\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t}\left[\mathbb{E}_{c_t\sim P_t}\left[\mathbb{E}_{s_{t+1}}\left[\widetilde{V}_{t+1}^{\star,\sigma}(s_{t+1})\right]\right]\right]\right\} \\
&= \inf_{P_t\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}\left[r_t(s,a_t) + \mathbb{E}_{c_t\sim P_t}\mathbb{E}_{s_{t+1}}\left[\left[\widetilde{V}_{t+1}^{\star,\sigma}(s_{t+1})\right] \,|\, a_t = \widetilde{\pi}_t(s)\right]\right], \qquad (4.93)
\end{aligned}
$$

where (i) holds by the operator $\inf_{P_{+(t+1)}\in\mathcal{U}^\sigma\left(P^c_{+(t+1)}\right)}$ is independent from π_t conditioned on a fixed distribution of s_{t+1}, (ii) arises from the Markov property such that the rewards $\{r_k(s_k,a_k)\}_{t+1\leq k\leq T}$ conditioned on $(S_t,A_t,R_t,C_t,S_{t+1})$ or S_{t+1} are the same, and the last equality follows from the definition of $\widetilde{\pi}$ in equation 4.92.

Step 3: Completing the proof by applying recursion.

Applying equation 4.93 recursively for $t+1, \cdots T$, we arrive at

$$\begin{aligned}
\widetilde{V}_t^{\star,\sigma}(s) &\leq \inf_{P_t \in \mathcal{U}^\sigma(P_t^c)} \mathbb{E}\left[r_t(s,a_t) + \mathbb{E}_{c_t \sim P_t}\mathbb{E}_{s_{t+1}}\left[\left[\widetilde{V}_{t+1}^{\star,\sigma}(s_{t+1})\right] \mid a_t = \widetilde{\pi}_t(s)\right]\right] \\
&\leq \inf_{P_t \in \mathcal{U}^\sigma(P_t^c)} \inf_{P_{t+1} \in \mathcal{U}^\sigma(P_{t+1}^c)} \mathbb{E}\Bigg[r_t(s,a_t) + \mathbb{E}_{c_t \sim P_t}\Big[\mathbb{E}_{s_{t+1}}\Big[\\
&\quad r_{t+1}(s_{t+1},a_{t+1}) + \mathbb{E}_{c_{t+1}\sim P_{t+1}}\left[\mathbb{E}_{s_{t+2}}\left[\widetilde{V}_{t+2}^{\star,\sigma}(s_{t+2})\right]\right]\Big]\Big] \Big| (a_t, a_{t+1}) = (\widetilde{\pi}_t(s), \widetilde{\pi}_{t+1}(s_{t+1}))\Bigg] \\
&\leq \cdots \leq \inf_{P_{+t} \in \mathcal{U}^\sigma(P_{+t}^c)} \mathbb{E}_{\pi, P_{+t}}\left[\sum_{k=t}^{T} r_k(s_k,a_k)\right] = \widetilde{V}_t^{\widetilde{\pi},\sigma}(s). \qquad (4.94)
\end{aligned}$$

Observing from equation 4.94 that

$$\forall s \in \mathcal{S}: \quad \widetilde{V}_t^{\star,\sigma}(s) \leq \widetilde{V}_t^{\widetilde{\pi},\sigma}(s) \leq \sup_{\pi \in \Pi} \widetilde{V}_t^{\pi,\sigma}(s) = \widetilde{V}_t^{\star,\sigma}(s), \qquad (4.95)$$

which directly verifies the first assertion in equation 4.89 $\widetilde{V}_t^{\widetilde{\pi},\sigma}(s) = \widetilde{V}_t^{\star,\sigma}(s)$ for all $s \in \mathcal{S}$. The second assertion in equation 4.89 can be achieved analogously. Until now, we verify that there exists at least a policy $\widetilde{\pi}$ that obeys equation 4.89, which we refer to as an optimal policy since its value is equal to or larger than any other non-stationary and stochastic policies over all states $s \in \mathcal{S}$.

Proof of Theorem 4

We establish the proof by separating it into several key steps.

Step 1: Constructing a hard instance $\mathcal{M}$ of standard MDP. In this section, we consider the following standard MDP instance $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P^0, T, r\}$ where $\mathcal{S} = \{[0,0],[0,1],[1,0],[1,1]\}$ is the state space consisting of four elements in dimension $n = 2$, and $\mathcal{A} = \{0,1\}$ is the action space with only two options. The transition kernel $P^0 = \{P_t^0\}_{1 \leq t \leq T}$ at different time steps $1 \leq t \leq T$ is defined as

$$P_1^0(s' \mid s,a) = \begin{cases} \mathbb{1}(s' = [0,0])\mathbb{1}(a=0) + \mathbb{1}(s'=[0,1])\mathbb{1}(a=1) & \text{if } s = [0,0] \\ \mathbb{1}(s'=s) & \text{otherwise} \end{cases}, \qquad (4.96)$$

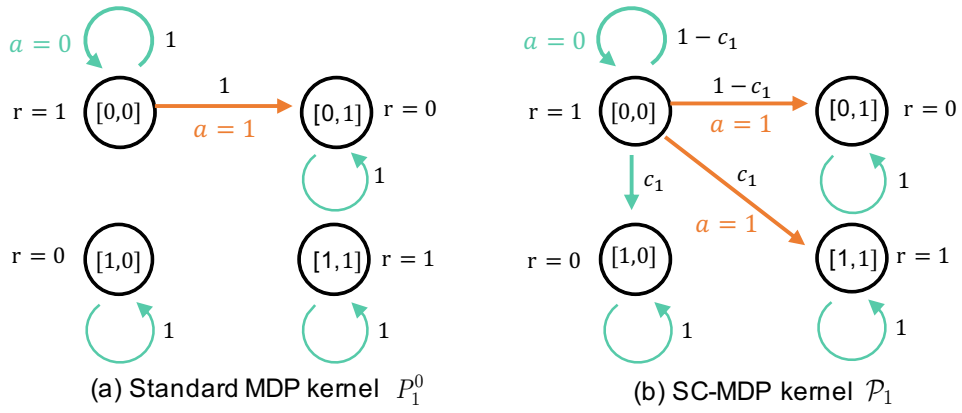


Figure 4.21: The illustration of the transition kernels of the standard MDP $\mathcal{M}$ and the proposed SC-MDP $\mathcal{M}_{\text{sc}}$ at the first time step $t = 1$, i.e., P_t^0 and $\mathcal{P}_1$ respectively.

which is illustrated in Figure 4.21(a), and

$$P_t^0(s' \,|\, s, a) = \mathbb{1}(s' = s), \quad \forall (t, s, a) \in \{2, 3, \cdots, T\} \times \mathcal{S} \times \mathcal{A}. \tag{4.97}$$

Note that this transition kernel P^0 ensures that the next state transitioned from the state $[0, 0]$ is $[0, 0]$ or $[0, 1]$. The reward function is specified as follows: for all time steps $1 \le t \le T$,

$$r_t(s, a) = \begin{cases} 1, & \text{if } s = [0, 0] \text{ or } s = [1, 1] \\ 0, & \text{otherwise} \end{cases}, \tag{4.98}$$

Step 2: The equivalence between $\mathcal{M}$ and one SC-MDP. Next, we shall show that the constructed standard MDP $\mathcal{M}$ can be equivalently represented by one SC-MDP $\mathcal{M}_{\text{sc}} = \{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, P^c\}$ with $\mathcal{C} := \{0, 1\}$. The equivalence is defined as the sequential observations $\{s_t, a_t, r_t\}_{1 \le t \le T}$ induced by any policy and any initial state distribution in two Markov processes are identical. To specify, $\mathcal{S}, \mathcal{A}, T, r$ are kept the same as $\mathcal{M}$. Here, $\{\mathcal{P}_t^i\}$ shall be specified in a while, which determines the transition to each dimension of the next state conditioned on the current state, action, and confounder distribution for all time steps, that is, $s_{t+1}^i \sim \mathbb{E}_{c_t \sim P_t^c}[\mathcal{P}_t^i(\cdot \,|\, s_t, a_t, c_t)]$ for any i-th dimension of the state ($i \in \{1, 2\}$) and for all time steps $1 \le t \le T$. For convenience, we denote $\mathcal{P}_t := [\mathcal{P}_t^1, \mathcal{P}_t^2] \in \Delta(\mathcal{S})$ as the transition kernel towards the next state, that is, $s_{t+1} \sim \mathbb{E}_{c_t \sim P_t^c}[\mathcal{P}_t(\cdot \,|\, s_t, a_t, c_t)]$.

Then we simply set the nominal distribution of the confounder as follows:

$$P_t^c(c_t) = \mathbb{1}(c_t = 0), \quad \forall 1 \le t \le T, c_t \in \mathcal{C}. \tag{4.99}$$

In addition, before introducing the transition kernel $\{\mathcal{P}_t^i\}$ of the SC-MDP $\mathcal{M}_{\mathsf{sc}}$, we introduce an auxiliary transition kernel $P^{\text{sc}} = \{P_t^{\text{sc}}\}$ as follows:

$$P_1^{\text{sc}}(s' \,|\, s, a) = \begin{cases} \mathbb{1}(s' = [1,0])\mathbb{1}(a = 0) + \mathbb{1}(s' = [1,1])\mathbb{1}(a = 1) & \text{if } (s,a) = ([0,0], 0) \\ \mathbb{1}(s' = s) & \text{otherwise} \end{cases}, \tag{4.100}$$

and

$$P_t^{\text{sc}}(s' \,|\, s, a) = \mathbb{1}(s' = s), \quad \forall (t, s, a) \in \{2, 3, \cdots, T\} \times \mathcal{S} \times \mathcal{A}. \tag{4.101}$$

It can be seen that P^{sc} is similar to P^0, except for the transition in the state $[0, 0]$.

Armed with this transition kernel P^{sc}, the $\{\mathcal{P}_t^i\}$ of the SC-MDP $\mathcal{M}_{\mathsf{sc}}$ is set to obey

$$\mathcal{P}_1(s' \,|\, s, a, c_1) = \begin{cases} (1 - c_1)P_1^0(s' \,|\, s, a) + c_1 P_1^{\text{sc}}(s' \,|\, s, a) & \text{if } s = [0,0] \\ \mathbb{1}(s' = s) & \text{otherwise} \end{cases}, \tag{4.102}$$

which is illustrated in Figure 4.21(b), and

$$\mathcal{P}_t(s' \,|\, s, a, c_t) = \mathbb{1}(s' = s), \quad \forall (t, s, a, c_t) \in \{2, 3, \cdots, T\} \times \mathcal{S} \times \mathcal{A} \times \mathcal{C}. \tag{4.103}$$

With them in mind, we are ready to verify that the marginalized transition from the current state and action to the next state in the SC-MDP $\mathcal{M}_{\mathsf{sc}}$ is identical to the one in MDP $\mathcal{M}$: for all $(t, s_t, a_t, s_{t+1}) \in \{1, 2, \cdots, T\} \times \mathcal{S} \times \mathcal{A} \times \mathcal{S}$:

$$\mathbb{P}(s_{t+1} \,|\, s_t, a_t) = \mathbb{E}_{c_t \sim P_t^c}\left[\mathcal{P}_t(s_{t+1} \,|\, s_t, a_t, c_t)\right] = \mathcal{P}_t(s_{t+1} \,|\, s_t, a_t, 0) = P^0(s_{t+1} \,|\, s_t, a_t), \tag{4.104}$$

where the second equality holds by the definition of P^c in equation 4.99, and the last equality

holds by the definitions of $\mathcal{P}$ (see equation 4.102 and equation 4.103).

In summary, we verified that the standard MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P^0, T, r\}$ is equal to the SC-MDP specified above $\mathcal{M}_{\mathsf{sc}}$.

Step 3: Defining corresponding RMDP and RSC-MDP. Equipped with the equivalent standard MDP $\mathcal{M}$ and SC-MDP $\mathcal{M}_{\mathsf{sc}}$, we consider the robust variants of them respectively — a RMDP $\mathcal{M}_{\mathsf{rob}} = \{\mathcal{S}, \mathcal{A}, \mathcal{U}^{\sigma_1}(P^0), T, r\}$ with some uncertainty level σ_1, and the proposed RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}} = \{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, \mathcal{U}^{\sigma_2}(P^c)\}$ with some uncertainty level σ_2.

In this section, without loss of generality, we consider total variation as the 'distance' function ρ for the uncertainty sets of both RMDP $\mathcal{M}_{\mathsf{rob}}$ and RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}}$, i.e., for any probability vectors $P', P \in \Delta(\mathcal{C})$ (or $P', P \in \Delta(\mathcal{S})$), $\rho(P', P) \coloneqq \frac{1}{2}\|P' - P\|_1$. Consequently, for any uncertainty level $\sigma \in [0, 1]$, the uncertainty set $\mathcal{U}^{\sigma_1}(P^0)$ of the RMDP (see equation 4.82) and $\mathcal{U}^{\sigma_2}(P^c)$ of the RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}}$ (see equation 4.85) are defined as follow, respectively:

$$\begin{aligned}
&\mathcal{U}^{\sigma}(P^0) \coloneqq \otimes\, \mathcal{U}^{\sigma}(P^0_{t,s,a}), \qquad \mathcal{U}^{\sigma}(P^0_{t,s,a}) \coloneqq \left\{ P_{t,s,a} \in \Delta(\mathcal{S}) : \frac{1}{2}\left\|P_{t,s,a} - P^0_{t,s,a}\right\|_1 \le \sigma \right\}, \\
&\mathcal{U}^{\sigma}(P^c) \coloneqq \otimes\, \mathcal{U}^{\sigma}(P^c_t), \qquad \mathcal{U}^{\sigma}(P^c_t) \coloneqq \left\{ P \in \Delta(\mathcal{C}) : \frac{1}{2}\|P - P^c_t\|_1 \le \sigma \right\}.
\end{aligned} \tag{4.105}$$

Step 4: Comparing between the performance of the optimal policy of RMDP $\mathcal{M}_{\mathsf{rob}}$ ($\pi^{\star,\sigma_1}_{\mathsf{RMDP}}$) and that of RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}}$ ($\pi^{\star,\sigma_2}_{\mathsf{RSC}}$). To continue, we specify the robust optimal policy $\pi^{\star,\sigma_1}_{\mathsf{RMDP}}$ associated with $\mathcal{M}_{\mathsf{rob}}$ and $\pi^{\star,\sigma_2}_{\mathsf{RSC}}$ associated with $\mathcal{M}_{\mathsf{sc\text{-}rob}}$ and then compare their performance on RSC-MDP with some initial state distribution.

To begin, we introduce the following lemma about the robust optimal policy $\pi^{\star,\sigma_1}_{\mathsf{RMDP}}$ associated with the RMDP $\mathcal{M}_{\mathsf{rob}}$.

Lemma 7. *For any $\sigma_1 \in (0, 1]$, the robust optimal policy of $\mathcal{M}_{\mathsf{rob}}$ obeys*

$$\forall s \in \mathcal{S}: \quad \left[\pi^{\star,\sigma_1}_{\mathsf{RMDP}}\right]_1(0 \,|\, s) = 1. \tag{4.106a}$$

In addition, we characterize the robust SC-value functions of the RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}}$ associated with any policy, combined with the optimal policy and its optimal robust SC-value functions, shown in the following lemma.

Lemma 8. *Consider any $\sigma_2 \in (\frac{1}{2}, 1]$ and the RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}} = \big\{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, \mathcal{U}^{\sigma_2}(P^c)\big\}$. For any policy π, the corresponding robust SC-value functions satisfy*

$$\widetilde{V}_1^{\pi,\sigma_2}([0,0]) = 1 + (T-1) \inf_{P \in \mathcal{U}^{\sigma}(P_1^c)} \mathbb{E}_{c_1 \sim P}\Big[\pi_1(0 \,|\, [0,0])(1-c_1) + \pi_1(1 \,|\, [0,0])c_1\Big]. \quad (4.107a)$$

In addition, the optimal robust SC-value function and the robust optimal policy $\pi_{\mathsf{RSC}}^{\star,\sigma_2}$ of the RMDP $\mathcal{M}_{\mathsf{sc\text{-}rob}}$ obeys:

$$\widetilde{V}_1^{\pi_{\mathsf{RSC}}^{\star,\sigma_2},\sigma_2}([0,0]) = \widetilde{V}_1^{\star,\sigma_2}([0,0]) = 1 + \frac{T-1}{2}. \quad (4.108)$$

Armed with above lemmas, applying Lemma 8 with policy $\pi = \pi_{\mathsf{RMDP}}^{\star,\sigma_1}$ obeying $\big[\pi_{\mathsf{RMDP}}^{\star,\sigma_1}\big]_1(0 \,|\, s) = 1$ in Lemma 7, one has

$$\begin{aligned}\widetilde{V}_1^{\pi_{\mathsf{RMDP}}^{\star,\sigma_1},\sigma_2}([0,0]) &= 1 + (T-1) \inf_{P \in \mathcal{U}^{\sigma}(P_1^c)} \mathbb{E}_{c_1 \sim P}\Big[1 - c_1\Big] \\ &\leq 1 + (T-1)\left[\frac{1}{4} \cdot 1 + \frac{3}{4} \cdot 0\right] = 1 + \frac{T-1}{4}, \end{aligned} \quad (4.109)$$

where the inequality holds by the fact that the probability distribution P obeying $P_1(0) = \frac{1}{4}$ and $P_1(1) = \frac{3}{4}$ is inside the uncertainty set $\mathcal{U}^{\sigma_2}(P_1^c)$ (recall that $\sigma_2 \in (\frac{1}{2}, 1]$ and $P_1^c(0) = 1$).

Finally, combining equation 4.109 and equation 4.108 together, we complete the proof by showing that with the initial state distribution ϕ defined as $\phi([0,0]) = 1$, we arrive at

$$\widetilde{V}_1^{\pi_{\mathsf{RSC}}^{\star,\sigma_2},\sigma_2}(\phi) - \widetilde{V}_1^{\pi_{\mathsf{RMDP}}^{\star,\sigma_1},\sigma_2}([0,0]) = \widetilde{V}_1^{\star,\sigma_2}(\phi) - \widetilde{V}_1^{\pi_{\mathsf{RMDP}}^{\star,\sigma_1},\sigma_2}([0,0]) \geq \frac{T-1}{4} \geq \frac{T}{8}, \quad (4.110)$$

where the last inequality holds by $T \geq 2$.

Proof of auxiliary results

Proof of Lemma 7

Step 1: specifying the minimum of the robust value functions over states. For any uncertainty set $\sigma_1 \in (0, 1]$, we first characterize the robust value function of any policy π over different states.

To start, we denote the minimum of the robust value function over states at each time step t as below:

$$V_{\min,t}^{\pi,\sigma_1} \coloneqq \min_{s\in\mathcal{S}} V_t^{\pi,\sigma_1}(s) \geq 0, \tag{4.111}$$

where the last inequality holds that the reward function defined in equation 4.98 is always non-negative. Obviously, there exists at least one state $s_{\min,t}^{\pi}$ that satisfies $V_t^{\pi,\sigma_1}(s_{\min,t}^{\pi}) = V_{\min,t}^{\pi,\sigma_1}$.

With this in mind, we shall verify that for any policy π,

$$\forall 1 \leq t \leq T: \quad V_t^{\pi,\sigma_1}([0,1]) = V_t^{\pi,\sigma_1}([1,0]) = 0. \tag{4.112}$$

To achieve this, we use a recursive argument. First, the base case can be verified since when $t+1 = T+1$, the value functions are all zeros in the $T+1$ step, that is, $V_{T+1}^{\pi,\sigma_1}(s) = 0$ for all $s \in \mathcal{S}$. Then, the goal is to verify the following fact

$$V_t^{\pi,\sigma_1}([0,1]) = V_t^{\pi,\sigma_1}([1,0]) = 0, \tag{4.113}$$

with the assumption that $V_{t+1}^{\pi,\sigma_1}(s) = 0$ for any state $s = \{[0,1],[1,0]\}$. It is easily observed that for any policy π, the robust value function when state $s = \{[0,1],[1,0]\}$ at any time step t obeys

$$\begin{aligned} 0 \leq V_t^{\pi,\sigma_1}(s) &= \mathbb{E}_{a\sim\pi_t(\cdot\,|\,s)}\left[r_t(s,a) + \inf_{P\in\mathcal{U}^{\sigma_1}(P_{t,s,a}^0)} PV_{t+1}^{\pi,\sigma_1}\right] \\ &\overset{\text{(i)}}{=} 0 + (1-\sigma_1)V_{t+1}^{\pi,\sigma_1}(s) + \sigma_1 V_{\min,t+1}^{\pi,\sigma_1} \overset{\text{(ii)}}{=} 0 + \sigma_1 V_{\min,t+1}^{\pi,\sigma_1} \\ &\leq 0 + \sigma_1 V_{t+1}^{\pi,\sigma_1}(s) = 0 \end{aligned} \tag{4.114}$$

where (i) holds by $r_t(s,a) = 0$ for all $s = \{[0,1],[1,0]\}$, the fact $P_t^0(s\,|\,s,a) = 1$ for $s \in \mathcal{S}$ (see equation 4.96 and equation 4.97), and the definition of the uncertainty set $\mathcal{U}^{\sigma_1}(P^0)$ in equation 4.105. Here, (ii) follows from the recursive assumption $V_{t+1}^{\pi,\sigma_1}(s) = 0$ for any state $s = \{[0,1],[1,0]\}$, and the last equality holds by $V_{\min,t+1}^{\pi,\sigma_1} \leq V_{t+1}^{\pi,\sigma_1}([0,1])$ (see equation 4.111). Until now, we have completed the proof for equation 4.113 and then verified equation 4.112.

Note that equation 4.112 direcly leads to

$$\forall 1 \leq t \leq T: \quad V_{\min,t}^{\pi,\sigma_1} = 0. \tag{4.115}$$

Step 2: Considering the robust value function at state $[0,0]$. Armed with the above facts, we are now ready to derive the robust value function for state $[0,0]$.

When $2 \leq t \leq T$, one has

$$\begin{aligned}
V_t^{\pi,\sigma_1}([0,0]) &= \mathbb{E}_{a\sim\pi_t(\cdot\,|\,[0,0])}\left[r_t([0,0],a) + \inf_{P\in\mathcal{U}^{\sigma_1}(P_{t,[0,0],a})} PV_{t+1}^{\pi,\sigma_1}\right] \\
&\overset{\text{(i)}}{=} 1 + \left[(1-\sigma_1)V_{t+1}^{\pi,\sigma_1}([0,0]) + \sigma_1 V_{\min,t+1}^{\pi,\sigma_1}\right] \\
&= 1 + (1-\sigma_1)V_{t+1}^{\pi,\sigma_1}([0,0]),
\end{aligned} \tag{4.116}$$

where (i) holds by $r_t([0,0],a) = 1$ for all $a \in \{0,1\}$ and the definition of P^0 (see equation 4.97), and the last equality arises from equation 4.115 .

Applying equation 4.116 recursively for $t, t+1, \cdots, T$ yields that

$$V_t^{\pi,\sigma_1}([0,0]) = \sum_{k=t}^{T}(1-\sigma_1)^{k-t} \geq 1. \tag{4.117}$$

When $t = 1$, the robust value function obeys:

$$\begin{aligned}
V_1^{\pi,\sigma_1}([0,0]) &= \mathbb{E}_{a\sim\pi_1(\cdot\,|\,[0,0])}\left[r_1([0,0],a) + \inf_{P\in\mathcal{U}^{\sigma_1}(P_{1,[0,0],a})} PV_2^{\pi,\sigma_1}\right] \\
&\overset{\text{(i)}}{=} 1 + \pi_1(0\,|\,[0,0]) \inf_{P\in\mathcal{U}^{\sigma_1}(P_{1,[0,0],0})} PV_2^{\pi,\sigma_1} + \pi_1(1\,|\,[0,0]) \inf_{P\in\mathcal{U}^{\sigma_1}(P_{1,[0,0],1})} PV_2^{\pi,\sigma_1} \\
&\overset{\text{(ii)}}{=} 1 + \pi_1(0\,|\,[0,0])\left[(1-\sigma_1)V_2^{\pi,\sigma_1}([0,0]) + \sigma_1 V_{\min,2}^{\pi,\sigma_1}\right] \\
&\quad + \pi_1(1\,|\,[0,0])\left[(1-\sigma_1)V_2^{\pi,\sigma_1}([0,1]) + \sigma_1 V_{\min,2}^{\pi,\sigma_1}\right] \\
&= 1 + \pi_1(0\,|\,[0,0])(1-\sigma_1)V_2^{\pi,\sigma_1}([0,0]),
\end{aligned} \tag{4.118}$$

where (i) holds by $r_1([0,0],a) = 1$ for all $a \in \{0,1\}$, (ii) follows from the definition of P^0 (see equation 4.96), and the last equality arises from equation 4.112 and equation 4.115.

Step 3: the optimal policy $\pi^{\star,\sigma_1}_{\mathsf{RMDP}}$. Observing that $V_1^{\pi,\sigma_1}([0,0])$ is increasing monotically as $\pi_1(0\,|\,[0,0])$ is larger, we directly have that $\pi^{\star,\sigma_1}_{\mathsf{RMDP}}(0\,|\,[0,0]) = 1$.

Considering that the action does not influence the state transition for $t = 2, 3, \cdots, T$ and all other states $s \neq [0,0]$, without loss of generality, we choose the robust optimal policy as

$$\forall s \in \mathcal{S}: \quad \left[\pi^{\star,\sigma_1}_{\mathsf{RMDP}}\right]_1(0\,|\,s) = 1. \tag{4.119}$$

Proof of Lemma 8

To begin with, for any uncertainty level $\sigma_2 \in (\frac{1}{2}, 1]$ and any policy $\pi = \{\pi_t\}$, we consider the robust SC-value function $\widetilde{V}_t^{\pi,\sigma_2}$ of the RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}}$.

Step 1: deriving $\widetilde{V}_t^{\pi,\sigma_2}$ for $2 \leq t \leq T$. Towards this, for any $2 \leq t \leq T$ and $s \in \mathcal{S}$, one has

$$\begin{aligned}
\widetilde{V}_t^{\pi,\sigma_2}(s) &= \mathbb{E}_{a\sim\pi_t(s)}\left[\widetilde{Q}_t^{\pi,\sigma_2}(s,a)\right] \\
&\overset{\text{(i)}}{=} \mathbb{E}_{a\sim\pi_t(s)}\left[r_t(s,a) + \inf_{P\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{c_t\sim P}\left[\mathcal{P}_{t,s,a,c_t}\widetilde{V}_{t+1}^{\pi,\sigma_2}\right]\right] \\
&\overset{\text{(ii)}}{=} r_t(s,a) + \inf_{P\in\mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{c_t\sim P}\left[\mathcal{P}_{t,s,a,c_t}\widetilde{V}_{t+1}^{\pi,\sigma_2}\right] \\
&= r_t(s,a) + \widetilde{V}_{t+1}^{\pi,\sigma}(s),
\end{aligned} \tag{4.120}$$

where (i) follows from the *state-confounded* Bellman consistency equation in equation 4.129, (ii) holds by that the reward function r_t and $\mathcal{P}_t$ are all independent from the action (see equation 4.98 and equation 4.103), and the last inequality holds by $\mathcal{P}_t(s'\,|\,s,a,c_t) = \mathbb{1}(s'=s)$ is independent from c_t (see equation 4.103).

Applying the above fact recursively for $t, t+1, \cdots, T$ leads to that for any $s \in \mathcal{S}$,

$$\begin{aligned}
\widetilde{V}_t^{\pi,\sigma_2}(s) &= r_t(s,a_t) + \widetilde{V}_{t+1}^{\pi,\sigma}(s) = r_t(s,a) + r_{t+1}(s,a_{t+1}) + \widetilde{V}_{t+2}^{\pi,\sigma}(s) \\
&= \cdots = r_t(s,a_t) + \sum_{k=t+1}^{T} r_k(s_k,a_k),
\end{aligned} \tag{4.121}$$

which directly yields (see reward r in equation 4.98)

$$\widetilde{V}_2^{\pi,\sigma_2}([0,0]) = \widetilde{V}_2^{\pi,\sigma_2}([1,1]) = T-1 \quad \text{and} \quad \widetilde{V}_2^{\pi,\sigma_2}([0,1]) = \widetilde{V}_2^{\pi,\sigma_2}([1,0]) = 0. \tag{4.122}$$

Step 2: characterizing $\widetilde{V}_1^{\pi,\sigma_2}([0,0])$ for any policy π. In this section, we consider the value of $\widetilde{V}_1^{\pi,\sigma_2}$ on the state $[0,0]$. To proceed, one has

$$\begin{aligned}
\widetilde{V}_1^{\pi,\sigma_2}([0,0]) &= \mathbb{E}_{a\sim\pi_1([0,0])}\left[\widetilde{Q}_1^{\pi,\sigma_2}([0,0],a)\right] \\
&\overset{\text{(i)}}{=} \mathbb{E}_{a\sim\pi_1([0,0])}\left[r_1([0,0],a) + \inf_{P\in\mathcal{U}^\sigma(P_1^c)} \mathbb{E}_{c_1\sim P}\left[\mathcal{P}_{1,[0,0],a,c_1}\widetilde{V}_2^{\pi,\sigma_2}\right]\right] \\
&\overset{\text{(ii)}}{=} 1 + \inf_{P\in\mathcal{U}^\sigma(P_1^c)} \mathbb{E}_{c_1\sim P}\left[\left(\pi_1(0\,|\,[0,0])\mathcal{P}_{1,[0,0],0,c_1} + \pi_t(1\,|\,[0,0])\mathcal{P}_{1,[0,0],1,c_1}\right)\widetilde{V}_2^{\pi,\sigma}\right] \\
&\overset{\text{(iii)}}{=} 1 + \inf_{P\in\mathcal{U}^\sigma(P_1^c)} \mathbb{E}_{c_1\sim P}\Bigg[\pi_1(0\,|\,[0,0])\left((1-c_1)P^0_{1,[0,0],0} + c_1 P^{\text{sc}}_{1,[0,0],0}\right)\widetilde{V}_2^{\pi,\sigma} \\
&\quad + \pi_1(1\,|\,[0,0])\left((1-c_1)P^0_{1,[0,0],1} + c_1 P^{\text{sc}}_{1,[0,0],1}\right)\widetilde{V}_2^{\pi,\sigma}\Bigg] \\
&\overset{\text{(iv)}}{=} 1 + \inf_{P\in\mathcal{U}^\sigma(P_1^c)} \mathbb{E}_{c_1\sim P}\Bigg[\pi_1(0\,|\,[0,0])\Big((1-c_1)\widetilde{V}_2^{\pi,\sigma}([0,0]) + c_1\widetilde{V}_2^{\pi,\sigma}([1,0])\Big) \\
&\quad + \pi_1(1\,|\,[0,0])\Big((1-c_1)\widetilde{V}_2^{\pi,\sigma}([0,1]) + c_1\widetilde{V}_2^{\pi,\sigma}([1,1])\Big)\Bigg] \\
&= 1 + (T-1)\inf_{P\in\mathcal{U}^\sigma(P_1^c)} \mathbb{E}_{c_1\sim P}\Big[\pi_1(0\,|\,[0,0])(1-c_1) + \pi_1(1\,|\,[0,0])c_1\Big] \\
&= 1 + (T-1)\pi_1(0\,|\,[0,0]) + (T-1)\inf_{P\in\mathcal{U}^\sigma(P_1^c)} \mathbb{E}_{c_1\sim P}\Big[c_1\big(1-2\pi_1(0\,|\,[0,0])\big)\Big],
\end{aligned} \tag{4.123}$$

where (i) holds by *robust state-confounded* Bellman consistency equation in equation 4.129, (ii) follows from $r_1([0,0],a) = 1$ for all $a \in \{0,1\}$ which is independent from c_t. (iii) arises from the definition of $\mathcal{P}$ in equation 4.102, (iv) can be verified by plugging in the definitions from equation 4.96 and equation 4.100, and the penultimate equality holds by equation 4.122.

Step 3: characterizing the optimal robust SC-value functions. Before proceeding, we recall the fact that $\mathcal{U}^\sigma(P_1^c) = \left\{P \in \Delta(\mathcal{C}) : \frac{1}{2}\left\|P - P_1^c\right\|_1 \le \sigma_2\right\}$.

Observing from equation 4.123 that for any fixed $\pi_1(0\,|\,[0,0])$, $c_1\big(1-2\pi_1(0\,|\,[0,0])\big)$ is

monotonously increasing with c_1 when $1 - 2\pi_1(0 \,|\, [0,0]) \geq 0$ and decreasing with c_1 otherwise, it is easily verified that the maximum of the following function

$$f\big(\pi_1(0 \,|\, [0,0])\big) \coloneqq (T-1) \inf_{P \in \mathcal{U}^{\sigma}(P_1^c)} \mathbb{E}_{c_1 \sim P} \left[c_1 \big(1 - 2\pi_1(0 \,|\, [0,0])\big) \right] \tag{4.124}$$

obeys

$$\max f\big(\pi_1(0 \,|\, [0,0])\big) = \begin{cases} 0 & \text{if } \pi_1(0 \,|\, [0,0]) \geq \frac{1}{2} \\ (T-1)\sigma_2\big(1 - 2\pi_1(0 \,|\, [0,0])\big) & \text{otherwise} \end{cases}. \tag{4.125}$$

Then, note that the value of $\widetilde{V}_1^{\pi,\sigma_2}([0,0])$ only depends on $\pi_1(\cdot \,|\, [0,0])$ which can be represent by $\pi_1(0 \,|\, [0,0])$. Plugging in equation 4.125 into equation 4.123 arrives at when $\pi_1(0 \,|\, [0,0]) \geq \frac{1}{2}$,

$$\begin{aligned} &\max_{\pi_1(0 \,|\, [0,0]) \geq \frac{1}{2}} \widetilde{V}_1^{\pi,\sigma_2}([0,0]) \\ &= \max_{\pi_1(0 \,|\, [0,0]) \geq \frac{1}{2}} 1 + (T-1)\pi_1(0 \,|\, [0,0]) + (T-1)\sigma_2\big(1 - 2\pi_1(0 \,|\, [0,0])\big) \\ &= 1 + (T-1)\sigma_2 + (T-1) \max_{\pi_1(0 \,|\, [0,0]) \geq \frac{1}{2}} (1 - 2\sigma_2)\pi_1(0 \,|\, [0,0]) \\ &= 1 + (T-1)\sigma_2 + \frac{(T-1)(1-2\sigma_2)}{2} = 1 + \frac{T-1}{2}, \end{aligned} \tag{4.126}$$

where the penultimate equality holds by $\sigma_2 > \frac{1}{2}$ and letting $\pi_1(0 \,|\, [0,0]) = \frac{1}{2}$. Similarly, when $\pi_1(0 \,|\, [0,0]) < \frac{1}{2}$,

$$\max_{\pi_1(0 \,|\, [0,0]) < \frac{1}{2}} \widetilde{V}_1^{\pi,\sigma_2}([0,0]) = \max_{\pi_1(0 \,|\, [0,0]) < \frac{1}{2}} 1 + (T-1)\pi_1(0 \,|\, [0,0]) < 1 + \frac{T-1}{2}. \tag{4.127}$$

Consequently, combining equation 4.126 and equation 4.127, we conclude that

$$\widetilde{V}_1^{\pi_{\mathsf{RSC}}^{\star,\sigma_2},\sigma_2}([0,0]) = \widetilde{V}_1^{\star,\sigma_2}([0,0]) = \max_{\pi} \widetilde{V}_1^{\pi,\sigma_2}([0,0]) = 1 + \frac{T-1}{2}. \tag{4.128}$$

Auxiliary results of RSC-MDPs

It is easily verified that for any RSC-MDP $\mathcal{M}_{\mathsf{sc\text{-}rob}} = \big\{\mathcal{S}, \mathcal{A}, T, r, \mathcal{C}, \{\mathcal{P}_t^i\}, \mathcal{U}^{\sigma_2}(P^c)\big\}$, any

policy π and optimal policy $\pi^\star$ satisfy the corresponding *robust state-confounded* Bellman consistency equation and Bellman optimality equation shown below, respectively:

$$\widetilde{Q}_t^{\pi,\sigma}(s,a) = r_t(s,a) + \inf_{P \in \mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{c_t \sim P}\left[\mathcal{P}_{t,s,a,c_t} \widetilde{V}_{t+1}^{\pi,\sigma}\right],$$
$$\widetilde{Q}_t^{\star,\sigma}(s,a) = r_t(s,a) + \inf_{P \in \mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{c_t \sim P}\left[\mathcal{P}_{t,s,a,c_t} \widetilde{V}_{t+1}^{\star,\sigma}\right], \tag{4.129}$$

where $\mathcal{P}_{t,s,a,c_t} \in \mathbb{R}^{1\times S}$ such that $\mathcal{P}_{t,s,a,c_t}(s') \coloneqq \mathcal{P}_t(s' \,|\, s,a,c_t)$ for $s' \in \mathcal{S}$, and $\widetilde{V}_t^{\star,\sigma}(s) = \sup_{\pi_t \in \Delta(\mathcal{A})} \left\{ \mathbb{E}_{\pi_t}[r_t(s,a_t)] + \inf_{P_t \in \mathcal{U}^\sigma(P_t^c)} \mathbb{E}_{\pi_t}\left[\mathbb{E}_{c_t \sim P_t}\left[\mathcal{P}_{t,s,a,c_t} \widetilde{V}_{t+1}^{\star,\sigma}(s_{t+1})\right]\right]\right\}$.

4.3.4 An Empirical Algorithm to Solve RSC-MDPs: RSC-SAC

Algorithm 7: RSC-SAC Training

Input: policy π, data buffer $\mathcal{D}$, transition model P_θ, ratio of modified data β

for $t \in [1, T]$ **do**
- Sample action $a_t \sim \pi(\cdot|s_t)$
- $(s_{t+1}, r_t) \leftarrow \text{Env}(s_t, a_t)$
- Add buffer $\mathcal{D} = \mathcal{D} \cup \{s_t, a_t, s_{t+1}, r_t\}$
- **for** *sample batch* $\mathcal{B} \in \mathcal{D}$ **do**
  - Randomly select $\beta\%$ data in $\mathcal{B}$
  - Modify s_t in selected data with (4.130)
  - $(\hat{s}_{t+1}, \hat{r}_t) \sim P_\theta(s_t, a_t, \mathcal{G}_\phi)$
  - Replace data with $(s_t, a_t, \hat{s}_{t+1}, \hat{r}_t)$
  - $\mathcal{L} = \|s_{t+1} - \hat{s}_{t+1}\|_2^2 + \|r_t - \hat{r}_t\|_2^2$
  - Update θ and ϕ with $\mathcal{L} + \lambda\|\boldsymbol{G}\|_p$
  - Update π with the SAC algorithm

When addressing distributionally robust problems in RMDPs, the worst-case is typically defined within a prescribed uncertainty set in a clear and implementation-friendly manner, allowing for iterative or analytical solutions. However, solving RSC-MDPs could be challenging, as the structured uncertainty set is induced by the causal effect of perturbing the confounder. The precise characterization of this structured uncertainty set is difficult, since neither the unobserved

confounder nor the true causal graph of the observable variables is accessible, both of which are necessary for intervention or counterfactual reasoning. Therefore, we chose to approximate the causal effect of perturbing the confounder by learning from the data collected during training.

In this section, we propose an intuitive yet effective empirical approach named RSC-SAC to solve RSC-MDPs, which is outlined in Algorithm 7. We first estimate the effect of perturbing the distribution P^c of the confounder to generate new states (Section 12). Then, we learn the structural causal model $\mathcal{P}_t^i$ to predict the rewards and the next states given the perturbed states (Section 12). By combining these two components, we construct a data generator capable of simulating novel transitions (s_t, a_t, r_t, s_{t+1}) from the set of structured uncertainties. To learn the optimal policy, we construct the data buffer with a mixture of the original data and the generated data and then use the Soft Actor-Critic (SAC) algorithm (Haarnoja et al., 2018) to optimize the policy.

Distribution of confounder

As we have no prior knowledge about confounders, we choose to approximate the effect of perturbing them without explicitly estimating the distribution P^c. We first randomly select a dimension i from the state s_t to apply the perturbation and then assign the dimension i of s_t with a heuristic rule. We select the value from another sample s_k that has the most different value from s_t in dimension i and the most similar value to s_t in the remaining dimensions. Formally, this process solves the following optimization problem to select the sample k from a batch of K samples:

$$s_t^i \leftarrow s_k^i,\ k = \arg\max \frac{\|s_t^i - s_k^i\|_2^2}{\sum_{\neg i} \|s_t^{\neg i} - s_k^{\neg i}\|_2^2},\ \ k \in \{1, ..., K\}, \tag{4.130}$$

where s_t^i and $s_t^{\neg i}$ means dimension i of s_t and other dimensions of s_t except for i, respectively. Intuitively, permuting the dimension of two samples breaks the spurious correlation and remains the most semantic meaning of the state space. However, this permutation sometimes also breaks the true cause and effect between dimensions, leading to a performance drop. The trade-off between robustness and performance (Xu et al., 2023) is a long-standing dilemma in the robust optimization framework, which we will leave to future work.

Learning of structural causal model

With the perturbed state s_t, we then learn an SCM to predict the next state and reward considering the effect of the action on the previous state. This model contains a causal graph to achieve better generalization to unseen state-action pairs. Specifically, we simultaneously learn the model parameter and discover the underlying causal graph in a fully differentiable way with $(\hat{s}_{t+1}, \hat{r}_t) \sim P_\theta(s_t, a_t, \mathcal{G}_\phi)$, where θ is the parameter of the neural network of the dynamic model and $\phi \in \mathbb{R}^{(n+d_\mathcal{A})\times(n+1)}$ is the parameter that represents the causal graph $\mathcal{G}$ between $\{s_t, a_t\}$ and $\{s_{t+1}, r_t\}$. This graph is represented by a binary adjacency matrix $\boldsymbol{G}$, where $1/0$ means the existence/absence of an edge. P_θ has an encoder-decoder structure with matrix $\boldsymbol{G}$ as an intermediate linear transformation. The encoder takes in the state and action and outputs features $f_e \in \mathbb{R}^{(n+d_\mathcal{A})\times d_f}$ for each dimension, where d_f is the dimension of the feature. The causal graph is then multiplied to generate the feature for the decoder $f_d = f_e^T \boldsymbol{G} \in \mathbb{R}^{d_f \times (n+1)}$. The decoder takes in f_d and outputs the next state and reward. The detailed architecture of this causal transition model can be found in Appendix 12.

The objective of training this model consists of two parts, one is the supervision signal from collected data $\|s_{t+1} - \hat{s}_{t+1}\|_2^2 + \|r_t - \hat{r}_t\|_2^2$, and the other is a penalty term $\lambda\|\boldsymbol{G}\|_p$ with weight λ to encourage the sparsity of the matrix $\boldsymbol{G}$. The penalty is important to break the spurious correlation between dimensions of the state, since it forces the model to eliminate unnecessary inputs for prediction.

Architecture of the structural causal model

We plot the architecture of the structural causal model we used in our method in Figure 4.22. In normal neural networks, the input is treated as a whole to pass through linear layers or convolution layers. However, this structure blends all the information in the input, making the causal graph useless to separate cause and effect. Thus, in our model, we design an encoder that is shared across all dimensions of the input. Since different dimensions could have exactly the same values, we add a learnable position embedding to the input of the encoder. In summary, the input dimension of the encoder is $1+d_{pos}$, where d_{pos} is the dimension of the position embedding.

After using the encoder, we obtain a set of independent features for each dimension of the

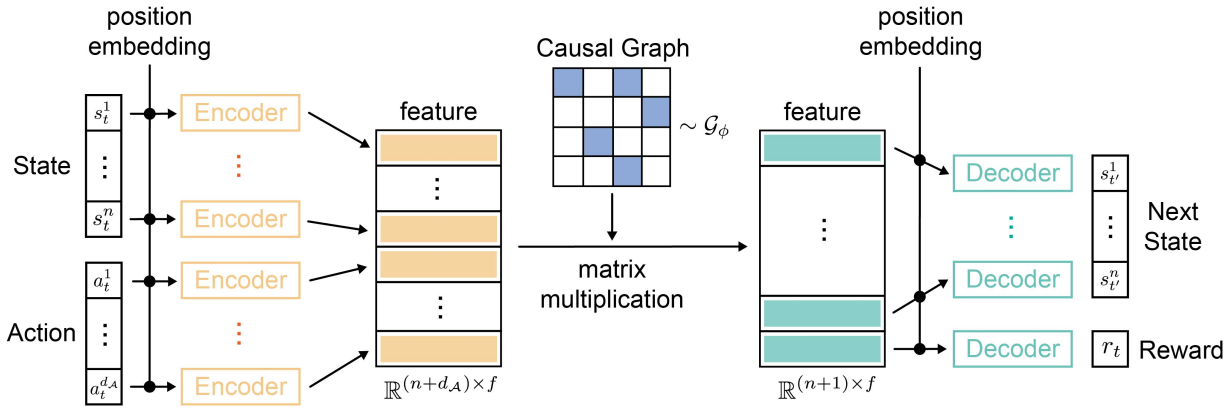


Figure 4.22: Model architecture of the structural causal model. Encoder, Decoder, position embedding, and Causal Graph are learnable during the training stage.

input. We now multiply the features with a learnable binary causal graph $\boldsymbol{G}$. The element (i, j) of the graph is sampled from a Gumbel-Softmax distribution with parameter $\phi_{i,j}$ to ensure the loss function is differentiable w.r.t. ϕ.

Multiplication of the causal graph and the input feature creates a linear combination of the input feature with respect to the causal graph. The obtained features are then passed through a decoder to predict the next state and reward. Again, the decoder is shared across all dimensions to avoid information leaking between dimensions. Position embedding is included in the input of the decoder and the output dimension of the decoder is 1.

4.3.5 Experiments and Analysis

In this section, we first provide a benchmark consisting of eight environments with spurious correlations, which may be of independent interest to robust RL. Then we evaluate the proposed algorithm RSC-SAC with comparisons to previous robust algorithms in RL.

Tasks with spurious correlation

To the best of our knowledge, no existing benchmark addresses the issues of spurious correlation in the RL state space. To bridge the gap, we designed a benchmark consisting of eight novel tasks in the self-driving and manipulation domains using the Carla (Dosovitskiy et al., 2017)

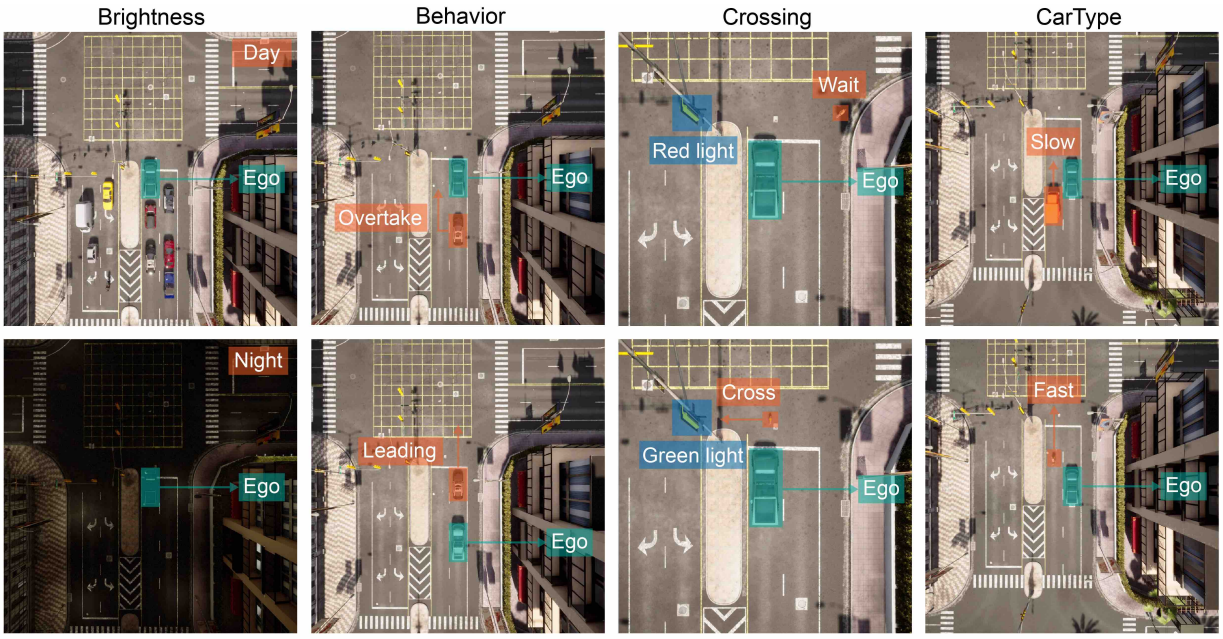


Figure 4.23: Illustration of tasks in the Carla simulator.

and Robosuite (Zhu et al., 2020) platforms (shown in Figure 4.23 and Figure 4.24). Tasks are designed to include spurious correlations in terms of human common sense, which is ubiquitous in decision-making applications and could cause safety issues. We categorize the tasks into *distraction correlation* and *composition correlation* according to the type of spurious correlation. We specify these two types of correlation below and also introduce the descriptions of the tasks.

- **Distraction correlation** is between task-relevant and task-irrelevant portions of the state. The task-irrelevant part could distract the policy model from learning important features and lead to a performance drop. A typical method to avoid distraction is background augmentation (Laskin et al., 2020; Yarats et al., 2021). We design four tasks with this category of correlation, i.e., *Lift*, *Wipe*, *Brightness*, and *CarType*.
- **Composition correlation** is between two task-relevant portions of the state. This correlation usually exists in compositional generalization, where states are recomposed to form novel tasks during testing. Typical examples are multi-task RL (Jiang et al., 2022; Lu et al., 2021) and hierarchical RL (Le et al., 2018; Yoo et al., 2022). We design four tasks with this category of correlation, i.e., *Stack*, *Door*, *Behavior*, and *Crossing*.

We designed four self-driving tasks in the Carla simulator (Dosovitskiy et al., 2017) and four

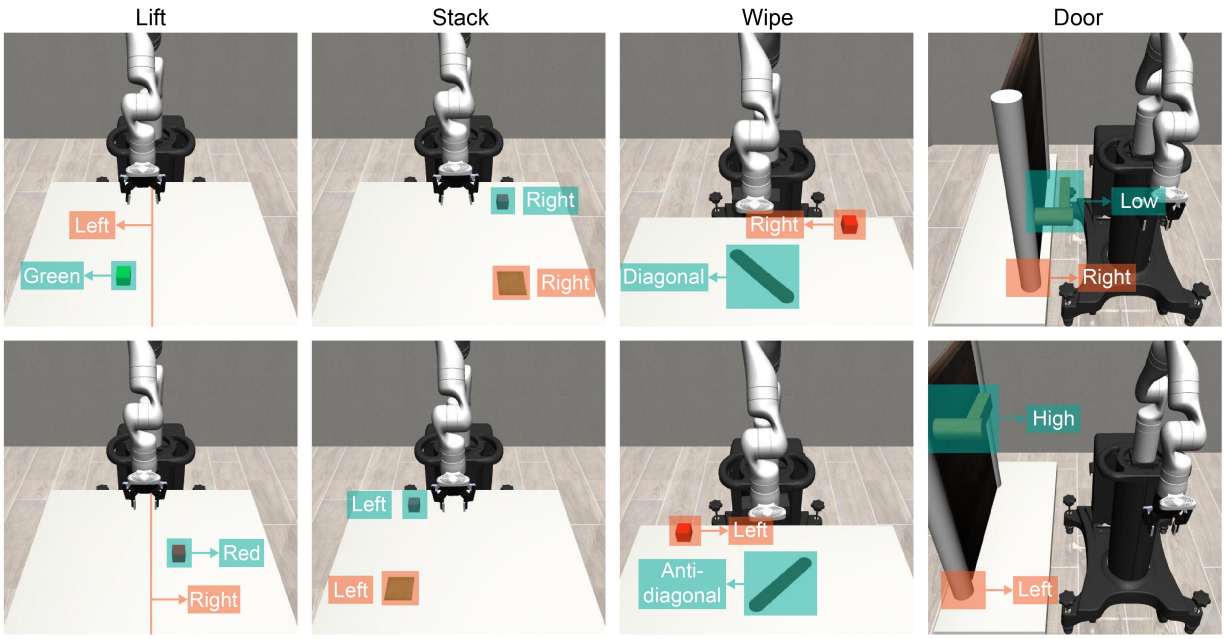


Figure 4.24: Illustration of tasks in the Robosuite simulator.

manipulation tasks on the Robosuite platform (Zhu et al., 2020). All of these realistic tasks contain strong spurious correlations that are explicitly observed in humans. We provide detailed descriptions of all of these environments in the following.

Brightness. The nominal environments are shown in the 1th column of Figure 4.23, where the brightness and the traffic density are correlated. When the ego vehicle drives during the day, there are many surrounding vehicles (first row). When the ego vehicle drives in the evening, there is no surrounding vehicle (second row). The shifted environment changes the brightness and traffic density in the nominal environment, that is, many surrounding vehicles in the evening and no surrounding vehicles in the day.

Behavior. The nominal environments are shown in the 2nd column of Figure 4.23, where the other vehicle has aggressive driving behavior. When the ego vehicle is in front of the other vehicle, the other vehicle always accelerates and overtakes the ego vehicle in the left lane. When the ego vehicle is behind the other vehicle, the other vehicle will always accelerate. In the shifted environment, the behavior of the other vehicle is conservative, i.e., the other vehicle always decelerates to block the ego vehicle.

Crossing. The nominal environments are shown in the 3rd column of Figure 4.23, where the

pedestrian follows the traffic rule and only crosses the road when the traffic light is green. In the shifted environment, the pedestrian disobeys the traffic rules and crosses the road when the traffic light is red.

CarType. The nominal environments are shown in the 4th column of Figure 4.23, where the type of vehicle and the speed of the vehicle are correlated. When the vehicle is a truck, the speed is low and when the vehicle is a motorcycle, the speed is high. In the shifted environment, the truck drives very fast and the motorcycle drives very slow.

Lift. The nominal environments are shown in the 1th column of Figure 4.24, where the position of the cube and the color of the cube are correlated. When the cube is in the left part of the table, the color of the cube is green, when the cube is in the right part of the table, the color of the cube is red. The shifted environment swaps the color and position of the cube in the nominal environment, i.e., the cube is green when it is in the right part and the cube is red when it is in the left part.

Stack. The nominal environments are shown in the 2nd column of Figure 4.24, where the position of the red cube and green plate are correlated. When the cube is in the left part of the table, the plate is also in the left part; when the cube is in the right part of the table, the plate is also in the right part. In the shifted environment, the relative position of the cube and the plate changes, i.e., When the cube is in the left part of the table, the plate is in the right part; when the cube is in the right part of the table, the plate is in the left part.

Wipe. The nominal environments are shown in the 3rd column of Figure 4.24, where the shape of the dirty region is correlated with the position of the cube. When the dirty region is diagonal, the cube is on the right-hand side of the robot arm. When the dirty region is anti-diagonal, the cube is on the left-hand side of the robot arm. In the shifted environment, the correlation changes, i.e., when the dirty region is diagonal, the cube is on the left-hand side of the robot arm; when the dirty region is anti-diagonal, the cube is on the right-hand side of the robot arm.

Door. The nominal environments are shown in the 4th column of Figure 4.24, where the height of the handle and the position of the door are correlated. When the door is closed to the robot arm, the handle is in a low position. When the door is far from the robot arm, the handle

Table 4.8: Testing reward on shifted environments. Bold font means the best reward.

Method	Brightness	Behavior	Crossing	CarType	Lift	Stack	Wipe	Door
SAC	0.56±0.13	0.13±0.03	0.81±0.13	0.63±0.14	0.58±0.13	0.26±0.12	0.16±0.20	0.08±0.07
RMDP-G	0.55±0.15	0.16±0.04	0.47±0.13	0.53±0.16	0.31±0.08	0.33±0.15	0.06±0.17	0.07±0.03
RMDP-U	0.54±0.19	0.13±0.05	0.60±0.15	0.39±0.13	0.51±0.17	0.23±0.11	0.06±0.17	0.10±0.13
MoCoDA	0.50±0.14	0.16±0.05	0.22±0.14	0.23±0.12	0.46±0.14	0.29±0.11	0.01±0.24	0.09±0.14
ATLA	0.48±0.11	0.14±0.03	0.61±0.14	0.52±0.14	0.61±0.18	0.21±0.12	0.29±0.18	0.28±0.19
DBC	0.52±0.18	0.16±0.03	0.68±0.12	0.45±0.10	0.12±0.02	0.03±0.02	0.19±0.35	0.01±0.01
Active	0.47±0.14	0.14±0.03	0.83±0.09	0.77±0.14	0.35±0.09	0.24±0.12	0.17±0.17	0.05±0.02
RSC-SAC	**0.99**±0.11	**1.02**±0.09	**1.04**±0.02	**1.03**±0.02	**0.98**±0.04	**0.77**±0.20	**0.85**±0.12	**0.61**±0.17

is in a high position. In the shifted environment, the correlation changes, i.e., when the door is closed to the robot arm, the handle is in a high position; when the door is far from the robot arm, the handle is in a low position.

Baselines

Robustness in RL has been explored in terms of various uncertainty sets over state, action, or transition kernels. Regarding this, we use a non-robust RL and four representative algorithms of robust RL as baselines, all of which are implemented on top of the SAC (Haarnoja et al., 2018) algorithm. **Non-robust RL (SAC):** This serves as a basic baseline without considering any robustness during training; **Solving robust MDP:** We generate samples to cover the uncertainty set over the state space by adding perturbation around the nominal states that follows two distributions, i.e., uniform distribution (RMDP-U) and Gaussian distribution (RMDP-G). **Solving SA-MDP:** We compare ATLA (Zhang et al., 2021), a powerful algorithm that generates adversarial states using an optimal adversary within the uncertainty set. **Invariant feature learning:** We choose DBC (Zhang et al., 2020b) that learns invariant features using the bi-simulation metric (Larsen and Skou, 1989) and (Gupta et al., 2023) (Active) that actively sample uncertain transitions to reduce causal confusion. **Counterfactual data augmentation:** We select MoCoDA (Pitis et al., 2022), which identifies local causality to change components and generate counterfactual samples to cover the targeted uncertainty set. We adapt this algorithm using an approximate causal graph rather than the true causal graph.

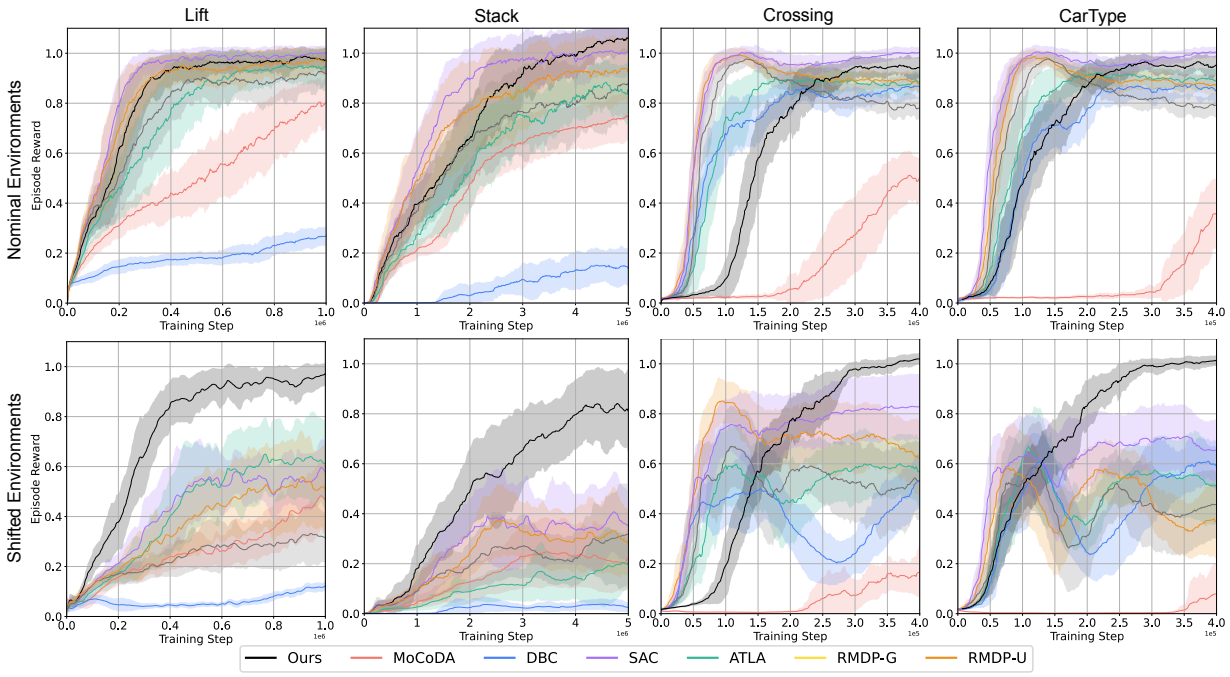


Figure 4.25: The first row shows the testing reward on the nominal environments, while the second row shows the testing reward on the shifted environments.

Results and Analysis

To comprehensively evaluate the performance of the proposed method RSC-SAC, we perform experiments to answer the following questions: **Q1.** Can RSC-SAC eliminate the harmful effect of spurious correlation in learned policy? **Q2.** Does the robustness of RSC-SAC only come from the sparsity of the model? **Q3.** How does RSC-SAC perform in nominal environments compared to non-robust algorithms? **Q4.** Which module is critical in our empirical algorithm? **Q5.** Is RSC-SAC robust to other types of uncertainty and perturbation of the model? **Q6.** How does RSC-SAC balance the trade-off between performance and robustness? We analyze the results and answer these questions below.

R1. RSC-SAC **is robust against spurious correlation.** The results of our proposed method testing with comparisons to baselines are presented in Table 4.8, where the rewards are normalized by the reward for the episode of SAC in the nominal environment. The results reveal that RSC-SAC significantly outperforms other baselines in shifted test environments, exhibiting performance comparable to that of vanilla SAC in the nominal environment in 5 out of 8 tasks.

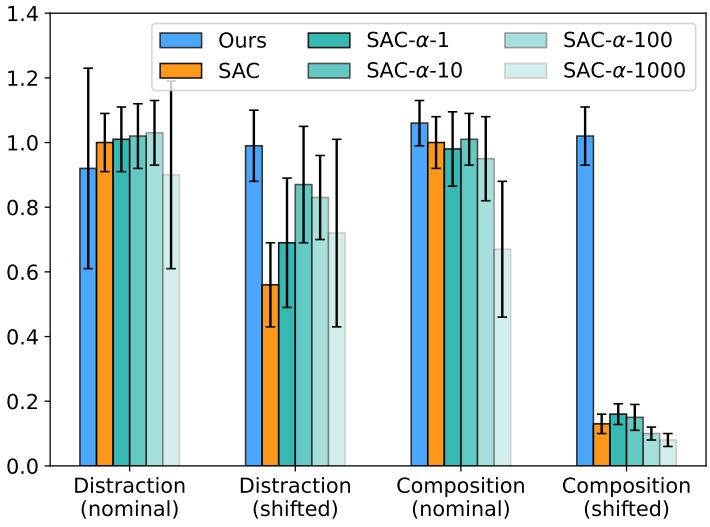


Figure 4.26: Comparison between SAC-Sparse and our method. α is the regularization weight.

An interesting and even surprising finding, as shown in Table 4.8, is that although RMDP-G, RMDP-U, and ATLA are trained desired to be robust against small perturbations, their performance tends to drop more than non-robust SAC in some tasks. This indicates that using the samples generated from the traditional robust algorithms could harm the policy performance when the test environment is outside of the prescribed uncertainty set considered in the robust algorithms.

R2. Sparsity of the model is only one reason for the robustness of RSC-SAC**.** As the existing literature shows (Park et al., 2021), sparsity regularization benefits the elimination of spurious correlation and causal confusion. Therefore, we compare our method with a sparse version of SAC (SAC-Sparse): we add an additional penalty $\alpha|W|_1$ during optimization, where W is the parameter of the first linear layer of the policy and value networks and α is the weight. The results of both *Distraction* and *Composition* are shown in Figure 4.26. We have two important findings based on the results: (1) The sparsity improves the robustness of SAC in the setting of distraction spurious correlation, which is consistent with the findings in (Park et al., 2021). (2) The sparsity does not help with the composition type of spurious correlation, which indicates that purely using sparsity regularization cannot explain the improvement of our RSC-SAC. In fact, the semantic perturbation in our method plays an important role in enhancing the composition

Table 4.9: Testing reward on nominal environments. Underline means the reward is over 0.9.

Method	Brightness	Behavior	Crossing	CarType	Lift	Stack	Wipe	Door
SAC	1.00±0.09	1.00±0.08	1.00±0.02	1.00±0.03	1.00±0.03	1.00±0.09	1.00±0.12	1.00±0.03
RMDP-G	1.04±0.09	1.00±0.11	0.78±0.05	0.79±0.05	0.92±0.07	0.86±0.14	0.99±0.13	0.99±0.06
RMDP-U	1.02±0.09	1.04±0.07	0.90±0.03	0.88±0.03	0.97±0.05	0.92±0.12	0.97±0.14	0.88±0.31
MoCoDA	0.65±0.17	0.78±0.15	0.57±0.07	0.55±0.13	0.79±0.11	0.72±0.08	0.69±0.13	0.41±0.22
ATLA	0.99±0.11	0.98±0.11	0.89±0.05	0.88±0.04	0.94±0.08	0.88±0.10	0.96±0.12	0.97±0.05
DBC	0.75±0.12	0.78±0.10	0.85±0.08	0.86±0.06	0.27±0.04	0.12±0.08	0.31±0.21	0.01±0.01
Active	1.02±0.10	1.08±0.06	1.00±0.02	1.00±0.02	0.99±0.03	0.90±0.12	0.93±0.20	0.99±0.05
RSC-SAC	0.92±0.31	1.06±0.07	0.96±0.03	0.96±0.03	0.96±0.05	1.04±0.08	0.92±0.14	0.98±0.05

generalization.

R3. RSC-SAC **maintains a high performance in nominal environments.** Previous literature (Xu et al., 2023) finds that there usually exists a trade-off between the performance in the nominal environment and the robustness against uncertainty. To evaluate the performance of RSC-SAC in the nominal environment, we perform experiments and summarize the results in Table 4.9, which shows that RSC-SAC still performs well in the training environment. Furthermore, the training curves are shown in Figure 4.25, showing that RSC-SAC achieves similar rewards compared to non-robust SAC, although converges slower than it.

Table 4.10: Influence of modules

Method	Lift	Behavior	Crossing
w/o $\boldsymbol{G}_\phi$	0.79±0.15	0.51±0.24	0.87±0.10
w/o P_θ	0.75±0.13	0.41±0.28	0.89±0.08
w/o P^c	0.90±0.09	0.66±0.21	0.96±0.04
Full model	0.98±0.04	1.02±0.09	1.04±0.02

R4. Both the confounder distribution and the structural causal model are critical. To assess the impact of each module in our algorithm, we perform three additional ablation studies (in Table 4.10), where we remove the causal graph $\boldsymbol{G}_\phi$, the transition model P_θ, and the confounder distribution P^c, respectively. The results demonstrate that the learnable causal graph $\boldsymbol{G}_\phi$ is critical to the performance that improves the prediction of the next state and rewards, thus facilitating the generation of high-quality next states with the current perturbed states. The transition model without $\boldsymbol{G}_\phi$ may still retain numerous spurious correlations, resulting in a performance drop

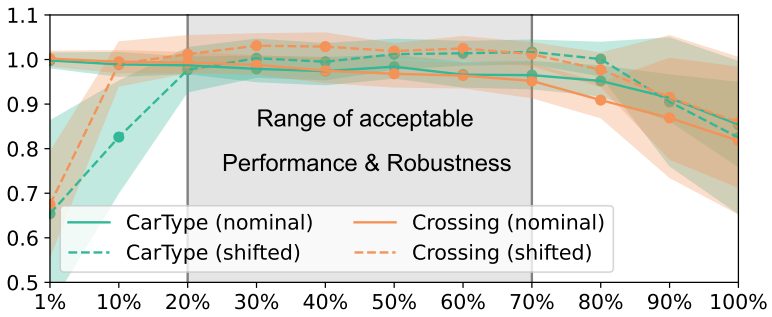


Figure 4.27: Performance-robustness tradeoff with different augmentation ratio β.

similar to that without P_θ, which does not alter the next state and reward. In the third row of Table 4.10, the performance drop indicates that the confounder P^c also plays a crucial role in preserving semantic meaning and avoiding distractions from policy training.

Table 4.11: Random purterbuation

Method	Lift-0	Lift-0.01	Lift-0.1
SAC	1.00±0.03	0.77±0.13	0.46±0.23
RMDP-0.01	0.97±0.05	0.96±0.06	0.51±0.21
RMDP-0.1	0.85±0.12	0.82±0.09	0.39±0.15
RSC-SAC	0.96±0.05	0.94±0.06	0.44±0.18

R5. RSC-SAC **is also robust to random perturbation.** The final investigation aims to assess the generalizability of our method to cope with random perturbation that is widely considered in robust RL (RMDPs). Toward this, we evaluated the proposed algorithm in the test environments added with random noise under the Gaussian distribution with two varying scales in the *Lift* environment. In Table 4.11, *Lift-0* indicates the nominal training environment, while *Lift-0.01* and *Lift-0.1* represent the environments perturbed by Gaussian noise with standard derivation 0.01 and 0.1, respectively. The results indicate that our RSC-SAC achieves comparable robustness compared to RMDP-0.01 in both large and small perturbation settings and outperforms RMDP methods in the nominal training environment.

R6. RSC-SAC **maintains good performance and robustness for a wide range of** β**.** As shown in Figure 4.27, the proposed RSC-SAC performs well in both nominal and shifted settings, maintaining good performance in the nominal setting and achieving robustness for a wide

range of (20%-70%). When the proportion of perturbed data is very small (1%), RSC-SAC almost achieves the same results as vanilla SAC in nominal settings, and there is no robustness in shifted settings. As it increases (considering more robustness), the performance of RSC-SAC in the nominal setting gradually gets worse, while reversely it gets better in the shifted settings (more robust). However, when the ratio is too large ($> 80\%$), the performance of RSC-SAC in both settings degrades a lot, as the policy is too conservative and, therefore, fails in all environments.

Example of Generated Data by Perturbations

We show an example of generated trajectories in the Lift task to demonstrate the reason why our method obtains robustness against spurious correlation. In Figure 4.28 (a), we show the collected trajectories from the data buffer. Since the green block is always generated on the left side of the table, the trajectories of the green block appear mainly on the left side of the table. In Figure 4.28 (b), we generate new trajectories from a trained transition model and observe that the distribution of trajectories follows the collected data. In Figure 4.28 (c), we directly perturbed the dimensions of the state and used the same transition model to generate new trajectories. We find that the generated trajectories blend the color, but fail to maintain the spatial distribution of the original data. In Figure 4.28 (d), we use the causal-based transition model to generate new trajectories and we find that the results not only follow the spatial distribution but also blend the color.

The results shown in Figure 4.28 illustrate that the data generated by our method eliminate the spurious correlation between the color and position of the block, therefore allowing the policy model to generalize to the shifted environment.

4.4 Summary

In this chapter, we discuss three of my previous works that use causal graphs to improve the generation process (Ding et al., 2021c, 2022a, 2024). In the first work, we assume that the causal graph is known by extracting human knowledge. In the second work, we propose a framework

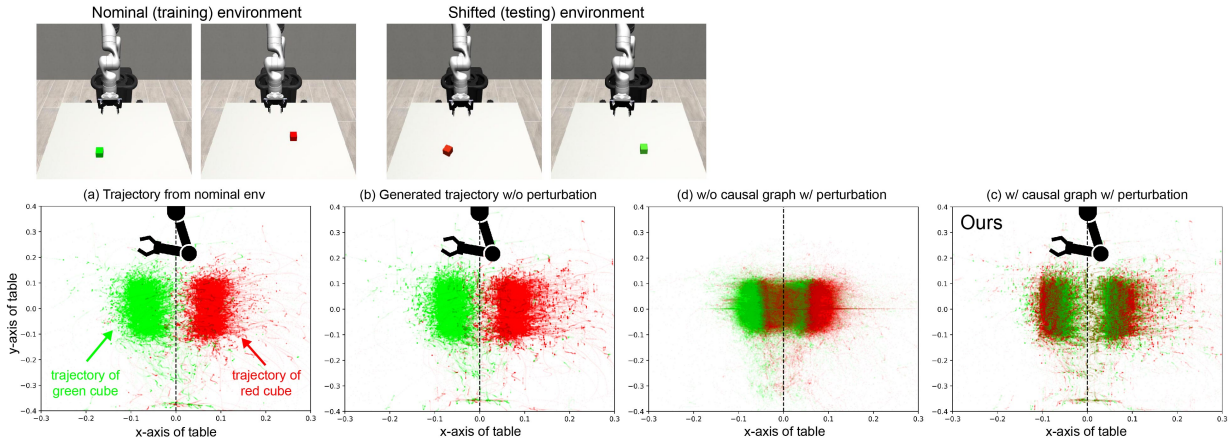


Figure 4.28: The generated transition data from different perturbation methods. (a) Trajectories collected from the policy interact with the nominal environment. (b) Generated trajectories without any perturbation. (c) Generated trajectories with perturbation but without the causal graph. (d) Generated trajectories with perturbation and with the causal graph.

for simultaneously discovering the causal graph and using the graph for generation. The causal graphs embed the interactive knowledge about how one agent influences another, providing two main advantages: (1) making the model learn features that are relevant to the task and ignore irrelevant features, and (2) making the model generalize to unseen situations. In the third work, we further investigate how to use the generative model to break the spurious correlation in state space, which provides a way to use causal discovery to deal with safety-critical scenarios.

One limitation of most of the current causality work still focuses on the cause-and-effect in a short temporal duration, that is, the influence of the last time step on the next time step. The extension of current work to a longer temporal duration to analyze the causality between events will generalize the usage of causality in scenario generation and the digital twin.

Chapter 5

Conclusion

In my Ph.D. research, I proposed several safety-critical scenario generation methods and categorized them into three types. In this chapter, I summarize the important message that the readers can take away, including why safety-critical scenario generation is important, how to select generation algorithms from so many existing methods, and what are future directions to improve existing algorithms.

5.1 Combination of Data, Adversary, and Knowledge

Although I provide a taxonomy to categorize my previous work according to the source of the generation, it is still unclear how to select a specific algorithm according to different situations. The most valuable takeaway from this categorization is that combining information is important for generating critical scenarios because different sources improve different perspectives of generation. Specifically, structured data can be used as prior regularization, downstream tasks can be used as feedback, and human knowledge can provide additional constraints. For example, the following combinations can be used as typical solutions:

- **Data-driven + Adversarial generation**. If the goal is to broadly evaluate your system under diverse scenarios to discover the weakness, combining multi-modal density models and adversarial training is preferred. Randomly sample generative models and search for safety-critical scenarios with adversarial generation.

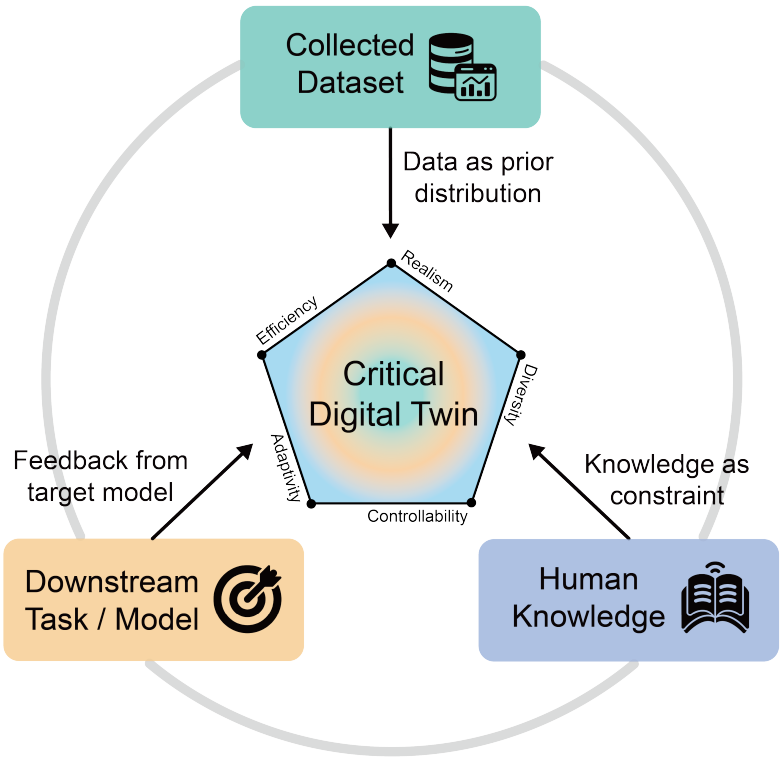


Figure 5.1: Combine as much information as possible to build a critical digital twin.

- **Data-driven + Knowledge-based generation**. When there are specific requirements that can be converted into constraints, using constraint optimization to manipulate existing scenarios (e.g., trajectories) is preferred. In that case, the scenarios will concentrate on one single cluster with low diversity, which is helpful if the objective is to test the driving system on specific scenarios.
- **Adversarial + Knowledge-based generation** If we already have rules that can design safety-critical scenarios but also want to increase the diversity of generated scenarios by automatically learning the parameters, then the combination of adversarial generation and the rule-based method is preferred.

Related tools for the combination. We want to emphasize that traffic scenarios are created by the relations of objects and physical laws, rather than being designed by the human mind. Relying solely on neural networks or optimization methods is not the ultimate solution to generate realistic and critical scenarios. Instead, the model should take advantage of as much external

knowledge and rules as possible to make the scenarios generated interpretable and satisfy the objective. For example, the representation of the scenario is crucial and usually determines the quality of the generation. A representation that naturally embeds rules and laws within a structure could be easily optimized by considering the complex relationship between objects. Additionally, correctly injecting the distribution of real-world data is important to ensure the reality of the generated scenarios. Using Offline RL (Prudencio et al., 2022) and imitation learning could be a potential direction to achieve this goal.

5.2 Future Directions of Critical Digital Twins

Use scenarios to increase robustness and safety Another aspect worth investigating is to effectively use the generated scenarios to improve robustness and safety. The most intuitive way is to train the autonomous system against generated safety-critical scenarios under the adversarial training framework. However, adversarial scenarios usually represent the worst cases, therefore, learning to a robust yet conservative system. It is not easy to select the difficulty and category of scenarios due to the problems of imbalanced data and over-fitting. These problems bridge the topic discussed in this survey to other areas, such as robust optimization (Beyer and Sendhoff, 2007) and distributional robust optimization (Rahimian and Mehrotra, 2019), which have a broad literature to explore.

Use scenarios to improve generalization. In addition to increasing robustness, generated scenarios could also be used to improve the generalization of AVs. For example, gradually training AVs with increasing risk levels under the curriculum learning (Soviany et al., 2021) framework may help systems easily generalize to more types of safety-critical scenarios. A recent survey (Kirk et al., 2021) that investigates the generalization problem in RL emphasizes the importance of environment generation to increase the similarity between the training and testing domains. This direction extends the scenario generation from safety to broader views that require the generation of a goal-conditioned environment.

Knowledge-guided reasoning of safety. In my previous work, I mainly focused on safety-critical scenario generation, where the definition of safety is important throughout the frame-

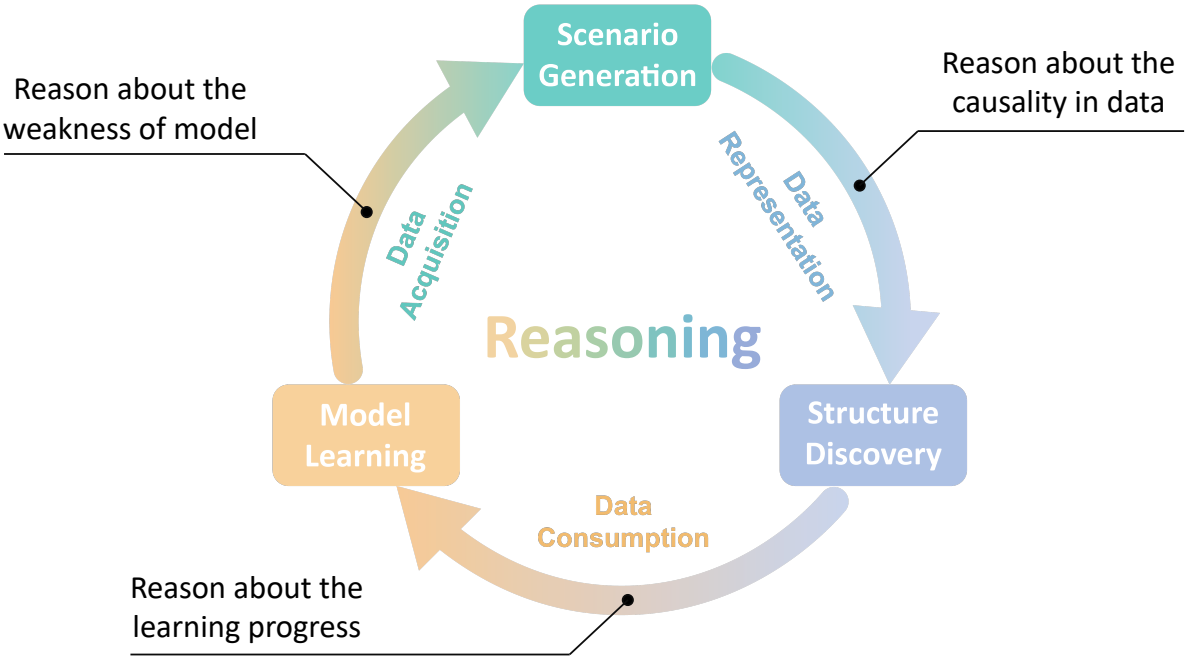


Figure 5.2: Reasoning in the data flywheel.

work. According to a standard criterion for automotive (Heires, 2008), safety is the absence of unreasonable risk due to a hazard caused by functional insufficiencies. However, the definition of an unreasonable risk could be quite difficult and varies case by case. To make autonomous systems work in human society, it is important to align the definition of unreasonable risk with human value, which requires human knowledge. Therefore, it will be an important direction to explore the reasoning capability to discover and categorize risk using powerful tools such as Large Language Models.

The importance of reasoning in the whole data flywheel. In recent years, the concept of data flywheel has attracted more and more attention, as data collection in the physical world could be difficult, but the development of large foundation models requires a large amount of data. The data flywheel is an approach in data science that emphasizes the self-reinforcing effects of collecting and using data, which requires less energy to maintain its speed or even accelerate. As shown in Figure 5.2, we separate the entire data flywheel into three components: data acquisition, data representation, and data consumption. From a high-level point of view, all of these components require reasoning to achieve better performance and greater efficiency. In my Ph.D.

research, I explored the first two components using scenario generation and causal structure discovery, where the first branch investigates the reasoning about the weakness of the model, and the second branch investigates the causality of the data. In the future, I am very interested in the last component, which requires the reasoning of learning progress. For example, the process of using data for parameter-efficient training, fine-tuning, and in-context learning. Finally, critical digital twins will be built upon the data flywheel to enable the prediction and diagnosis of physical systems, providing decision-making guidance to broad applications.

Bibliography

Abdelfattah, M., Yuan, K., Wang, Z. J., and Ward, R. (2021). Towards universal physical attacks on cascaded camera-lidar 3d object detection models. *arXiv preprint arXiv:2101.10747*. 1.3.4

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. (2018). Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*. 4.2.2

Abel, D. (2022). A theory of abstraction in reinforcement learning. *arXiv preprint arXiv:2203.00397*. 4.2, 4.2.1

Abel, D., Arumugam, D., Lehnert, L., and Littman, M. (2018). State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 10–19. PMLR. 4.2, 4.2.1

Abeysirigoonawardena, Y., Shkurti, F., and Dudek, G. (2019). Generating adversarial driving scenarios in high-fidelity simulators. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8271–8277. IEEE. 1.3.4

Acid, S. and de Campos, L. M. (2003). Searching for bayesian network structures in the space of restricted acyclic partially directed graphs. *Journal of Artificial Intelligence Research*, 18:445–490. 1

Addanki, R., Kasiviswanathan, S., Mcgregor, A., and Musco, C. (2020). Efficient intervention design for causal discovery with latents. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 63–73. PMLR. 4.2.3

Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. (2019). Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32. 4.3.3

Ahmed, O., Träuble, F., Goyal, A., Neitz, A., Bengio, Y., Schölkopf, B., Wüthrich, M., and Bauer, S. (2020). Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*. 4.2.4

Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. (2016). Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971. 2.2.3

Allen, B. L., Shin, B. T., and Cooper, P. J. (1978). Analysis of traffic conflicts and collisions. Technical report, McMaster University, Hamilton, Ontario. 1.2.3

Almqvist, S., Hyden, C., and Risser, R. (1991). Use of speed limiters in cars for increased safety and a better environment. *Transportation Research Record*, 1(1318). 1.2.3

Althoff, M. and Lutz, S. (2018). Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE. 1.3.4

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. (2017). Hindsight experience replay. *Advances in neural information processing systems*, 30. 4.2.2

Appenzeller, T. (2017). The scientists' apprentice. *American Association for the Advancement of Science*, 357(6346):16–17. 2.1

Arief, M., Bai, Y., Ding, W., He, S., Huang, Z., Lam, H., and Zhao, D. (2021a). Certifiable deep importance sampling for rare-event simulation of black-box systems. *arXiv preprint arXiv:2111.02204*. 1.3.4

Arief, M., Glynn, P., and Zhao, D. (2018). An accelerated approach to safely and efficiently test pre-production autonomous vehicles on public streets. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2006–2011. IEEE. 1.3.2

Arief, M., Huang, Z., Kumar, G. K. S., Bai, Y., He, S., Ding, W., Lam, H., and Zhao, D. (2020). Deep probabilistic accelerated evaluation: A certifiable rare-event simulation methodology for black-box autonomy. *arXiv preprint arXiv:2006.15722*. 1.3.4

Arief, M., Huang, Z., Kumar, G. K. S., Bai, Y., He, S., Ding, W., Lam, H., and Zhao, D. (2021b). Deep probabilistic accelerated evaluation: A robust certifiable rare-event simulation methodology for black-box safety-critical systems. In *International Conference on Artificial Intelligence and Statistics*, pages 595–603. PMLR. 1.4

Azevedo, C. L., Deshmukh, N. M., Marimuthu, B., Oh, S., Marczuk, K., Soh, H., Basak, K., Toledo, T., Peh, L.-S., and Ben-Akiva, M. E. (2017). Simmobility short-term: An integrated microscopic mobility simulator. *Transportation Research Record*, 2622(1):13–23. 1.2

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*. 2.3.1

Bagschik, G., Menzel, T., and Maurer, M. (2018). Ontology based scene creation for the development of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1813–1820. IEEE. 1.3.2

Bansal, M., Krizhevsky, A., and Ogale, A. (2018). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*. 2.2.3

Barnes, D., Gadd, M., Murcutt, P., Newman, P., and Posner, I. (2020). The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris. 1.1

Bennett, J. (2010). *OpenStreetMap*. Packt Publishing Ltd. 1.3.4

Berthelot, D., Raffel, C., Roy, A., and Goodfellow, I. (2018). Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*. 2.2.3, 2.2, 2.2.3

Best, A., Narang, S., Pasqualin, L., Barber, D., and Manocha, D. (2018). Autonovi-sim: Autonomous vehicle simulation platform with weather, sensing, and traffic control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1048–1056. 1.2

Beyer, H.-G. and Sendhoff, B. (2007). Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33-34):3190–3218. 5.2

Bica, I., Jarrett, D., and van der Schaar, M. (2021). Invariant causal imitation learning for generalizable policies. *Advances in Neural Information Processing Systems*, 34. 4.2, 4.2.4

Bock, J., Krajewski, R., Moers, T., Runde, S., Vater, L., and Eckstein, L. (2020). The ind dataset: A drone dataset of naturalistic road user trajectories at german intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1929–1934. IEEE. (document), 1.1, 1.3.5, 3.16, 3.2.3

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*. 1.3.3, 4.3

Boutilier, C., Dearden, R., and Goldszmidt, M. (2000). Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2):49–107. 4.2.1

Brostow, G. J., Fauqueur, J., and Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97. 1.1

Brouillard, P., Lachapelle, S., Lacoste, A., Lacoste-Julien, S., and Drouin, A. (2020). Differentiable causal discovery from interventional data. *arXiv preprint arXiv:2007.01754*. 4.2.2, 4.2.4

Burgess, C. P., Higgins, I., Pal, A., Matthey, L., Watters, N., Desjardins, G., and Lerchner, A. (2018). Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*. 2.1.1

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631. 1.1, 1.3.5, 2.2, 2.3.3, 2.3.3

Cai, P., Lee, Y., Luo, Y., and Hsu, D. (2020). Summit: A simulator for urban driving in massive mixed traffic. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4023–4029. IEEE. 1.3.2, 1.2, 1.3.5

Camacho, E. F. and Alba, C. B. (2013). *Model predictive control*. Springer science & business media. 4.2.2

Canonne, C. L., Diakonikolas, I., Kane, D. M., and Stewart, A. (2018). Testing conditional independence of discrete distributions. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–57. IEEE. 4.2.2

Cao, Y., Xiao, C., Yang, D., Fang, J., Yang, R., Liu, M., and Li, B. (2019). Adversarial objects against lidar-based autonomous driving systems. *arXiv preprint arXiv:1907.05418*. 1.3.4

Cao, Y., Xu, D., Weng, X., Mao, Z., Anandkumar, A., Xiao, C., and Pavone, M. (2022). Robust trajectory prediction against adversarial attacks. *arXiv preprint arXiv:2208.00094*. 1.3.4

Cérou, F. and Guyader, A. (2007). Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25(2):417–443. 1.3.4

Chalupka, K., Perona, P., and Eberhardt, F. (2018). Fast conditional independence test for vector variables with large sample sizes. *arXiv preprint arXiv:1804.02747*. 4.2.2, 4.2.2, 4.2.2

Chandra, R., Bhattacharya, U., Bera, A., and Manocha, D. (2019). Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8483–8492. 1.1

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019a). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757. 1.3.2, 2.2, 2.2.3

Chang, M.-F., Lambert, J. W., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. (2019b). Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 1.1

Chen, B., Chen, X., Wu, Q., and Li, L. (2021a). Adversarial evaluation of autonomous vehicles in lane-change scenarios. *IEEE Transactions on Intelligent Transportation Systems*. 1.3.4

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017a). Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*. 2.3

Chen, H., Wang, J., Shao, K., Liu, F., Hao, J., Guan, C., Chen, G., and Heng, P.-A. (2023). Traj-mae: Masked autoencoders for trajectory prediction. *arXiv preprint arXiv:2303.06697*.

2.3.3

Chen, J., Yuan, B., and Tomizuka, M. (2019). Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2884–2890. IEEE. 1.3.3

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. (2017b). Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pages 15–26. 3.2

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. *arXiv preprint arXiv:1806.07366*. 1.3.3

Chen, Y., Rong, F., Duggal, S., Wang, S., Yan, X., Manivasagam, S., Xue, S., Yumer, E., and Urtasun, R. (2021b). Geosim: Realistic video simulation via geometry-aware composition for self-driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7230–7240. 1.3.3

Cheng, J., Mei, X., and Liu, M. (2023). Forecast-mae: Self-supervised pre-training for motion forecasting with masked autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8679–8689. 2.3.3

Chevalier-Boisvert, M., Willems, L., and Pal, S. (2018). Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`. 4.2.4

Chickering, D. M. (2002). Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554. 4.2.2

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*. 2.1

Choi, Y., Kim, N., Hwang, S., Park, K., Yoon, J. S., An, K., and Kweon, I. S. (2018). Kaist multi-spectral day/night data set for autonomous and assisted driving. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):934–948. 1.1

Chua, K., Calandra, R., McAllister, R., and Levine, S. (2018). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31. 4.2.4

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. 4.2.2

Contributors, S. R. (2019). Carla Scenario Runner. `https://github.com/carla-simulator/scenario_runner`. 1.3.2, 1.3.5

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223. 1.1

Cranmer, M., Sanchez-Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. (2006). Discovering symbolic models from deep learning with inductive biases (2020). *arXiv preprint arXiv:2006.11287*. 4.2

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. (2019). Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277. 1.3.5, 3.1, 3.2

Cuturi, M. (2013). Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26. 2.3.3

De Haan, P., Jayaraman, D., and Levine, S. (2019). Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32. 4.3

Denker, J., Gardner, W., Graf, H., Henderson, D., Howard, R., Hubbard, W., Jackel, L. D., Baird, H., and Guyon, I. (1988). Neural network recognizer for hand-written zip code digits. *Advances in neural information processing systems*, 1. 2.2.3

Deo, N. and Trivedi, M. M. (2018). Convolutional social pooling for vehicle trajectory prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1468–1476. 2.2.3

Devaranjan, J., Kar, A., and Fidler, S. (2020). Meta-sim2: Unsupervised learning of scene structure for synthetic data generation. In *European Conference on Computer Vision*, pages 715–733. Springer. 1.3.1

Ding, W., Cao, Y., Zhao, D., Xiao, C., and Pavone, M. (2023). Realgen: Retrieval augmented generation for controllable traffic scenarios. *arXiv preprint arXiv:2312.13303*. 2.4

Ding, W., Chen, B., Li, B., Eun, K. J., and Zhao, D. (2021a). Multimodal safety-critical scenarios generation for decision-making algorithms evaluation. *IEEE Robotics and Automation Letters*, 6(2):1551–1558. 3.3, 4.1, 4.1.3

Ding, W., Lin, H., Li, B., Eun, K. J., and Zhao, D. (2021b). Semantically adversarial driving scenario generation with explicit knowledge integration. *arXiv e-prints*, pages arXiv–2106. 2.3, 4.1

Ding, W., Lin, H., Li, B., and Zhao, D. (2021c). Causalaf: Causal autoregressive flow for goal-directed safety-critical scenes generation. *arXiv preprint arXiv:2110.13939*. 4.1, 4.4

Ding, W., Lin, H., Li, B., and Zhao, D. (2022a). Generalizing goal-conditioned reinforcement learning with variational causal reasoning. In *Neural Information Processing Systems (NeurIPS)*. 4.2, 4.3, 4.4

Ding, W., Shi, L., Chi, Y., and Zhao, D. (2024). Seeing is not believing: Robust reinforcement learning against spurious correlation. *Advances in Neural Information Processing Systems*, 36. 4.4

Ding, W., Wang, W., and Zhao, D. (2018). A new multi-vehicle trajectory generator to simulate vehicle-to-vehicle encounters. *arXiv preprint arXiv:1809.05680*. 1.3.3, 2.4, 3.1

Ding, W., Wang, W., and Zhao, D. (2019). Multi-vehicle trajectories generation for vehicle-to-vehicle encounters. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*. 2.2.3

Ding, W., Xu, C., Arief, M., Lin, H., Li, B., and Zhao, D. (2022b). A survey on safety-critical driving scenario generation-a methodological perspective. *CoRR*. 4.1, 4.2.4

Ding, W., Xu, M., and Zhao, D. (2020a). Cmts: Conditional multiple trajectory synthesizer

for generating safety-critical driving scenarios. In *International Conference on Robotics and Automation (ICRA)*. IEEE. 1.3.3, 2.4

Ding, W., Xu, M., and Zhao, D. (2020b). Learning to collide: An adaptive safety-critical scenarios generating method. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 3.2, 3.2.3, 3.2.3, 3.2, 3.3, 4.1, 4.1.3

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2016). Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*. 1.3.3, 3.2.2, 3.2.2

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR. 1.3.2, 1.2, 1.3.5, 3.1.3, 3.2.3, 4.3.5, 4.3.5

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. (2023). Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*. 1

drive Contributors, D. (2021). DI-drive: OpenDILab decision intelligence platform for autonomous driving simulation. `https://github.com/opendilab/DI-drive`. 1.3.2, 1.3.5

Ehrhardt, S., Groth, O., Monszpart, A., Engelcke, M., Posner, I., Mitra, N., and Vedaldi, A. (2020). Relate: Physically plausible multi-object scene synthesis using structured latent spaces. *arXiv preprint arXiv:2007.01272*. 1.3.3

Fang, J., Zhou, D., Yan, F., Zhao, T., Zhang, F., Ma, Y., Wang, L., and Yang, R. (2020). Augmented lidar simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938. 1.3.2

Fawzi, A., Fawzi, H., and Fawzi, O. (2018). Adversarial vulnerability for any classifier. *Advances in neural information processing systems*, 31. 3.2

Feng, L., Li, Q., Peng, Z., Tan, S., and Zhou, B. (2022). Trafficgen: Learning to generate diverse and realistic traffic scenarios. *arXiv preprint arXiv:2210.06609*. 1.3.3

Feng, L., Li, Q., Peng, Z., Tan, S., and Zhou, B. (2023a). Trafficgen: Learning to generate

diverse and realistic traffic scenarios. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3567–3575. IEEE. 1.3.3, 2.3.3

Feng, S., Sun, H., Yan, X., Zhu, H., Zou, Z., Shen, S., and Liu, H. X. (2023b). Dense reinforcement learning for safety validation of autonomous vehicles. *Nature*, 615(7953):620–627. 1.3.4

Feng, S., Yan, X., Sun, H., Feng, Y., and Liu, H. X. (2021). Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment. *Nature communications*, 12(1):1–14. 1.3.4, 4.1

Fernández Llorca, D. and Gómez, E. (2021). Trustworthy autonomous vehicles. Technical report, Joint Research Centre (Seville site). 3

Feydy, J., Séjourné, T., Vialard, F.-X., Amari, S.-i., Trouve, A., and Peyré, G. (2019). Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2681–2690. 2.3.3

Fox, E. B., Sudderth, E. B., Jordan, M. I., and Willsky, A. S. (2008). An hdp-hmm for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pages 312–319. 1.3.2

Frazier, P. I. (2018). A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*. 1.3.4

Fremont, D. J., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., and Seshia, S. A. (2019). Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 63–78. 1.3.1

Fremont, D. J., Kim, E., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., and Seshia, S. A. (2022). Scenic: A language for scenario specification and data generation. *Machine Learning*, pages 1–45. 1.3.5

Fremont, D. J., Kim, E., Pant, Y. V., Seshia, S. A., Acharya, A., Bruso, X., Wells, P., Lemke, S., Lu, Q., and Mehta, S. (2020). Formal scenario-based testing of autonomous vehicles: From simulation to the real world. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8. IEEE. 1.3.2

Fryman, J. and Matthias, B. (2012). Safety of industrial robots: From conventional to collaborative applications. In *ROBOTIK 2012; 7th German Conference on Robotics*, pages 1–5. VDE. 3.2

Garcıa, J. and Fernández, F. (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480. 3.2

Gasse, M., Grasset, D., Gaudron, G., and Oudeyer, P.-Y. (2021). Causal reinforcement learning using observational and interventional data. *arXiv preprint arXiv:2106.14421*. 4.2

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237. 1.1, 1.3.5, 1.3.5

Gershman, S. J. (2017). Reinforcement learning and causal models. *The Oxford handbook of causal reasoning*, 1:295. 4.2

Ghahramani, Z. (1997). Learning dynamic bayesian networks. *International School on Neural Networks, Initiated by IIASS and EMFCSC*, pages 168–197. 1.3.1

Ghodsi, Z., Hari, S. K. S., Frosio, I., Tsai, T., Troccoli, A., Keckler, S. W., Garg, S., and Anandkumar, A. (2021). Generating and characterizing scenarios for safety testing of autonomous vehicles. *arXiv preprint arXiv:2103.07403*. 1.3.4

Gilboa, D., Chang, B., Chen, M., Yang, G., Schoenholz, S. S., Chi, E. H., and Pennington, J. (2019). Dynamical isometry and a mean field theory of lstms and grus. *ArXiv*, abs/1901.08987. 4.2.3, 4.2.3

Girgis, R., Golemo, F., Codevilla, F., Weiss, M., D'Souza, J. A., Kahou, S. E., Heide, F., and Pal, C. (2021). Latent variable sequential set transformers for joint multi-agent motion prediction. *arXiv preprint arXiv:2104.00563*. 2.3.3

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27. 1.3.3, 2.2, 3.1

Görür, D. and Edward Rasmussen, C. (2010). Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25(4):653–664. 2.2.3

Goslin, M. and Mine, M. R. (2004). The panda3d graphics engine. *Computer*, 37(10):112–114. 1.3.5

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773. 2.3.3

Gulino, C., Fu, J., Luo, W., Tucker, G., Bronstein, E., Lu, Y., Harb, J., Pan, X., Wang, Y., Chen, X., et al. (2023). Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *arXiv preprint arXiv:2310.08710*. 2.3

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. (2017). Improved training of wasserstein gans. *Advances in neural information processing systems*, 30. 2.1

Guo, C., Gardner, J., You, Y., Wilson, A. G., and Weinberger, K. (2019a). Simple black-box adversarial attacks. In *International Conference on Machine Learning*, pages 2484–2493. PMLR. 1.4

Guo, Y., Kalidindi, V. V., Arief, M., Wang, W., Zhu, J., Peng, H., and Zhao, D. (2019b). Modeling multi-vehicle interaction scenarios using gaussian random field. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3974–3980. IEEE. 1.3.1

Gupta, G., Rudner, T. G., McAllister, R. T., Gaidon, A., and Gal, Y. (2023). Can active sampling reduce causal confusion in offline reinforcement learning? In *2nd Conference on Causal Learning and Reasoning*. 4.3.5

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. (2020). Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR. 2.3

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR. 11, 3.2.3, 4.2.4, 12, 4.3.5

Håkansson, M. and Wall, J. (2021). Driving scenario generation using generative adversarial networks. *Master Thesis*. 1.3.3

Han, B., Zheng, C., Chan, H., Paster, K., Zhang, M., and Ba, J. (2021). Learning domain invariant

representations in goal-conditioned block mdps. *Advances in Neural Information Processing Systems*, 34. 4.2

Han, J. S. H. S. T. and Zhou, B. (2020). Neuro-symbolic program search for autonomous driving decision module design. In *Conference on Robot Learning (CoRL)*. 4.2

Han, S., Su, S., He, S., Han, S., Yang, H., and Miao, F. (2022). What is the solution for state adversarial multi-agent reinforcement learning? *arXiv preprint arXiv:2212.02705*. 4.3

Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., and Geiger, A. (2022). King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. *arXiv preprint arXiv:2204.13683*. 1.3.4

Harshvardhan, G., Gourisaria, M. K., Pandey, M., and Rautaray, S. S. (2020). A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38:100285. 1.3.3

Hayward, J. C. (1972). Near miss determination through use of a scale of danger. *Pennsylvania State University*. 1.2.3

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. (2022). Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009. 2.3.3

Heires, M. (2008). The international organization for standardization (iso). *New Political Economy*, 13(3):357–367. 5.2

Herman, J., Francis, J., Ganju, S., Chen, B., Koul, A., Gupta, A., Skabelkin, A., Zhukov, I., Kumskoy, M., and Nyberg, E. (2021). Learn-to-race: A multimodal control environment for autonomous racing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9793–9802. 1.2, 1.3.5

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*. 2.1, 2.1.1, 2.1.3

Ho, J. and Ermon, S. (2016). Generative adversarial imitation learning. *Advances in neural*

information processing systems, 29:4565–4573. 1.3.3, 1.3.4

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *arXiv preprint arXiv:2006.11239*. 1.3.3, 2.3

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. 1.3.3, 2.1

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2021). Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169. 2.3.2

Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. (2020). One thousand and one hours: Self-driving motion prediction dataset. *arXiv preprint arXiv:2006.14480*. 1.1

Hu, A., Corrado, G., Griffiths, N., Murez, Z., Gurau, C., Yeo, H., Kendall, A., Cipolla, R., and Shotton, J. (2022). Model-based imitation learning for urban driving. *Advances in Neural Information Processing Systems*, 35:20703–20716. 2.3

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018a). Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087. PMLR. 4.1.1

Huang, X. and Belongie, S. (2017). Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision*, pages 1501–1510. 2.2.2

Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., and Yang, R. (2019). The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2702–2719. 1.1

Huang, Z., Arief, M., Lam, H., and Zhao, D. (2018b). Synthesis of different autonomous vehicles test approaches. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2000–2005. IEEE. 1.3.1

Huang, Z., Guo, Y., Arief, M., Lam, H., and Zhao, D. (2018c). A versatile approach to evaluating and testing automated vehicles based on kernel methods. In *2018 Annual American Control*

Conference (ACC), pages 4796–4802. IEEE. 1.3.4

Huang, Z., Lam, H., and Zhao, D. (2018d). Rare-event simulation without structural information: a learning-based approach. In *2018 Winter Simulation Conference (WSC)*, pages 1826–1837. IEEE. 1.3.4

Huang, Z., Zhao, D., Lam, H., LeBlanc, D. J., and Peng, H. (2017). Evaluation of automated vehicles in the frontal cut-in scenario—an enhanced approach using piecewise mixture models. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 197–202. IEEE. 1.3.4

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. (2017). Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35. 1.3.3

Ivanovic, B., Song, G., Gilitschenski, I., and Pavone, M. (2023). trajdata: A unified interface to multiple human trajectory datasets. *arXiv preprint arXiv:2307.13924*. 2.3.3

Iyengar, G. N. (2005). Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280. 4.3, 4.3.1

Jain, L., Chandrasekaran, V., Jang, U., Wu, W., Lee, A., Yan, A., Chen, S., Jha, S., and Seshia, S. A. (2019). Analyzing and improving neural networks by generating semantic counterexamples through differentiable rendering. *arXiv preprint arXiv:1910.00727*. 1.3.4

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. (2022). Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*. 4.3.5

Jullien, J.-M., Martel, C., Vignollet, L., and Wentland, M. (2009). Openscenario: a flexible integrated environment to develop educational activities based on pedagogical scenarios. In *2009 Ninth IEEE International Conference on Advanced Learning Technologies*, pages 509–513. IEEE. 1.3.5

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. (2019). Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*. 3.2.3

Kalra, N. and Paddock, S. M. (2016). Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193. 3.2

Kantorovich, L. V. (1960). Mathematical methods of organizing and planning production. *Management science*, 6(4):366–422. 1.4

Kar, A., Prakash, A., Liu, M.-Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A., and Fidler, S. (2019). Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4551–4560. 1.3.1, 3.1.1

Ke, N. R., Didolkar, A., Mittal, S., Goyal, A., Lajoie, G., Bauer, S., Rezende, D., Bengio, Y., Mozer, M., and Pal, C. (2021). Systematic evaluation of causal discovery in visual model based reinforcement learning. *arXiv preprint arXiv:2107.00848*. 4.2.4, 4.2.4, 4.2.4

Khirodkar, R., Yoo, D., and Kitani, K. M. (2018). Vadra: Visual adversarial domain randomization and augmentation. *arXiv preprint arXiv:1812.00491*. 1.3.4

Kim, H. and Mnih, A. (2018). Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR. 2.1, 2.1.3, 2.1.4

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2.3.3, 3.1.3

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31. 2.2, 3.1.3

Kingma, D. P., Mohamed, S., Jimenez Rezende, D., and Welling, M. (2014). Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27. 2.2.2

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*. 1.3.3, 2.1, 2.1.1, 2.1.2, 2.2, 2.2.1, 2.2, 3.1, 3.1.1, 3.1.3

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. (2021). A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*. 5.2

Klischat, M. and Althoff, M. (2019). Generating critical test scenarios for automated vehicles

with evolutionary algorithms. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2352–2358. IEEE. 1.3.4

Klischat, M., Liu, E. I., Holtke, F., and Althoff, M. (2020). Scenario factory: Creating safety-critical traffic scenarios for automated vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7. IEEE. 1.3.4

Knies, C. and Diermeyer, F. (2020). Data-driven test scenario generation for cooperative maneuver planning on highways. *Applied Sciences*, 10(22):8154. 1.3.2

Koren, M., Alsaif, S., Lee, R., and Kochenderfer, M. J. (2018). Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE. 1.3.3

Koren, M. and Kochenderfer, M. J. (2019). Efficient autonomy validation in simulation with adaptive stress testing. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4178–4183. IEEE. 1.3.3

Koren, M. and Kochenderfer, M. J. (2020). Adaptive stress testing without domain heuristics using go-explore. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE. 1.3.3

Koren, M., Nassar, A., and Kochenderfer, M. J. (2021). Finding failures in high-fidelity simulation using adaptive stress testing and the backward algorithm. *arXiv preprint arXiv:2107.12940*. 1.3.3

Kou, Y., Peng, H., and Jung, D. (2008). Worst-case evaluation for integrated chassis control systems. *Vehicle System Dynamics*, 46(S1):329–340. 3.1

Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. (2018). The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE. 1.1, 1.3.5

Krajewski, R., Moers, T., Bock, J., Vater, L., and Eckstein, L. (2020). The round dataset: A drone dataset of road user trajectories at roundabouts in germany. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE. 1.1, 1.3.5

Kruber, F., Wurst, J., and Botsch, M. (2018). An unsupervised random forest clustering technique for automatic traffic scenario categorization. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2811–2818. IEEE. 1.3.2

Kruber, F., Wurst, J., Morales, E. S., Chakraborty, S., and Botsch, M. (2019). Unsupervised and supervised learning with the random forest algorithm for traffic scenario clustering and classification. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2463–2470. IEEE. 1.3.2

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86. 1.4

Kuutti, S., Fallah, S., and Bowden, R. (2020). Training adversarial agents to exploit weaknesses in deep control policies. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 108–114. IEEE. 1.3.4, 3.2

Lamsweerde, A. v. (2000). Formal specification: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 147–159. 1.3.2

Landajuela, M., Petersen, B. K., Kim, S., Santiago, C. P., Glatt, R., Mundhenk, N., Pettit, J. F., and Faissol, D. (2021). Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR. 4.2

Larsen, K. G. and Skou, A. (1989). Bisimulation through probabilistic testing (preliminary report). In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 344–352. 4.3.5

Laskin, M., Srinivas, A., and Abbeel, P. (2020). Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR. 4.3.5

Le, H., Jiang, N., Agarwal, A., Dudík, M., Yue, Y., and Daumé III, H. (2018). Hierarchical imitation and reinforcement learning. In *International conference on machine learning*, pages 2917–2926. PMLR. 4.3.5

Lee, H. B., Lee, H., Na, D., Kim, S., Park, M., Yang, E., and Hwang, S. J. (2019). Learning to balance: Bayesian meta-learning for imbalanced and out-of-distribution tasks. *arXiv preprint*

arXiv:1905.12917. 1.3.5, 3.1

Lee, R., Kochenderfer, M. J., Mengshoel, O. J., Brat, G. P., and Owen, M. P. (2015). Adaptive stress testing of airborne collision avoidance systems. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 6C2–1. IEEE. 1.3.3

Leurent, E. (2018). An environment for autonomous driving decision-making. `https://github.com/eleurent/highway-env`. 1.2, 1.3.5, 4.2.4

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*. 4.2, 4.2.2

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474. 2.3

Li, Q., Huang, S., Hong, Y., Chen, Y., Wu, Y. N., and Zhu, S.-C. (2020). Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *International Conference on Machine Learning (ICML)*. 4.2

Li, Q., Peng, Z., Feng, L., Duan, C., Mo, W., Zhou, B., et al. (2023). Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *arXiv preprint arXiv:2306.12241*. 2.3

Li, Q., Peng, Z., Xue, Z., Zhang, Q., and Zhou, B. (2021). Metadrive: Composing diverse driving scenarios for generalizable reinforcement learning. *arXiv preprint arXiv:2109.12674*. 1.2.3, 1.3.2, 1.2, 1.3.5

Li, W., Pan, C., Zhang, R., Ren, J., Ma, Y., Fang, J., Yan, F., Geng, Q., Huang, X., Gong, H., et al. (2019). Aads: Augmented autonomous driving simulation using data-driven algorithms. *Science robotics*, 4(28). 1.3.3

Li, X., Flohr, F., Yang, Y., Xiong, H., Braun, M., Pan, S., Li, K., and Gavrila, D. M. (2016). A new benchmark for vision-based cyclist detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 1028–1033. IEEE. 1.1

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wier-

stra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*. 1.3.4, 3.2.3

Liu, M., Zhu, M., and Zhang, W. (2022). Goal-conditioned reinforcement learning: Problems and solutions. *arXiv preprint arXiv:2201.08299*. 4.2

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE. 1.2, 1.3.5

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*. 1.3.4

Lu, Y., Shen, Y., Zhou, S., Courville, A., Tenenbaum, J. B., and Gan, C. (2021). Learning task decomposition with ordered memory policy network. *arXiv preprint arXiv:2103.10972*. 4.2, 4.3.5

MacKay, D. J. et al. (1998). Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166. 3.2.3

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15. 1.1

Mahmood, A. R., Korenkevych, D., Vasan, G., Ma, W., and Bergstra, J. (2018). Benchmarking reinforcement learning algorithms on real-world robots. In *Conference on robot learning*, pages 561–591. PMLR. 4.3

Mahmud, S. S., Ferreira, L., Hoque, M. S., and Tavassoli, A. (2017). Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs. *IATSS research*, 41(4):153–163. 1.2.3

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*. 2.2

Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., Yang, B., Ma, W.-C.,

and Urtasun, R. (2020). Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11167–11176. 1.3.2

Mao, J., Gan, C., Kohli, P., Tenenbaum, J. B., and Wu, J. (2019). The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*. 1.4

Mao, J., Niu, M., Jiang, C., Liang, H., Chen, J., Liang, X., Li, Y., Ye, C., Zhang, W., Li, Z., et al. (2021). One million scenes for autonomous driving: Once dataset. *arXiv preprint arXiv:2106.11037*. 1.1

Marino, J. and Yue, Y. (2019). An inference perspective on model-based reinforcement learning. In *ICML Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI*. 4.2.2

McDuff, D., Song, Y., Lee, J., Vineet, V., Vemprala, S., Gyde, N., Salman, H., Ma, S., Sohn, K., and Kapoor, A. (2021). Causalcity: Complex simulations with agency for causal discovery and reasoning. *arXiv preprint arXiv:2106.13364*. 1.3.2, 1.2

Menzel, T., Bagschik, G., and Maurer, M. (2018). Scenarios for development, test and validation of automated vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1821–1827. IEEE. 1.3.2

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42. 1.2

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2020). Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer. 1.4

Minderhoud, M. M. and Bovy, P. H. (2001). Extended time-to-collision measures for road traffic safety assessment. *Accident Analysis & Prevention*, 33(1):89–97. 1.2.3

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*. 1.4, 2.1

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR. 1.3.4, 3.2.3

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*. 3.2.3

Moos, J., Hansel, K., Abdulsamad, H., Stark, S., Clever, D., and Peters, J. (2022). Robust reinforcement learning: A review of foundations and recent advances. *Machine Learning and Knowledge Extraction*, 4(1):276–315. 4.3

Müller, M., Casser, V., Lahoud, J., Smith, N., and Ghanem, B. (2018). Sim4cv: A photo-realistic simulator for computer vision applications. *International Journal of Computer Vision*, 126(9):902–919. 1.2

Nair, S., Zhu, Y., Savarese, S., and Fei-Fei, L. (2019). Causal induction from visual observations for goal directed tasks. *arXiv preprint arXiv:1910.01751*. 4.2, 4.2.4

Najm, W. G., Ranganathan, R., Srinivasan, G., Smith, J. D., Toma, S., Swanson, E., Burgett, A., et al. (2013). Description of light-vehicle pre-crash scenarios for safety applications based on vehicle-to-vehicle communications. Technical report, United States. National Highway Traffic Safety Administration. 4.1.3

Najm, W. G., Smith, J. D., Yanagisawa, M., et al. (2007). Pre-crash scenario typology for crash avoidance research. Technical report, United States. National Highway Traffic Safety Administration. 3.1.3, 3.2.3

Nazemi, A. and Fieguth, P. (2019). Potential adversarial samples for white-box attacks. *arXiv preprint arXiv:1912.06409*. 3.2

Neuhold, G., Ollmann, T., Rota Bulo, S., and Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999. 1.1

NHTSA (2023). Nhtsa crash viewer. 2.3.3

Nonnengart, A., Klusch, M., and Müller, C. (2019). Crisgen: Constraint-based generation of critical scenarios for autonomous vehicles. In *International Symposium on Formal Methods*, pages 233–248. Springer. 1.3.4

Norden, J., O'Kelly, M., and Sinha, A. (2019). Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*. 3.2.3

O'Kelly, M., Sinha, A., Namkoong, H., Tedrake, R., and Duchi, J. C. (2018). Scalable end-to-end autonomous vehicle testing via rare-event simulation. *Advances in neural information processing systems*, 31. 1.3.4

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*. 1.3.3

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*. 2.3, 2.3.1

OpenAI (2023). Gpt-4 technical report. 1, 4.3

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*. 1.3.3

Ozbay, K., Yang, H., Bartin, B., and Mudigonda, S. (2008). Derivation and validation of new simulation-based surrogate safety measure. *Transportation research record*, 2083(1):105–113. 1.2.3

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. 1.4

Park, J., Seo, Y., Liu, C., Zhao, L., Qin, T., Shin, J., and Liu, T.-Y. (2021). Object-aware regularization for addressing causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 34:3029–3042. 4.3.5

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance

deep learning library. *Advances in neural information processing systems*, 32. 3.1.3

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR. 3.2.2

Patil, A., Malla, S., Gang, H., and Chen, Y.-T. (2019). The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9552–9557. IEEE. 1.1

Pearl, J. (2009). *Causality*. Cambridge university press. 4.1, 4.3.2, 4.3.2

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of causal inference: foundations and learning algorithms*. The MIT Press. 3.3, 5, 6, 4.2.1, 10, 11, 4.2.1, 4.2.2, 4.3.2

Pham, H. and Zhang, X. (2003). Nhpp software reliability and cost models with testing coverage. *European Journal of Operational Research*, 145(2):443–454. 1.3.2

Pham, Q.-H., Sevestre, P., Pahwa, R. S., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V., and Lin, J. (2020). A* 3d dataset: Towards autonomous driving in challenging environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2267–2273. IEEE. 1.1

Pitis, S., Creager, E., and Garg, A. (2020). Counterfactual data augmentation using locally factored dynamics. *Advances in Neural Information Processing Systems*, 33:3976–3990. 4.2.1

Pitis, S., Creager, E., Mandlekar, A., and Garg, A. (2022). Mocoda: Model-based counterfactual data augmentation. *arXiv preprint arXiv:2210.11287*. 4.3.5

Poggenhans, F., Pauls, J.-H., Janosovits, J., Orf, S., Naumann, M., Kuhnt, F., and Mayr, M. (2018). Lanelet2: A high-definition map framework for the future of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1672–1679. IEEE. 1.3.5

Prakash, A., Boochoon, S., Brophy, M., Acuna, D., Cameracci, E., State, G., Shapira, O., and Birchfield, S. (2019). Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *2019 International Conference on Robotics and Automation (ICRA)*,

pages 7249–7255. IEEE. 1.3.4

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. (2022). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*. 5.1

Rahimian, H. and Mehrotra, S. (2019). Distributionally robust optimization: A review. *arXiv preprint arXiv:1908.05659*. 5.2

Rana, A. and Malhi, A. (2021). Building safer autonomous agents by leveraging risky driving behavior knowledge. *arXiv preprint arXiv:2103.10245*. 1.3.2

Ray, A., Achiam, J., and Amodei, D. (2019). Benchmarking safe exploration in deep reinforcement learning. arxiv. *arXiv preprint arXiv:1910.01708*, 7. 3.2.3

Rempe, D., Philion, J., Guibas, L. J., Fidler, S., and Litany, O. (2022). Generating useful accident-prone driving scenarios via a learned traffic prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17305–17315. 1.3.4, 4.1.3

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR. 2.2.1

Richards, A. G. (2005). *Robust constrained model predictive control*. PhD thesis, Massachusetts Institute of Technology. 4.2.2

Richter, S. R., Hayder, Z., and Koltun, V. (2017). Playing for benchmarks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2213–2222. 1.1, 1.2

Robicquet, A., Sadeghian, A., Alahi, A., and Savarese, S. (2016). Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, pages 549–565. Springer. 1.1

Rocklage, E., Kraft, H., Karatas, A., and Seewig, J. (2017). Automated scenario generation for regression testing of autonomous vehicles. In *2017 ieee 20th international conference on intelligent transportation systems (itsc)*, pages 476–483. IEEE. 1.3.4

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on*

computer vision and pattern recognition, pages 10684–10695. 1

Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., et al. (2020). Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE. 1.2, 1.3.5

Ros, G., Koltun, V., Codevilla, F., and Lopez, A. (2022). Carla autonomous driving challenge. [EB/OL]. `https://carlachallenge.org/`. 1.2.3, 1.3.2

Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243. 1.1

Ross, S., Gordon, G. J., and Bagnell, J. A. (2011). No-regret reductions for imitation learning and structured prediction. In *In AISTATS*. Citeseer. 4.2.4

Rubinstein, R. Y. and Kroese, D. P. (2004). *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer. 3.2

Ruiz, N., Schulter, S., and Chandraker, M. (2018). Learning to simulate. *arXiv preprint arXiv:1810.02513*. 1.3.4

Samak, T. V., Samak, C. V., and Xie, M. (2021). Autodrive simulator: A simulator for scaled autonomous vehicle research and education. *arXiv preprint arXiv:2103.10030*. 1.2

Savkin, A., Ellouze, R., Navab, N., and Tombari, F. (2021). Unsupervised traffic scene generation with synthetic 3d scene graphs. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1229–1235. IEEE. 1.3.1

Scanlon, J. M., Kusano, K. D., Daniel, T., Alderson, C., Ogle, A., and Victor, T. (2021). Waymo simulated driving behavior in reconstructed fatal crashes within an autonomous vehicle operating domain. 1.3.2

Schlichtkrull, M., Kipf, T. N., Bloem, P., Berg, R. v. d., Titov, I., and Welling, M. (2018). Model-

ing relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer. 4.2.4

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117. 2.3.1

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*. 1.3.3, 3.2.3

Schwarting, W., Seyde, T., Gilitschenski, I., Liebenwein, L., Sander, R., Karaman, S., and Rus, D. (2020). Deep latent competition: Learning to race using visual control policies in latent space. In *Conference on Robot Learning*. 1.2

Seitzer, M., Schölkopf, B., and Martius, G. (2021). Causal influence detection for improving efficiency in reinforcement learning. *Advances in Neural Information Processing Systems*, 34. 4.2, 4.2.1, 4.2.1

Shafahi, A., Huang, W. R., Studer, C., Feizi, S., and Goldstein, T. (2018). Are adversarial examples inevitable? *arXiv preprint arXiv:1809.02104*. 3.2

Shah, R. D. and Peters, J. (2020). The hardness of conditional independence testing and the generalised covariance measure. *The Annals of Statistics*, 48(3):1514–1538. 4.2.2

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer. 1.2, 1.3.5

Shanahan, M. and Mitchell, M. (2022). Abstraction for deep reinforcement learning. *arXiv preprint arXiv:2202.05839*. 4.2, 4.2.1

Shi, L. and Chi, Y. (2022). Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity. *arXiv preprint arXiv:2208.05767*. 4.3, 4.3.1

Shi, S., Jiang, L., Dai, D., and Schiele, B. (2022). Motion transformer with global intention localization and local movement refinement. *Advances in Neural Information Processing Systems*, 35:6531–6543. 2.3.2

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser,

J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489. 4.3

Sinha, A., O'Kelly, M., Tedrake, R., and Duchi, J. C. (2020). Neural bridge sampling for evaluating safety-critical autonomous systems. *Advances in Neural Information Processing Systems*, 33:6402–6416. 1.3.4

Sodhani, S., Levine, S., and Zhang, A. (2022). Improving generalization with approximate factored value functions. In *ICLR2022 Workshop on the Elements of Reasoning: Objects, Structure and Causality*. 4.2

Sohn, K., Lee, H., and Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491. 1.4, 2.2.2, 2.2.3

Song, J., Meng, C., and Ermon, S. (2020). Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*. 1.3.3

Sontakke, S. A., Mehrjou, A., Itti, L., and Schölkopf, B. (2021). Causal curiosity: Rl agents discovering self-supervised experiments for causal representation learning. In *International Conference on Machine Learning*, pages 9848–9858. PMLR. 4.2

Soviany, P., Ionescu, R. T., Rota, P., and Sebe, N. (2021). Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*. 5.2

Spirtes, P., Glymour, C. N., Scheines, R., and Heckerman, D. (2000). *Causation, prediction, and search*. MIT press. 4.2.1

Spirtes, P. and Zhang, K. (2016). Causal discovery and inference: concepts and recent methodological advances. In *Applied informatics*, volume 3, pages 1–28. SpringerOpen. 1.4

Sun, H., Feng, S., Yan, X., and Liu, H. X. (2021). Corner case generation and analysis for safety assessment of autonomous vehicles. *arXiv preprint arXiv:2102.03483*. 1.3.4

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern*

Recognition, pages 2446–2454. 1.1

Sun, S.-H., Wu, T.-L., and Lim, J. J. (2019). Program guided agent. In *International Conference on Learning Representations*. 4.2

Suo, S., Regalado, S., Casas, S., and Urtasun, R. (2021). Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409. 1.3.3

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press. 1.3.4, 4.2

Tan, K. L., Esfandiari, Y., Lee, X. Y., and Sarkar, S. (2020). Robustifying reinforcement learning agents via action space adversarial training. In *2020 American control conference (ACC)*, pages 3959–3964. IEEE. 4.3

Tan, S., Ivanovic, B., Weng, X., Pavone, M., and Kraehenbuehl, P. (2023). Language conditioned traffic generation. *arXiv preprint arXiv:2307.07947*. 1.3.3, 2.3, 2.3.3, 2.3.3, 2.3.3, 2.3.3

Tan, S., Wong, K., Wang, S., Manivasagam, S., Ren, M., and Urtasun, R. (2021). Scenegen: Learning to generate realistic traffic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 892–901. 1.3.3

Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P. P., Barron, J. T., and Kretzschmar, H. (2022). Block-nerf: Scalable large scene neural view synthesis. *arXiv preprint arXiv:2202.05263*. 1.4

Tavares, Z., Koppel, J., Zhang, X., Das, R., and Solar-Lezama, A. (2021). A language for counterfactual generative models. In *International Conference on Machine Learning*, pages 10173–10182. PMLR. 4.2.4

team, A. (2018a). Apollo Simulation. `http://apollo.auto/platform/simulation.html`. 1.2

team, D. (2018b). Deepdrive Simulation. `https://deepdrive.io/`. 1.2

team, E. (2018c). Environment Simulator Minimalistic (esmini). `https://github.com/esmini/esmini`. 1.2

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. (2021). Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*. 1.2.1

Team, O. G. (2016). Openai gym car racing. 1.2

team, S. (2018d). SUMO NETEDIT. `https://sumo.dlr.de/docs/Netedit/index.html`. 1.3.5

team, U. (2018e). Udacity Dataset. `https://github.com/udacity/self-driving-car/tree/master/datasets`. 1.1

Tessler, C., Efroni, Y., and Mannor, S. (2019). Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*, pages 6215–6224. PMLR. 4.3

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. (2017). Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE. 1.3.4

Tomar, M., Zhang, A., Calandra, R., Taylor, M. E., and Pineau, J. (2021). Model-invariant state abstractions for model-based reinforcement learning. *arXiv preprint arXiv:2102.09850*. 4.2

Tsamardinos, I., Brown, L. E., and Aliferis, C. F. (2006). The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78. 4.2.4

Tu, J., Ren, M., Manivasagam, S., Liang, M., Yang, B., Du, R., Cheng, F., and Urtasun, R. (2020). Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725. 1.3.4

Van Den Oord, A., Vinyals, O., et al. (2017). Neural discrete representation learning. *Advances in neural information processing systems*, 30. 2.2

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11). 2.2.3

van der Made, R., Tideman, M., Lages, U., Katz, R., and Spencer, M. (2015). Automated generation of virtual driving scenarios from test drive data. In *24th International Technical Conference on the Enhanced Safety of Vehicles (ESV) National Highway Traffic Safety Administration*, pages 15–0268. 1.3.2

Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. (2016). Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR. 1.3.3

Van Ratingen, M., Williams, A., Lie, A., Seeck, A., Castaing, P., Kolke, R., Adriaenssens, G., and Miller, A. (2016). The european new car assessment programme: a historical review. *Chinese journal of traumatology*, 19(2):63–69. 4.1.3

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. 2.3.1

Villani, C. et al. (2009). *Optimal transport: old and new*, volume 338. Springer. 2.3.1, 2.3.3

Volodin, S., Wichers, N., and Nixon, J. (2020). Resolving spurious correlations in causal models of environments via interventions. *arXiv preprint arXiv:2002.05217*. 4.2

Wachi, A. (2019). Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving. *arXiv preprint arXiv:1903.10654*. 1.3.4

Wang, H., Xu, M., Zhu, F., Deng, Z., Li, Y., and Zhou, B. (2018). Shadow traffic: A unified model for abnormal traffic behavior simulation. *Computers & Graphics*, 70:235–241. 3.1

Wang, J., Pun, A., Tu, J., Manivasagam, S., Sadat, A., Casas, S., Ren, M., and Urtasun, R. (2021a). Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918. 1.3.4

Wang, L., Yang, Z., and Wang, Z. (2021b). Provably efficient causal reinforcement learning with confounded observational data. *Advances in Neural Information Processing Systems*, 34:21164–21175. 4.3.2

Wang, S. X. (2001). *Maximum weighted likelihood estimation*. PhD thesis, University of British Columbia. 3.2, 3.2.2

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. (2019a). Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*. 4.2.2

Wang, W., Liu, C., and Zhao, D. (2017). How much data are enough? a statistical approach with case study on longitudinal driving behavior. *IEEE Transactions on Intelligent Vehicles*, 2(2):85–98. 2.1, 2.1.4

Wang, W. and Zhao, D. (2018). Extracting traffic primitives directly from naturalistically logged data for self-driving applications. *IEEE Robotics and Automation Letters*, 3(2):1223–1229. 1.3.1, 1.3.2, 2.1

Wang, X., Krasowski, H., and Althoff, M. (2021c). Commonroad-rl: a configurable reinforcement learning environment for motion planning of autonomous vehicles. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 466–472. IEEE. 1.3.4, 1.3.5

Wang, X., Peng, H., and Zhao, D. (2019b). Combining reachability analysis and importance sampling for accelerated evaluation of highly automated vehicles at pedestrian crossing. In *Dynamic Systems and Control Conference*, volume 59162, page V003T18A011. American Society of Mechanical Engineers. 1.3.4

Wang, Y., Su, Z., Guo, S., Dai, M., Luan, T. H., and Liu, Y. (2023). A survey on digital twins: architecture, enabling technologies, security and privacy, and future prospects. *IEEE Internet of Things Journal*. 1

Wang, Z., Xiao, X., Zhu, Y., and Stone, P. (2021d). Task-independent causal state abstraction. *Workshop on Robot Learning: Self-Supervised and Lifelong Learning, NeurIPS*. 4.2, 4.2.1, 4.2.4

Webb, N., Smith, D., Ludwick, C., Victor, T., Hommes, Q., Favaro, F., Ivanov, G., and Daniel, T. (2020). Waymo's safety methodologies and safety readiness determinations. *arXiv preprint arXiv:2011.00054*. 1.3.2

Wen, M., Park, J., and Cho, K. (2020). A scenario generation pipeline for autonomous vehicle simulators. *Human-centric Computing and Information Sciences*, 10:1–15. 1.3.3

Wheeler, T. A. and Kochenderfer, M. J. (2016). Factor graph scene distributions for automotive safety analysis. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1035–1040. IEEE. 1.3.1, 3.1, 3.1.1

Wheeler, T. A. and Kochenderfer, M. J. (2019). Critical factor graph situation clusters for accelerated automotive safety validation. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2133–2139. IEEE. 1.3.2

Wheeler, T. A., Kochenderfer, M. J., and Robbel, P. (2015). Initial scene configurations for highway traffic propagation. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 279–284. IEEE. 1.3.1

Wierstra, D., Schaul, T., Glasmachers, T., Sun, Y., Peters, J., and Schmidhuber, J. (2014). Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980. 3.2

Wiesemann, W., Kuhn, D., and Rustem, B. (2013). Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183. 4.3.1

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256. 1.3.4, 11, 13

Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., et al. (2023). Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*. 1.1, 1.3.5

Winkler, C., Worrall, D., Hoogeboom, E., and Welling, M. (2019). Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*. 3.2.2

Wu, P., Majumdar, A., Stone, K., Lin, Y., Mordatch, I., Abbeel, P., and Rajeswaran, A. (2023). Masked trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*. 2.3.3

Wulfe, B., Chintakindi, S., Choi, S.-C. T., Hartong-Redden, R., Kodali, A., and Kochenderfer, M. J. (2018). Real-time prediction of intermediate-horizon automotive collision risk. *arXiv*

preprint arXiv:1802.01532. 1.3.2

Wymann, B., Espié, E., Guionneau, C., Dimitrakakis, C., Coulom, R., and Sumner, A. (2000). Torcs, the open racing car simulator. *Software available at http://torcs. sourceforge. net*, 4(6):2. 1.2

Xiao, A., Huang, J., Guan, D., Zhan, F., and Lu, S. (2021a). Synlidar: Learning from synthetic lidar sequential point cloud for semantic segmentation. *arXiv preprint arXiv:2107.05399*. 1.3.1

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. (2018). Spatially transformed adversarial examples. *arXiv preprint arXiv:1801.02612*. 1.4

Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., et al. (2021b). Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE. 1.1

Xu, C., Ding, W., Lyu, W., Liu, Z., Wang, S., He, Y., Hu, H., Zhao, D., and Li, B. (2022). Safebench: A benchmarking platform for safety evaluation of autonomous vehicles. In *Neural Information Processing Systems (NeurIPS)*. 1.3.2, 1.3.5

Xu, K., Liu, S., Zhao, P., Chen, P.-Y., Zhang, H., Fan, Q., Erdogmus, D., Wang, Y., and Lin, X. (2018). Structured adversarial attack: Towards general implementation and better interpretability. *arXiv preprint arXiv:1808.01664*. 1.4

Xu, K., Srivastava, A., Gutfreund, D., Sosa, F., Ullman, T., Tenenbaum, J., and Sutton, C. (2021a). A bayesian-symbolic approach to reasoning and learning in intuitive physics. *Advances in Neural Information Processing Systems*, 34. 4.2

Xu, M., Huang, P., Li, F., Zhu, J., Qi, X., Oguchi, K., Huang, Z., Lam, H., and Zhao, D. (2021b). Accelerated policy evaluation: Learning adversarial environments with adaptive importance sampling. *arXiv preprint arXiv:2106.10566*. 1.3.4

Xu, M., Huang, P., Niu, Y., Kumar, V., Qiu, J., Fang, C., Lee, K.-H., Qi, X., Lam, H., Li, B., et al. (2023). Group distributionally robust reinforcement learning with hierarchical latent variables. In *International Conference on Artificial Intelligence and Statistics*, pages 2677–2703. PMLR.

12, 4.3.5

Yan, M., Li, A., Kalakrishnan, M., and Pastor, P. (2019). Learning probabilistic multi-modal actor models for vision-based robotic grasping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 4804–4810. IEEE. 3.2

Yang, K. D., Katoff, A., and Uhler, C. (2018a). Characterizing and learning equivalence classes of causal dags under interventions. In *ICML*. 1

Yang, S., Wang, W., Liu, C., and Deng, W. (2018b). Scene understanding in deep learning-based end-to-end controllers for autonomous vehicles. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):53–63. 2.1

Yang, W., Zhang, L., and Zhang, Z. (2022). Toward theoretical understandings of robust markov decision processes: Sample complexity and asymptotics. *The Annals of Statistics*, 50(6):3223–3248. 4.3

Yang, Y., Inala, J. P., Bastani, O., Pu, Y., Solar-Lezama, A., and Rinard, M. (2021). Program synthesis guided reinforcement learning for partially observed environments. *Advances in Neural Information Processing Systems*, 34. 4.2

Yang, Y., Zhang, Q., Gilles, T., Batool, N., and Folkesson, J. (2023). Rmp: A random mask pretrain framework for motion prediction. *arXiv preprint arXiv:2309.08989*. 2.3.3

Yang, Z., Chai, Y., Anguelov, D., Zhou, Y., Sun, P., Erhan, D., Rafferty, S., and Kretzschmar, H. (2020). Surfelgan: Synthesizing realistic sensor data for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11118–11127. 1.3.3

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. (2021). Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*. 4.3.5

Yin, P., Xu, L., Li, X., Yin, C., Li, Y., Srivatsan, R. A., Li, L., Ji, J., and He, Y. (2019). A multi-domain feature learning method for visual place recognition. In *2019 International conference on robotics and automation (ICRA)*, pages 319–324. IEEE. 2.2.2

Yoo, M., Cho, S., and Woo, H. (2022). Skills regularized task decomposition for multi-task of-

fline reinforcement learning. *Advances in Neural Information Processing Systems*, 35:37432–37444. 4.3.5

Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. (2020). Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645. 1.1

Zhan, W., Sun, L., Wang, D., Shi, H., Clausse, A., Naumann, M., Kümmerle, J., Königshof, H., Stiller, C., de La Fortelle, A., and Tomizuka, M. (2019). INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps. *arXiv:1910.03088 [cs, eess]*. 1.1, 1.3.5

Zhang, A., Lyle, C., Sodhani, S., Filos, A., Kwiatkowska, M., Pineau, J., Gal, Y., and Precup, D. (2020a). Invariant causal prediction for block mdps. In *International Conference on Machine Learning*, pages 11214–11224. PMLR. 4.2

Zhang, A., McAllister, R., Calandra, R., Gal, Y., and Levine, S. (2020b). Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*. 4.3.5

Zhang, H., Chen, H., Boning, D., and Hsieh, C.-J. (2021). Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452*. 4.3, 4.3, 4.3.5

Zhang, H., Chen, H., Xiao, C., Li, B., Liu, M., Boning, D., and Hsieh, C.-J. (2020c). Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems*, 33:21024–21037. 4.3

Zhang, Q., Hu, S., Sun, J., Chen, Q. A., and Mao, Z. M. (2022). On adversarial robustness of trajectory prediction for autonomous vehicles. *arXiv preprint arXiv:2201.05057*. 1.3.4, 4.1

Zhao, D., Huang, X., Peng, H., Lam, H., and LeBlanc, D. J. (2017). Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):733–744. 3.1

Zhao, D., Huang, X., Peng, H., Lam, H., and LeBlanc, D. J. (2018). Accelerated evaluation of automated vehicles in car-following maneuvers. *IEEE Transactions on Intelligent Transporta-*

tion Systems, 19(3):733–744. 1.3.4

Zhao, D., Lam, H., Peng, H., Bao, S., LeBlanc, D. J., Nobukawa, K., and Pan, C. S. (2016). Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques. *IEEE transactions on intelligent transportation systems*, 18(3):595–607. 1.3.4, 3.1

Zhao, D., Peng, H., Lam, H., Bao, S., Nobukawa, K., LeBlanc, D. J., and Pan, C. S. (2015). Accelerated evaluation of automated vehicles in lane change scenarios. In *ASME 2015 Dynamic Systems and Control Conference*, pages V001T17A002—-V001T17A002. American Society of Mechanical Engineers. 1.3.4

Zhao, Z., Samel, K., Chen, B., et al. (2021). Proto: Program-guided transformer for program-guided tasks. *Advances in Neural Information Processing Systems*, 34. 4.2

Zhong, Z., Rempe, D., Chen, Y., Ivanovic, B., Cao, Y., Xu, D., Pavone, M., and Ray, B. (2023a). Language-guided traffic simulation via scene-level diffusion. *arXiv preprint arXiv:2306.06344*. 1.3.3, 2.3

Zhong, Z., Rempe, D., Xu, D., Chen, Y., Veer, S., Che, T., Ray, B., and Pavone, M. (2023b). Guided conditional diffusion for controllable traffic simulation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3560–3566. IEEE. 2.3, 2.3.3

Zhou, M., Luo, J., Villella, J., Yang, Y., Rusu, D., Miao, J., Zhang, W., Alban, M., Fadakar, I., Chen, Z., et al. (2020). Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. *arXiv preprint arXiv:2010.09776*. 1.3.2, 1.2, 1.3.5

Zhu, B., Liu, J. Z., Cauley, S. F., Rosen, B. R., and Rosen, M. S. (2018). Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492. 2.2

Zhu, Y., Miao, C., Hajiaghajani, F., Huai, M., Su, L., and Qiao, C. (2021). Adversarial attacks against lidar semantic segmentation in autonomous driving. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 329–342. 1.3.4

Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Nasiriany, S., and Zhu, Y. (2020). robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*. 4.3.5, 4.3.5

Zhu, Z.-M., Chen, X.-H., Tian, H.-L., Zhang, K., and Yu, Y. (2022). Offline reinforcement learning with causal structured world models. *arXiv preprint arXiv:2206.01474*. 4.2.4

Zyner, A., Worrall, S., and Nebot, E. M. (2019). Acfr five roundabouts dataset: Naturalistic driving at unsignalized intersections. *IEEE Intelligent Transportation Systems Magazine*, 11(4):8–18. 1.1