Learning to Combine Top-Down and Bottom-Up Signals in Recurrent Neural Networks with Attention over Modules

Sarthak Mittal [1] **Alex Lamb** [2] **Anirudh Goyal** [2] **Vikram Voleti** [1 2] **Murray Shanahan** [3] **Guillaume Lajoie** [1 2]
Michael Mozer [4] **Yoshua Bengio** [1 2]

Abstract

Robust perception relies on both bottom-up and top-down signals. Bottom-up signals consist of what's directly observed through sensation. Top-down signals consist of beliefs and expectations based on past experience and the current reportable short-term memory, such as how the phrase 'peanut butter and ...' will be completed. The optimal combination of bottom-up and top-down information remains an open question, but the manner of combination must be dynamic and both context and task dependent. To effectively utilize the wealth of potential top-down information available, and to prevent the cacophony of intermixed signals in a bidirectional architecture, mechanisms are needed to restrict information flow. We explore deep recurrent neural net architectures in which bottom-up and top-down signals are dynamically combined using attention. Modularity of the architecture further restricts the sharing and communication of information. Together, attention and modularity direct information flow, which leads to reliable performance improvements in perceptual and language tasks, and in particular improves robustness to distractions and noisy data. We demonstrate on a variety of benchmarks in language modeling, sequential image classification, video prediction and reinforcement learning that the *bidirectional* information flow can improve results over strong baselines.

1. Introduction

Deep learning emerged from the goal of learning representational hierarchies, with the higher levels corresponding to abstract features used for decision-making (Hinton et al., 2006; Bengio et al., 2007; Salakhutdinov & Hinton, 2009). Hierarchical models are often taken to imply that computation proceeds in a feedforward or *bottom up* fashion, i.e., stage-wise information processing in which low-level (sensory) representations construct or modulate high level (conceptual) representations. However, they could equally well support the flow of information in a feedback or *top down* fashion, i.e., information processing in which high-level representations modulate lower-level representations. Neuroscientists have noted that reciprocity of connectivity among anatomically distinct areas of neocortex is common (Felleman & Van Essen, 1991; Rockland, 2015), causing early visual areas to be modulated by later stages of processing (Bastos et al., 2015). The same is true for other sensory modalities as well (Manita et al., 2015). Neuroimaging research has found evidence for distinct bidirectional activity flows with functional consequences (Dijkstra et al., 2017; Nielsen et al., 1999), and neurophysiological and neuroanatomical studies indicate the important role of top-down information for guiding processing resources to behaviorally relevant events (Baluch & Itti, 2011; Gilbert, 2013). The Global Workspace theory (Baars, 1997; Dehaene et al., 2017) posits that top-down signals are necessary to broadcast information throughout neocortex, which enables conscious states and verbal behavior. Nonetheless, the neural mechanisms of interaction between bottom-up and top-down pathways are still poorly understood. One goal of our research is to conduct machine-learning experiments to explore the value of top-down mechanisms for learning, achieving robustness to distributional shifts, and guiding the development of novel and improved neural architectures.

In the cognitive science community, the relative contributions of bottom-up and top-down signals has been an ongoing subject of debate for over 40 years (Kinchla & Wolfe, 1979; Rauss & Pourtois, 2013). The McClelland & Rumelhart (1981) model of printed-word reading consisted of a hierarchy of detectors, from letter fragments to letters to words, with bidirectional connectivity. The top-down connectivity imposed orthographic constraints of the vocabulary and helped to explain human proficiency in reading. The model also accounted for puzzling behavioral phenomena,

[1]MILA [2]Université de Montréal [3]Imperial College London [4]Google Research, Brain Team. Correspondence to: Anirudh Goyal <anirudhgoyal9119@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020.

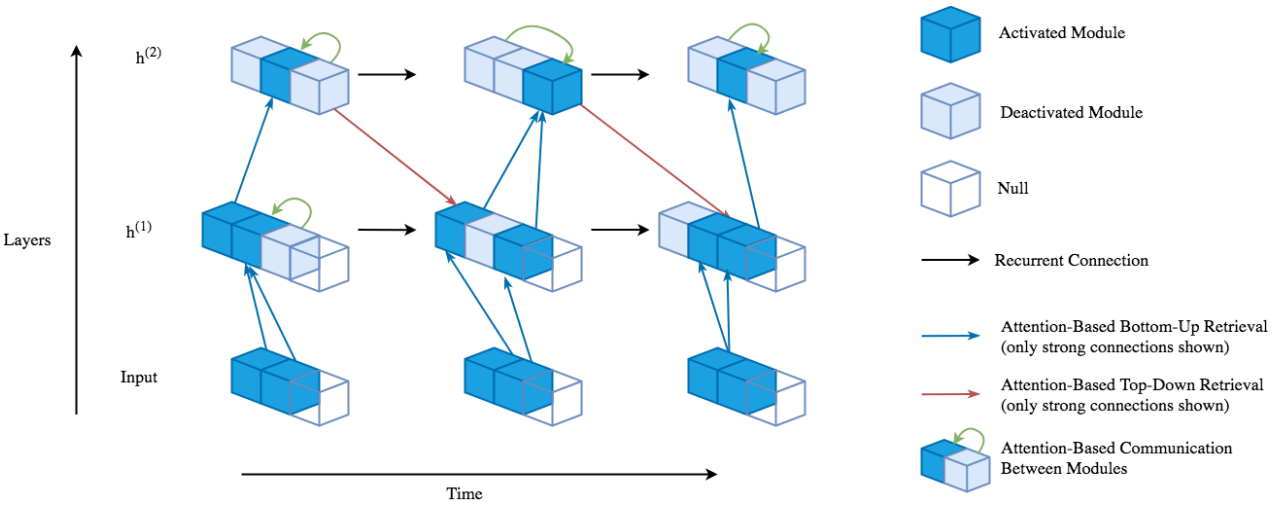

Figure 1. Overall layout of the proposed Bidirectional Recurrent Independent Mechanisms (BRIMs) model. Information is passed forward in time using recurrent connections and information passed between layers using attention. Modules attend to both the lower layer on the current time step as well as the higher layer on the previous time step, along with *null*, which refers to a vector of zeros.

such as the fact that it is easier for people to perceive a letter embedded in a word—such as the A in READ—than the same letter presented in isolation. In the model, partial activation from representations of the word READ provide top-down support for the activation of the letter A in the context of the word, but not of the isolated A.

If top-down pathways provide expectations, they can be considered as priors on subsequent inputs, and uncertainty plays a key role in how information is combined from bottom-up and top-down sources (Weiss et al., 2002; Kersten et al., 2004). For instance, if one enters a familiar but dark room, one's expectations will guide perception and behavior, whereas expectations are less relevant in the light of day. Even without prior expectations, top-down pathways can leverage simultaneous contextual information to resolve local uncertainty, e.g., the interpretation of the middle letters of these two words: THE CAT. Thus, top-down processing leverages goals, temporal and concurrent context, and expectations (conscious or otherwise) to steer or focus the interpretation of sensory signals. Top-down processing helps to overcome intrinsically noisy or ambiguous sensory data, whereas bottom-up processing allows an agent to be reactive to unexpected or surprising stimuli.

Information Flow in Deep Neural Networks. Models having within-layer recurrence are common (e.g., LSTM layers), but such recurrence involves a particular level of representation interacting with itself, as opposed to the strict top-down notion of higher levels acting on lower levels. In contrast, a layered architecture with feedforward and feedback connections makes a structural distinction between higher and lower levels of representations, and it requires combination of information from two distinct sources.

Bidirectional layered models have long existed (Dayan et al., 1995; Larochelle & Bengio, 2008; Salakhutdinov & Hinton, 2009), but these models have not received the same intense scrutiny and development as feedforward models. The goal of this paper is to revisit bidirectional layered models with the aim of raising them to state-of-the-art in performance. Simply taking existing models and incorporating bidirectional connectivity introduces challenges (Iuzzolino et al., 2019). Instead, we explore the notion that bidirectional information flow must be coupled with additional mechanisms to effectively select from potential top-down signals, modulate bottom-up signals and prevent a cacophony of intermixed signals. We propose two specific mechanisms from the deep learning toolkit to dynamically route information flow: *modularity* and *attention*. With these two mechanisms, we outperform state-of-the-art feedforward models on challenging vision and language tasks.

Dynamic Information Flow using Attention. Goyal et al. (2019) offer evidence that modularity, coupled with a flexible means of communication between modules, can dramatically improve generalization performance. They describe a recurrent independent mechanisms (RIMs) architecture that dynamically controls information flow in a modular RNN. The recurrently connected modules compete for external input and communicate sets objects (associated with a key and value) *sparingly* via differentiable attention mechanisms driven by the match between keys (from a source module) and queries (from a destination module). Computation is sparse and modular, meaning that only a subset of the neural modules are active at any time. Composition of computations is dynamic and *plug-and-play* rather than static, in the sense that the attention mechanism uses

context to select the subset of modules that are activated and what is communicated to which module. The motivating intuition behind this increased flexibility is that if the relationship between modules changes between training and evaluation, a model which keeps the information in these modules sufficiently separate should still be able to correctly recombine information, even though the overall data distribution differs from what was seen during training.

Link to Global Workspace Theory. The consciousness prior (Bengio, 2017) attempts to provide a machine learning motivation for the Global Workspace Theory or GWT (Baars, 1997; Dehaene et al., 2017). This prior states that high-level variables have a joint distribution which can be captured by a sparse factor graph, with the same parameterized computations (factors, in the factor graph nomenclature) being applicable (like rules) to many possible instances of high-level random variables. Inference in this sparse factor graph would naturally be performed by considering only a few elements at the time, which would correspond to the current conscious content. The GWT proposes that this high-level conscious content is broadcast across the brain via top-down connections forming an active circuit with the bottom-up connections. What has been missing from RIMs and which we propose to study here is therefore a notion of hierarchy of levels enabling this kind of bidirectional information flow mediated by attention.

Combining Top-Down and Bottom-Up Information with Attention over Modules. We now present our central proposal, which is that top-down and bottom-up signals should be combined explicitly and selectively by using attention. On a given time step, only a fraction of the high-level concepts being detected are likely to be relevant for a particular stimulus. For example, if a person is trying to classify an object in a dark room (which they are familiar with), the most relevant top-down signal is the person's prior knowledge of what that object looks like, as opposed to other knowledge about the room. This motivates breaking up the hidden state into multiple modules, such that the top-down and bottom-up interactions can be appropriately focused. Thus we consider the use of modules at multiple levels of abstraction to be a key ingredient for our method. We elect to build upon the Recurrent Independent Mechanisms (RIMs) framework for the modular building blocks, as their end-to-end training demonstrated successful specialization between modules (Goyal et al., 2019). However, while network modularity revealed considerable computational advantages, the modules in the RIMs architecture are fully interconnected and thus there is no notion of layered or hierarchical structure.

To address this issue, we propose a new architecture, which we call *Bidirectional Recurrent Independent Mechanisms (BRIMs)*. This new approach endows the attention-based, modular structure with a hierarchy of layers composed of competing modules, yielding organized contextual computations during training. The rationale is to provide an architectural backbone such that each layer of the hierarchy can send information in both a bottom-up direction as well as in a top-down direction. We find that this inductive bias, combined with the modularity mechanisms developed in Goyal et al. (2019), provides considerable and further advantages for tasks where there is a change of input distribution from training to testing.

2. Preliminaries

Multi-layer Stacked Recurrent Networks. The most common multi-layer RNN architecture is bottom-up and feed-forward, in the sense that higher layers are supplied with the states of the lower layers as inputs. An L-layer deep RNN is then concisely summarized as:

$$\mathbf{y}_t = D(\mathbf{h}_t^L) \tag{1}$$

$$\mathbf{h}_t^l = F^l(\mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^l) \tag{2}$$

$$\mathbf{h}_t^0 = E(\mathbf{x}_t) \tag{3}$$

with $l = 0, 1, ..., L$. For a given time t, $\mathbf{y}_t$ denotes the model prediction, $\mathbf{x}_t$ the input and $\mathbf{h}_t^l$ the hidden state of the model at layer l. D and E denote the Decoder and Encoder for the model. F^l represents the recurrent dynamics at the hierarchy level l (e.g., an LSTM or GRU).

Key-Value Attention. Key-value Attention (also sometimes called Scaled Dot Product attention), defines the backbone of updates to the hidden states in the proposed model. This form of attention is widely used in self-attention models and performs well on a wide array of tasks (Vaswani et al., 2017; Santoro et al., 2018). Given a set of queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$, an attention score $\mathbf{A}_S$ and an attention modulated result $\mathbf{A}_R$ are computed as

$$\mathbf{A}_{\mathrm{S}}\left(\mathbf{Q}, \mathbf{K}\right) = \mathrm{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right) \tag{4}$$

$$\mathbf{A}_{\mathrm{R}}\left(\mathbf{Q}, \mathbf{K}, \mathbf{V}\right) = \mathbf{A}_{\mathrm{S}}\ \mathbf{V} \tag{5}$$

Recurrent Independent Mechanisms. RIMs (Goyal et al., 2019) consist of a single layered recurrent structure where the hidden state $\mathbf{h}_t$ is decomposed into n modules, $\mathbf{h}_{t,k}$ for $k = 1, ...n$. It also has the property that on a given time step, only a subset of modules is activated. In RIMs, the updates for the hidden state follow a three-step process. First, a subset of modules is selectively activated based on their determination of the relevance of their input. Second, the activated modules independently process the information made available to them. Third, the active modules gather contextual information from all the other modules and consolidate this information in their hidden state.

Selective Activation. Each module creates queries $\bar{\mathbf{Q}} = \mathbf{Q}_{inp}\left(\mathbf{h}_{t-1,k}\right)$ which are then combined with the keys $\bar{\mathbf{K}} = \mathbf{K}_{inp}\left(\mathbf{\emptyset}, \mathbf{x}_t\right)$ and values $\bar{\mathbf{V}} = \mathbf{V}_{inp}\left(\mathbf{\emptyset}, \mathbf{x}_t\right)$ obtained from the input $\mathbf{x}_t$ and zero vectors $\mathbf{\emptyset}$ to get both the attention score and attention modulated input as per equations (4) and (5). Based on this attention score, a fixed number of modules m are activated for which the input information is most relevant (where the null module, which provides no additional information, has low attention score). We refer to this activated set per time-step as $\mathcal{S}_t$.

Independent Dynamics. Given the attention modulated input obtained above, each activated module then undergoes an update in its hidden state:

$$\bar{\mathbf{h}}_{t,k} = \begin{cases} F_k\left[\mathbf{A}_R\left(\bar{\mathbf{Q}}, \bar{\mathbf{K}}, \bar{\mathbf{V}}\right), \mathbf{h}_{t-1,k}\right] & k \in \mathcal{S}_t \\ \mathbf{h}_{t-1,k} & k \notin \mathcal{S}_t \end{cases} \tag{6}$$

F_k here stands for any update procedure, ex. GRU or LSTM.

Communication. After an independent update step, each module then consolidates information from all the other modules. RIMs again utilizes the attention mechanism to perform this consolidation. Active modules create queries $\hat{\mathbf{Q}} = \mathbf{Q}_{com}(\bar{\mathbf{h}}_{t,k})$ which act with the keys $\hat{\mathbf{K}} = \mathbf{K}_{com}(\bar{\mathbf{h}}_t)$ and values $\hat{\mathbf{V}} = \mathbf{V}_{com}(\bar{\mathbf{h}}_t)$ generated by all modules and the result of attention appends the state for that time step:

$$\mathbf{h}_{t,k} = \begin{cases} \bar{\mathbf{h}}_{t,k} + \mathbf{A}_R\left(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}\right) & k \in \mathcal{S}_t \\ \bar{\mathbf{h}}_{t,k} & k \notin \mathcal{S}_t \end{cases} \tag{7}$$

3. Proposed Method

Our contribution is to extend RIMs to a multilayered bidirectional architecture, which we refer to as BRIMs. Each layer is a modified version of the RIMs architecture (Goyal et al., 2019). A concise representation of our architecture is depicted in Figure 1. We now describe the dynamic bottom-up and top-down flow of information in BRIMs.

3.1. Composition of Modules

We use the procedure provided in RIMs to decompose the hidden state $\mathbf{h}_t^l$ on each layer l and time t into separate modules. Thus, instead of representing the state as just a fixed dimensional vector $\mathbf{h}_t^l$, we choose to represent it as $\{(\mathbf{h}_{t,k}^l)_{k=1}^{n_l}, \mathcal{S}_t^l)\}$ where n_l denotes the number of modules in layer l and $\mathcal{S}_t^l$ is the set of modules that are active at time t in layer l. $|\mathcal{S}_t^l| = m_l$, where m_l is a hyperparameter specifying the number of modules active in layer l at any time. Each layer can potentially have different number of modules active. Typically, setting m_l to be roughly half the value of n_l works well.

3.2. Communication Between Layers

We dynamically establish communication links between multiple layers using key-value attention, in a way which differs radically from RIMs (Goyal et al., 2019). While many RNNs build a strictly bottom-up multi-layer dependency using (2), we instead build multi-layer dependency by considering queries $\bar{\mathbf{Q}} = \mathbf{Q}_{lay}\left(\mathbf{h}_{t,k}^l\right)$ from modules and keys $\bar{\mathbf{K}} = \mathbf{K}_{lay}\left(\mathbf{\emptyset}, \mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^{l+1}\right)$ and values $\bar{\mathbf{V}} = \mathbf{V}_{lay}\left(\mathbf{\emptyset}, \mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^{l+1}\right)$ from all the modules in the lower and higher layer (blue and red arrows respectively; Figure 1). Note that in the deepest layer, only the lower layer is used and on the first layer, the input's embedded state serves as the lower layer. Also note that the attention receiving information from the higher layer looks at the previous time step, whereas the attention receiving information from lower layer (or input) looks at the current time step.

Based on the attention score $\mathbf{A}_S$, the set S_t^l is constructed which comprises modules for which null information is least relevant. Every activated module gets its own separate version of input (as its dependent on its query, which is a function of the hidden state of the module) which is obtained through the attention mechanism. Concretely, for each activated module, this can be represented as:

$$\bar{\mathbf{h}}_{t,k}^l = F_k^l\left[\mathbf{A}_R(\bar{\mathbf{Q}}, \bar{\mathbf{K}}, \bar{\mathbf{V}}), \mathbf{h}_{t-1,k}^l\right] \quad k \in \mathcal{S}_t^l \tag{8}$$

where F_k^l denotes the recurrent update procedure.

3.3. Communication Within Layers

We also perform communication between the different modules within each layer (green arrows; Figure 1). In order to enable this communication, we again make use of key-value attention. This communication between modules within a layer allows them to share information, albeit in a limited way through the bottleneck of attention. We create queries $\hat{\mathbf{Q}} = \mathbf{Q}_{com}\left(\bar{\mathbf{h}}_{t,k}^l\right)$ from active modules and keys $\hat{\mathbf{K}} = \mathbf{K}_{com}\left(\bar{\mathbf{h}}_t^l\right)$ and values $\hat{\mathbf{V}} = \mathbf{V}_{com}\left(\bar{\mathbf{h}}_t^l\right)$ from all the modules to get the final update to the module state as follows:

$$\mathbf{h}_{t,k}^l = \begin{cases} \bar{\mathbf{h}}_{t,k}^l + \mathbf{A}_R(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) & k \in \mathcal{S}_t^l \\ \mathbf{h}_{t-1,k}^l & k \notin \mathcal{S}_t^l \end{cases} \tag{9}$$

3.4. Training

The architecture we propose introduces changes in both the structure of the hidden state as well as the dynamics of updates. It doesn't rely on additional losses and can thus be used as a drop-in substitute for LSTMs and GRUs. For training we consider task-specific losses which range from classification and video prediction losses to RL losses depending on the problem.

4. Related Work

Deep Boltzmann Machines: Deep Boltzmann machines (Salakhutdinov & Hinton, 2009) resemble our motivating intuitions, as they are undirected and thus have both top-down and bottom-up feedback through an energy function. Additionally, the model captures a probability distribution over the hidden states, such that the role of top-down feedback becomes most important when the bottom-up signal has the most uncertainty. However, a significant problem in deep Boltzmann machines, and with undirected graphical models in general, is that sampling and inference are both difficult. Sampling generally requires an iterative procedure, which may converge slowly.

Helmholtz Machine: Like auto-encoders, the Helmholtz machine (Dayan et al., 1995) contains two separate networks: a generative network and a discriminative network, and the wake-sleep learning algorithm is applied to discover a latent representation of the data in an unsupervised way. However, unlike in our proposal, the top-down path only influences the learning of the bottom-up path, and not its actual current computation.

Transformers: The Transformer architecture (Vaswani et al., 2017) eschews the use of a recurrent state in favor of using attention to pass information between different positions. It lacks an explicit inductive bias towards a bottom-up followed by top-down flow, since at the lowest-layers of the network, the positions in the distant past can only undergo a small amount of processing before they can influence the lower levels of processing of later inputs.

Hierarchical Multiscale Recurrent Neural Network: This paper explored an architecture in which the higher layer at previous steps is given as additional inputs to an RNN's lower layers (Chung et al., 2016). Additionally the higher level information is flushed, copied, or updated, using a sparse gating operation. However a key difference from our work is that we use attention to determine whether and what to read from the higher or lower levels, as opposed to concatenating into the input. Additionally, our approach uses a modular recurrent state.

Generative Classifiers: These classify by selecting a class y to maximize $p(x|y)p(y)$. This is a top-down form of classification, since a key factor is estimating the likelihood of a given sample x under a generative model $p(x|y)$. Notably, this purely top-down approach often struggles as learning a model of $p(x|y)$ is generally much harder and much more complex than a direct model of $p(y|x)$ (Ng & Jordan, 2002).

5. Experiments

The main goal of our experiments is to explore the effect of BRIMs on generalization. In particular, we primarily focus on tasks where the training and testing distribution differ systematically, as we believe that the improved top-down modulation of BRIMs will help the model to adapt to variations unseen during training. Additionally, we provide ablations and analysis which explore how the different components of the BRIMs architecture affect results.

We show that BRIMs improve out-of-distribution generalization over sequence length, on sequential MNIST and CIFAR classification, and moving MNIST generation. We then show that BRIMs better reason about physical movement in a synthetic bouncing balls dataset. In all of these tasks, the test distribution differs dramatically from the training distribution. We also show that BRIMs improves results on both language modeling and reinforcement learning. As the data distribution implicitly changes in reinforcement learning as a result of the policy changing during training, this provides further evidence that BRIMs improves out-of-distribution generalization in complex settings.

5.1. Baselines

In our experiments, we consider multiple baselines which use some aspects of BRIMs, such as stacked recurrent layers and modularity. We compare our architecture against various state of the art models (Transformers (Vaswani et al., 2017), Relational RNNs (Santoro et al., 2018)) while also providing ablation studies based on LSTM (Hochreiter & Schmidhuber, 1997) backbone. By outperforming these baselines, we demonstrate the necessity of the contributions introduced in the BRIMs architecture. In particular, we try to disentangle the contributions of attention [A], hierarchy [H], modularity [M], and bidirectional [B] structure.

LSTM variants: We consider variants of LSTMs based on different subsets of important properties. In particular, we experimented with standard LSTMs as well as hierarchical LSTMs without feedback [H], with feedback [H+B], with attention [H+A], and with both [H+A+B].

Transformers [H+A]: Self-attention based multi-layer architecture (Vaswani et al., 2017).

RMC, a relational RNN [A]: Memory based recurrent model with attention communicating between saved memory and hidden states (Santoro et al., 2018).

Recurrent Independent Mechanisms (RIMs) [A+M]: Modular memory based single layered recurrent model with attention modulated input and communication between modules (Goyal et al., 2019).

Hierarchical RIMs [H+A+M]: Two layered modular architecture with attention based communication between the modules in the same layer and between different layers. Here the flow of information is unidirectional i.e information only flows from bottom to top, and hence no top-down

Algorithm	*Properties*	*19 x 19*	*24 x 24*	*32 x 32*
LSTM	—	54.4	44.0	32.2
LSTM	H	57.0	46.8	33.2
LSTM	H+B	56.5	52.2	42.1
LSTM	H+A	56.7	51.5	40.0
LSTM	H+A+B	59.9	54.6	43.0
RMC	A	49.9	44.3	31.3
RIMs	A+M	56.9	51.4	40.1
Hierarchical RIMs	H+A+M	57.2	54.6	46.8
MLD-RIMs	H+A+M	56.8	53.1	44.5
BRIMs (ours)	H+A+B+M	60.1	57.7	52.2

Table 1. Performance on **Sequential CIFAR generalization:** Test Accuracy % after 100 epochs. Both the proposed and the Baseline model (LSTM) were trained on 16x16 resolution but evaluated at different resolutions; results averaged over 3 different trials.

Algorithm	*Properties*	*16 x 16*	*19 x 19*	*24 x 24*
LSTM	—	86.8	42.3	25.2
LSTM	H	87.2	43.5	22.9
LSTM	H+B	83.2	44.4	25.3
LSTM	H+A	84.3	47.5	31.0
LSTM	H+A+B	83.2	40.1	20.8
RMC	A	89.6	54.2	27.8
Transformers	H+A+B	91.2	51.6	22.9
RIMs	A+M	88.9	67.1	38.1
Hierarchical RIMs	H+A+M	85.4	72.0	50.3
MLD-RIMs	H+A+M	88.8	69.1	45.3
BRIMs (ours)	H+A+B+M	88.6	74.2	51.4

Table 2. Performance on the **Sequential MNIST resolution generalization:** Test Accuracy % after 100 epochs. Both the proposed and the Baseline model (LSTM) were trained on 14x14 resolution but evaluated at different resolutions; results averaged over 3 different trials.

information is being used.

MLD-RIMs, RIMs with Multilayer Dynamics [H+A+M]: The same as hierarchical RIMs except that the activation of RIMs on the higher layer is copied from the lower layer instead of being computed at the higher layer. This more tightly couples the behavior of the lower and higher layers.

BRIMs [H+A+M+B]: Our model incorporates all these ingredients: layered structure, where each layer is composed of modules, sparingly interacting with the bottleneck of attention, and allowing top down information flow.

For more detailed description about the baselines, we ask the reader to refer to section A.1 in Appendix.

5.2. Model Setup

We use a 2-layered setup with each layer consisting of a set of modules. The proposed architecture has 2 degrees of freedom: the number of modules in each layer and number of active modules in each layer. For the supervised loss, the final state of the recurrent model is passed through a feed-forward network before making a prediction. For RL experiments, the concatenation of the state of lower level modules as well as higher level modules is used to compute the policy. For video prediction experiments, the state of the lower level module is fed into the decoder, and used for generating the image. Unless otherwise indicated we always (a) learn an embedding for input tokens before feeding it to the RNNs, (b) use Adam with a learning rate of 0.001 and momentum of 0.9. We include more experimental results and the code in the Supplementary Material.

5.3. Sequential MNIST and CIFAR

These tasks involve feeding a recurrent model the pixels of an image in a scan-line order and producing a classification label at the end of the sequence. To study the generalization capacity of different recurrent models, we train on sMNIST at a resolution of (14,14) and test on the resolutions (16,16), (19,19) and (24,24). Similarly, we perform training for sCIFAR at resolution (16,16) and testing on (19,19), (24,24) and (32,32). Because these images contain many distracting and uninformative regions, a model which is better able to ignore these patterns should generalize better to longer sequences. Additionally many details in these images are difficult to understand without having good top-down priors. These will be most salient when the specific character of these details changes as, for example, when we increase the resolution. See Tables 1, 2 for results and appendix section A.2 for more details.

5.4. Moving MNIST: Video Prediction

We use the Stochastic Moving MNIST (SM-MNIST) dataset introduced by Denton & Fergus (2018) which consists of sequences of frames of size 64 x 64, containing one or two MNIST digits moving and bouncing off the edge of the frame (walls). Training sequences were generated on the fly by sampling two different MNIST digits from the training set (60k total digits). We trained the proposed model on SM-MNIST by conditioning on 5 frames and training the model to predict the next 10 frames in the sequence, and compared it to the SVG-LP model (Denton & Fergus, 2018).

To test out-of-distribution generalization capability, we trained models on image sequences from the SM-MNIST dataset containing only digits 0-7, and tested them on sequences containing only digits 8 & 9. We compared the Structural Similarity index (SSIM) and the Mean Squared Error (MSE) of the best generated sequences from the baseline SVG-LP model, and those from ours. We followed the same evaluation protocol as SVG, and the results in Figure 2 demonstrate that our proposed method performs consistently better. For details about the experimental setup, we ask the reader to refer to section A.5 in Appendix.

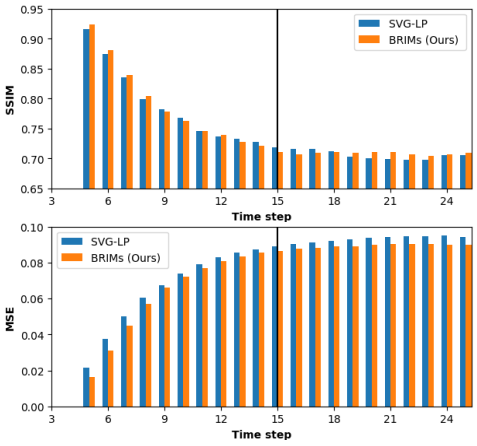

Figure 2. Comparison of BRIMs and SVG-LP (Denton & Fergus, 2018) on Stochastic Moving MNIST, evaluated using (a) SSIM (higher is better) and (b) MSE (lower is better). Models are trained on digits 0-7 and tested with digits 8-9. The models are conditioned on 5 time steps, and trained for 10 more time steps. SSIM is calculated by rolling out the models into future time steps.

5.5. Handling Occlusion in Bouncing Balls

We study the ability of the proposed model to model the physical reasoning capabilities, on the bouncing balls task, a standard environment for evaluating physical reasoning capabilities exhibiting complex non-linear physical dynamics. We train BRIMs on sequences of 6464 binary images over 51 time-steps that contain four bouncing balls with different masses corresponding to their radii. The balls are initialized with random initial positions, masses and velocities. Balls bounce elastically against each other and the image window. We also test the proposed method in the scenario when an invisible "curtain" (Van Steenkiste et al., 2018) is present, which further tests the degree to which the model correctly understands the underlying dynamics of the scene. For vizualizations about the predictions from the proposed model, refer to section A.6 in Appendix.

5.6. Language Modelling

We consider word-level language modeling on the WikiText-103 dataset which have been used as benchmarks in previous work (Santoro et al., 2018). Language has an organic hierarchical and recurrent structure which is both noisy and flexible, making it an interesting and apt task to study for the proposed model. Table 3 shows Validation and Test set perplexities for various models. We find that the proposed method improve over the standard LSTM as well as RMCs.

5.7. Reinforcement Learning

Top-down and bottom-up signals are clearly differentiated in the case of reinforcement learning (or in active agents, more generally) in that they lead to different optimal behavior. Since top-down signals change more slowly, they often suggest long-term and deliberative planning. On the other hand, bottom-up signals which are surprising can often require an immediate reaction. Thus, in selecting actions, points in which the top-down system dominates may benefit from activating a system with a long-term plan, whereas points where the bottom-up information dominates could benefit from a system which immediately takes action to resolve an unexpected threat. On the whole, we expect that dynamically modulating activation based on bottom-up and top-down signal importance will be especially important in reinforcement learning. Additionally, in reinforcement learning the data distribution implicitly changes as the policy evolves over the course of training, testing the models ability to correctly handle distribution shift.

Algorithm	*Properties*	*Valid*	*Test*
LSTM	—	38.2	41.8
RMC*	H	36.2	38.3
Hierarchical RIMs	H+A+M	36.1	38.1
BRIMs (ours)	H+A+B+M	35.5	36.8

Table 3. Perplexities on the WikiText-103 dataset. $(*)$ refers to our implementation of the model.

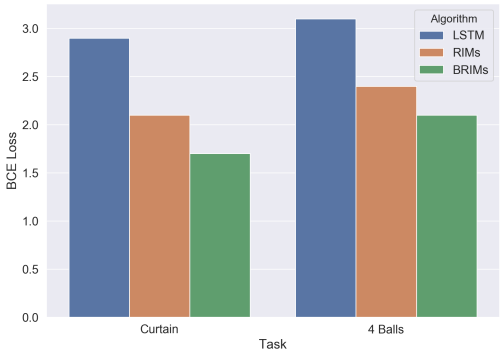

Figure 3. We study the performance of the proposed model as compared to the LSTM as well as RIMs baseline. We feed the ground truth for the first 15 time steps, and then we ask the model to predict the next 35 time steps. During rollout phase, the proposed model performs better in terms of accurately predicting the dynamics of the balls for the 4Balls scenario, as well as for the more difficult occlusion scenario, as reflected by the lower BCE.

To investigate this we use an RL agent trained using Proximal Policy Optimization (PPO) (Schulman et al., 2017) with a recurrent network producing the policy. We employ an LSTM as well as RIMs architecture (Goyal et al., 2019) as baseline and compare results to the proposed architecture. This was a simple drop-in replacement and did not require changing any of the hyperparameters for PPO. We experimented on 10 Atari games (chosen alphabetically) and ran three independent trials for each game (reporting mean and standard deviation). We found that simply using BRIMs for the recurrent policy improves performance (Table 4). We also found an improvement on Ms. Pacman, which requires

Environment	*LSTM*	*RIMs*	*BRIMs (ours)*
Alien	1612 ± 44	2152 ± 81	4102 ± 400
Amidar	1000 ± 58	1800 ± 43	2454 ± 100
Assault	4000 ± 213	5400 ± 312	5700 ± 320
Asterix	3090 ± 420	21040 ± 548	30700 ± 3200
Asteroids	1611 ± 200	3801 ± 89	2000 ± 300
Atlantis	3.28M ± 0.20M	3.5M ± 0.12M	3.9M ± 0.05M
BankHeist	1153 ± 23	1195 ± 4	1155 ± 20
BattleZone	21000 ± 232	22000 ± 324	25000 ± 414
BeamRider	698 ± 100	5320 ± 300	4000 ± 323
MsPacMan	4598 ± 100	3920 ± 500	5900 ± 1000

Table 4. Results on the Atari Reinforcement learning task using PPO with different recurrent policy networks.

careful planning and is not purely reactive.

6. Analysis and Ablation Studies

We have demonstrated substantially improved results using BRIMs. However, it is essential to understand which aspects of the newly proposed model are important for performance. For example, BRIMs does introduce additional capacity, and thus it is critical to demonstrate that the performance improvement is not replicated by increasing capacity in simpler ways. To this end we conduct ablation studies to demonstrate the necessity of bidirectional structure, attention, modularity, and hierarchy.

6.1. Role of Bidirectional Information Flow

We demonstrate that in different architectures, having top down information flow helps in generalization. Tables 1, 2, 3 demonstrate the effectiveness of using top down connections in standard LSTMs, attention based LSTMs and hierarchical modules. We maintain that higher layers are able to filter out important information from lower layers and thus conditioning on it provides much more context as to what is relevant at any time in a sequence. However, the results still fall short of BRIMs, indicating that this component alone is not sufficient.

6.2. Consistent Utilization of the Higher Level

We also investigated the nature of the learned bidirectional flow on sequential CIFAR-10 classification. The lower level can dynamically choose whether to use information from input or the higher level. The lower level should query the higher level only if the latter contains information which might be relevant for solving the task at hand. We can analyze the frequency with which the lower level queries information from the higher level, i.e., whether it exploits top-down information.

We experimentally verified that almost every example attends to the higher level, yet only a small fraction of steps attend to the higher level on typical examples. Quantitatively, 95.1% of images accessed the higher level at least five times, while on average only 2.86% of the total attention (averaged over all modules) was to the higher level.

6.3. Role of Attention

Tables 1, 2, 3, 4 and Figure 2 show the importance of having attention, as Transformers, RMC, attention-modulated LSTMs, RIMs and BRIMs lead to much better performance and generalization in domains ranging from supervised learning to reinforcement learning than their non attention counterparts. This suggests that key-value attention mechanism is better able to extract out important information and dependency as opposed to standard connections.

6.4. Role of Modularity

Tables 1, 2, 3, 4 indicate that having layers as a composition of smaller modules with their own independent dynamics has better performance compared to architectures without modular decomposition. In particular, we note that RIMs, its hierarchical variants and BRIMs perform much better than their LSTM counterparts. This shows that having independent dynamics for modules as well as a communication channel between them are important ingredients for generalization.

6.5. Role of Hierarchy

We consistently found that models with hierarchical structure perform much better than their shallow counterparts (Tables 1, 2, 4). This relates to the discussion in Section 6.1 that higher layers are able to abstract and filter out important information and concepts learned in lower layers.

7. Conclusion

Top-down and bottom-up information are both critical to robust and accurate perception. How to combine these signals has been a central topic of focus within not just deep learning, but also the entire field of cognitive science for several decades. Our work focuses on the idea that attention can be used to give models explicit control over the combination of top-down and bottom-up signals which is both dynamic and context dependent. Our simulation experiments have shown a critical role for network modularity: ensuring that specific top-down signals combine with specific bottom-up signals. Using these insights we have proposed a new algorithm, *Bidirectional Recurrent Independent Mechanisms (BRIMs)*, which achieves substantial improvements on sequence-length generalization tasks, language modeling, video generation, and reinforcement learning on Atari. A detailed ablation study gives evidence that combining top-down and bottom-up signals through attention is the critical component for achieving these improved results.

References

Baars, B. J. In the theatre of consciousness. global workspace theory, a rigorous scientific theory of consciousness. *Journal of Consciousness Studies*, 4(4):292–309, 1997.

Baluch, F. and Itti, L. Mechanisms of top-down attention. *Trends in Neurosciences*, 34(4):210–224, 2011.

Bastos, A. M., Vezoli, J., Bosman, C. A., Schoffelen, J.-M., Oostenveld, R., Dowdall, J. R., De Weerd, P., Kennedy, H., and Fries, P. Visual Areas Exert Feedforward and Feedback Influences through Distinct Frequency Channels. *Neuron*, 85(2):390–401, January 2015.

Bengio, Y. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. Greedy layer-wise training of deep networks. In *NIPS'2006*, 2007.

Chung, J., Ahn, S., and Bengio, Y. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Dehaene, S., Lau, H., and Kouider, S. What is consciousness, and could machines have it? *Science*, 358(6362): 486–492, 2017.

Denton, E. and Fergus, R. Stochastic video generation with a learned prior. *arXiv preprint arXiv:1802.07687*, 2018.

Dijkstra, N., Zeidman, P., Ondobaka, S., and Friston, K. Distinct top-down and bottom-up brain connectivity during visual perception and imagery. *Scientific Reports*, 7: 5677, 2017.

Felleman, D. J. and Van Essen, D. C. Distributed Hierarchical Processing in the Primate Cerebral Cortex. *Cerebral Cortex*, 1(1):1–47, 01 1991.

Gilbert, C. D. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5):350–363, 2013.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*, 2019.

Hinton, G. E., Osindero, S., and Teh, Y. A fast learning algorithm for deep belief nets. *Neural Computation*, 18: 1527–1554, 2006.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Iuzzolino, M., Singer, Y., and Mozer, M. C. Convolutional bipartite attractor networks, 2019.

Kersten, D., Mamassian, P., and Yuille, A. Object Perception as Bayesian Inference. *Annual Review of Psychology*, 55 (1):271–304, February 2004.

Kinchla, R. A. and Wolfe, J. M. The order of visual processing: top-down, bottom-up, or middle-out. *Perception & Psychophysics*, 25:225–231, 1979.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Larochelle, H. and Bengio, Y. Classification using discriminative restricted boltzmann machines. pp. 536–543, 01 2008. doi: 10.1145/1390156.1390224.

Manita, S., Suzuki, T., Homma, C., Matsumoto, T., Odagawa, M., Yamada, K., Ota, K., Matsubara, C., Inutsuka, A., Sato, M., Ohkura, M., Yamanaka, A., Yanagawa, Y., Nakai, J., Hayashi, Y., Larkum, M. E., and Murayama, M. A Top-Down Cortical Circuit for Accurate Sensory Perception. *Neuron*, 86(5):1304–1316, June 2015.

McClelland, J. L. and Rumelhart, D. E. An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological review*, 88 (5):375, 1981.

Ng, A. Y. and Jordan, M. I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Dietterich, T. G., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems 14*, pp. 841–848. MIT Press, 2002.

Nielsen, M. L., Tanabe, H., Imaruoka, T., Sekiyama, K., Tashiro, T., and Miyauchi, S. Reciprocal connectivity in visual cortex: evidence from fmri. In *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, volume 2, pp. 28–33 vol.2, 1999.

Rauss, K. and Pourtois, G. What is bottom-up and what is top-down in predictive coding? *Frontiers in Psychology*, 4:276, 2013.

Rockland, K. S. About connections. *Frontiers in Neuroanatomy*, 9:61, 2015.

Salakhutdinov, R. and Hinton, G. Deep boltzmann machines. In *Artificial intelligence and statistics*, pp. 448–455, 2009.

Santoro, A., Faulkner, R., Raposo, D., Rae, J. W., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. P. Relational recurrent neural networks. *CoRR*, abs/1806.01822, 2018. URL http://arxiv.org/abs/1806.01822.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Weiss, Y., Simoncelli, E. P., and Adelson, E. H. Motion illusions as optimal percepts. *Nature Neuroscience*, 5(6): 598–604, 2002.

A. Appendix

Algorithm 1: Single recurrent step for an L layered BRIMs model

Result: RNN Cell forward for L layered BRIMs
x : shape (d_{in})
h : List of L elements where l^{th} element is of size (n_l, d_{model})
n_l : Number of modules in layer l
m_l : Number of modules kept active in layer l
ϕ : Null vector
All **Query**, **Key**, **Value** networks are linear networks
Note: Unless specified, all indexing start with 1

Function `BRIMsCell` (*x, h*) **:**
  $h[0] = x$

  for *l = 1 to L* **do**
    for *k = 1 to* n_l **do**
      Q[k] = **Input Query**$_{l,k}$(h[l, k])
    end
    K[0], V[0] = **Null Key Value**$_l(\phi)$
    K[1], V[1] = **Input Key Value**$_l$(h[l-1])
    K[2], V[2] = **Top-Down Key Value**$_l$(h[l+1]) (if available)
    S = Softmax($QK^T/\sqrt{d_{att}}$)
    input = SV

    for *k = 1 to* n_l **do**
      Q[k] = **Input Query**$_{l,k}$(h[l, k])
    end
    K[0], V[0] = **Null Key Value**$_l(\phi)$
    for *k = 1 to* n_{l-1} **do**
      K[k], V[k] = **Input Key Value**$_{l,k}$(h[l-1, k])
    end
    for *k = 1 to* n_{l+1} **do**
      K[$n_{l-1}+k$], V[$n_{l-1}+k$] = **Top-Down Key Value**$_{l,k}$(h[l+1, k]) (if available)
    end
    S = Softmax($QK^T/\sqrt{d_{att}}$)
    input = SV

    Sort S[:,0] and take lowest m_l as active
    for *k s.t. module k is active* **do**
      h[l,k] = **RNN**$_{l,k}$(input[k], h[l,k]) (can use GRU or LSTM)
    end

    for *k = 1 to* n_l **do**
      Q[k] = **Communication Query**$_{l,k}$(h[l,k])
      K[k] = **Communication Key**$_{l,k}$(h[l,k])
      V[k] = **Communication Value**$_{l,k}$(h[l,k])
    end
    comm = Softmax($QK^T/\sqrt{d_{att}}$) V
    for *k s.t. module k is active* **do**
      h[l,k] += comm[k]
    end
  end
  return h

Communication Between Layers

Communication Between Layers

Sparse Activation

Communication Within Layer

Task	Hidden Dimensions	Number of RIMs (n_l)	Number of Active RIMs (m_l)
Sequential MNIST	(600 , 300)	(6 , 3)	(4 , 2)
Sequential CIFAR	(300 , 300)	(6 , 6)	(4 , 4)
Moving MNIST	(300 , 300)	(6 , 6)	(4 , 4)
Bouncing Balls	(300 , 300)	(5 , 5)	(3 , 3)
Language Modelling	(300 , 300)	(6 , 6)	(4 , 4)
Reinforcement Learning	(300 , 300)	(6 , 6)	(4 , 4)
Adding	(300 , 300)	(5 , 5)	(3 , 3)

Table 5. Hyperparameters for experiments. We list the number of hidden units (combined over all RIMs) as well as the number of RIMs on each layer.

A.1. Baseline Description

In our experiments, we consider multiple baselines which use some aspects of BRIMs, such as stacked recurrent layers and modularity. We compare our architecture against various state of the art models (Transformers (Vaswani et al., 2017), Relational RNNs (Santoro et al., 2018)) while also providing ablation studies based on LSTM (Hochreiter & Schmidhuber, 1997) backbone. By outperforming these baselines, we demonstrate the necessity of the contributions introduced in the BRIMs architecture. In particular, we try to disentangle the contributions of attention [A], hierarchy [H], modularity [M], and bidirectional [B] structure.

LSTM variants: We consider variants of LSTMs based on different subsets of important properties. In particular, we experimented with standard LSTMs as well as hierarchical LSTMs without feedback [H], with feedback [H+B], with attention [H+A], and with both [H+A+B].

Transformers [H+A]: Self-attention based multi-layer architecture (Vaswani et al., 2017).

RMC, a relational RNN [A]: Memory based recurrent model with attention communicating between saved memory and hidden states (Santoro et al., 2018).

Recurrent Independent Mechanisms (RIMs) [A+M]: Modular memory based single layered recurrent model with attention modulated input and communication between modules (Goyal et al., 2019).

Hierarchical RIMs [H+A+M]: Two layered modular architecture with attention based communication between the modules in the same layer and between different layers. Here the flow of information is unidirectional i.e information only flows from bottom to top, and hence no top-down information is being used.

MLD-RIMs, RIMs with Multilayer Dynamics [H+A+M]: The same as hierarchical RIMs except that the activation of RIMs on the higher layer is copied from the lower layer instead of being computed at the higher layer. This more tightly couples the behavior of the lower and higher layers.

BRIMs [H+A+M+B]: Our proposed model incorporates all these ingredients: layered structure, where each layer is composed of modules, sparingly interacting with the bottleneck of attention, and allowing top down information flow.

A.2. sMNIST

For the baseline LSTM variants, we perform hyperparameter tuning by considering the dimension of hidden state in each layer to be chosen from $\{300,600\}$. We also experimented with learning rates of 0.001, 0.0007 and 0.0003. For RIMs, Hierarchical RIMs and MLD-RIMs, we experiment with the same hidden state size (= sum of sizes of all modules of the layer) and learning rate. Number of modules are chosen from the set $\{3,5,6\}$ and the number active at any time is roughly around half of the total number of modules.

We use an encoder to embed the input pixels to a 300 dimensional vector. We consider the lower hidden state to comprise of 6 modules and the higher state of 3 modules. The modules at lower levels are of 50 dimensions while at higher levels they are of 100 dimensions. We maintain 4 modules active at lower level and 2 at higher level at any given time.

We train the model using Adam optimizer with a learning rate of 0.0007. We clip the gradients at 1.0 for stability and train the model with dropout of 0.5 for 100 epochs. Evaluation is obtained according to best performance on a validation split.

A.3. sCIFAR

For the baseline LSTM variants, we perform hyperparameter tuning by considering the dimension of hidden state in each layer to be chosen from {300,600}. We also experimented with learning rates of 0.001, 0.0007 and 0.0003. For RIMs, Hierarchical RIMs and MLD-RIMs, we experiment with the same hidden state size and the same learning rates. We choose the number of modules from the set {3,5,6} and the number active at any time is roughly around half of the total number of modules.

We train a model which takes CIFAR images as a sequence of pixels and predicts the class label. We downsample to 16x16 (a sequence length of 256) during training and use nearest-neighbor downsampling.

We provide the model with all of the three colors as inputs on each step we use a separate encoder for each channel of the RGB image. We maintain the size of each module at both the lower and higher layer to be 50 and the number of modules in both the layers 6. We restrict the number of activated modules at any time step to be 4 for both the layers. The inputs received by the model are encoded into a 300 dimensional vector.

We train the model using Adam optimizer with learning rate 0.0007. We further clip the gradients at 1.0 in order to stabilize training. We train the model with embedding dropout of 0.5 for 100 epochs and use a validation split to obtain the results on the test set corresponding to best accuracy on validation split.

In Figure 4 we show that if we train normally but make some of the pixels random (uniformly random noise) at test time, then the model puts more of its attention on the higher level, as opposed to the input. This is evidence that the model learns to rely more heavily on expectations and prior knowledge when the input sequence is less reliable.

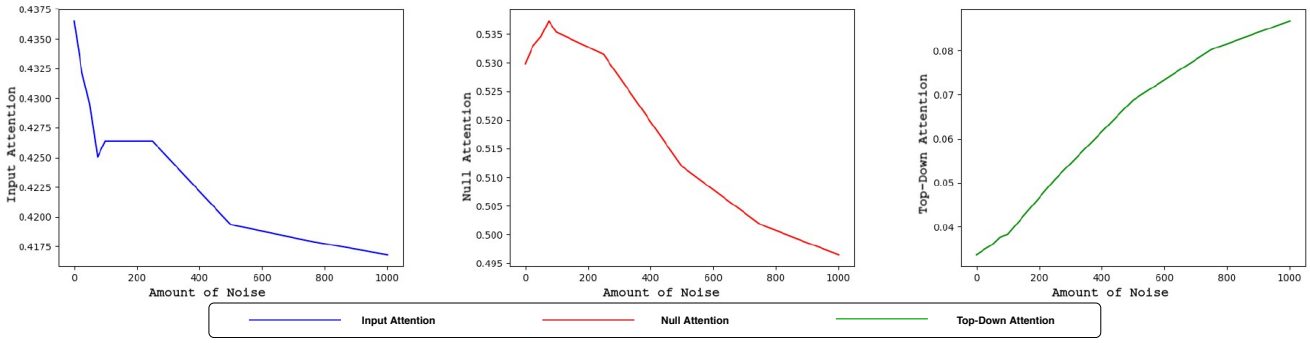

Figure 4. On sequential CIFAR-10, we evaluate on 32x32 test images and change a certain number of the pixels to random values. We show the **average activation weight (y-axis)** on Input (top), Null (middle) and Higher Layer (bottom) plotted against the **number of random pixels (x-axis)** (total number of pixels = 1024). We see that as more pixels are set to random values, the model becomes increasingly reliant on the higher-level information.

A.4. Adding

For all the variants, we perform hyperparameter search for learning rate from the set {0.001, 0.0007, 0.0003}. Apart from learning rate, we didn't perform any other kind of hyperparameter search.

In the adding task we consider a stream of numbers as inputs (given as real-values) and then indicate which two numbers should be added together as a set of two input streams which varies randomly between examples. The length of the input sequence during testing is longer than during training. This is a simple test of the model's ability to ignore the numbers which it is not tasked with adding together. We show that BRIMs provides substantially faster convergence on the adding task. We also evaluate the model's performance on adding multiple numbers even when it is trained on adding only two. We demonstrate that BRIMs generalize better for longer testing sequences as well as when the number of numbers to be added changes between training and evaluation (Table 6 and Table 7).

For the task we consider a linear encoder that encodes the three input streams into a 300 dimensional vector and a linear decoder that gives the final output given the final hidden state at the higher layer. We use 5 modules, each comprising of a 60 dimensional vector, at both the lower and higher layers. We also maintain that 3 modules remain active at any time at both the lower and higher layers.

Number of Values	*Random Prediction*	*LSTM*	*BRIMs (ours)*
2	0.500	0.2032	0.0003
3	1.000	0.2678	0.0002
4	1.333	0.3536	0.0002
5	2.500	0.5505	0.0058
10	9.161	3.5709	2.078

Table 6. We train on sequences with either 2 or 4 values to be summed, and on a sequence length of 50. We test on a sequence length 200 with different numbers of values to be summed. This demonstrates that BRIMs improves generalization when jointly varying both the sequence length and the numbers of values to be summed.

Number of Values	*Random Prediction*	*LSTM*	*BRIMs (ours)*
2	0.500	0.0842	0.0000
3	1.000	0.4601	0.0999
4	1.333	1.352	0.4564
5	2.500	2.738	1.232
10	9.161	17.246	11.90

Table 7. We train on sequences of length 1000 where 2 values must be summed. We test on sequence length 2000 and vary the number of values to be summed between 2 and 10. We note much better generalization when using BRIMs.

The model is trained end to end with a learning rate of 0.001 using Adam Optimizer. We clip the gradients at 0.1 and use dropout of 0.1. We perform two experiments, outlined below:

- We train the model to add two numbers from 1000 length sequences and evaluate it on adding variable number of numbers (2 - 10) on 2000 length sequences. (Table 6)
- We train the model to add a mixture of two and four numbers from 50 length sequences and evaluate it on adding variable number of numbers (2 - 10) on 200 length sequences. (Table 7)

A.5. Moving MNIST

We use the Stochastic Moving MNIST (SM-MNIST) (Denton & Fergus, 2018) dataset which consists of sequences of frames of size 64×64, containing one or two MNIST digits moving and bouncing off the walls. Training sequences are generated on the fly by sampling two different MNIST digits from the training set (60k total digits) and two distinct trajectories. Trajectories change randomly every time a digit hits a wall.

We trained the model on sequences only containing digits 0-7, and validate it on sequences with digits 8 and 9. We ran the proposed algorithm with 2 layers, each with 6 RIMs and 4 active RIMs. We used a batch size of 100, and ran 600 batches per epoch (60000 samples per epoch). We trained for 300 epochs using Adam as optimizer (Kingma & Ba, 2014) with a learning rate of 0.002.

A.6. Bouncing Balls

We use the bouncing-ball dataset from (Van Steenkiste et al., 2018). The dataset consists of 50,000 training examples and 10,000 test examples showing $\sim$50 frames of either 4 solid balls bouncing in a confined square geometry, 6-8 balls bouncing in a confined geometry, or 3 balls bouncing in a confined geometry with a random occluded region. In all cases, the balls bounce off the wall as well as off one another. We train baselines as well as proposed model for about 100 epochs using 0.0007 as learning rate and using Adam as optimizer (Kingma & Ba, 2014). We use the same architecture for encoder as well as decoder as in (Van Steenkiste et al., 2018). We train the proposed model as well as the baselines for 100 epochs.

A.7. Language Modelling

Hyperparameters are tuned for the vanilla LSTM, which consist of 1, 2 LSTM layer, 0.3 embedding dropout, no layer norm, and shared, input/output embedding parameters. We use a hidden state of the size 2048 for the vanilla LSTM. Training was

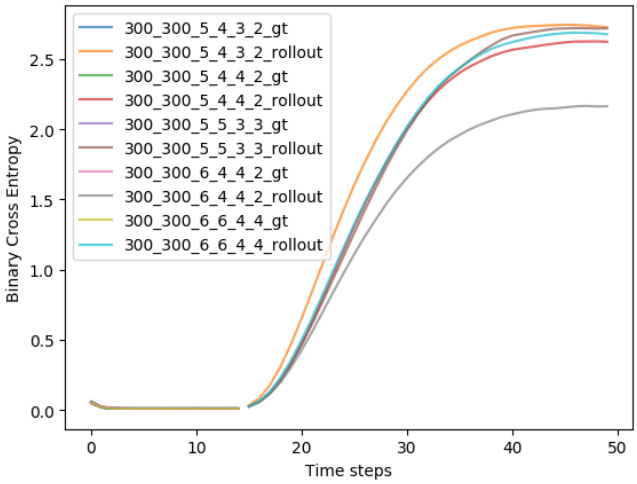

Figure 5. We show the performance of the proposed model for different values of number of RIMs on each layer, as well as number of active RIMs. The first 15 frames of ground truth are fed in and then the system is rolled out for the next 35 time steps. The legend format is: (number of units on first layer, number of total units on second layer, total number of RIMs on first layer, total RIMs on second layer, activated RIMs on first layer, activated RIMs on second layer). The best result is achieved by using sparse activation on both the higher and lower layer.

performed with Adam at learning rate 1e-3, gradients clipped to 0.1, sequence length 128, and batch size 128. We ran the proposed algorithm with 6 RIMs on each layer and kept the number of activated RIMs to 4 on each layer, with number of layers = 2. We have not done any hyper-parameter search for these experiments either for the baseline or for the proposed model.

A.8. Atari

We used an open-source implementation of PPO from (Kostrikov, 2018) with default parameters. We ran the proposed algorihtm with 6 RIMs on each layer and kept the number of activated RIMs to 4 on each layer, with number of layers = 2. We have not done any hyper-parameter search for Atari experiments.