Université de Montréal

Towards Computationally Efficient Neural Networks with Adaptive and Dynamic Computations

par

Taesup Kim

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Orientation intelligence artificielle

Août 2021

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Towards Computationally Efficient Neural Networks with Adaptive and Dynamic Computations

présentée par

Taesup Kim

a été évaluée par un jury composé des personnes suivantes :

Simon Lacoste-Julien
(président-rapporteur)

Yoshua Bengio
(directeur de recherche)

Aaron Courville
(membre du jury)

Chelsea Finn
(examinateur externe)

Guillaume Lajoie
(représentant du doyen de la FESP)

Résumé

Ces dernières années, l'intelligence artificielle a été considérablement avancée et l'apprentissage en profondeur, où des réseaux de neurones profonds sont utilisés pour tenter d'imiter vaguement le cerveau humain, y a contribué de manière significative. Les réseaux de neurones profonds sont désormais capables d'obtenir un grand succès sur la base d'une grande quantité de données et de ressources de calcul suffisantes. Malgré leur succès, leur capacité à s'adapter rapidement à de nouveaux concepts, tâches et environnements est assez limitée voire inexistante. Dans cette thèse, nous nous intéressons à la façon dont les réseaux de neurones profonds peuvent s'adapter à des circonstances en constante évolution ou totalement nouvelles, de la même manière que l'intelligence humaine, et introduisons en outre des modules architecturaux adaptatifs et dynamiques ou des cadres de méta-apprentissage pour que cela se produise de manière efficace sur le plan informatique. Cette thèse consiste en une série d'études proposant des méthodes pour utiliser des calculs adaptatifs et dynamiques pour aborder les problèmes d'adaptation qui sont étudiés sous différentes perspectives telles que les adaptations au niveau de la tâche, au niveau temporel et au niveau du contexte.

Dans le premier article, nous nous concentrons sur l'adaptation rapide des tâches basée sur un cadre de méta-apprentissage. Plus précisément, nous étudions l'incertitude du modèle induite par l'adaptation rapide à une nouvelle tâche avec quelques exemples. Ce problème est atténué en combinant un méta-apprentissage efficace basé sur des gradients avec une inférence variationnelle non paramétrique dans un cadre probabiliste fondé sur des principes. C'est une étape importante vers un méta-apprentissage robuste que nous développons une méthode d'apprentissage bayésienne à quelques exemples pour éviter le surapprentissage au niveau des tâches.

Dans le deuxième article, nous essayons d'améliorer les performances de la prédiction de la séquence (c'est-à-dire du futur) en introduisant une prédiction du futur sauteur basée sur la taille du pas adaptatif. C'est une capacité critique pour un agent intelligent d'explorer un environnement qui permet un apprentissage efficace avec une imagination sauteur futur. Nous rendons cela possible en introduisant le modèle hiérarchique d'espace d'état récurrent

(HRSSM) qui peut découvrir la structure temporelle latente (par exemple, les sous-séquences) tout en modélisant ses transitions d'état stochastiques de manière hiérarchique.

Enfin, dans le dernier article, nous étudions un cadre qui peut capturer le contexte global dans les données d'image de manière adaptative et traiter davantage les données en fonction de ces informations. Nous implémentons ce cadre en extrayant des concepts visuels de haut niveau à travers des modules d'attention et en utilisant un raisonnement basé sur des graphes pour en saisir le contexte global. De plus, des transformations au niveau des caractéristiques sont utilisées pour propager le contexte global à tous les descripteurs locaux de manière adaptative.

mots-clés: l'apprentissage en profondeur, réseaux de neurones profonds, réseaux de neurones adaptatifs, calcul adaptatif, calcul dynamique, apprentissage quelques-shot, méta-apprentissage, apprendre pour apprendre, mécanisme d'attention, modulation de feature, raisonnement concept, abstraction temporelle.

Summary

Over the past few years, artificial intelligence has been greatly advanced, and deep learning, where deep neural networks are used to attempt to loosely emulate the human brain, has significantly contributed to it. Deep neural networks are now able to achieve great success based on a large amount of data and sufficient computational resources. Despite their success, their ability to quickly adapt to new concepts, tasks, and environments is quite limited or even non-existent. In this thesis, we are interested in how deep neural networks can become adaptive to continually changing or totally new circumstances, similarly to human intelligence, and further introduce adaptive and dynamic architectural modules or meta-learning frameworks to make it happen in computationally efficient ways. This thesis consists of a series of studies proposing methods to utilize adaptive and dynamic computations to tackle adaptation problems that are investigated from different perspectives such as task-level, temporal-level, and context-level adaptations.

In the first article, we focus on task-level fast adaptation based on a meta-learning framework. More specifically, we investigate the inherent model uncertainty that is induced from quickly adapting to a new task with a few examples. This problem is alleviated by combining the efficient gradient-based meta-learning with nonparametric variational inference in a principled probabilistic framework. It is an important step towards robust meta-learning that we develop a Bayesian few-shot learning method to prevent task-level overfitting.

In the second article, we attempt to improve the performance of sequence (i.e. future) prediction by introducing a jumpy future prediction that is based on the adaptive step size. It is a critical ability for an intelligent agent to explore an environment that enables efficient option-learning and jumpy future imagination. We make this possible by introducing the Hierarchical Recurrent State Space Model (HRSSM) that can discover the latent temporal structure (e.g. subsequences) while also modeling its stochastic state transitions hierarchically.

Finally, in the last article, we investigate a framework that can capture the global context in image data in an adaptive way and further process the data based on that information. We implement this framework by extracting high-level visual concepts through attention modules and using graph-based reasoning to capture the global context from them. In

addition, feature-wise transformations are used to propagate the global context to all local descriptors in an adaptive way.

Keywords: deep learning, deep neural networks, adaptive neural networks, adaptive computation, dynamic computation, few-shot learning, meta-learning, learn-to-learn, attention mechanism, feature modulation, concept reasoning, temporal abstraction.

Contents

List of Figures

List of Tables

List of Abbreviations

Adam	Adaptive Moment Estimation
AP	Average Precision
ASI	Adaptive Skip Intervals
BFA	Bayesian Fast Adaptation
BMAML	Bayesian Model-Agnostic Meta-Learning
BN	Batch Normalization
BNN	Bayesian Neural Network
CAM	Class Activation Mapping
CBAM	Convolutional Block Attention Module
CM	Concept Modulator
CNN	Convolutional Neural Network
CNP	Conditional Neural Process
CPPN	Compositional Pattern Producing Network
CR	Concept Reasoner
CS	Concept Sampler
DLN	Dynamic Layer Normalization
EGNN	Edge-labeling Graph Neural Network
ELBO	Evidence Lower Bound
ELU	Exponential Linear Unit
EMA	Exponential Moving Average
EMAML	Ensemble Model-Agnostic Meta-Learning
FiLM	Feature-wise Linear Modulation
FPN	Feature Pyramid Network
GAP	Global Average Pooling
GC	Global-Context

GE	Gather-Excite
GFLOPs	Giga Floating Point Operations
GloRe	Global Reasoning
GNN	Graph Neural Network
GPU	Graphics Processing Unit
GRU	Gated Recurrent Units
HMC	Hamiltonian Monte Carlo
HMM	Hidden Markov Model
HMRNN	Hierarchical Multiscale Recurrent Neural Network
HRSSM	Hierarchical Recurrent State Space Model
IoU	Intersection over Union
KEYIN	Key Frame Intermediate Model
K-FAC	Kronecker-factored Approximate Curvature
KL	Kullback-Leibler
LLAMA	Lightweight Laplace Approximation for Meta-Adaptation
LSTM	Long Short-Term Memory
MAML	Model-Agnostic Meta-Learning
MCMC	Markov Chain Monte Carlo
MCTS	Monte Carlo Tree Search
MLP	Multi Layer Perceptron
MoE	Mixture of Experts
MSE	Mean Squared Error
NAS	Neural Architecture Search
R-CNN	Region Based Convolutional Neural Network
ReLU	Rectified Linear Unit
REINFORCE	REward Increment = Nonnegativev Factor times Offset Reinforcement times Characteristic Eligibility
ResNet	Residual Network
RBF	Radial Basis Function
RGB	Red Green Blue
RL	Reinforcement Learning

RNN	Recurrent Neural Network
RSSM	Recurrent State Space Model
SE	Squeeze-and-Excitation
SGD	Stochastic Gradient Descent
SGLD	Stochastic Gradient Langevin Dynamics
SNAIL	Simple Neural Attentive Meta-Learner
SSM	State Space Model
SVGD	Stein Variational Gradient Descent
SVPG	Stein Variational Policy Gradient
tanh	Hyperbolic Tangent
TAP	Time-Agnostic Predictor
TD-VAE	Temporal Difference Variational Auto-Encoder
trn	Train
TRPO	Trust Region Policy Optimization
t-SNE	t-distributed Stochastic Neighbor Embedding
tst	Test
val	Validation
VCRNET	Visual Concept Reasoning Network
VHRED	Variational Hierarchical Recurrent Encoder-Decoder
VPG	Vanilla Policy Gradient
VQA	Visual Question Answering
VRNN	Variational Recurrent Neural Network
VTA	Variational Temporal Abstraction

Acknowledgments

I would like to first thank my advisor, Yoshua Bengio, for guiding me to become an independent researcher. Yoshua gave me the independence to do my research during my PhD, and this PhD thesis would not have been possible without his advice and support. It was an honor to be Yoshua's student for my PhD.

I was extremely lucky to be part of Mila with incredible colleagues, and I want to say thanks to all of them. I could not possibly have completed my long PhD journey without these colleagues.

During my PhD, I was fortunate to have several opportunities to work at many different research groups. First, I would like to thank Kakao Brain for giving me a wonderful experience to do amazing research with great people in Korea. Microsoft Research and Element AI also hosted me as a research intern, and I really enjoyed and learned a lot from both groups and want to say thanks. Lastly, Sungjin Ahn invited me as an academic visitor to his lab at Rutgers University for a year, and it was my pleasure to work with him.

Among many who lent help along the way, I will give my dearest thanks to my parents for their unwavering support. Finally, I would like to thank my wife, Minjung Kim, and my daughter, Loah Kim. Minjung endlessly supported my studies and dreams, and indeed I could not possibly have completed my PhD without her. Loah thankfully came to us at the end of my PhD, and she is my energy and my life. This PhD thesis is rightly dedicated to both my ladies. I love you and thank you so much.

Chapter 1

Introduction

Human intelligence can be simply described by a very general mental capability that involves the ability to reason, plan, and solve problems. By attempting to emulate this human intelligence with a large amount of data and sufficient computational resources, deep learning methods have shown significant practical successes in many different applications such as computer vision, speech recognition, natural language processing, and robotics. For example, convolutional neural networks may have the ability to recognize thousands of objects with superhuman accuracy (He et al., 2016; Simonyan and Zisserman, 2015; Tan and Le, 2019), and artificial agents can play many Atari games and also Go at the superhuman level (Mnih et al., 2016; Silver et al., 2016). In general, these methods are limited to the use of deep neural networks only to learn specific pre-defined settings or tasks without having any adaptability or flexibility. This can unfortunately lead to performance degradation due to unexpected changes in data or requiring additional computation to adapt to them.

Interestingly, in many different definitions of human intelligence, learning and adaption are also commonly and strongly emphasized as one of the defining features of it, and this implies that humans are mostly able to deal with the environment that is only partially known. In other words, humans can easily and quickly learn or adapt so as to perform as well as possible over a wide range of environments, situations, and tasks, even though they are unseen or unexperienced before (Gottfredson, 1997; Legg and Hutter, 2007). Inspired by this context of human intelligence, there have been many research efforts carried out to make deep neural networks adaptive and flexible in many different circumstances by introducing some adaptive and dynamic computations such as attention mechanisms (Bahdanau et al., 2015; Vaswani et al., 2017) or learning frameworks such as meta-learning (Bengio et al., 1991; Vinyals et al., 2016; Finn et al., 2017). In many practical applications, these are critical components to make deep neural networks be computationally efficient in terms of the learning and adaptation procedures and also be robust to unexpected circumstance changes.

This thesis explores and focuses deeply on this topic to answer the question about how deep neural networks can have the ability to quickly learn new tasks or easily adapt to different circumstances in computationally efficient ways. By investigating adaptation problems from different perspectives such as task-level, temporal-level, and context-level adaptations, the thesis attempts to provide specific solutions that apply to each of the perspectives.

Outline of Thesis. In Chapter 2, we explain adaptation methods in the deep learning literature and categorize them into three different categories corresponding to different perspectives such as (1) task-level, (2) temporal-level, and (3) context-level. It will provide the necessary background details on adaptation methods to understand this thesis. Moreover, the motivations are also provided to explain why adaptation methods are required in different settings and how they can improve deep neural networks in terms of not only prediction performance, but also computational efficiency. After that, each chapter covers an adaptation method that applies to a specific perspective on adaptation problems.

Chapter 3 discusses *task-level* adaptation based on a meta-learning framework. In particular, we investigate the uncertainty induced during fast-adaptation with a few examples and present a Bayesian meta-learning framework that is able to approximate the complex multimodal task-posterior distribution using particle filters.

In Chapter 4, temporal abstraction is discussed, which is referred to as *temporal-level* adaptation. To improve long-term future prediction, we introduce a jumpy prediction that can adaptively define the optimal step size to predict forward by learning a temporally hierarchical structure in sequence data.

Chapter 5, we investigate *context-level* adaptation particularly in visual recognition tasks. Attention mechanism and graph neural network are used to adaptively extract the global context, and feature-wise transformations give feedback to local descriptors to update them according to the global context.

Finally, the last chapter concludes the thesis by summarizing the contributions and outlining some future research directions in the context of this thesis.

Chapter 2

Background: Adaptation in Deep Learning

In this chapter, we provide the background on adaptation methods in deep learning, with deep neural networks specially designed to learn or adapt so as to perform as well as possible over a wide range of environments, situations, and tasks, even though they are unseen or unexperienced before. More specifically, we investigate adaptation problems from different perspectives such as task-level, temporal-level, and context-level by providing motivations and related works.

2.1. Task-Level Adaptation

The general setting in deep learning is to learn a single specific task from scratch using a supervised learning framework. It necessarily requires a large amount of data, which is expected to be collected or simulated with label information, and also huge computational power to train potentially large and deep neural networks. This severely constrains the performance of deep neural networks, although the goal is to learn only a single task. Moreover, in this setting, trained models are not able to seamlessly adapt to unexpected changes or rapidly learn new tasks using only a few examples. For example, robots in the real-world are typically deployed with pre-defined behaviors or skills to make them succeed in specific tasks or environments, and this leads to failure in unexpected changes. Another example is object detection systems that are only able to detect objects provided during the learning phase, and the whole system has to be retrained from scratch if a new class of objects is additionally provided to be detected as well.

In contrast, humans are successful in quickly adapting to new environments and learning new concepts and tasks. For example, two-year-old children can infer a new category from only one instance (Smith and Slone, 2017). This is presumed to be because during early learning a human brain develops foundational structures such as the "shape bias" in order to learn the learning procedure (Landau et al., 1988). This ability is also described as one of the defining features of human intelligence that humans can efficiently leverage knowledge

acquired from past experiences to easily learn new things with fewer examples or less trial-and-error (Jankowski et al., 2013; Lake et al., 2015) . It is known as *learning to learn* or *meta-learning* (Biggs, 1985; Bengio et al., 1990) in machine learning literature that has recently obtained much attention by formulating it as *few-shot learning* (Lake et al., 2015; Vinyals et al., 2016). In this thesis, we also describe this ability as *task-level adaptation*.

In the following section, we introduce meta-learning more formally and categorize meta-learning methods into (1) metric-based, (2) model-based, and (3) optimization-based methods. In particular, the methods are explained by focusing on the setting of few-shot learning.

2.1.1. Meta-Learning

Meta-learning provides an alternative learning paradigm that a machine learning model (in this thesis, we are specifically interested in deep neural networks) is trained by learning how to learn, where it acquires experiences from multiple tasks and utilizes them to improve the learning of unseen tasks. In other words, during the learning phase, the model extracts meaningful task-agnostic knowledge from a given distribution of tasks and learns how to leverage it to improve the efficiency of learning new tasks. In this setting, which is different from the general learning setting, the learning is primarily defined on task-level (e.g. episodes) rather than instance-level (e.g. data samples), and it is conducted in two different levels: (1) *inner*-level and (2) *outer*-level.

The inner-level can be exactly considered as the general learning setting that is defined to learn a single task τ (i.e. regular supervised learning) and it is described by optimizing:

$$\theta_\tau^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}\left(f_\theta, \mathcal{D}_\tau; \phi\right), \tag{2.1.1}$$

where the loss function $\mathcal{L}$ measures the prediction error of the model f_θ that is trained for the task τ with its dataset D_τ. This is optimized by the meta-knowledge ϕ that can be referred to as '*how to learn*' (Hospedales et al., 2020). It can be characterized not only by optimization methods such as initialization method for model parameters, cross-validation setting, and optimizer, but also by the type of models such as a model architecture. In the general learning setting, the meta-knowledge ϕ is hand-crafted or pre-specified without any learning procedure, but it can possibly be optimized through hyper-parameter searching.

One of the main goals of meta-learning is to find the optimal meta-knowledge ϕ^*, which is learned from multiple tasks, that can improve both data and computational efficiency of learning of new tasks. The meta-knowledge ϕ is therefore optimized at the outer-level as:

$$\begin{aligned} \phi^* &= \underset{\phi}{\operatorname{argmin}}\, \mathbb{E}_{\tau\sim p(\mathcal{T})}\left[\mathcal{L}^{\text{meta}}\left(f_{\theta_\tau^*(\phi)}, \mathcal{D}_\tau^{\text{val}}\right)\right] \qquad \text{(outer-level)}, \\ \text{where} \quad \theta_\tau^*(\phi) &= \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}\left(f_\theta, \mathcal{D}_\tau^{\text{trn}}; \phi\right) \qquad \text{(inner-level)}. \end{aligned} \tag{2.1.2}$$

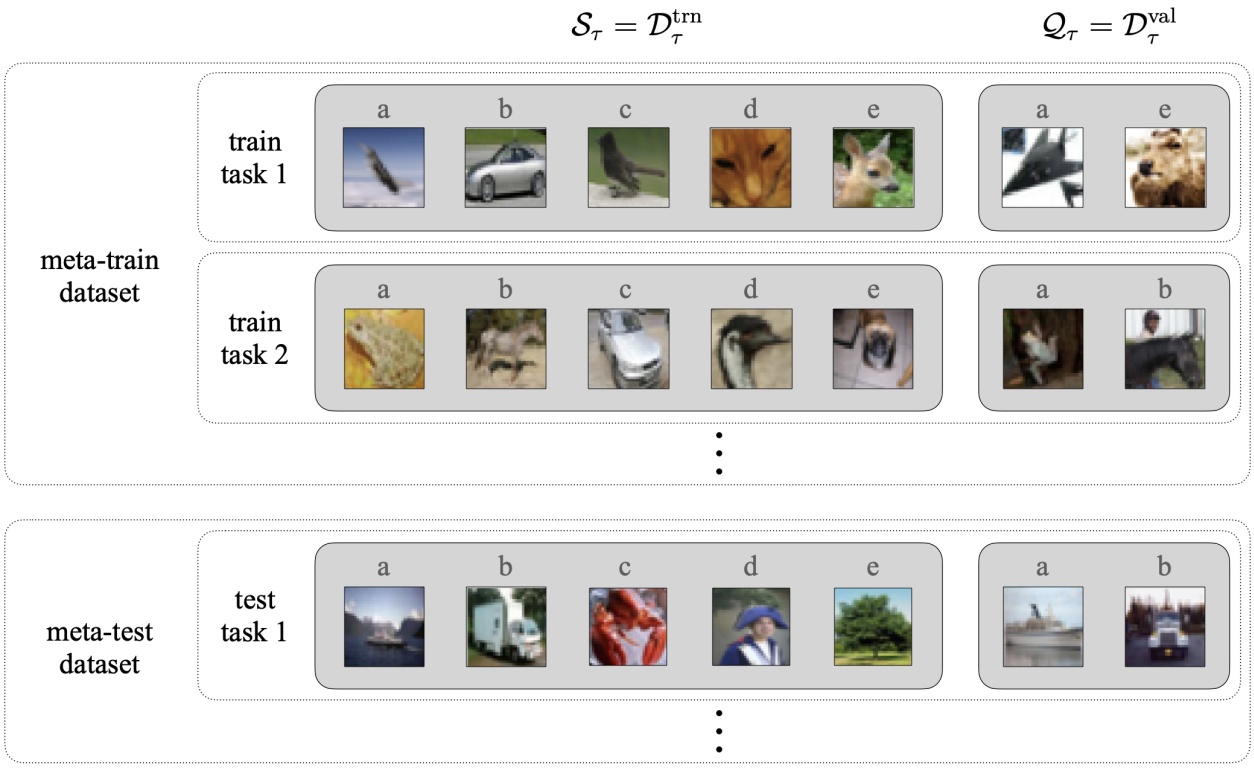


Fig. 2.1. Few-shot learning: N-way, K-shot classification ($N = 5, K = 1$). The dataset is defined on task-level, where each task consists of support and query set.

This optimizes the meta loss $\mathcal{L}^{\text{meta}}$ such as a generalization (validation) performance over multiple tasks $p(\mathcal{T})$, where each task τ is optimized in the inner-level. Note that this is considered as a bi-level optimization problem that the inner-level is conditioned on the meta-knowledge ϕ defined by the outer-level, and the outer-level is conversely conditioned on the models $\{\theta^*(\phi)\}_{\tau \sim p(\mathcal{T})}$ optimized by the inner-level. In general, both levels are sequentially optimized: the inner-level is first optimized over multiple tasks, and the outer-level is subsequently optimized upon the results from it, and this procedure is iteratively carried out until convergence. This optimization is referred to as a *meta-train* stage, where the meta-learning algorithm is mainly applied to learn how to learn. *Meta-validation* and *meta-test* stages are also defined to measure the meta-generalization performance to tune hyper-parameters and evaluate the final performance, respectively. Here, each stage is conducted by a separate dataset with disjoint tasks (e.g. labels), but all tasks are presumably sampled from the common task distribution $p(\mathcal{T})$. A general formulation of this meta-learning setting is few-shot learning, and we further describe meta-learning in the context of it.

Few-Shot Learning. Few-shot classification involves learning a classifier when only a few training samples per each class are given. Therefore, each few-shot classification task τ contains a *support set* $\mathcal{S}_\tau (= \mathcal{D}_\tau^{\text{trn}})$, a labeled set of input-label pairs, and a *query set*

$\mathcal{Q}_\tau (= \mathcal{D}_\tau^{\text{val}})$, an unlabeled set on which the learned classifier is evaluated. If the support set $\mathcal{S}_\tau$ contains K labeled samples for each of N unique classes, the problem is called *N-way K-shot* classification problem.

As an efficient way of meta-learning, *episodic training* (Santoro et al., 2016a; Vinyals et al., 2016) is commonly employed in the literature (Snell et al., 2017; Finn and Levine, 2017; Yang et al., 2018). Given a relatively large class-labeled training dataset, the idea of episodic training is to sample training tasks (episodes) that mimic the few-shot learning setting of test tasks. Since the distribution of training tasks is assumed to be similar to that of test tasks, the performances of the test tasks can be improved by learning a model to work well on the training tasks.

More concretely, in episodic training, all training, validation, and test datasets of the N-way K-shot problem are formed as follows so that each task τ has a dataset $\mathcal{D}_\tau$:

$$\begin{gathered} \mathcal{D}_\tau = \mathcal{S}_\tau \bigcup \mathcal{Q}_\tau \\ \text{where } \mathcal{S}_\tau = \mathcal{D}_\tau^{\text{trn}} = \{(x_i^\tau, y_i^\tau)\}_{i=1}^{N \times K} \text{ and } \mathcal{Q}_\tau = \mathcal{D}_\tau^{\text{val}} = \{(x_i^\tau, y_i^\tau)\}_{i=N\times K+1}^{N\times K+|\mathcal{Q}_\tau|}. \end{gathered} \tag{2.1.3}$$

Here, x_i^τ is the ith input data and $y_i^\tau \in \{C_1^\tau, \cdots, C_N^\tau\} = \mathcal{C}_\tau \subset \mathcal{C}$ is its label, where $\mathcal{C}$ is the set of all classes from training, validation, and test datasets. The classes from all datasets are mutually exclusive (i.e. $\mathcal{C}^{\text{train}} \cap \mathcal{C}^{\text{valid}} = \emptyset$, $\mathcal{C}^{\text{train}} \cap \mathcal{C}^{\text{test}} = \emptyset$, and $\mathcal{C}^{\text{valid}} \cap \mathcal{C}^{\text{test}} = \emptyset$).

2.1.2. Metric-Based Meta-learning

The metric-based meta-learning mainly aims to learn a meaningful feature space that can be useful to discriminate between different classes. That way, the meta-knowledge ϕ in Equation 2.1.2 is described by the *meta-learned* feature space, and the model parameter θ is non-parametric that is defined by samples (x_i^τ, y_i^τ) in the support set $\mathcal{S}_\tau$.

The few-shot classifier is based on a probability distribution over classes in task τ that is defined by the support set $\mathcal{S}_\tau$ as:

$$P(y|x, \mathcal{S}_\tau; \phi) \propto \sum_{(x_i^\tau, y_i^\tau) \in \mathcal{S}_\tau} k\left(f_\phi(x_i^\tau), f_\phi(x)\right) y_i^\tau, \tag{2.1.4}$$

where the label y_j^τ is one-hot encoded and the kernel $k(\cdot, \cdot)$ is defined by a similarity kernel, such as cosine similarity. The inner-level is straightforward and simply defines the classifier as described in Equation 2.1.4 and it doesn't require any optimization. The exact optimization only occurs at the outer-level, where the feature space is optimized with the similarity kernel to improve the meta-generalization. This simplicity is the main strength of metric-based approaches.

A siamese neural network (Bromley et al., 1993; Koch et al., 2015) is a simple example of metric-based methods that consists of feature extractor and distance layer to extract feature vectors and measure the similarity between a pair of them, respectively. Matching

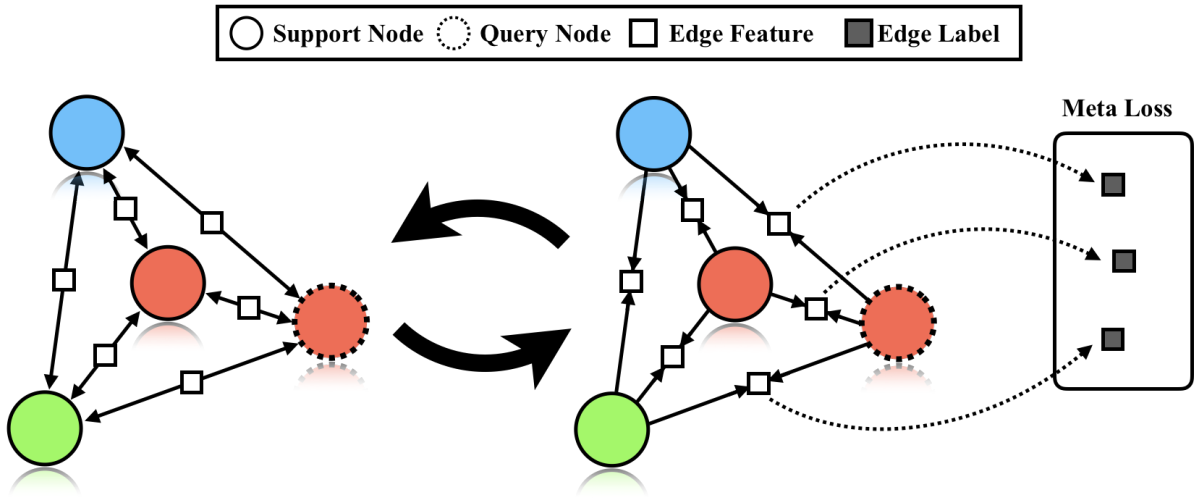


Fig. 2.2. Edge-labeling graph neural network (EGNN) with alternative node and edge feature updates. (source: Kim et al. (2019))

networks (Vinyals et al., 2016) are similar to siamese neural networks, but using an attention mechanism to compute normalized similarities, and the model is optimized by using the episodic training framework. Prototypical networks (Snell et al., 2017) derive class-wise prototypes (centroids), which are aggregated representations of classes, and the prediction becomes simpler as it is done by computing the similarity with N prototypes instead of $N \times K$ samples in the support set.

Graph neural networks (GNN) can be used to extract task-specific feature vectors by iteratively propagating task-specific information to the nodes in a graph (Garcia and Bruna, 2017). An edge-labeling GNN (EGNN, Kim et al. (2019)[2.1]) works similarly, but instead predicts the edge-labels in the graph by jointly learning the intra-cluster similarity and the inter-cluster dissimilarity. In this way, as shown in Figure 2.2, the edge features are adjusted reflecting both similarity and dissimilarity between nodes, and the node feature is also updated accordingly.

2.1.3. Model-Based Meta-Learning

Model-based meta-learning is based on models that are designed to rapidly adapt to a new task via some internal adaptive computations. More specifically, each task is encoded into some internal states capturing task-specific knowledge, and they are used to control the adaptive computations to enable fast task adaption. The internal states can be treated as

[2.1]"Edge-Labeling Graph Neural Network for Few-shot Learning". Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Appeared in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019)*. I contributed significantly to this article as the second author by implementing the codes and also formulating the proposed method.

the task-specific knowledge that is extracted from the support set $\mathcal{S}_\tau$. As the model-based approaches do prediction based on internal mechanisms, which are not externally exposed, they are also called black-box methods.

The few-shot classifier is directly defined by the model f_ϕ that is considered as a deep neural network parameterized by ϕ:

$$P(y|x, \mathcal{S}_\tau; \phi) = f_\phi\left(x; \mathcal{S}_\tau\right). \tag{2.1.5}$$

Inside the model f_ϕ, the support set $\mathcal{S}_\tau$ is used to extract task-specific knowledge, and this is internally combined and jointly processed with input data x to get the prediction. In this way, the inner-level is considered to only compute the internal states based on the support set, and the outer-level is to optimize the model parameter ϕ to improve the overall adaptation power. This introduces more flexibility and broader applicability of meta-learning than other metric-based and optimization-based methods.

Memory-augmented neural networks (Santoro et al., 2016b) are using an external memory with a controller to store and read task-specific knowledge that will be used for making predictions. Simple neural attentive meta-learner (SNAIL, Mishra et al. (2017)) also possess an external memory and consists of temporal convolutions and attention modules that are efficient to extract and read task-specific knowledge. In contrast to previous methods, conditional neural process (CNP, Garnelo et al. (2018)) and neural statistician (Edwards and Storkey, 2016) do not rely on external memory, but instead use a compact latent representation to encode task-specific knowledge.

2.1.4. Optimization-Based Meta-Learning

The core idea in optimization-based meta-learning is to improve task adaptation by learning the optimization procedures of it. The learned optimization procedures are considered as the meta-knowledge ϕ and are expected to be simpler and more computationally efficient than the general gradient-based optimizer. In general, optimization-based methods are exactly based on the bi-level optimization as described in Equation 2.1.2. The inner-level is defined to derive task-specific models with the learned optimizer ϕ, which is optimized over multiple tasks at the outer-level, and the classifier is defined as:

$$P(y|x, \mathcal{S}_\tau; \phi) = f_{\theta(\mathcal{S}_\tau, \phi)}\left(x\right) \tag{2.1.6}$$

where the model parameter $\theta(\mathcal{S}_\tau, \phi)$ is derived by the learned optimization method ϕ with the support set $\mathcal{S}_\tau$. This leads to achieving better generalization than other metric-based and model-based methods (Finn and Levine, 2017), but it is computational less efficient as task-specific models are only obtainable through the learned optimization procedure.

The LSTM meta-learner (Ravi and Larochelle, 2017) is a good example of optimization-based method that explicitly learns a meta-optimizer. It is based on an LSTM recurrent

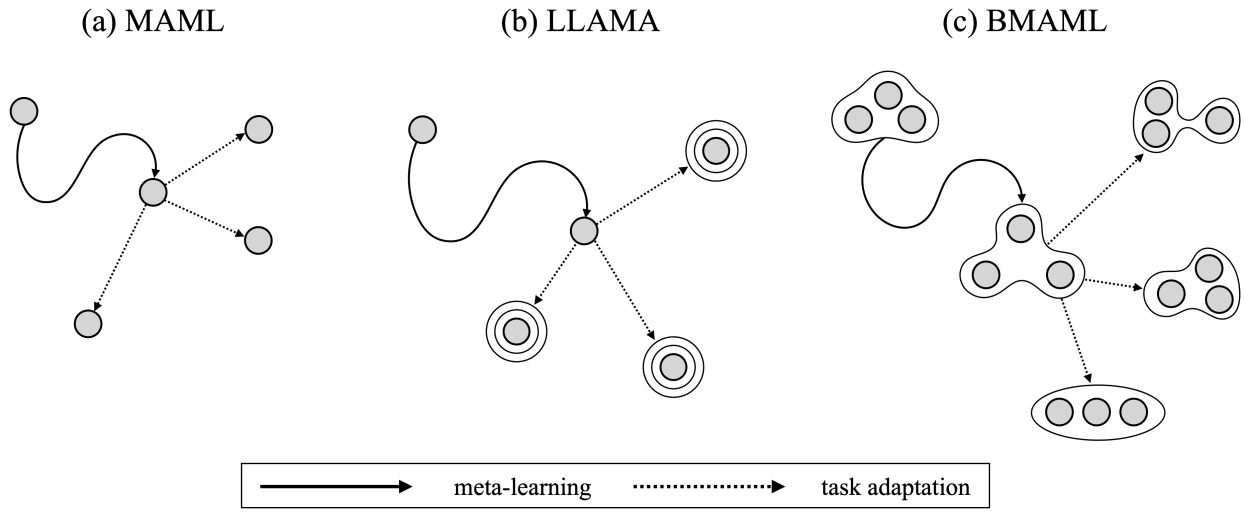


Fig. 2.3. Comparison between optimization-based meta-learning methods based on MAML: all methods learn initialization parameters, but differently handle uncertainty. (a) MAML: point estimate, (b) LLAMA: Gaussian approximation, (c) BMAML: complex multimodal

network, where the cell state defines the task-specific model parameters θ, and its update rule is used to optimize task-specific models. Model-agnostic meta-learning (MAML, Finn et al. (2017)) is a fairly general optimization-based method as it is simple and generally applicable to any model as long as the gradient can be estimated. It works purely by gradient-based optimization without requiring any additional parameter or model modification. Reptile (Nichol et al., 2018) is another optimization-based method similar to MAML, but attempts to learn a good initialization of the model parameters θ in a simpler way.

In Chapter 3, we propose an extended version of MAML, Bayesian model-agnostics meta-learning (BMAML), that improves the performance by taking into account the uncertainty induced during fast task adaptation with a few examples. Instead of approximating the task-posterior with point estimates or local Laplace approximation (LLAMA, Grant et al. (2018)), it is able to approximate the complex multimodal task-posterior using particles, as illustrated in Figure 2.3

2.2. Temporal-Level Adaptation

Sequential data includes text streams, audio and video clips, and also any type of time-series data and it is becoming more ubiquitous in a wide spectrum of application scenarios. Most commonly, it is assumed to be collected or sampled at successive equally spaced points in time and the data points or frames are indexed in time order.

Many practical applications including speech recognition, video prediction, machine comprehension, and many others are fundamentally based on the definition of sequential problems that are formulated as sequential data processing. That way, a system is designed to receive

sequence data with an arbitrary length and process input data x_t at every time step t and also optionally produce prediction data y_t if necessary. Here, the prediction data y_t can be a word label as in language modeling, image data as in video prediction, or an action in the context of sequential decision making based on reinforcement learning. Both input data x_t and output data y_t might depend on any of the previous time step $t' < t$ or even all of the previous time steps. This makes the sequential problems particularly more challenging than the standard prediction problems handling non-sequential data. It is therefore important to learn and understand the dynamic behavior and the causality relationships across sequential data.

In deep learning, recurrent neural networks (RNNs) are a type of neural network architectures that are a temporal generalization of classical feed-forward neural networks. RNNs have some hidden-to-hidden recurrent connections that enable them to aggregate information over time and store it in the hidden state acting as a memory. In general, all input data in a sequence are sequentially processed through RNNs one by one and summarized by iteratively updating the hidden state in RNNs. However, as the length of the sequence increases, this setting can unfortunately cause some potential problems degrading the prediction performance.

- Due to the problem of vanishing and exploding gradients, RNNs can fail to properly capture long-term dependencies across sequential data (Hochreiter, 1991; Bengio et al., 1994).
- The issue of error accumulation can become severe in some sequence prediction problems, when the output influences or is the next input (e.g. video prediction, speech recognition/synthesis, machine translation, long-term forecast).
- The computational cost linearly increases with the length of the sequence.

There exist some solutions for these problems. For example, certain recurrent architectures such as long short-term memory networks (LSTM, Hochreiter and Schmidhuber (1997)) and gated recurrent units (GRU, Cho et al. (2014)) can mitigate the problem of vanishing and exploding gradients by controlling the information flow by gating units. By modifying the training process or objective of RNNs, the error accumulation can also be alleviated, and this results in more robust long-term predictions (Bengio et al., 2015b; Lamb et al., 2016). The computational cost clearly depends on the length of the sequence, and therefore a solution is to reduce the length of it by skipping some steps or frames under certain rules.

In this thesis, we investigate these problems by introducing the concept of *temporal-level adaptation*. This requires learning the underlying structure and dynamics in the sequence so that we can expect models that can adaptively skip some unnecessary steps to process and predict only necessary steps. This is also referred to as temporal abstraction, which is inspired by human intelligence and adaptability. For example, humans are able to do skim reading that fixates on some words and skips others during reading some long text. Another

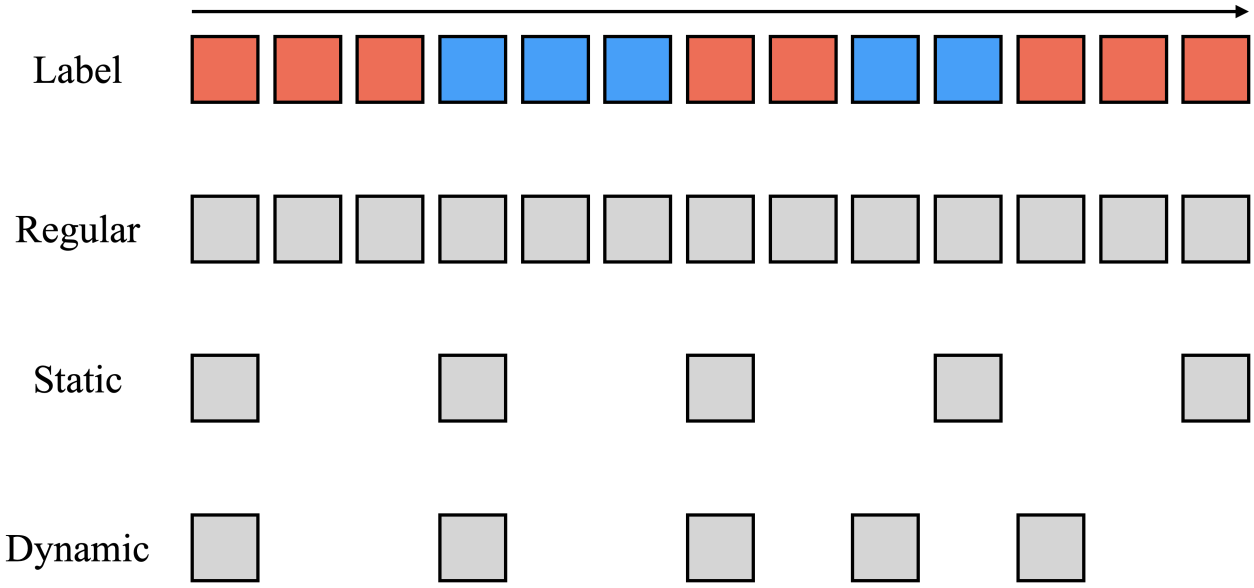


Fig. 2.4. Static vs. dynamic frame skipping. Dynamic frame skipping is proposed to adaptively skip frames by ensuring the alignment with the temporal structure in the label sequence.

example is that making a plan for one-hour driving from one point to another does not involve predictions at the scale of every second. These examples suggest that we should look for models that can do temporal-level adaptation to improve the processing of sequential data.

In the following sections, we introduce two methods, (1) adaptive skipping and (2) jumpy future imagination, that are based on the temporal-level adaptation.

2.2.1. Adaptive Skipping

In this section, we introduce methods that learn to adaptively skip steps or frames in sequence data or reduce the length of it in other ways. Especially for speech recognition systems, it is critical to reducing the computational cost as it is generally used in real-time applications or on mobile devices. Frame skipping has been broadly used in neural acoustic models by lowering the frame rate and interpolating the predicted labels in the sequence (Vanhoucke et al., 2013; Miao et al., 2016). This approach is referred to as *static* frame skipping, where the skip size is predefined and fixed without considering any information variability in speech signals as shown in Figure 2.4. It can reduce the computational cost but can not avoid performance degradation. The more adaptive strategy is *dynamic* frame skipping (Song et al., 2018) [2.2]. A neural acoustic model using dynamic frame skipping

[2.2]"Dynamic Frame Skipping for Fast Speech Recognition in Recurrent Neural Network Based Acoustic Models". Inchul Song, Junyoung Chung, Taesup Kim, and Yoshua Bengio. Appeared in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*. I contributed as the main proposer of using the concept of adaptive frame skipping.

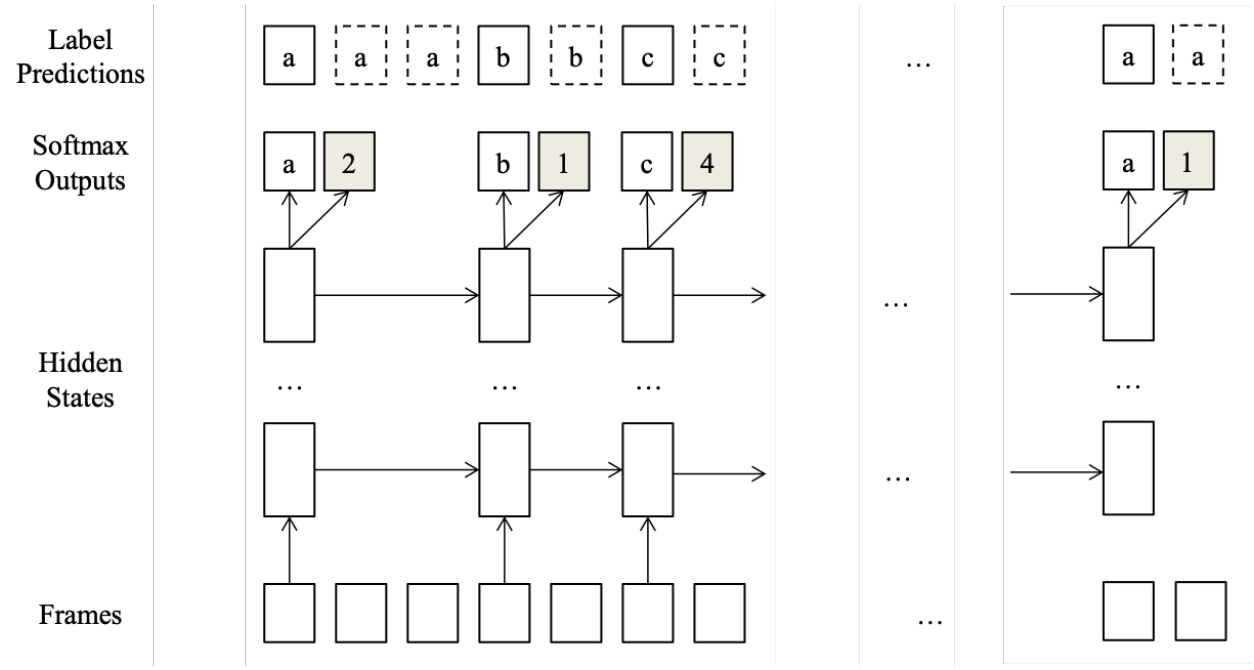


Fig. 2.5. Dynamic frame skipping is based on a skip-policy network. It is implemented on the last layer of RNNs so that the model can jointly predict the label and also the size of a segment (i.e. skip size). (source: Song et al. (2018))

can adaptively skip frames by predicting label changes as shown in Figure 2.4. It is based on a skip-policy network that samples the skip size as shown in Figure 2.5 and is jointly trained with the neural acoustic model. It tries to avoid reading redundant frames, and therefore the computational cost can be reduced while preserving performance.

The concept of dynamic frame skipping is also used in reinforcement learning particularly for deep Q-networks (Srinivas et al., 2016). Here, the Q-network is designed with a larger action space, where the actions are decomposed into an executive action and the duration (i.e. repeating times) of it, and thus can improve the computational efficiency by avoiding the redundant action sampling procedure.

In a similar way, several methods have been proposed in natural language processing to learn to skim as humans do during reading long texts. It is based on making a sequence of decisions whether to fixate or skip words in the text. Hahn and Keller (2016) introduced a method that trains a model in a completely unsupervised manner only requiring unlabeled text during training. Here, an attention network is defined to sample binary actions that indicate whether to read or skip each of the words in the text sequence. This can reduce the number of updates in RNNs, but the binary action has to be sampled at every step. The overall network is based on an auto-encoder framework with an objective to minimize the reconstruction cost and also maximize the number of skipped words. Another work is based on a strategy that learns when to skim and when to read (Johansen and Socher, 2017), but

does not use a skipping mechanism. It is based on two models having different computational complexities, and they are adaptively used not only to reduce the computational cost but also to improve prediction power. If the given text sequence is determined to be easily processed, a computationally cheap bag-of-words model is used to process it as a form of skimming. Otherwise, a model based on RNNs is used to carefully read words one by one. Lei et al. (2016) use a hard attention mechanism to select candidate rationales from a text and only take them for the final prediction rather than using the full text. This can be treated as a pre-processing to reduce the size of the text, and it also provides some interpretability of the prediction process.

2.2.2. Jumpy Future Imagination

For an intelligent agent exploring complex environments, we assume that it should have an imagination module or a mental simulator of the environment to act and plan in a model-based fashion. It is critical to learn and operate on compact and abstract state representations to make predictions about the future of the environment efficiently and accurately. However, in general, the planning based on long-term predictions is implemented in a way that all future moments or steps are sequentially predicted one by one without any temporal abstraction. This is not cognitively and computationally sensible and not only increases computational cost but also degrades prediction performance.

In this section, we provide more explanations of models that can make *jumpy* predictions to improve long-term predictions with less computational complexity. Temporally sub-sampled sequence data can be used to build state-space models with jumpy state transitions (Buesing et al., 2018), but it only works for a fixed (static) jump size and performs poorly due to dropping the information contained in the skipped steps. The temporal difference variational auto-encoder (TDVAE, Gregor et al. (2019)) is similarly based on stochastic state-space models, but with a modified evidence lower bound (ELBO). It is formulated with a deterministic belief state to ensure that all the observed information is being used during prediction. Moreover, temporal abstraction is realized in the model by extending the loss to relate not only neighboring time steps but also arbitrarily distant time steps in the sequence. Although this allows the model to predict arbitrary future steps that are defined in some finite range, the optimal step size at each time step cannot be obtained and therefore has to be preset for the future prediction.

Both adaptive skip intervals (ASI, Neitz et al. (2018)) and time-agnostic predictors (TAP, Jayaraman et al. (2019)) allow the model to dynamically adjust the step size based on the input data by defining the target as the most predictable future frame. The models learn to converge to nearly deterministic state transitions that are the easiest to predict and also considered to have the lowest uncertainty (see Figure 2.6). This is implemented by using a

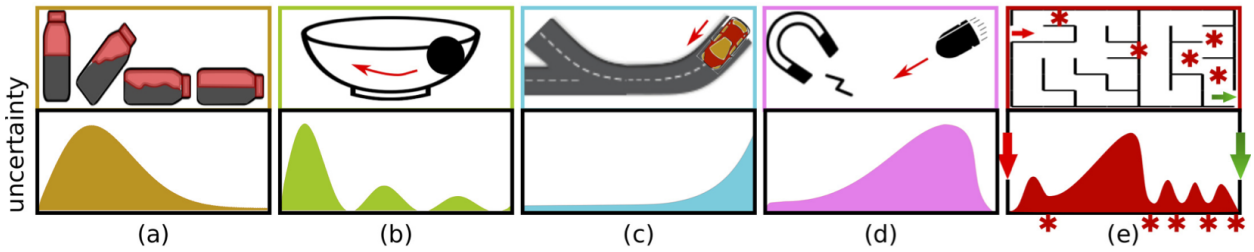


Fig. 2.6. Uncertainty profiles corresponding to various scenarios. Adaptive skip intervals (ASI) and time-agnostic predictors (TAP) are designed to learn transitions between moments with less uncertainty. (source: Jayaraman et al. (2019))

minimum-over-time loss so that the penalty for predictions $G(x_t)$ is adaptively determined based on its closest matching over a set of future time steps $\{t+\delta\}_{\delta \in T=[1,2,\ldots]}$:

$$G^* = \underset{G}{\operatorname{argmin}}\, \mathcal{L}\left(G\right) = \underset{G}{\operatorname{argmin}}\, \mathbb{E}_{x_t}\left[\min_{\delta \in T} d_x\left(G(x_t), x_{t+\delta}\right)\right] \tag{2.2.1}$$

where the function d_x measures a distance between prediction $G(x_t)$ and $x_{t+\delta}$ such as the mean squared error (MSE). Additionally, ASI uses an exploration curriculum strategy to soften the winner-takes-all optimization setting. However, both approaches are based on predictions directly on the observation-level without learning state representations and this potentially cannot capture the full context from the previous observations.

A model can also be designed by a different perspective, where sequence data can be decomposed into a set of time steps representing moments of interest, which are highly stochastic, and other steps in between them as described in Figure 2.7. That way, a hierarchical keyframe intermediate model (KEYIN, Pertsch et al. (2019)) stochastically predicts keyframes in image sequences and then uses these predicted keyframes to deterministically predict the intermediate frames.

In Chapter 4, we propose a hierarchical recurrent state-space model, Variational Temporal Abstraction (VTA), that is also based on the same assumption of KEYIN. It is a fully stochastic sequence model that is able to discover the latent temporal abstraction structure by decomposing the sequence data into non-overlapping subsequences. The inference of the hierarchical state representations, which are decomposed into frame-level and subsequence-level, is conducted by using the discovered temporal structure. The jumpy future prediction is then implemented by the state transitions, where the states represent subsequences.

2.3. Context-Level Adaptation

Deep neural networks are mainly composed of two types of variables that are activation and weight variables. The activation variables represent the internal status or activities based on the input data and are dynamically computed or extracted rather than being learned. On the other hand, in general, the weight variables are learned to capture consistent regularities

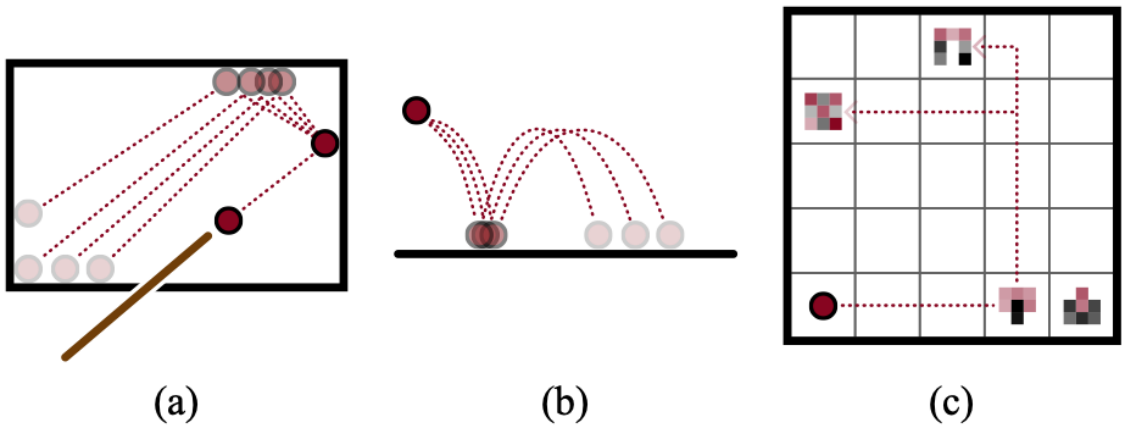


Fig. 2.7. Sequence data can be decomposed into subsequences that are defined by some keyframes with high uncertainty. Trajectories with dotted lines are considered to be nearly deterministic. The model then learns to do jumpy prediction by discovering keyframes and learning the state transitions between them. (source: Pertsch et al. (2019))

among input data, output data, and some others. They are treated as static variables that are not modified after the learning phase, and this is based on the assumption that the learned regularities can be well-generalized also to unseen data. However, input data may have many different characteristics that have to be processed in different ways. This suggests that deep neural networks might benefit from some weight variables that can adaptively change as the activation variables. This concept is also known as *fast weights* (Schmidhuber, 1992, 1993), where one network dynamically generates context-dependent weight variables for the main target network. In this thesis, we define this concept as *context-level adaptation* and investigate methods that are able to dynamically modify the weight variables or select the subset of them to rapidly adapt to some conditional context. Here, the context can be defined by some auxiliary conditional data or extracted from input data.

A mixture of experts (MoE, Jacobs et al. (1991)) takes advantage of using multiple models (i.e. experts), where each model is assumed to respond to particular characteristics with greater specialization. Most importantly, a set of experts is differently and adaptively combined based on input data so that the aggregated output data is obtained in an adaptive manner. Here, the weighting factors of experts are considered as fast weights that realize the context-level adaptation. Conditional computation (Bengio et al., 2013, 2015a; Shazeer et al., 2017) is a special case of MoE, where the weighting factors of experts are sparse and discrete to ensure that only a few expert models are activated. This is to efficiently increase the total model capacity without a proportional increase in computation.

Attention mechanism and feature-wise transformation are more generalized frameworks of context-level adaptation that are broadly used with deep neural networks in many application settings, and in the following sections, we focus on these methods by providing more details and related works.

2.3.1. Attention Mechanism

Standard deep neural networks can be considered computationally expensive and inefficient, because they examine all hidden units or all local positions at the same level of detail. Although such brute-force approaches have shown impressive results so far, they are less interpretable and potentially become ineffective as both network and dataset sizes rapidly increase. Interestingly, humans are able to process a large amount of visual information in sequence through a visual system that is based on a biological *attention* mechanism (Rensink, 2000; Shulman and Corbetta, 2002). Rather than processing an entire image simultaneously, it selectively attends to the most informative parts of it. In other words, it dynamically allocates computation resources to the most informative parts.

Inspired by this human visual system, there have been many ways to integrate attention mechanisms into deep neural networks. In general, attention modules are designed to aggregate or summarize a set of information (e.g. representations) by differently weighting them, where the attention weights are adaptively computed by some conditional data, and learned in an end-to-end manner. More specifically, the weights can be deterministically computed in a soft way using a softmax function, and this allows the model to be easily optimized with back-propagation as usual. Otherwise, the weights are binary random variables that are stochastically sampled, and this makes the optimization difficult requiring some tricks such as REINFORCE (Williams, 1992), straight-through (ST, Bengio et al. (2013)) and categorical reparameterization (Jang et al., 2017; Maddison et al., 2017).

In the context of neural machine translation, the decoder generates each target word differently and adaptively aggregating the encoded source sequence (Bahdanau et al., 2015). That way, the alignment between source and target sequences can be learned in an adaptive manner that is described by attention weights. Also in computer vision problems such as image captioning (Xu et al., 2015), the source image is iteratively attended in different ways, where different objects are being focused at different steps, to generate a caption in a sequential manner.

Attention mechanisms furthermore have become an integral part of the compelling sequence and image models in various tasks that can efficiently model the internal dependencies in an adaptive manner without regard to their distance in the data. Especially, self-attention modules are mostly used to process input data $X \in \mathbb{R}^{N \times d}$, where N is the number of elements (e.g. the number of pixels or the sequence length) and d is the feature size, by using the following operation:

$$\begin{gathered} \text{Attention}\left(K, Q, V\right) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \\ \text{where } K = XW^k, Q = XW^q, \text{ and } V = XW^v. \end{gathered} \tag{2.3.1}$$

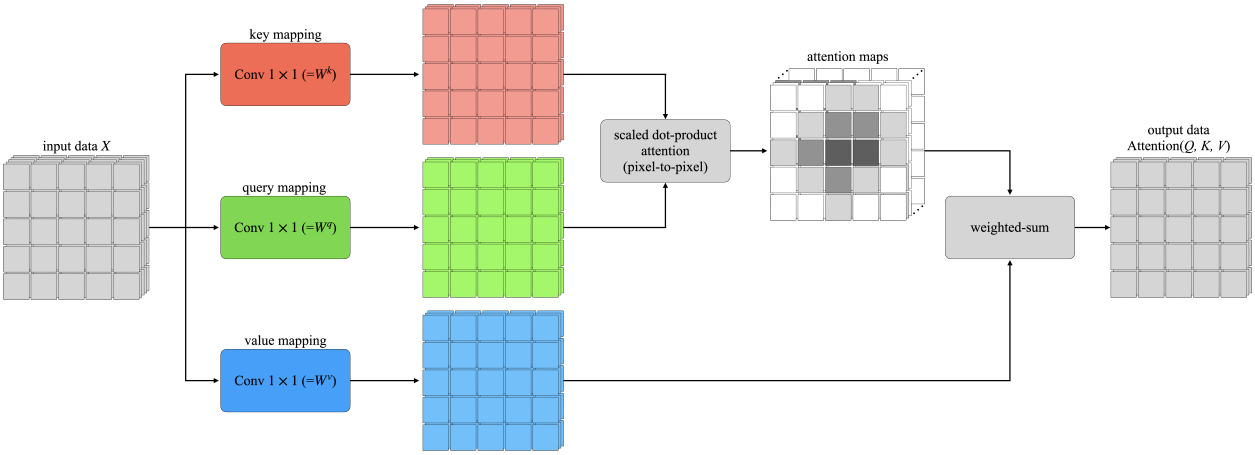


Fig. 2.8. Self-attention for image data. Input data X is mapped to keys K, queries Q, and values V by using 1×1 convolutions. Attention maps are computed for each of pixels and used to compute output data.

The weights $W^k, W^q \in \mathbb{R}^{d \times \tilde{d}}$, and $W^v \in \mathbb{R}^{d \times \hat{d}}$ are learned linear transformations that map input data X into keys K, queries Q, and values V (see Figure 2.8).

The transformer (Vaswani et al., 2017) is the model that entirely consists of self-attention operations to compute representations without using any typical sequence-aligned recurrent neural networks or convolution neural networks with local connectivity. Its advantages are characterized not only by capturing the long-term dependencies, but also by relaxing the weight-sharing property in RNNs and CNNs. In this way, transformers are broadly used in various tasks by taking the place of RNNs and CNNs (Parmar et al., 2018; Dong et al., 2018; Devlin et al., 2019; Bello et al., 2019; Dosovitskiy et al., 2020; Carion et al., 2020).

2.3.2. Feature-wise Transformations

In this section, we investigate a specific type of model that defines context-based processing by doing feature modulation based on information extracted from some conditional data (i.e. the conditional context data). In general, context-based computations can be implemented by simply concatenating, adding, or multiplying the conditional context data to hidden representations in the model. This can be further generalized into a single unified framework as a conditional affine transformation, which is also well known as *feature-wise transformations*, that was shown in feature-wise linear modulation (FiLM, Perez et al. (2018)).

$$\tilde{h} = \alpha(c) \odot h + \beta(c) \tag{2.3.2}$$

where the hidden representation $h \in \mathbb{R}^d$ is scaled and shifted by $\alpha(c) \in \mathbb{R}^d$ and $\beta(c) \in \mathbb{R}^d$, respectively, before a non-linearity is applied. Moreover, both α and β are functions of the conditional context data c that directly generates affine parameters. We have two networks

that one is the main (target) network and the other is the parameter generating (auxiliary) network. That way, in the Hyper-NEAT framework (Stanley et al., 2009), compositional pattern producing networks (CPPNs) are evolved to generate the weight structure of the much larger main network. In a similar way, Ha et al. (2016) introduced a hyper-network, which particularly generates some parameters in recurrent networks to relax their weight sharing property, that is jointly trained with the main network in an end-to-end manner. Feature-wise transformations have been broadly applied in many different problem settings thanks to their simplicity and effectiveness. In the following, we explain some examples in the deep learning literature.

In neural style transfer, a style transfer network is conditioned on a chosen style image, and the network has to learn the style-specific parameters of it. Instead of learning them, a conditional instance normalization (Dumoulin et al., 2017), which is based on the feature-wise transformations, allows the style transfer network to use dynamically generated affine parameters that are based on the embedding of the chosen style image.

Visual reasoning models are generally conditioned on linguistic questions to obtain the corresponding answers, and there are a series of works that showcased the effectiveness of the feature-wise transformations. Conditional batch normalizations (de Vries et al., 2017) are implemented in a pre-trained residual network (ResNet) to modulate convolutional feature maps by generating the related parameters conditioned on a linguistic embedding to solve visual question answering (VQA) tasks. Furthermore, Perez et al. (2018) proposed FiLM that is defined in a more generalized way.

Similarly, but in a self-conditioning manner, typical convolutional neural networks such as ResNet have been improved by augmenting some modules that are able to adaptively modulate feature responses. For example, squeeze-and-excitation (SE) blocks (Hu et al., 2018) utilize global information, which is simply extracted from a pooling operation, to adaptively adjust inter-dependencies between channels by implementing a self-gating mechanism on the channel-wise feature maps. Moreover, convolutional block attention modules (CBAM, Woo et al. (2018)) adaptively modulate the feature maps along not only a channel dimension but also spatial dimensions.

In speech recognition, Kim et al. (2017)[2.3] introduced the feature-wise transformations into neural acoustic modeling to efficiently handle acoustic variabilities arising from various factors such as speakers, channel noises, and environments. These variabilities are generally handled by explicitly learning them from some auxiliary labeled data (e.g. speaker representation and identity). This inefficiency is resolved by implementing the dynamic layer

[2.3]"Dynamic Layer Normalization for Adaptive Neural Acoustic Modeling in Speech Recognition". Taesup Kim, Inchul Song, and Yoshua Bengio. Appeared in *Proceedings of Conference of the International Speech Communication Association (INTERSPEECH 2017)*.I contributed as the first author that proposed the main concept and implemented the proposed method.

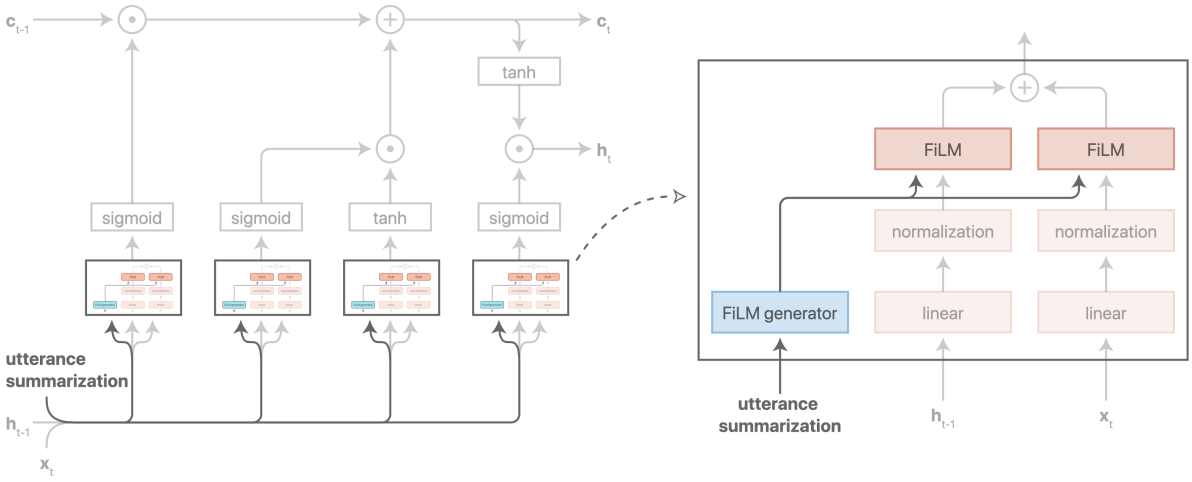


Fig. 2.9. Dynamic layer normalization (DLN) for adaptive neural acoustic modeling (Kim et al., 2017). The utterance summarization feature is used as the conditional context data and it modulates the hidden states in LSTM cells. (source: Kim et al. (2017) and Dumoulin et al. (2018))

normalization (DLN) in deep bidirectional LSTM networks as depicted in Figure 2.9. It does not require any auxiliary or external data, but internally extracts an utterance summarization feature to dynamically adjust the main network in a self-conditioning manner.

In Chapter 5, we also propose a convolutional neural network, Visual Concept Reasoning Networks (VCRNet), that is based on both attention mechanisms and feature-wise transformations to further enhance the ability of context-level adaption. This network is able to dynamically extract the global context in the input image and adaptively refine the local features based on it. That way, it makes the network do some kind of visual concept reasoning that is based on interactions between high-level visual concepts.

Chapter 3

Bayesian Model-Agnostic Meta-Learning

Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio and Sungjin Ahn

Appeared in:

- *Proceedings of Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*

Personal Contribution. Sungjin Ahn initially proposed an idea to combine gradient-based meta-learning (MAML) with nonparametric variational inference (SVGD) to model uncertainty during meta-learning, and he also set up the initial framework of it. To make this work, I and Sungjin Ahn came up with several types of novel meta-loss functions, which was also discussed with Sungwoong Kim, and I implemented all related codes to run all experiments to validate those loss functions. Based on the initial experimental results, we formulated a chaser loss as our meta-loss shown in the paper. I particularly ran all supervised learning and active learning experiments based on it. Ousmane Dia supported us by initiating the experimental setup at the beginning to make the project move forward smoothly. Jaesik Yoon, who is also the first and equal author/contributor, mainly focused on the reinforcement learning part, and he made our method properly work with reinforcement learning tasks. He also ran all those related experiments by himself. I also participated heavily in the writing with Sungjin Ahn, who contributed significantly to the introductory part, and Yoshua Bengio did the final editing. Furthermore, this paper was selected as a spotlight presentation, and I mainly prepared and gave the presentation for it.

Context. Due to inherent model uncertainty, learning to infer a Bayesian posterior from a few-shot dataset is an important step towards robust meta-learning (i.e *task-level adaptation*). However, at that moment, most of the related works were focusing on the meta-learning learning framework without considering the uncertainty. For this reason, we proposed a novel Bayesian model-agnostic meta-learning method. The proposed method combines efficient gradient-based meta-learning with nonparametric variational inference in a principled probabilistic framework. Unlike previous methods, during fast adaptation, the

method is capable of learning complex uncertainty structure beyond a simple Gaussian approximation, and during meta-update, a novel Bayesian mechanism prevents meta-level overfitting. As our method remains a gradient-based method, it is also a Bayesian model-agnostic method applicable to various tasks including reinforcement learning. Experimental results show the accuracy and robustness of the proposed method in sinusoidal regression, image classification, active learning, and reinforcement learning.

Keywords: meta-learning, few-shot learning, fast task adaptation, uncertainty, Bayesian inference, MAML, SVGD

3.1. Introduction

Two-year-old children can infer a new category from only one instance (Smith and Slone, 2017). This is presumed to be because during early learning, a human brain develops foundational structures such as the "shape bias" in order to learn the learning procedure (Landau et al., 1988). This ability, also known as *learning to learn* or *meta-learning* (Biggs, 1985; Bengio et al., 1990), has recently obtained much attention in machine learning by formulating it as few-shot learning (Lake et al., 2015; Vinyals et al., 2016). Because, initiating the learning from scratch, a neural network can hardly learn anything meaningful from such a few data points, a learning algorithm should be able to extract the statistical regularity from past tasks to enable warm-start for subsequent tasks.

Learning a new task from a few examples inherently induces a significant amount of uncertainty. This is apparent when we train a complex model such as a neural network using only a few examples. It is also empirically supported by the fact that a challenge in existing few-shot learning algorithms is their tendency to overfit (Mishra et al., 2017). A robust meta-learning algorithm therefore must be able to systematically deal with such uncertainty in order to be applicable to critical problems such as healthcare and self-driving cars. Bayesian inference provides a principled way to address this issue. It brings us not only robustness to overfitting but also numerous benefits such as improved prediction accuracy by Bayesian ensembling (Balan et al., 2015), active learning (Gal et al., 2016), and principled/safe exploration in reinforcement learning (Houthooft et al., 2016). Therefore, developing a Bayesian few-shot learning method is an important step towards robust meta-learning.

Motivated by the above arguments, in this paper we propose a Bayesian meta-learning method, called Bayesian MAML. By introducing Bayesian methods for both fast adaptation and meta-update, the proposed method learns to quickly obtain an approximate posterior of a given unseen task and provides the benefits of having access to uncertainty. Being an efficient and scalable gradient-based meta-learner which encodes the meta-level statistical regularity in the initial model parameters, our method remains a model-agnostic method applicable to various tasks including reinforcement learning. Combining an efficient non-parametric variational inference method with gradient-based meta-learning in a principled probabilistic framework, it can learn complex uncertainty structures while remaining simple to implement.

The main contributions of the paper are as follows. We propose a novel Bayesian method for meta-learning. The proposed method is based on a novel Bayesian fast adaptation method and a new meta-update loss called the Chaser loss. To our knowledge, the Bayesian fast adaptation is the first in meta-learning that provides flexible capability to capture the complex uncertainty curvature of the task-posterior beyond a simple Gaussian approximation. Furthermore, unlike the previous methods, the Chaser loss prevents meta-level overfitting. In

experiments, we show that our method is efficient, accurate, robust, and applicable to various problems: sinusoidal regression, image classification, reinforcement learning, and active learning.

3.2. Preliminaries

Consider a model $y = f_\theta(x)$ parameterized by and differentiable w.r.t. θ. Task τ is specified by a K-shot dataset $\mathcal{D}_\tau$ that consists of a small number of training examples, e.g., K pairs (x_k,y_k) per class for classification. We assume that tasks are sampled from a task distribution $\tau \sim p(\mathcal{T})$ such that the sampled tasks share the statistical regularity of the task distribution. A meta-learning algorithm leverages this regularity to improve the learning efficiency of subsequent tasks. The whole *dataset of tasks* is divided into training/validation/test *tasksets*, and the dataset of each task is further divided into task-training/task-validation/task-test *datasets*.

Model-Agnostic Meta Learning (MAML) proposed by Finn et al. (2017) is a gradient-based meta-learning framework. Because it works purely by gradient-based optimization without requiring additional parameters or a problem specialized model, it is simple and generally applicable to any model as long as the gradient can be estimated.

Algorithm 3.1 MAML

Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
for each task $\tau \in \mathcal{T}_t$ **do**
  $\theta_\tau \leftarrow \mathrm{GD}_n(\theta_0; \mathcal{D}_\tau^{\mathrm{trn}}, \alpha)$
end for
$\theta_0 \leftarrow \theta_0 - \beta \nabla_{\theta_0} \sum_{\tau \in \mathcal{T}_t} \mathcal{L}(\theta_\tau; \mathcal{D}_\tau^{\mathrm{val}})$

In Algorithm 3.1, we briefly review MAML. At each meta-train iteration t, it performs:

(1) *Task-Sampling*: a mini-batch $\mathcal{T}_t$ of tasks is sampled from the task distribution $p(\mathcal{T})$. Each task $\tau \in \mathcal{T}_t$ provides task-train data $\mathcal{D}_\tau^{\mathrm{trn}}$ and task-validation data $\mathcal{D}_\tau^{\mathrm{val}}$.

(2) *Fast Adaptation* (or *Inner-Update*): the parameter for each task τ in $\mathcal{T}_t$ is updated by starting from the *current* generic initial model θ_0 and then performing n gradient descent steps on the task-train loss, an operation which we denote by $\mathrm{GD}_n\left(\theta_0; \mathcal{D}_\tau^{\mathrm{trn}}, \alpha\right)$ with α being a step size.

(3) *Meta-Update* (or *Outer-Update*): the generic initial parameter θ_0 is updated by gradient descent. The meta-loss is the summation of task-validation losses for all tasks in $\mathcal{T}_t$, i.e., $\sum \mathcal{L}\left(\theta_\tau; \mathcal{D}_\tau^{\mathrm{val}}\right)$ where the summation is over all $\tau \in \mathcal{T}_t$.

At meta-test time, given an unseen test-task $\bar{\tau} \sim p(\mathcal{T})$, starting from the optimized initial model θ_0^*, we obtain a model $\theta_{\bar{\tau}}$ by taking a small number of inner-update steps using K-shot

task-training data $\mathcal{D}_{\bar{\tau}}^{\text{trn}}$. Then, the learned model $\theta_{\bar{\tau}}$ is evaluated on the *task-test* dataset $\mathcal{D}_{\bar{\tau}}^{\text{tst}}$.

Stein Variational Gradient Descent (SVGD) (Liu and Wang, 2016) is a recently proposed nonparametric variational inference method. SVGD combines the strengths of MCMC and variational inference. Unlike traditional variational inference, SVGD does not confine the family of approximate distributions within tractable parametric distributions while remaining a simple algorithm. Also, it converges faster than MCMC because its update rule is deterministic and leverages the gradient of the target distribution. Specifically, to obtain M samples from target distribution $p(\theta)$, SVGD maintains M instances of model parameters, called *particles*. We denote the particles by $\Theta = \{\theta^m\}_{m=1}^M$. At iteration t, each particle $\theta_t \in \Theta_t$ is updated by the following rule:

$$\begin{gathered} \theta_{t+1} \leftarrow \theta_t + \epsilon_t \phi(\theta_t), \\ \text{where } \phi(\theta_t) = \frac{1}{M} \sum_{j=1}^{M} \left[k(\theta_t^j, \theta_t) \nabla_{\theta_t^j} \log p(\theta_t^j) + \nabla_{\theta_t^j} k(\theta_t^j, \theta_t) \right], \end{gathered} \tag{3.2.1}$$

ϵ_t is step-size and $k(x,x')$ is a positive-definite kernel. We can see that a particle consults with other particles by asking their gradients, and thereby determines its own update direction. The importance of other particles is weighted according to the kernel distance, relying more on closer particles. The last term $\nabla_{\theta^j} k(\theta^j, \theta^m)$ enforces repulsive force between particles so that they do not collapse to a point. The resulting particles can be used to obtain the posterior predictive distribution as:

$$p(y|x,\mathcal{D}^\tau) = \int p(y|x,\theta) p(\theta|\mathcal{D}^\tau) \mathrm{d}\theta \approx \frac{1}{M} \sum_m p(y|x,\theta^m). \tag{3.2.2}$$

where each particle θ^m is sampled from $p(\theta|\mathcal{D}^\tau)$.

A few properties of SVGD are particularly relevant to the proposed method: (i) when the number of particles M equals 1, SVGD becomes standard gradient ascent on the objective $\log p(\theta)$, (ii) under a certain condition, an SVGD-update increasingly reduces the distance between the approximate distribution defined by the particles and the target distribution, in the sense of Kullback-Leibler (KL) divergence (Liu and Wang, 2016), and finally (iii) it is straightforward to apply to reinforcement learning by using Stein Variational Policy Gradient (SVPG) (Liu et al., 2017).

3.3. Proposed Method

3.3.1. Bayesian Fast Adaptation

Our goal is to *learn to infer* by developing an efficient Bayesian gradient-based meta-learning method to efficiently obtain the task-posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}})$ of a novel task. As our

method is in the same class as MAML—in the sense that it encodes the meta-knowledge in the initial model by gradient-based optimization—we first consider the following probabilistic interpretation of MAML with one inner-update step,

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) = \prod_{\tau \in \mathcal{T}} p(\mathcal{D}_{\tau}^{\text{val}} \mid \theta'_{\tau} = \theta_0 + \alpha \nabla_{\theta_0} \log p(\mathcal{D}_{\tau}^{\text{trn}} \mid \theta_0)), \tag{3.3.1}$$

where $p(\mathcal{D}_{\tau}^{\text{val}}|\theta'_{\tau}) = \prod_{i=1}^{|\mathcal{D}_{\tau}^{\text{val}}|} p(y_i|x_i,\theta'_{\tau})$, $\mathcal{D}_{\mathcal{T}}^{\text{trn}}$ denotes all task-train sets in training taskset $\mathcal{T}$, and $\mathcal{D}_{\mathcal{T}}^{\text{val}}$ has the same meaning but for task-validation sets. From the above, we can see that the inner-update step of MAML amounts to obtaining task model θ'_{τ} from which the likelihood of the task-validation set $\mathcal{D}_{\tau}^{\text{val}}$ is computed. The meta-update step is then to perform maximum likelihood estimation of this model w.r.t. the initial parameter θ_0. This probabilistic interpretation can be extended further to applying empirical Bayes to a hierarchical probabilistic model (Grant et al., 2018) as follows:

$$p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) = \prod_{\tau \in \mathcal{T}} \left(\int p(\mathcal{D}_{\tau}^{\text{val}} \mid \theta_{\tau}) p(\theta_{\tau} \mid \mathcal{D}_{\tau}^{\text{trn}}, \theta_0) \mathrm{d}\theta_{\tau} \right). \tag{3.3.2}$$

We see that the probabilistic MAML model in Equation 3.3.1 is a special case of Equation 3.3.2 that approximates the task-train posterior $p(\theta_{\tau}|\theta_0, \mathcal{D}_{\tau}^{\text{trn}})$ by a point estimate θ'_{τ}. That is, $p(\theta_{\tau}|\mathcal{D}_{\tau}^{\text{trn}}, \theta_0) = \delta_{\theta'_{\tau}}(\theta_{\tau})$ where $\delta_y(x) = 1$ if $x = y$, and 0 otherwise. To model the task-train posterior which also becomes the prior of task-validation set, Grant et al. (2018) used an isotropic Gaussian distribution with a fixed variance.

"*Can we use a more flexible task-train posterior than a point estimate or a simple Gaussian distribution while maintaining the efficiency of gradient-based meta-learning*?" This is an important question because as discussed in Grant et al. (2018), the task-train posterior of a Bayesian neural network (BNN) trained with a few-shot dataset would have a significant amount of uncertainty which, according to the Bayesian central limit theorem (Le Cam, 1986; Ahn et al., 2012), cannot be well approximated by a Gaussian distribution.

Our first step for designing such an algorithm starts by noticing that SVGD performs deterministic updates and thus gradients can be backpropagated through the particles. This means that we now maintain M initial particles Θ_0 and by obtaining samples from the task-train posterior $p(\theta_{\tau}|\mathcal{D}_{\tau}^{\text{trn}}, \Theta_0)$ using SVGD (which is now conditioned on Θ_0 instead of θ_0), we can optimize the following Monte Carlo approximation of Equation 3.3.2 by computing the gradient of the meta-loss $\log p(\mathcal{D}_{\mathcal{T}}^{\text{val}}|\Theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}})$ w.r.t. Θ_0,

$$\begin{gathered} p(\mathcal{D}_{\mathcal{T}}^{\text{val}} \mid \Theta_0, \mathcal{D}_{\mathcal{T}}^{\text{trn}}) \approx \prod_{\tau \in \mathcal{T}} \left(\frac{1}{M} \sum_{m=1}^{M} p(\mathcal{D}_{\tau}^{\text{val}} \mid \theta_{\tau}^m) \right), \\ \text{where } \theta_{\tau}^m \sim p(\theta_{\tau} \mid \mathcal{D}_{\tau}^{\text{trn}}, \Theta_0). \end{gathered} \tag{3.3.3}$$

Algorithm 3.2 Bayesian Fast Adaptation

Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
for each task $\tau \in \mathcal{T}_t$ **do**
  $\Theta_\tau(\Theta_0) \leftarrow \text{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\text{trn}}, \alpha)$
end for
$\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_{\text{BFA}} \left(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}} \right)$

Being updated by gradient descent, it hence remains an efficient meta-learning method while providing a more flexible way to capture the complex uncertainty structure of the task-train posterior than a point estimate or a simple Gaussian approximation.

Algorithm 3.2 describes an implementation of the above model. Specifically, at iteration t, for each task τ in a sampled mini-batch $\mathcal{T}_t$, the particles initialized to Θ_0 are updated for n steps by applying the SVGD updater, denoted by $\text{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$ – the target distribution (the $p(\theta_t^j)$ in Equation 3.2.1 is set to the task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto p(\mathcal{D}_\tau^{\text{trn}}|\theta_\tau)p(\theta_\tau)$ [3.1]. This results in task-wise particles Θ_τ for each task $\tau \in \mathcal{T}_t$. Then, for the meta-update, we can use the following meta-loss:

$$\begin{aligned} \log p(\mathcal{D}_{\mathcal{T}_t}^{\text{val}}|\Theta_0, \mathcal{D}_{\mathcal{T}_t}^{\text{trn}}) &\approx \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}}), \\ \text{where } \mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}}) &= \log \left[\frac{1}{M} \sum_{m=1}^{M} p(\mathcal{D}_\tau^{\text{val}}|\theta_\tau^m) \right]. \end{aligned} \tag{3.3.4}$$

Here, we use $\Theta_\tau(\Theta_0)$ to explicitly denote that Θ_τ is a function of Θ_0. Note that, by the above model, all the initial particles in Θ_0 are *jointly* updated in such a way as to find the best joint-formation among them. From this optimized initial particles, the task-posterior of a new task can be obtained quickly, i.e., by taking a small number of update steps, and efficiently, i.e, with a small number of samples. We call this Bayesian Fast Adaptation (BFA). The method can be considered a Bayesian ensemble in which, unlike non-Bayesian ensemble methods, the particles interact with each other to find the best formation representing the task-train posterior. Because SVGD with a single particle, i.e., $M = 1$, is equal to gradient ascent, Algorithm 3.2 reduces to MAML when $M = 1$.

Although the above algorithm brings the power of Bayesian inference to fast adaptation, it can be numerically unstable due to the product of the task-validation likelihood terms. More importantly, for meta-update it is not performing Bayesian inference. Instead, it looks for the initial prior Θ_0 such that SVGD-updates lead to minimizing the empirical loss on task-validation sets. Therefore, like other meta-learning methods, the BFA model can still suffer from overfitting despite the fact that we use a flexible Bayesian inference in the inner

[3.1] In our experiments, we put hyperprior on the variance of the prior (mean is set to 0). Thus, the posterior of hyperparameter is automatically learned also by SVGD, i.e., the particle vectors include the prior parameters.

update. The reason is somewhat apparent. Because we perform only a small number of inner-updates while the meta-update is based on empirical risk minimization, the initial model Θ_0 can overfit to the task-validation sets when we use highly complex models like deep neural networks. Therefore, to become a fully robust meta-learning approach, it is desired for the method to retain the uncertainty during the meta-update as well while remaining an efficient gradient-based method.

3.3.2. Bayesian Meta-Learning with Chaser Loss

Motivated by the above observation, we propose a novel meta-loss. For this, we start by defining the loss as the dissimilarity between approximate task-train posterior $p_\tau^n \equiv p_n(\theta_\tau|\mathcal{D}_\tau^{\text{trn}};\Theta_0)$ and true task-posterior $p_\tau^\infty \equiv p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}} \cup \mathcal{D}_\tau^{\text{val}})$. Note that p_τ^n is obtained by taking n fast-adaptation steps from the initial model. Assuming that we can obtain samples Θ_τ^n and Θ_τ^∞ respectively from these two distributions, the new meta-learning objective can be written as

$$\underset{\Theta_0}{\operatorname{argmin}} \sum_\tau d_p(p_\tau^n \parallel p_\tau^\infty) \approx \underset{\Theta_0}{\operatorname{argmin}} \sum_\tau d_s(\Theta_\tau^n(\Theta_0) \parallel \Theta_\tau^\infty). \tag{3.3.5}$$

Here, $d_p(p\|q)$ is a dissimilarity between two distributions p and q, and $d_s(s_1\|s_2)$ a distance between two sample sets. We then want to minimize this distance using gradient w.r.t. Θ_0. This is to find optimized Θ_0 from which the task-train posterior can be obtained quickly and closely to the true task-posterior. However, this is intractable because we neither have access to the true posterior p_τ^∞ nor its samples Θ_τ^∞.

To this end, we approximate Θ_τ^∞ by Θ_τ^{n+s}. This is done by (i) obtaining Θ_τ^n from $p_n(\theta_\tau|\mathcal{D}_\tau^{\text{trn}};\Theta_0)$ and then (ii) taking s additional SVGD steps with the updated target distribution $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}} \cup \mathcal{D}_\tau^{\text{val}})$, i.e., augmented with additional observation $\mathcal{D}_\tau^{\text{val}}$. Although it is valid in theory not to augment the leader with the validation set, to help fast convergence we take advantage of it like other meta-learning methods. Note that, because SVGD-updates provide increasingly better approximations of the target distribution as s increases, the *leader* Θ_τ^{n+s} becomes closer to the target distribution Θ_τ^∞ than the *chaser* Θ_τ^n. This gives us the following meta-loss:

$$\mathcal{L}_{\text{BMAML}}(\Theta_0) = \sum_{\tau\in\mathcal{T}_t} d_s(\Theta_\tau^n \parallel \Theta_\tau^{n+s}) = \sum_{\tau\in\mathcal{T}_t} \sum_{m=1}^{M} \|\theta_\tau^{n,m} - \theta_\tau^{n+s,m}\|_2^2. \tag{3.3.6}$$

Here, to compute the distance between the two sample sets, we make a one-to-one mapping between the leader particles and the chaser particles and compute the Euclidean distance between the paired particles. Note that we do not back-propagate through the leader particles because we use them as targets that the chaser particles follow. A more sophisticated method like maximum mean discrepancy (Borgwardt et al., 2006) can also be used here. In our experiments, setting n and s to a small number like $n = s = 1$ worked well.

Algorithm 3.3 Bayesian Meta-Learning with Chaser Loss (BMAML)

Initialize Θ_0
for $t = 0, \ldots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Compute chaser $\Theta_\tau^n(\Theta_0) = \mathrm{SVGD}_n(\Theta_0; \mathcal{D}_\tau^{\mathrm{trn}}, \alpha)$
    Compute leader $\Theta_\tau^{n+s}(\Theta_0) = \mathrm{SVGD}_s(\Theta_\tau^n(\Theta_0); \mathcal{D}_\tau^{\mathrm{trn}} \cup \mathcal{D}_\tau^{\mathrm{val}}, \alpha)$
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} d_s \left(\Theta_\tau^n(\Theta_0) \parallel \mathrm{stopgrad}(\Theta_\tau^{n+s}(\Theta_0))\right)$
end for

Minimizing the above loss w.r.t. Θ_0 places Θ_0 in a region where the chaser Θ_τ^n can efficiently *chase* the leader Θ_τ^{n+s} in n SVGD-update steps starting from Θ_0. Thus, we call this meta-loss the *Chaser* loss. Because the leader converges to the posterior distribution instead of doing empirical risk minimization, it retains a proper level of uncertainty and thus prevents from meta-level overfitting. In Algorithm 3.3, we describe the algorithm for supervised learning. One limitation of the method is that, like other ensemble methods, it needs to maintain M model instances. Because this could sometimes be an issue when training a large model, in the Experiment section we introduce a way to share parameters among the particles.

3.4. Related Works

There have been many studies in the past that formulate meta-learning and learning-to-learn from a probabilistic modeling perspective (Tenenbaum, 1999; Fe-Fei et al., 2003; Lawrence and Platt, 2004; Daumé III, 2009). Since then, the remarkable advances in deep neural networks (Krizhevsky et al., 2012; Goodfellow et al., 2016) and the introduction of new few-shot learning datasets (Lake et al., 2015; Ravi and Larochelle, 2017), have rekindled the interest in this problem from the perspective of deep networks for few-shot learning (Santoro et al., 2016b; Vinyals et al., 2016; Snell et al., 2017; Duan et al., 2016; Finn et al., 2017; Mishra et al., 2017). Among these, Finn et al. (2017) proposed MAML that formulates meta-learning as gradient-based optimization.

Grant et al. (2018) reinterpreted MAML as a hierarchical Bayesian model, and proposed a way to perform an implicit posterior inference. However, unlike our proposed model, the posterior on validation set is approximated by local Laplace approximation and used a relatively complex 2^{nd}-order optimization using K-FAC (Martens and Grosse, 2015). The fast adaptation is also approximated by a simple isotropic Gaussian with fixed variance. As pointed by Grant et al. (2018), this approximation would not work well for skewed distributions, which is likely to be the case of BNNs trained on a few-shot dataset. The authors also pointed that their method is limited in that the predictive distribution over new data-points

is approximated by a point estimate. Our method resolves these limitations. Although it can be expensive when training many large networks, we mitigate this cost by parameter sharing among the particles. In addition, Bauer et al. (2017) also proposed Gaussian approximation of the task-posterior and a scheme of splitting the feature network and the classifier which is similar to what we used for the image classification task. Lacoste et al. (2017) proposed learning a distribution of stochastic input noise while fixing the BNN model parameter.

3.5. Experiments

We evaluated our proposed model (BMAML) in various few-shot learning tasks: sinusoidal regression, image classification, active learning, and reinforcement learning. Because our method is a Bayesian ensemble, as a baseline model we used an ensemble of independent MAML models (EMAML) from which we can easily recover regular MAML by setting the number of particles to 1. In all our experiments, we configured BMAML and EMAML to have the same network architecture and used the RBF kernel. The experiments are designed in such a way to see the effects of uncertainty in various ways such as accuracy, robustness, and efficient exploration.

3.5.1. Regression

The population of the tasks is defined by a sinusoidal function $y = A\sin(wx + b) + \epsilon$ which is parameterized by amplitude A, frequency w, and phase b, and observation noise ϵ. To sample a task, we sample the parameters uniformly randomly $A \in [0.1, 5.0]$, $b \in [0.0, 2\pi]$, $w \in [0.5, 2.0]$ and add observation noise from $\epsilon \sim \mathcal{N}(0, (0.01A)^2)$. The K-shot dataset is obtained by sampling x from $[-5.0, 5.0]$ and then by computing its corresponding y with noise ϵ. Note that, because of the highly varying frequency and observation noise, this is a more challenging setting containing more uncertainty than the setting used in Finn et al. (2017). For the regression model, we used a neural network with 3 layers each of which consists of 40 hidden units.

In Figure 3.1, we show the mean squared error (MSE) performance on the test tasks. To see the effect of the degree of uncertainty, we controlled the number of training tasks $|\mathcal{T}|$ to 100 and 1000, and the number of observation shots K to 5 and 10. The lower number of training tasks and observation shots is expected to induce a larger degree of uncertainty. We observe, as we claimed, that both MAML (which is EMAML with $M = 1$) and EMAML overfit severely in the settings with high uncertainty although EMAML with multiple particles seems to be slightly better than MAML. BMAML with the same number of particles provides significantly better robustness and accuracy for all settings. Also, having more particles tends to improve further.

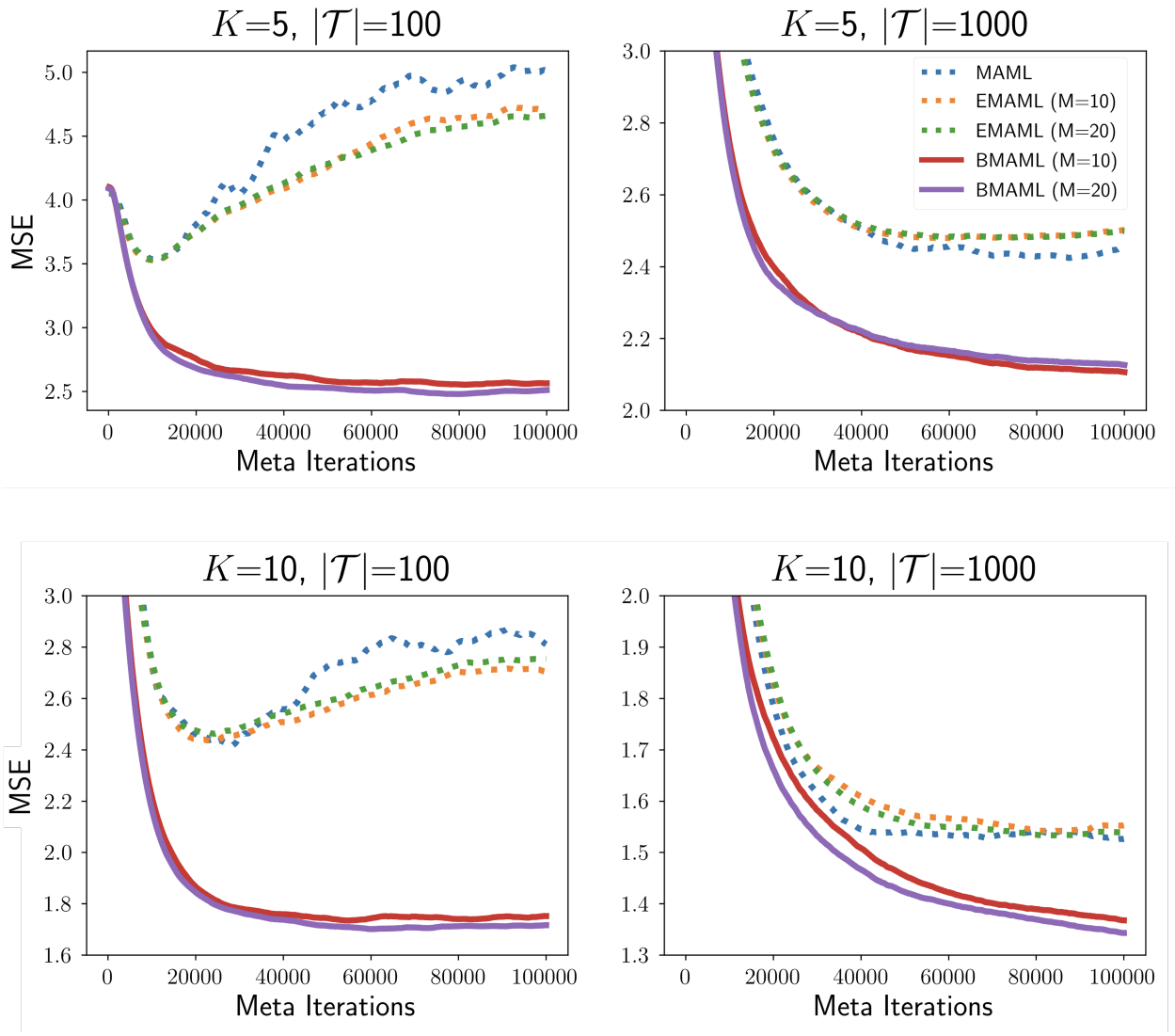


Fig. 3.1. Sinusoidal regression experimental results (meta-testing performance) by varying the number of examples (K-shot) given for each task and the number of tasks $|\mathcal{T}|$ used for meta-training.

3.5.2. Classification

To evaluate the proposed method on a more complex model, we test the performance on the *mini*Imagenet classification task (Vinyals et al., 2016) involving task adaptation of 5-way classification with 1 shot. The dataset consists of 60,000 color images of 84×84 dimension. The images consist of total 100 classes and each of the classes contains 600 examples. The entire classes are split into 64, 12, and 24 classes for meta-train, meta-validation, and meta-test, respectively. We generated the tasks following the same procedure as in Finn et al. (2017).

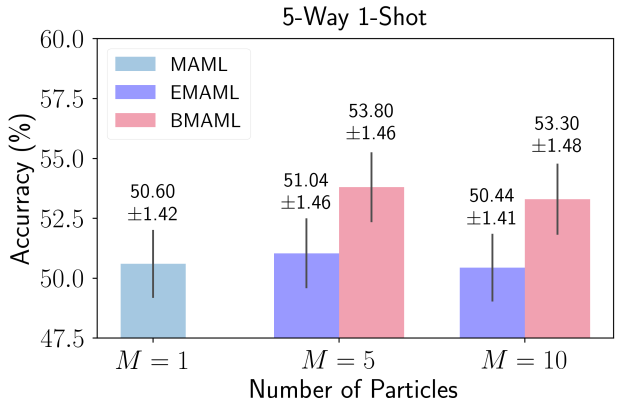
In order to reduce the space and time complexity of the ensemble models (i.e., BMAML and EMAML) in this large network setting, we used the following parameter sharing scheme among the particles, similarly to Bauer et al. (2017). We split the network architecture into the feature extractor layers and the classifier. The feature extractor is a convolutional network with 5 hidden layers with 64 filters. The classifier is a single-layer fully-connected network with softmax output. The output of the feature extractor which has 256 dimensions is input to the classifier. We share the feature extractor across all the particles while each particle has its own classifier. Therefore, the space complexity of the network is $\mathcal{O}(|\theta_{\text{feature}}|+M|\theta_{\text{classifier}}|)$. Both the classifier and feature extractor are updated during meta-update, but for inner-update only the classifier is updated. The baseline models are updated in the same manner. We describe more details of the setting in Appendix.

We can see from Figure 3.2 (a) that for both $M = 5$ and $M = 10$ BMAML provides more accurate predictions than EMAML. However, the performance of both BMAML and EMAML with 10 particles is slightly lower than having 5 particles[3.2]. Because a similar instability is also observed in the SVGD paper (Liu and Wang, 2016), we presume that one possible reason is the instability of SVGD such as sensitivity to kernel function parameters. To increase the inherent uncertainty further, in Figure 3.2 (b), we reduced the number of training tasks $|\mathcal{T}|$ from 800K to 10K. We see that BMAML provides robust predictions even for such a small number of training tasks while EMAML overfits easily.

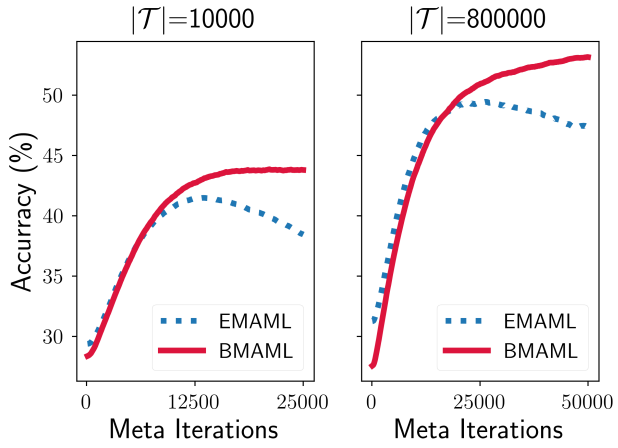
3.5.3. Active Learning

In addition to the ensembled prediction accuracy, we can also evaluate the effectiveness of the measured uncertainty by applying it to active learning. To demonstrate, we use the *mini*Imagenet classification task. To do this, given an unseen task τ at test time, we first run a fast adaptation from the meta-trained initial particles Θ_0^* to obtain Θ_τ of the task-train posterior $p(\theta_\tau|\mathcal{D}_\tau;\Theta_0^*)$. For this we used 5-way 1-shot labeled dataset. Then, from a pool of unlabeled data $\mathcal{X}_\tau = \{x_1,\ldots,x_{20}\}$, we choose an item x^* that has the maximum predictive

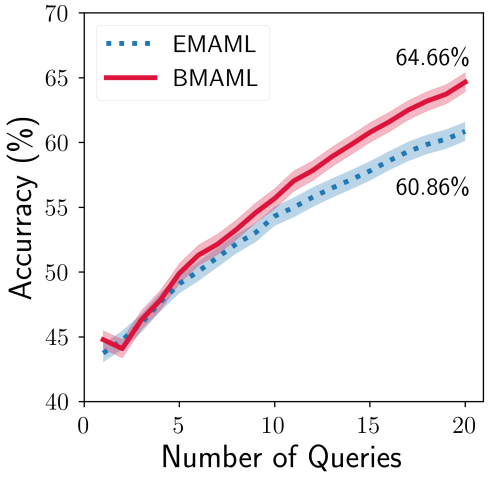
[3.2]We found a similar instability in the relationship between the number of particles and the prediction accuracy from the original implementation by the authors of the SVGD paper.



(a) Few-shot image classification using different number of particles.



(b) Few-shot image classification using different number of tasks for meta-training.



(c) Active learning setting.

Fig. 3.2. Experimental results in *mini*Imagenet dataset.

entropy as:

$$x^* = \underset{x \in \mathcal{X}_\tau}{\operatorname{argmax}} \mathbb{H}[y|x, \mathcal{D}_\tau] = -\sum_{y'} p(y'|x, \mathcal{D}_\tau) \log p(y'|x, \mathcal{D}_\tau). \tag{3.5.1}$$

The chosen item x^* is then removed from $\mathcal{X}_\tau$ and added to $\mathcal{D}_\tau$ along with its label. We repeat this process until we consume all the data in $\mathcal{X}_\tau$. We set M to 5. As we can see from Figure 3.2 (c), active learning using the Bayesian fast adaptation provides consistently better results than EMAML. Particularly, the performance gap increases as more examples are added. This shows that the examples picked by BMAML so as to reduce the uncertainty, provides proper discriminative information by capturing a reasonable approximation of the task-posterior. We presume that the performance degradation observed in the early stage might be due to the class imbalance induced by choosing examples without considering the class balance.

3.5.4. Reinforcement Learning

SVPG is a simple way to apply SVGD to policy optimization. Liu et al. (2017) showed that the maximum entropy policy optimization can be recast to Bayesian inference. In this framework, the particle update rule (a particle is now parameters of a policy) is simply to replace the target distribution $\log p(\theta)$ in Equation 3.2.1 with the objective of the maximum entropy policy optimization, i.e., $\mathbb{E}_{q(\theta)}[J(\theta)] + \eta\mathbb{H}[q]]$ where $q(\theta)$ is a distribution of policies, $J(\theta)$ is the expected return of policy θ, and η is a parameter for exploration control. Deploying multiple agents (particles) with a principled Bayesian exploration mechanism, SVPG encourages generating diverse policy behaviours while being easy to parallelize.

We test and compare the models on the same MuJoCo continuous control tasks (Todorov et al., 2012) as are used in Finn et al. (2017). In the goal velocity task, the agent receives higher rewards as its current velocity approaches the goal velocity of the task. In the goal direction task, the reward is the magnitude of the velocity in either the forward or backward direction. We tested these tasks for two simulated robots, the ant and the cheetah. The goal velocity is sampled uniformly at random from $[0.0, 2.0]$ for the cheetah and from [0.0,3.0] for the ant. As the goal velocity and the goal direction change per task, a meta learner is required to learn a given unseen task after trying K episodes. We implemented the policy network with two hidden-layers each with 100 ReLU units. We tested the number of particles for $M \in \{1, 5, 10\}$ with $M = 1$ only for non-ensembled MAML. We describe more details of the experiment setting in Appendix.

For meta-update, MAML uses TRPO (Schulman et al., 2015) which is designed with a special purpose to apply for reinforcement learning and uses a rather expensive 2^{nd}-order optimization. However, the meta-update by the chaser loss is general-purpose and based on

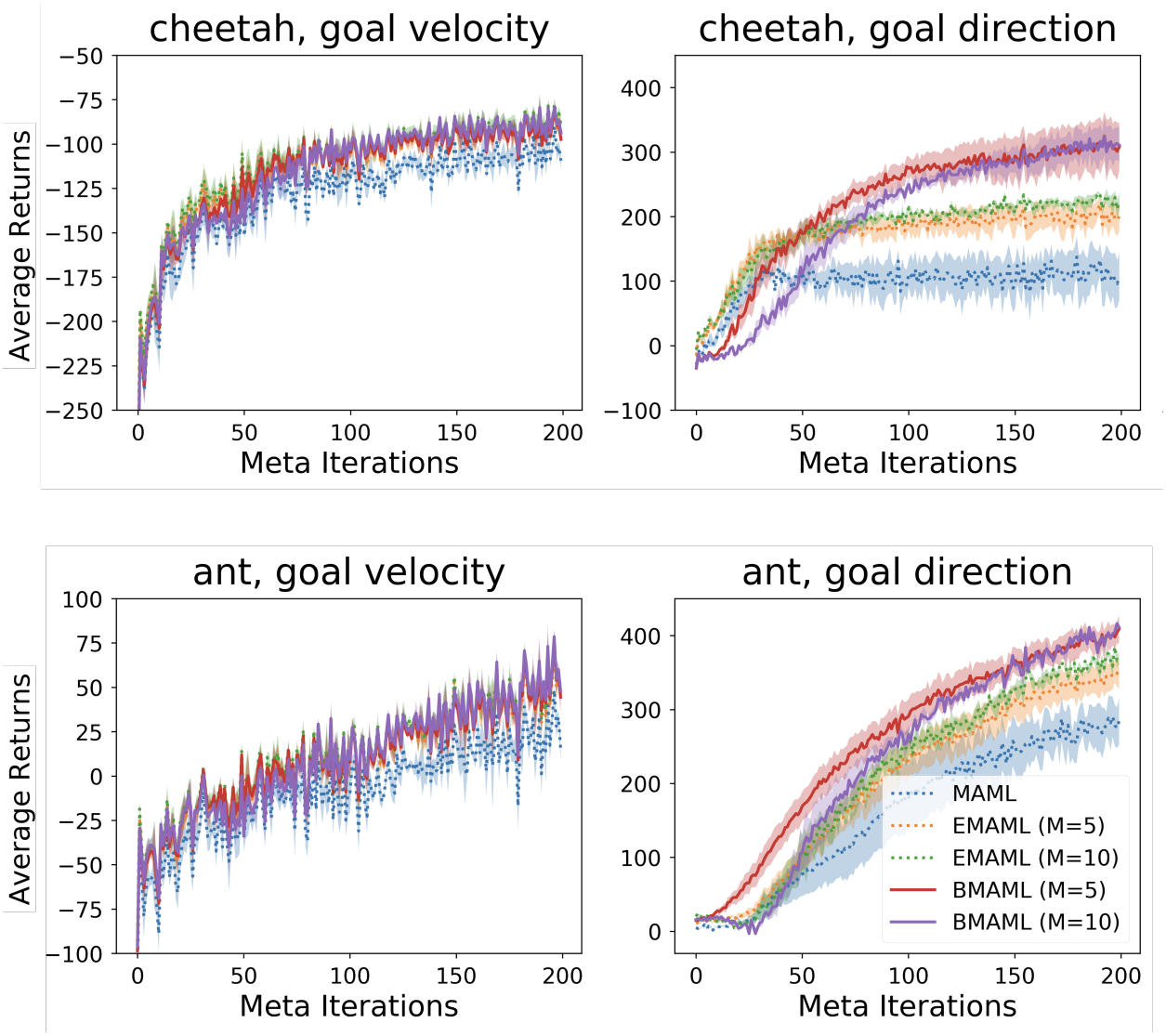


Fig. 3.3. Locomotion comparison results of SVPG-TRPO and VPG-TRPO

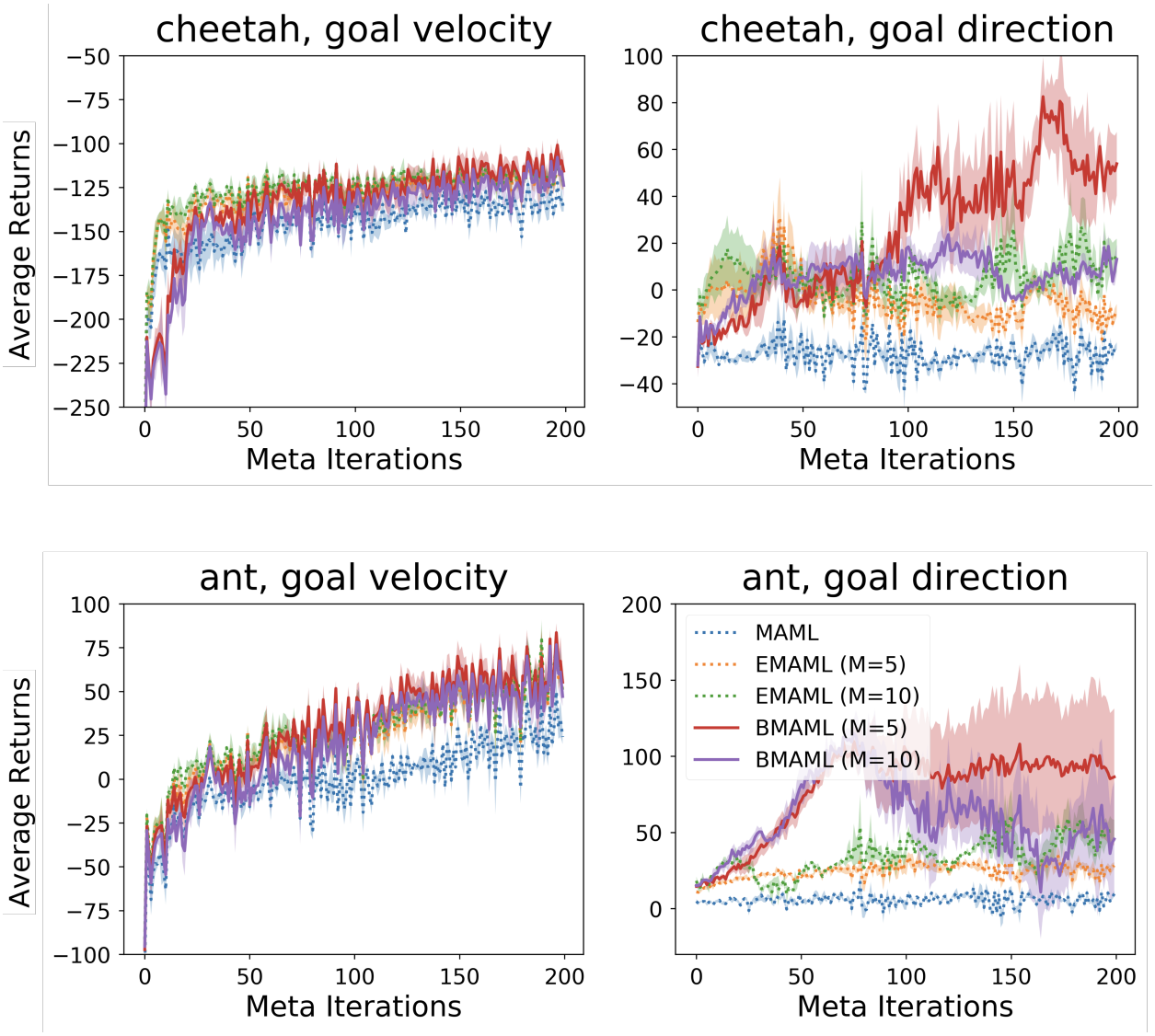


Fig. 3.4. Locomotion comparison results of SVPG-Chaser and VPG-Reptile

1^{st}-order optimization[3.3]. Thus, for a fair comparison, we consider the following two experiment designs. First, in order to evaluate the performance of the inner updates using Bayesian fast adaptation, we compare the standard MAML, which uses vanilla policy gradient (REINFORCE, Williams (1992)) for inner-updates and TRPO for meta-updates, with the Bayesian fast adaptation with TRPO meta-update. We label the former as VPG-TRPO and the later as SVPG-TRPO. Second, we compare SVPG-Chaser with VPG-Reptile. Because, similarly to the chaser loss, Reptile (Nichol et al., 2018) performs 1^{st}-order gradient optimization based on the distance in the model parameter space, this provides us a fair baseline to evaluate the chaser loss in RL. The VPG-TRPO and VPG-Reptile are implemented with independent multiple agents. We tested the comparing methods for $M = [1,5,10]$. More details of the experimental setting is provided in Appendix.

As shown in Figure 3.3 and Figure 3.4, we can see that overall, BMAML shows superior performance to EMAML. In particular, BMAML performs significantly and consistently better than EMAML in the case of using TRPO meta-updater. In addition, we can see that BMAML performs much better than EMAML for the goal direction tasks. We presume that this is because in the goal direction task, there is no goal velocity and thus a higher reward can always be obtained by searching for a better policy. This, therefore, demonstrates that BMAML can learn a better exploration policy than EMAML. In contrast, in the goal velocity task, exploration becomes less effective because it is not desired once a policy reaches the given goal velocity. This thus explains the results on the goal velocity task in which BMAML provides slightly better performance than EMAML. For some experiments, we also see that having more particles do not necessarily provides further improvements. As in the case of classification, we hypothesize that one of the reasons could be due to the instability of SVGD. In Appendix , we also provide the results on 2D Navigation task, where we observe similar superiority of BMAML to EMAML.

3.6. Discussions

BMAML is tied to SVGD? In principle, it could actually be more generally applicable to any inference algorithm that can provide *differentiable samples*. Gradient-based MCMC methods like HMC (Neal et al., 2011) or SGLD (Welling and Teh, 2011) are such methods. We however chose SVGD specifically for BMAML because jointly updating the particles altogether is more efficient for capturing the distribution quickly by a small number of update steps. In contrast, MCMC would require to wait for much more iterations until the chain mixes enough and a long backpropagation steps through the chain.

[3.3]When considering the inner update together, TRPO, Chaser and Reptile are $3^{rd}/2^{nd}/1^{st}$-order, respectively.

Parameter space v.s. prediction space? We defined the chaser loss by the distance in the model parameter space although it is also possible to define it in the prediction distance, i.e., by prediction error. We chose the parameter space because (1) we can save computation for the forward-pass for predictions, and (2) it empirically showed better performance for RL and similar performance for other tasks. The advantages of working in the parameter space is also discussed in Nichol et al. (2018).

Do the small number of SVGD steps converge to the posterior? In our small-data-big-network setting, a large area of a true task-posterior will be meaningless for other tasks. Thus, it is not desired to fully capture the task-posterior but instead we need to find an area which will be broadly useful for many tasks. This is the goal of hierarchical Bayes which our method approximate by finding such area and putting Θ_0 there. In theory, the task-posterior can be fully captured with infinite number of particles and update-steps, and thus dilute the initialization effect. In practice, the full coverage would, however, not be achievable (and not desired) because SVGD or MCMC would have difficulties in covering all areas of the complex multimodal task-posterior like that of a neural network.

3.7. Conclusion

Motivated by the hierarchical probabilistic modeling perspective to gradient-based meta-learning, we proposed a Bayesian gradient-based meta learning method. To do this, we combined the Stein Variational Gradient Descent with gradient-based meta learning in a probabilistic framework, and proposed the Bayesian Fast Adaptation and the Chaser loss for meta-update. As it remains a model-agnostic model, in experiments, we evaluated the method in various types of learning tasks including supervised learning, active learning, and reinforcement learning, and showed its superior performance in prediction accuracy, robustness to overfitting, and efficient exploration.

As a Bayesian ensemble method, along with its advantages, the proposed method also inherits the generic shortcomings of ensemble methods, particularly the space/time complexity proportional to the number of particles. Although we showed that our parameter sharing scheme is effective to mitigate this issue, it would still be interesting to improve the efficiency further in this direction. In addition, because the performance of SVGD can be sensitive to the parameters of the kernel function, incorporating the fast-adaptation of the kernel parameter into a part of meta-learning would also be an interesting future direction.

3.8. Appendix

3.8.1. Supervised Learning

Regression. We used 10 tasks for each meta-batch and the meta-validation dataset $\mathcal{D}_\tau^{\text{val}}$ is set to have the same size of the meta-training dataset $\mathcal{D}_\tau^{\text{trn}}$ ($|\mathcal{D}_\tau^{\text{trn}}| = |\mathcal{D}_\tau^{\text{val}}| = K$). During training, the number of steps n for chaser is set to $n = 1$ and also the number of steps s for leader is set to $s = 1$. We used different step sizes α for computing chaser and leader, 0.01 and 0.001, respectively. This allows the leader to stay nearby the chaser but toward the target posterior and stabilized the training. The models were trained with using different size of training dataset $|\mathcal{T}|$, the number of tasks observable during training, and we trained the model over 10000 epochs for $|\mathcal{T}| = 100$ and 1000 epochs for $|\mathcal{T}| = 1000$. In Figure 3.5, we show the qualitative results on randomly sampled sinusoid task and we used 5 update steps. The task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}})$ decomposes into the train data likelihood and parameter prior as $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto p(\mathcal{D}_\tau^{\text{trn}}|\theta_\tau)p(\theta_\tau)$ and this is formulated as:

$$p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto \prod_{(x,y)\in\mathcal{D}_\tau^{\text{trn}}} \mathcal{N}(y|f_W(x), \gamma^{-1}) \prod_{w\in W} \mathcal{N}(w|0, \lambda^{-1})\text{Gamma}(\gamma|a, b)\text{Gamma}(\lambda|a', b')$$

where θ_τ consists of network parameters W and scaling parameters γ, λ. In all experiments, we set Gamma distribution hyper-parameters as $a = 2.0, b = 0.2$ and $a' = 2.0, b' = 2.0$. During meta-update with chaser-loss, we used Adam optimizer (Kingma and Ba, 2014) with learning rate $\beta = 0.001$.

Classification. All models and experiments on the *mini*Imagenet classification task are trained with using the same network architecture and 16 tasks are used for each meta-batch during training. Each task is defined by randomly selected 5 classes with one instance of each class to adapt the model and it is evaluated on unseen instances within the selected 5 classes. We used the meta-validation dataset $\mathcal{D}_\tau^{\text{val}}$ containing one example per each class for the 5-way 1-shot setting. This reduced the computational cost and also improved the performance of all models. During training, the number of steps for chaser and leader both are set to 1 ($n = s = 1$). The chaser and leader used step size $\alpha = 0.01$ and $\alpha = 0.005$, respectively. The meta-update was done by using Adam optimizer ($\beta = 0.0005$). The models were trained with using different size of training dataset $|\mathcal{T}|$ and we trained the model with $|\mathcal{T}| = 800000$ and 1 epoch. With $|\mathcal{T}| = 10000$, the model was trained over 40 epochs. The task-train posterior $p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}})$ for classification is slightly different to the regression task due to using softmax for the data likelihood.

$$p(\theta_\tau|\mathcal{D}_\tau^{\text{trn}}) \propto \prod_{(x,y)\in\mathcal{D}_\tau^{\text{trn}}} p(y|f_W(x)) \prod_{w\in W} \mathcal{N}(w|0, \lambda^{-1})\text{Gamma}(\lambda|a, b)$$

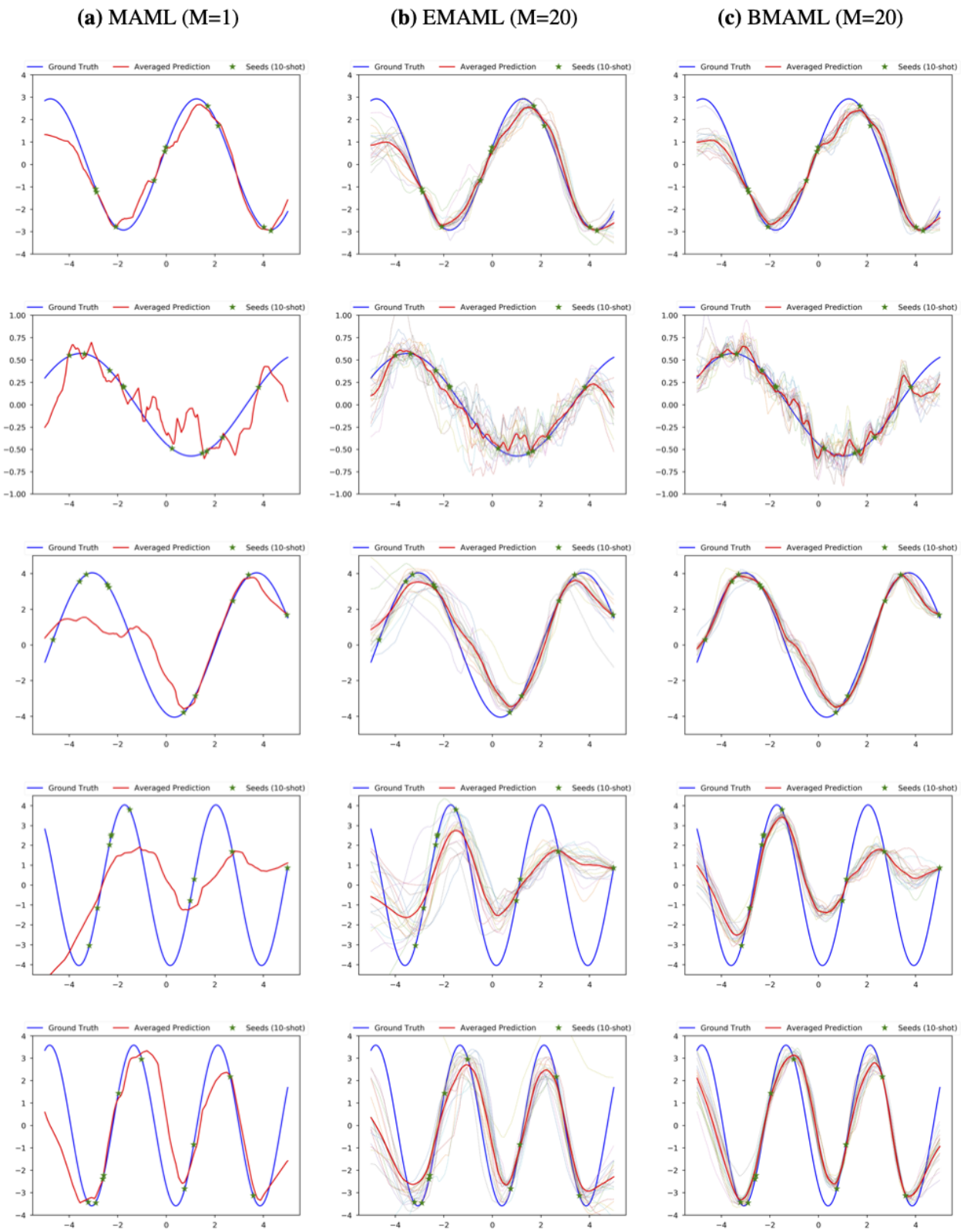


Fig. 3.5. Regression qualitative examples: randomly sampled tasks with 10 examples (10-shot) and 10 gradient updates for adaptation

where the hyper-parameters for Gamma distribution were set as $a = 2.0, b = 0.2$ or $a = 1.0, b = 0.1$ in our experiments.

3.8.2. Active Learning

Here we describe our algorithm for active learning, which is based on our BMAML and also EMAML. The details are shown in Algorithm 3.4.

Algorithm 3.4 Active Learning on Image Classification

Sample a few-shot labeled dataset $\mathcal{D}_\tau$ and a pool of unlabeled dataset $\mathcal{X}_\tau$ of task τ
Initialize $\Theta_\tau \leftarrow \Theta_0^*$
Update $\Theta_\tau \leftarrow \text{SVGD}_n(\Theta_\tau; \mathcal{D}_\tau, \alpha)$
while $\mathcal{X}_\tau$ is not empty **do**
  Select $x' \leftarrow \text{argmax}_{x \in \mathcal{X}_\tau} \mathbb{H}[y|x, \Theta_\tau]$ and remove x' from $\mathcal{X}_\tau$
  Request y' of x'
  Update $\mathcal{D}_\tau \leftarrow \mathcal{D}_\tau \cup \{(x', y')\}$
  Update $\Theta_\tau \leftarrow \text{SVGD}_n(\Theta_\tau; \mathcal{D}_\tau, \alpha)$
end while

3.8.3. Reinforcement Learning: Locomotion

The locomotion experiments require two simulated robots, a planar cheetah and 3D quadruped ones (called as ant), and two individual goals, to run in a particular direction or at a particular velocity. For the ant goal velocity, a positive bonus reward at each timestep is added to prevent the ant from ending the episode. In those experiments, the timestep in each episodes is 200, the number of episode per each inner update, K is 10 except the ant goal direction task, in which 40 episodes for each inner update is used, because of task complexity. The number of tasks per each meta update is 20, and the models are trained for up to 200 meta iterations.

We evaluate our proposed method on two cases, SVPG-TRPO vs VPG-TRPO and SVPG-Chaser vs VPG-Reptile. We describe the methods in this subsection, except VPG-TRPO, because this is MAML when $M = 1$.

SVPG-TRPO. This method is to use SVPG as inner update and TRPO as meta update, which is following to a simple Bayesian meta-learning manner. In K-shot reinforcement learning on this method, K episodes from each policy particles and task τ (total number of episode is KM), and the corresponding rewards are used for task learning on the task. This method gets the above data ($\mathcal{D}_\tau^{\text{trn}}$) from Θ_0, and updates the parameters Θ_0 to Θ_τ^n with $\mathcal{D}_\tau^{\text{trn}}$ and SVPG. After getting few-shot learned parameters (Θ_τ^n), our method get new data ($\mathcal{D}_\tau^{\text{val}}$) from Θ_τ^n. After all the materials for meta learning have been collected, our method finds the meta loss with few-shot learned particles and task-validation set, $\mathcal{D}_\tau^{val}$. On meta-learning,

TRPO (Schulman et al., 2015) is used as MAML (Finn et al., 2017) for validating inner Bayesian learning performance. The overall algorithm is described in Algorithm 3.5.

Algorithm 3.5 Simple Bayesian Meta-Learning for Reinforcement Learning

Initialize Θ_0
for $t = 0, \dots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Sample trajectories $\mathcal{D}_\tau^{\text{trn}}$ with Θ_0 in τ
    Compute chaser $\Theta_\tau^n = \text{SVPG}(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$
    Sample trajectories $\mathcal{D}_\tau^{\text{val}}$ with Θ_τ^n in τ
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} \mathcal{L}_\tau^{meta}(\Theta_\tau^n; \mathcal{D}_\tau^{\text{val}})$
end for

SVPG-Chaser. This method is to use SVPG as inner-update and chaser loss for meta-update to maintain uncertainty. Different to supervised learning, this method updates leader particles just with $\mathcal{D}_\tau^{\text{val}}$ in policy gradient update manner. Same chaser loss to supervised learning ones is consistently applied to evaluating the chaser loss extensibility. The chaser loss in RL changes the meta update from a policy gradient problem to a problem similar to imitation learning. Unlike conventional imitation learning with given expert agent, this method keeps the uncertainty provided by the SVPG by following one more updated agent, and ultimately ensures that the chaser agent is close to the expert. Compared to Algorithm 3.5, this method adds updating the leader and changes the method of meta update like Algorithm 3.6.

Algorithm 3.6 Bayesian Meta-Learning for Reinforcement Learning with Chaser Loss

Initialize Θ_0
for $t = 0, \dots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Sample trajectories $\mathcal{D}_\tau^{\text{trn}}$ with Θ_0 in τ
    Compute chaser $\Theta_\tau^n = \text{SVPG}(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$
    Sample trajectories $\mathcal{D}_\tau^{\text{val}}$ with Θ_τ^n in τ
    Compute leader $\Theta_\tau^{n+s} = \text{SVPG}(\Theta_\tau^n; \mathcal{D}_\tau^{\text{val}})$
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} d_s(\Theta_\tau^n || \text{stopgrad}(\Theta_\tau^{n+s}))$
end for

VPG-Reptile. Reptile (Nichol et al., 2018) solved meta learning problem by using only 1^{st}-order derivatives, and used meta update in parameter space. We design a version of meta loss similar to Reptile to verify the performance of chaser loss in RL problem. This method

computes the chaser Θ_τ^n using $\mathcal{D}_\tau^{\text{trn}}$ and then calculates the euclidean distance between this parameter and the global parameter as a meta loss (to prevent the gradient from being calculated through the chaser parameter to maintain the 1$^{\text{st}}$-order derivatives). The overall algorithm is described in Algorithm 3.7.

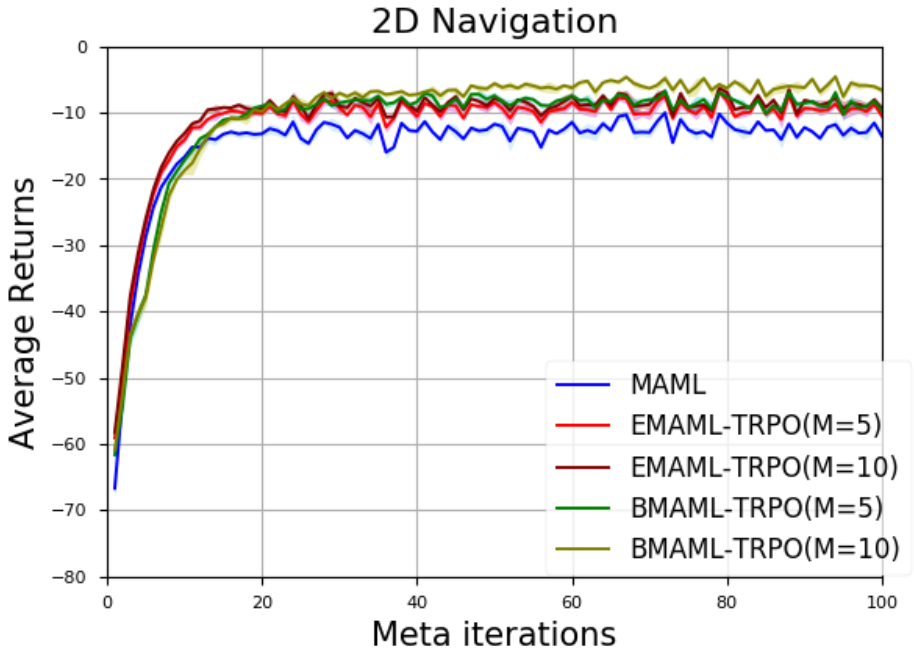
Algorithm 3.7 VPG-Reptile

Initialize Θ_0
for $t = 0, \dots$ until converge **do**
  Sample a mini-batch of tasks $\mathcal{T}_t$ from $p(\mathcal{T})$
  for each task $\tau \in \mathcal{T}_t$ **do**
    Sample trajectories $\mathcal{D}_\tau^{\text{trn}}$ with Θ_0 in τ
    Compute chaser $\Theta_\tau^n = \text{VPG}(\Theta_0; \mathcal{D}_\tau^{\text{trn}})$
  end for
  $\Theta_0 \leftarrow \Theta_0 - \beta \nabla_{\Theta_0} \sum_{\tau \in \mathcal{T}_t} d_s(\Theta_0 || \text{stopgrad}(\Theta_\tau^n))$
end for

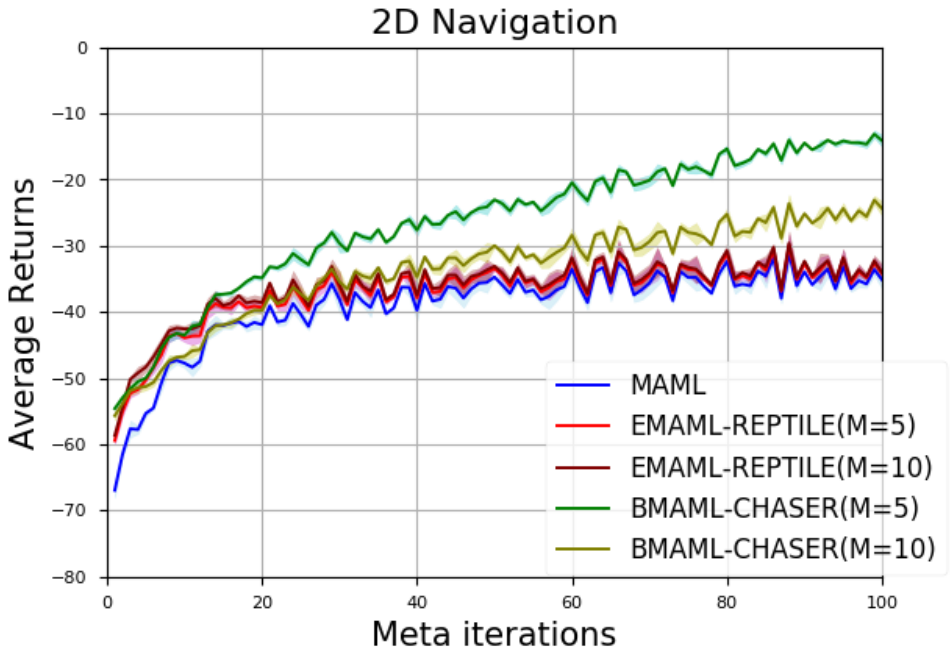
Experimental Details. Inner update learning rate and the number of inner update are set as 0.1 and 1 for all experiments, which are locomotion (ant/cheetah goal velocity and ant/cheetah goal direction) and 2D-Navigation experiments. Meta update learning rate is set as 0.1 for ant goal direction and 0.01 for other experiments. η, the parameter that controls the strength of exploration in SVPG is set as 0.1 for ant velocity experiment with SVPG-Chaser, ant goal direction and 2D Navigation with SVPG-Chaser, and 1.0 for other experiments. Each plots are based on an mean and a standard deviation from three different random seed. The subsumed results are plotted with the average reward of maximum task rewards in during of particles.

3.8.4. Reinforcement Learning: 2D Navigation

We also compare the models on the toy experiment designed in previous work (Finn et al., 2017), 2D Navigation. This experiment is a set of tasks where agent must move to different goal positions in 2D, which is randomly set for each task within a unit square. The observation is the current 2D position, and actions correspond to velocity clipped to be in the range [-0.1, 0.1]. The reward is the negative squared distance between the goal and the current position, and episodes terminate when the agent is within 0.01 of the goal or at the timestep = 100. We used 10 episodes per each inner update ($K = 10$) and 20 tasks per each meta update. The models are trained for up to 100 meta iterations. The policy network has two hidden layers each with 100 ReLU units. We tested the number of particles for $M \in \{1, 5, 10\}$ with $M = 1$ only for non-ensembled MAML. Same to above locomotion experiments, we compare the models as SVPG-TRPO vs VPG-TRPO and SVPG-Chaser vs VPG-Reptile. As shown in Figure 3.6, BMAML showed better performance than EMAML on both comparisons.



(a) SVPG-TRPO vs. VPG-TRPO



(b) SVPG-Chaser vs. VPG-Reptile

Fig. 3.6. 2D Navigation results (with three different random seeds).

Chapter 4

Variational Temporal Abstraction

Taesup Kim, Sungjin Ahn and Yoshua Bengio
Appeared in:

- *Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*

Personal Contribution. I was interested in video prediction with adaptive frame skipping, which is highly related to my previous work in neural acoustic modeling (Song et al., 2018), and Sungjin Ahn was trying to find a better video prediction model that can improve the imagination-augmented agent learning. In the end, we came up with an idea to implement a model with jumpy future predictions. Most of our proposed methods were designed by myself with guidance from Sungjin Ahn and Yoshua Bengio. Sungjin Ahn also worked hard to design our experiments that could properly show the efficiency of our proposed methods. In this work, I was the only person, who implemented and performed all of the experiments. Sungjin Ahn significantly contributed to writing the introductory part and Yoshua Bengio did the final editing. This work was also presented in *ICML Generative Modeling and Model-Based Reasoning for Robotics and AI* workshop 2019, and I prepared and gave the presentation for it.

Context. At this stage, while model-based reinforcement learning (RL) may not have clear commercial applications, its potential impact is enormous as an environment becomes more complex and adaptive. A key challenge in model-based RL is to synthesize computationally efficient and accurate environment models. That way, we mainly focused on the topic to make models computationally efficient by implementing jumpy future predictions based on the concept of *temporal-level adaptation* (temporal abstraction). We introduced a variational approach to learning and inference of temporally hierarchical structure and representation for sequential data by proposing the *Variational Temporal Abstraction* (VTA). It is a hierarchical recurrent state-space model that can infer the latent temporal structure and thus perform the stochastic state transition hierarchically. Furthermore, we applied this model to implement a jumpy imagination ability in imagination-augmented agent-learning in order

to improve the efficiency of the imagination. In experiments, we demonstrated that our proposed method can model 2D and 3D visual sequence datasets with interpretable temporal structure discovery and that its application to jumpy imagination enables more efficient agent-learning in a 3D navigation task.

Keywords: representation learning, temporal abstraction, temporally hierarchical structure, variational inference, imagination, agent-learning, model-based reinforcement learning (RL)

4.1. Introduction

Discovering temporally hierarchical structure and representation in sequential data is the key to many problems in machine learning. In particular, for an intelligent agent exploring an environment, it is critical to learn such spatio-temporal structure hierarchically because it can, for instance, enable efficient option-learning and jumpy future imagination, abilities critical to resolving the sample efficiency problem (Hamrick, 2019). Without such temporal abstraction, imagination would easily become inefficient; imagine a person planning one-hour driving from her office to home with future imagination at the scale of every second. It is also biologically evidenced that future imagination is the very fundamental function of the human brain (Mullally and Maguire, 2014; Buckner, 2010) which is believed to be implemented via hierarchical coding of the grid cells (Wei et al., 2015).

There have been approaches to learn such hierarchical structure in sequences such as the HMRNN (Chung et al., 2016). However, as a deterministic model, it has the main limitation that it cannot capture the stochastic nature prevailing in the data. In particular, this is a critical limitation to imagination-augmented agents because exploring various possible futures according to the uncertainty is what makes the imagination meaningful in many cases. There have been also many probabilistic sequence models that can deal with such stochastic nature in the sequential data (Chung et al., 2015; Krishnan et al., 2017; Fraccaro et al., 2017). However, unlike HMRNN, these models cannot automatically discover the temporal structure in the data.

In this paper, we propose the Hierarchical Recurrent State Space Model (HRSSM) that combines the advantages of both worlds: it can discover the latent temporal structure (e.g., subsequences) while also modeling its stochastic state transitions hierarchically. For its learning and inference, we introduce a variational approximate inference approach to deal with the intractability of the true posterior. We also propose to apply the HRSSM to implement efficient *jumpy imagination* for imagination-augmented agents. We note that the proposed HRSSM is a *generic* generative sequence model that is not tied to the specific application to the imagination-augmented agent but can be applied to any sequential data. In experiments, on 2D bouncing balls and 3D maze exploration, we show that the proposed model can model sequential data with interpretable temporal abstraction discovery. Then, we show that the model can be applied to improve the efficiency of imagination-augmented agent-learning.

The main contributions of the paper are:

- We propose the Hierarchical Recurrent State Space Model (HRSSM) that is the first stochastic sequence model that discovers the temporal abstraction structure.
- We propose the application of HRSSM to imagination-augmented agent so that it can perform efficient jumpy future imagination.

- In experiments, we showcase the temporal structure discovery and the benefit of HRSSM for agent learning.

4.2. Proposed Model

4.2.1. Hierarchical Recurrent State Space Models

In our model, we assume that a sequence $X = x_{1:T} = (x_1, \ldots, x_T)$ has a latent structure of temporal abstraction that can partition the sequence into N non-overlapping subsequences $X = (X_1, \ldots, X_N)$. A subsequence $X_i = x^i_{1:l_i}$ has length l_i such that $T = \sum_{i=1}^{T} l_i$ and $L = \{l_i\}$. Unlike previous works (Serban et al., 2017), we treat the number of subsequences N and the lengths of subsequences L as discrete latent variables rather than given parameters. This makes our model discover the underlying temporal structure adaptively and stochastically.

We also assume that a subsequence X_i is generated from a *temporal abstraction* z_i and an observation x_t has *observation abstraction* s_t. The temporal abstraction and observation abstraction have a hierarchical structure in such a way that all observations in X_i are governed by the temporal abstraction z_i in addition to the local observation abstraction s_t. As a temporal model, the two abstractions take temporal transitions. The transition of temporal abstraction occurs only at the subsequence scale while the observation transition is performed at every time step. This generative process can then be written as follows:

$$p(X,S,L,Z,N) = p(N) \prod_{i=1}^{N} p(X_i, S_i|z_i, l_i) p(l_i|z_i) p(z_i|z_{<i}) \tag{4.2.1}$$

where $S = \{s^i_j\}$ and $Z = \{z_i\}$ and the subsequence joint distribution $p(X_i, S_i|z_i, l_i)$ is:

$$p(X_i, S_i|z_i,l_i) = \prod_{j=1}^{l_i} p(x^i_j|s^i_j) p(s^i_j|s^i_{<j},z_i). \tag{4.2.2}$$

We note that it is also possible to use the traditional Markovian state space model in Equation 4.2.1 and Equation 4.2.2 which has some desirable properties such as modularity and interpretability as well as having a closed-form solution for a limited class of models like the linear Gaussian model. However, it is known that this Markovian model has difficulties in practice in capturing complex long-term dependencies (Auger-Méthé et al., 2016). Thus, in our model, we take the recurrent state space model (RSSM) approach (Zheng et al., 2017; Buesing et al., 2018; Hafner et al., 2018) which resolves this problem by adding a deterministic RNN path that can effectively encode the complex nonlinear long-term dependencies in the past, i.e., $z_{<i}$ and $s^i_{<j}$ in our model. Specifically, the transition is performed by the following updates: $c_i = f_{z\text{-rnn}}(z_{i-1}, c_{i-1}), z_i \sim p(z_i|c_i)$ for z_i, and $h^i_j = f_{s\text{-rnn}}(s^i_{j-1}||z_i, h^i_{j-1}), s^i_j \sim p(s^i_j|h^i_j)$ for s^i_j.

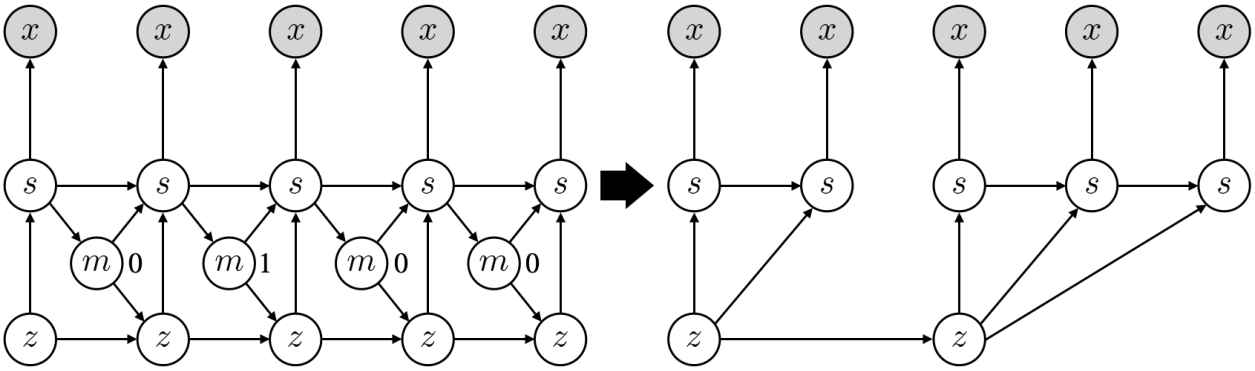


Fig. 4.1. Sequence generative procedure (recurrent deterministic paths are excluded). **Left**: The model with the boundary indicators $M = \{0, 1, 0, 0\}$. **Right**: The corresponding generative procedure with a temporal structure derived from the boundary indicators M

4.2.2. Binary Subsequence Indicator

Although the above modeling intuitively explains the actual generation process, the discrete latent random variables N and $\{l_i\}$—whose realization is an integer—raise difficulties in learning and inference. To alleviate this problem, we reformulate the model by replacing the integer latent variables by a sequence of binary random variables $M = m_{1:T}$, called the *boundary indicator*. As the name implies, the role of this binary variable is to indicate whether a new subsequence should start at the next time step or not. In other words, it specifies the end of a subsequence. This is a similar operation to the FLUSH operation in the HMRNN model (Chung et al., 2016). With the binary indicators, the generative process can be rewritten as follows:

$$p(X,Z,S,M) = \prod_{t=1}^{T} p(x_t|s_t)p(m_t|s_t)p(s_t|s_{<t}, z_t, m_{t-1})p(z_t|z_{<t}, m_{t-1}). \tag{4.2.3}$$

In this representation of the generative process, we can remove the subsequence hierarchy and make both transitions perform at every time step. Although this seemingly looks different to our original generation process, the control of the binary indicator—selecting either COPY or UPDATE—can make this equivalent to the original generation process, which we explain later in more detail. In Figure 4.1, we provide an illustration on how the binary indicators induce an equivalent structure represented by the discrete random variables N and L.

This reformulation has the following advantages. First, we do not need to treat the two different types of discrete random variables N and L separately but instead can unify them by using only one type of random variables M. Second, we do not need to deal with the variable range of N and L because each time step has finite states {0,1} while N and L depend on T that can be changed across sequences. Lastly, the decision can be made adaptively while observing the progress of the subsequence, instead of making a decision governing the whole subsequence.

4.2.3. Prior on Temporal Structure

We model the binary indicator $p(m_t|s_t)$ by a Bernoulli distribution parameterized by $\sigma(f_{m\text{-mlp}}(s_t))$ with a multi-layer perceptron (MLP) $f_{m\text{-mlp}}$ and a sigmoid function σ. In addition, it is convenient to explicitly express our prior knowledge or constraint on the temporal structure using the boundary distribution. For instance, it is convenient to specify the maximum number of subsequences $N_{\max}$ or the longest subsequence lengths $l_{\max}$ when we do not want too many or too long subsequences. To implement, at each time step t, we can compute the number of subsequences discovered so far by using a counter $n(m_{<t})$ as well as the length of current subsequence with another counter $l(m_{<t})$. Based on this, we can design the boundary distribution with our prior knowledge as follows:

$$p\left(m_t = 1|s_t\right) = \begin{cases} 0 & \text{if } n\left(m_{<t}\right) \geq N_{\max}, \\ 1 & \text{elseif } l\left(m_{<t}\right) \geq l_{\max}, \\ \sigma\left(f_{m\text{-mlp}}\left(s_t\right)\right) & \text{otherwise.} \end{cases} \tag{4.2.4}$$

4.2.4. Hierarchical Transitions

The transition of temporal abstraction should occur only a subsequence is completed. This timing is indicated by the boundary indicator. Specifically, the transition of temporal abstraction is implemented as follows:

$$p\left(z_t|z_{<t}, m_{t-1}\right) = \begin{cases} \delta(z_t = z_{t-1}) & \text{if } m_{t-1} = 0 \text{ (COPY)}, \\ \tilde{p}(z_t|c_t) & \text{otherwise (UPDATE)} \end{cases} \tag{4.2.5}$$

where c_t is the following RNN encoding of all previous temporal abstractions $z_{<t}$ (and $m_{<t}$):

$$c_t = \begin{cases} c_{t-1} & \text{if } m_{t-1} = 0 \text{ (COPY)}, \\ f_{z\text{-rnn}}\left(z_{t-1}, c_{t-1}\right) & \text{otherwise (UPDATE)}. \end{cases} \tag{4.2.6}$$

Specifically, having $m_{t-1} = 0$ indicates that the time step t is still in the same subsequence as the previous time step $t-1$ and thus the temporal abstraction should not be updated but copied. Otherwise, it indicates that the time step $t-1$ was the end of the previous subsequence and thus the temporal abstraction should be updated. This transition is implemented as a Gaussian distribution $\mathcal{N}(z_t|\mu_z(c_t), \sigma_z(c_t))$ where both μ_z and σ_z are implemented with MLPs.

At test time, we can use this transition of temporal abstraction without the COPY mode, i.e., every transition is UPDATE. This implements the *jumpy* future imagination which do not require to rollout at every raw time step and thus is computationally efficient.

The observation transition is similar to the transition of temporal abstraction except that we want to implement the fact that given the temporal abstraction z_i, a subsequence

is independent of other subsequences. The observation transition is implemented as follows:

$$p(s_t|s_{<t}, z_t, m_{t-1}) = \tilde{p}(s_t|h_t),$$
$$\text{where} \quad h_t = \begin{cases} f_{s\text{-rnn}}\left(s_{t-1}\|z_t, h_{t-1}\right) & \text{if } m_{t-1} = 0 \text{ (UPDATE)}, \\ f_{s\text{-mlp}}\left(z_t\right) & \text{otherwise (INIT)}. \end{cases} \tag{4.2.7}$$

Here, h_t is computed by using an RNN $f_{s\text{-rnn}}$ to update (UPDATE), and a MLP $f_{s\text{-mlp}}$ to initialize (INIT). The concatenation is denoted by $\|$. Note that if the subsequence is finished, i.e., $m_{t-1} = 1$, we sample a new observational abstraction s_t without conditioning on h_t. That is, the underlying RNN is initialized.

4.3. Learning and Inference

As the true posterior is intractable, we apply variational approximation which gives the following evidence lower bound (ELBO) objective:

$$\log p\left(X\right) \geq \sum_{M} \int_{Z,S} q_\phi\left(Z,S,M|X\right) \log \frac{p_\theta\left(X,Z,S,M\right)}{q_\phi\left(Z,S,M|X\right)} dZdS. \tag{4.3.1}$$

This is optimized w.r.t. θ and ϕ using the reparameterization trick (Kingma and Welling, 2014). In particular, we use the Gumbel-softmax (Jang et al., 2017; Maddison et al., 2017) with straight-through estimators (Bengio et al., 2013) for the discrete variables M. For the approximate posterior, we use the following factorization:

$$q_\phi(Z,S,M|X) = q(M|X)q(Z|M,X)q(S|Z,M,X). \tag{4.3.2}$$

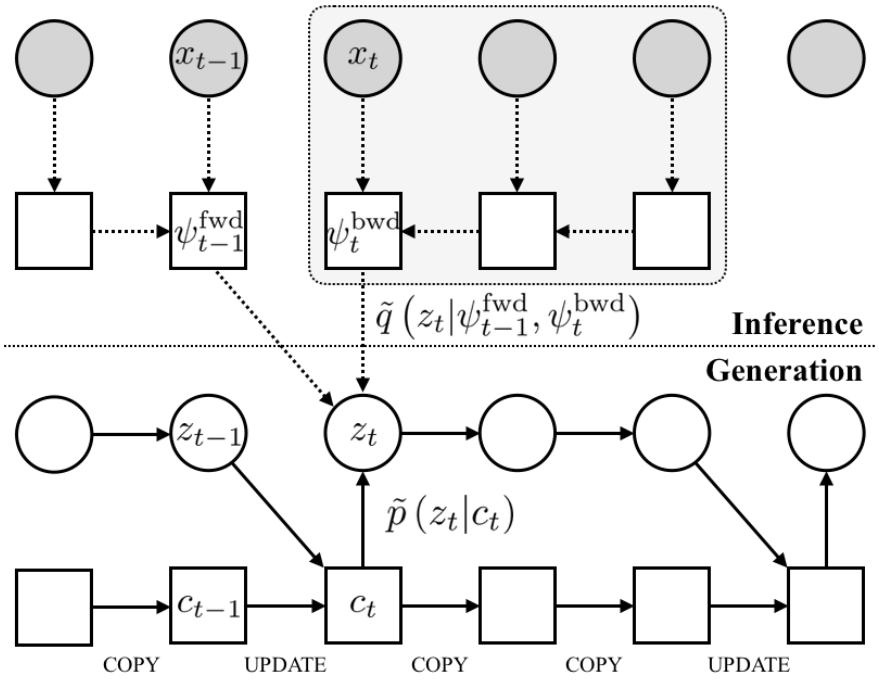
That is, by *sequence decomposition* $q(M|X)$, we first infer the boundary indicators independent of Z and S. Then, given the discovered boundary structure, we infer the two abstractions via the *state inference* $q(Z|M,X)$ and $q(S|Z,M,X)$.

Sequence Decomposition. Inferring the subsequence structure is important because the other state inference can be decomposed into independent subsequences. This sequence decomposition is implemented by the following decomposition:

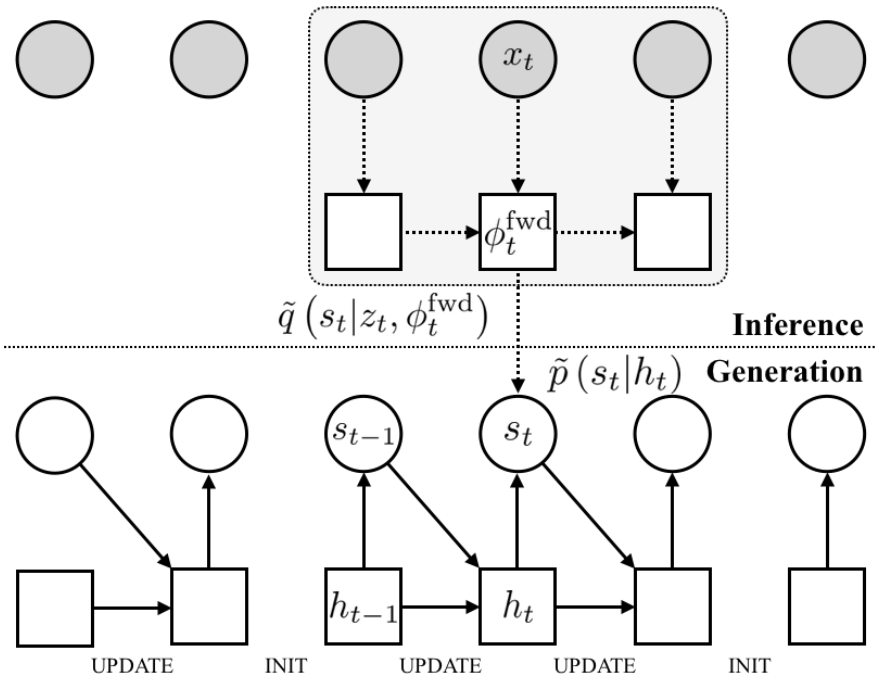
$$q(M|X) = \prod_{t=1}^{T} q(m_t|X) = \prod_{t=1}^{T} \text{Bern}\left(m_t|\sigma(\varphi(X))\right), \tag{4.3.3}$$

where φ is a convolutional neural network (CNN) applying convolutions over the temporal axis to extract dependencies between neighboring observations. This enables to sample all indicators M independently and simultaneously. Empirically, we found this CNN-based architecture working better than an RNN-based architecture.

State Inference. State inference is also performed hierarchically. The temporal abstraction predictor $q(Z|M,X) = \prod_{t=1}^{T} q(z_t|M,X)$ does inference by encoding subsequences



(a) Temporal abstraction



(b) Observation abstraction

Fig. 4.2. State transitions: inference and generation with a given hierarchical temporal structure based on the boundary indicators M.

determined by M and X. To use the same temporal abstraction across the time steps of a subsequence, the distribution $q(z_t|M, X)$ is also conditioned on the boundary indicator m_{t-1}:

$$q(z_t|M, X) = \begin{cases} \delta(z_t = z_{t-1}) & \text{if } m_{t-1} = 0 \text{ (COPY)}, \\ \tilde{q}(z_t|\psi_{t-1}^{\text{fwd}}, \psi_t^{\text{bwd}}) & \text{otherwise (UPDATE)}. \end{cases} \tag{4.3.4}$$

We use the distribution $\tilde{q}(z_t|\psi_{t-1}^{\text{fwd}}, \psi_t^{\text{bwd}})$ to update the state z_t. It is conditioned on all previous observations $x_{<t}$ and this is represented by a feature ψ_{t-1}^{fwd} extracted from a forward RNN $\psi^{\text{fwd}}(X)$. The other is a feature ψ_t^{bwd} representing the current step's subsequence that is extracted from a backward (masked) RNN $\psi^{\text{bwd}}(X, M)$. In particular, this RNN depends on M, which is used as a masking variable, to ensure independence between subsequences.

The observation abstraction predictor $q(S|Z, M, X) = \prod_{t=1}^{T} q(s_t|z_t, M, X)$ is factorized and each observational abstraction s_t is sampled from the distribution $q(s_t|z_t, M, X) = \tilde{q}\left(s_t|z_t, \phi_t^{\text{fwd}}\right)$. The feature ϕ_t^{fwd} is extracted from a forward (masked) RNN $\phi^{\text{fwd}}(X, M)$ that encodes the observation sequence X and resets hidden states when a new subsequence starts.

4.4. Related Works

The most similar work with our model is the HMRNN (Chung et al., 2016). While it is similar in the sense that both models discover the hierarchical temporal structure, HMRNN is a deterministic model and thus has a severe limitation to use for an imagination module. In the switching state-space model (Ghahramani and Hinton, 2000), the upper layer is a Hidden Markov Model (HMM) and the behavior of the lower layer is modulated by the discrete state of the upper layer, and thus gives hierarchical temporal structure. Linderman et al. (2016) proposed a new class of switching state-space models that discovers the dynamical units and also explains the switching behavior depending on observations or continuous latent states. The authors used inference based on message-passing. The hidden semi-Markov models (Yu, 2010; Dai et al., 2016) perform similar segmentation with discrete states. However, unlike our model, there is no states for temporal abstraction. Kipf et al. (2018) proposed soft-segmentation of sequence for compositional imitation learning.

The variational recurrent neural networks (VRNN) (Chung et al., 2015) is a latent variable RNN but uses auto-regressive state transition taking inputs from the observation. Thus, this can be computationally expensive to use as an imagination module. Also, the error can accumulate more severely in the high dimensional rollout. To resolve this problem, Krishnan et al. (2017) and Buesing et al. (2018) proposes to combine the traditional Markovian State Space Models with deep neural networks. Zheng et al. (2017) and Hafner et al. (2018) proposed to use an RNN path to encode the past making non-Markovian state-space models which can alleviate the limitation of the traditional SSMs. Serban et al. (2017) proposed

a hierarchical version of VRNN called Variational Hierarchical Recurrent Encoder-Decoder (VHRED) which results in a similar model as ours. However, it is a significant difference that our model learns the segment while VHRED uses a given structure. A closely related work is TDVAE (Gregor et al., 2019). TDVAE is trained on pairs of temporally separated time points. Jayaraman et al. (2019) and Neitz et al. (2018) proposed models that predict the future frames that, unlike our approach, have the lowest uncertainty. The resulting models predict a small number of easily predictable "bottleneck" frames through which any possible prediction must pass. Pertsch et al. (2019) proposed to predict the keyframes with their temporal offsets using stochastic prediction and deterministically interpolate the intermediate frames.

4.5. Experiments

We demonstrate our model on visual sequence datasets to show (1) how sequence data is decomposed into perceptually plausible subsequences without any supervision, (2) how jumpy future prediction is done with temporal abstraction and (3) how this jumpy future prediction can improve the planning as an imagination module in a navigation problem. Moreover, we test conditional generation $p(X|X_{\text{ctx}})$ where $X_{\text{ctx}} = x_{-(T_{\text{ctx}}-1):0}$ is the context observation of length T_{ctx}. With the context, we preset the state transition of the temporal abstraction by deterministically initializing $c_0 = f_{\text{ctx}}(X_{\text{ctx}})$ with f_{ctx} implemented by a forward RNN.

4.5.1. Bouncing Balls

We generated a synthetic 2D visual sequence dataset called *bouncing balls*. The dataset is composed of two colored balls that are designed to bounce in hitting the walls of a square box. Each ball is independently characterized with certain rules: (1) The color of each ball is randomly changed when it hits a wall and (2) the velocity (2D vector) is also slightly changed at every time steps with a small amount of noise. We trained a model to learn 1D state representations and all observation data $x_t \in \mathbb{R}^{32\times32\times3}$ are encoded and decoded by convolutional neural networks. During training, the length of observation sequence data X is set to $T = 20$ and the context length is $T_{\text{ctx}} = 5$. Hyper-parameters related to sequence decomposition are set as $N_{\max} = 5$ and $l_{\max} = 10$.

Our results in Figure 4.3 show that the sequence decomposer $q(M|X)$ predicts reasonable subsequences by setting a new subsequence when the color of balls is changed or the ball is bounced. At the beginning of training, the sequence decomposer is unstable with having large entropy and tends to define subsequences with a small number of frames. It then began to learn to increase the length of subsequences and this is controlled by annealing the temperature τ of Gumbel-softmax towards small values from 1.0 to 0.1. However, without our

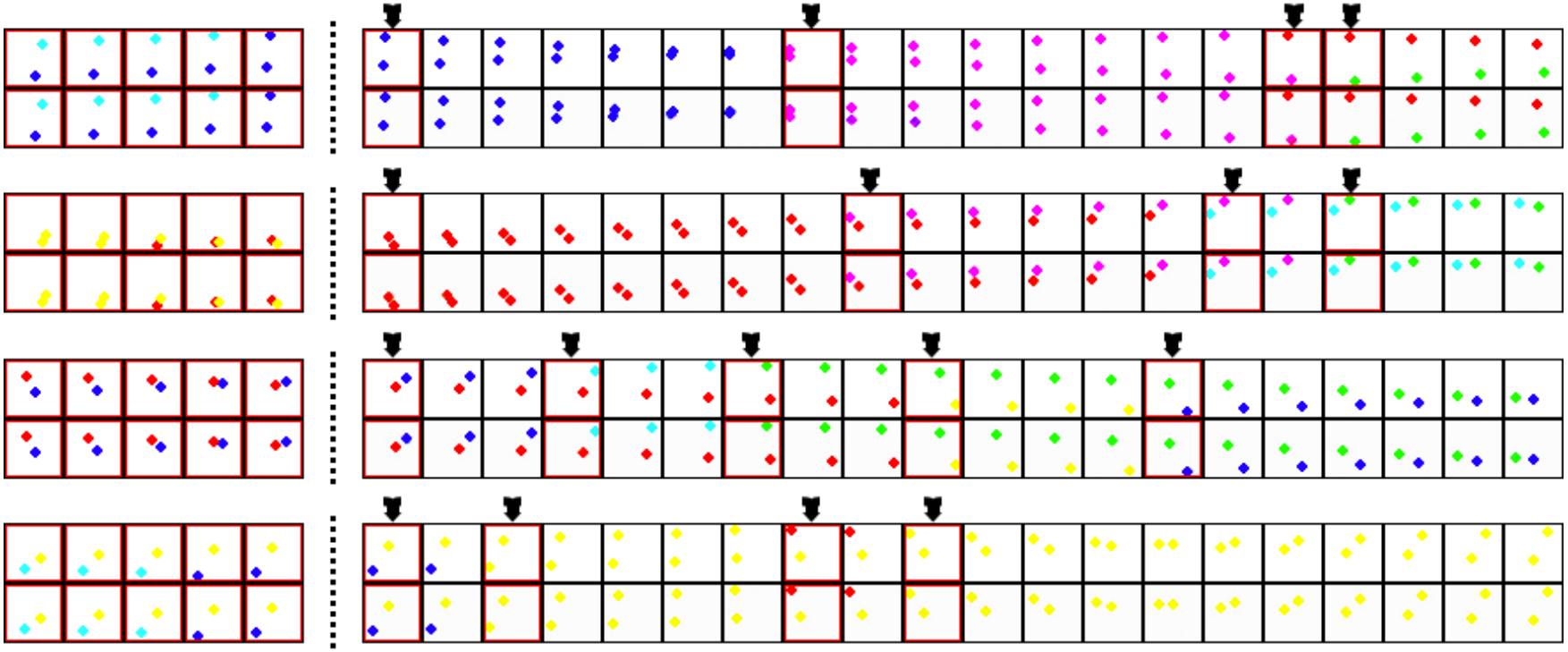


Fig. 4.3. Left: Previously observed (context) data X_{ctx}. **Right**: Each first row is the input observation sequence X and the second row is the corresponding reconstruction. The sequence decomposer $q(M|X)$ predicts the starting frames of subsequences and it is indicated by arrows (red squared frames). Subsequences are newly defined when a ball hits the wall by changing the color.

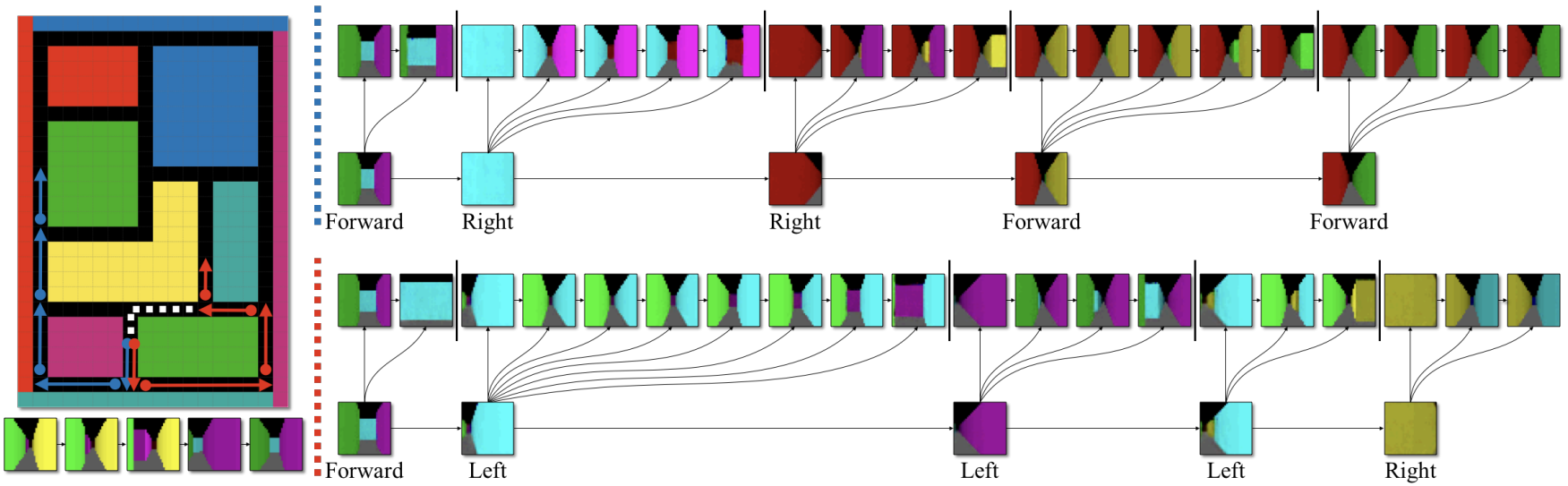


Fig. 4.4. Left: Bird's-eye view (map) of the 3D maze with generated navigation paths. White dotted lines indicate the given context path X_{ctx} and the corresponding frames are depicted below the map. Solid lines are the generated paths (blue: top, red: bottom) conditioned on the same context. Circles are the starting points of subsequences where the temporal abstract transitions exactly take place. **Right**: The generated sequence data is shown with its temporal structure. Both generations are conditioned on the same context but different input actions as indicated. Frame samples on each bottom row are generated with the temporal abstract transition $\tilde{p}\left(z_{t'}|c_{t'}\right)$ with $c_{t'} = f_{z\text{-rnn}}\left(z_{t'-1}, c_{t'-1}\right)$ and this shows how the jumpy future prediction is done. Other samples on top rows, which are not necessarily required for future prediction with our proposed HRSSM, are generated from the observation abstraction transition $\tilde{p}\left(s_t|h_t\right)$ with $h_t = f_{s\text{-rnn}}\left(s_{t-1}\|z_t, h_{t-1}\right)$. The boundaries between subsequences are determined by $p\left(m_t|s_t\right)$.

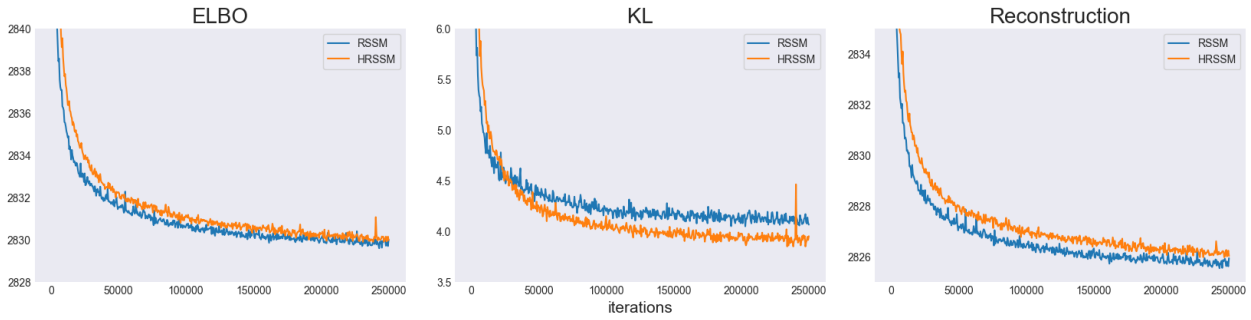


Fig. 4.5. The learning curve of RSSM and HRSSM: ELBO, KL-divergence and reconstruction loss.

proposed prior on temporal structure, the sequence decomposer fails to properly decompose sequences and our proposed model consequently converges into RSSM.

4.5.2. Navigation in 3D Maze

Another sequence dataset is generated from the 3D maze environment by an agent that navigates the maze. Each observation data $x_t \in \mathbb{R}^{32\times32\times3}$ is defined as a partially observed view observed by the agent. The maze consists of hallways with colored walls and is defined on a 26×18 grid map as shown in Figure 4.4. The agent is set to navigate around this environment and the viewpoint of the agent is constantly jittered with some noise. We set some constraints on the agent's action (*forward, left-turn, right-turn*) that the agent is not allowed to turn around when it is located on the hallway. However, it can turn around when it arrives nearby intersections between hallways. Due to these constraints, the agent without a policy can randomly navigate the maze environment and collect meaningful data.

To train an environment model, we collected 1M steps (frames) from the randomly navigating agent and used it to train both RSSM and our proposed HRSSM. For HRSSM, we used the same training setting as bouncing balls but different $N_{\max} = 5$ and $l_{\max} = 8$ for the sequence decomposition. The corresponding learning curves are shown in Figure 4.5 that both reached a similar ELBO. This suggests that our model does not lose the reconstruction performance while discovering the hierarchical structure. We trained state transitions to be action-conditioned and therefore this allows to perform action-controlled imagination. For HRSSM, only the temporal abstraction state transition is action-conditioned as we aim to execute the imagination only with the jumpy future prediction. The overall sequence generation procedure is described in Figure 4.4. The temporal structure of the generated sequence shows how the jumpy future prediction works and where the transitions of temporal abstraction occur. We see that our model learns to set each hallway as a subsequence and consequently to perform jumpy transitions between hallways without repeating or skipping a hallway. In Figure 4.6, a set of jumpy predicted sequences from the same context X_{ctx} and

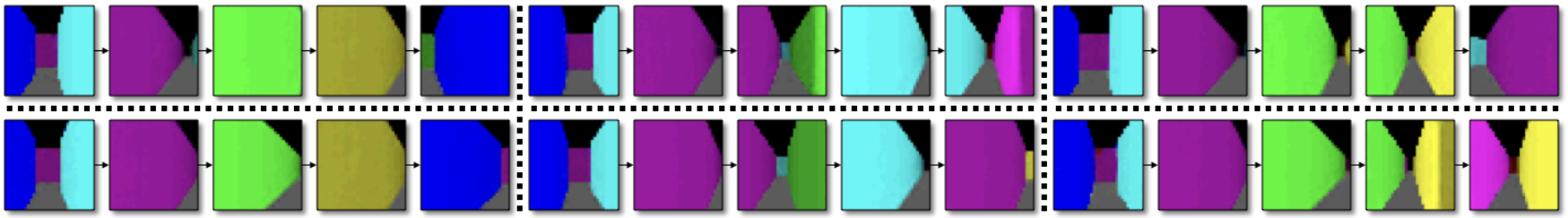


Fig. 4.6. Jumpy future prediction conditioned on the same context X_{ctx} and different input actions

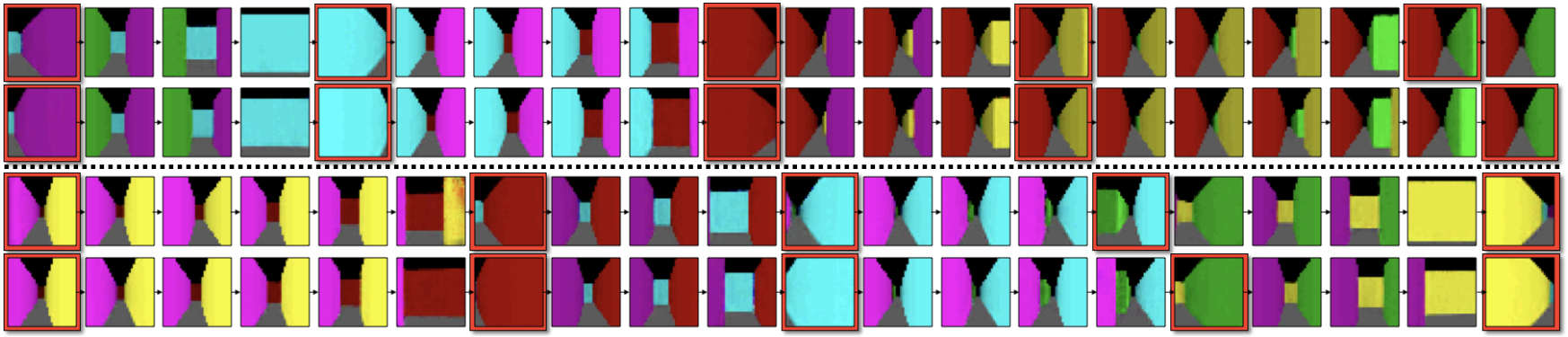


Fig. 4.7. Fully generate sequences conditioned on the same context X_{ctx} and same input actions: generated paths are equal but the viewpoint and the lengths of subsequences are varied (red squared frames are jumpy future predictions).

different input actions are shown and this can be interpreted as imaginations the agent can use for planning.

Goal-Oriented Navigation. We further use the trained model as an imagination module by augmenting it to an agent to perform the goal-oriented navigation. In this experiment, the task is to navigate to a randomly selected goal position within the given life steps. The goal position in the grid map is not provided to the agent, but a 3×3-cropped image around the goal position is given. To reach the goal fast, the agent is augmented with the imagination model and allowed to execute a rollout over a number of imagination trajectories (i.e., a sequence of temporal abstractions) by varying the input actions. Afterward, it decides the best trajectory that helps to reach the goal faster. To find the best trajectory, we use a simple strategy: a cosine-similarity based matching between all imagined state representations in imaginations and the feature of the goal image. The feature extractor for the goal image is jointly trained with the model. [4.1] This way, at every time step we let the agent choose the first action resulting in the best trajectory. This approach can be considered as a simple variant of the Monte Carlo Tree Search (MCTS) and the detailed overall procedure can be found in Appendix. Each episode is defined by randomly initializing the agent position and the goal position. The agent is allowed maximum 100 steps to reach the goal and the final reward is defined as the number of remaining steps when the agent reaches the goal or consumes all life-steps. The performance highly depends on the accuracy and the computationally efficiency of the model and we therefore compare between RSSM and HRSSM with varying the length of imagined trajectories. We measure the performance by randomly generated 5000 episodes and show how each setting performs across the episodes by plotting the reward distribution in Figure 4.8. It is shown that the HRSSM significantly improves the performance compared to the RSSM by having the same computational budget.

HRSSM showed consistent performance over different lengths of imagined trajectories and most episodes were solved within 50 steps. We believe that this is because HRSSM is able to abstract multiple time steps within a single state transition and this enables to reduce the computational cost for imaginations. The results also show that finding the best trajectory becomes difficult as the imagination length gets larger, i.e., the number of possible imagination trajectories increases. This suggests that imaginations with temporal abstraction can benefit both the accuracy and the computationally efficiency in effective ways.

[4.1] During training, the 3×3 window (image) around the agent position is always given as additional observation data and we trained feature extractor by maximizing the cosine-similarity between the extracted feature and the corresponding time step state representation.

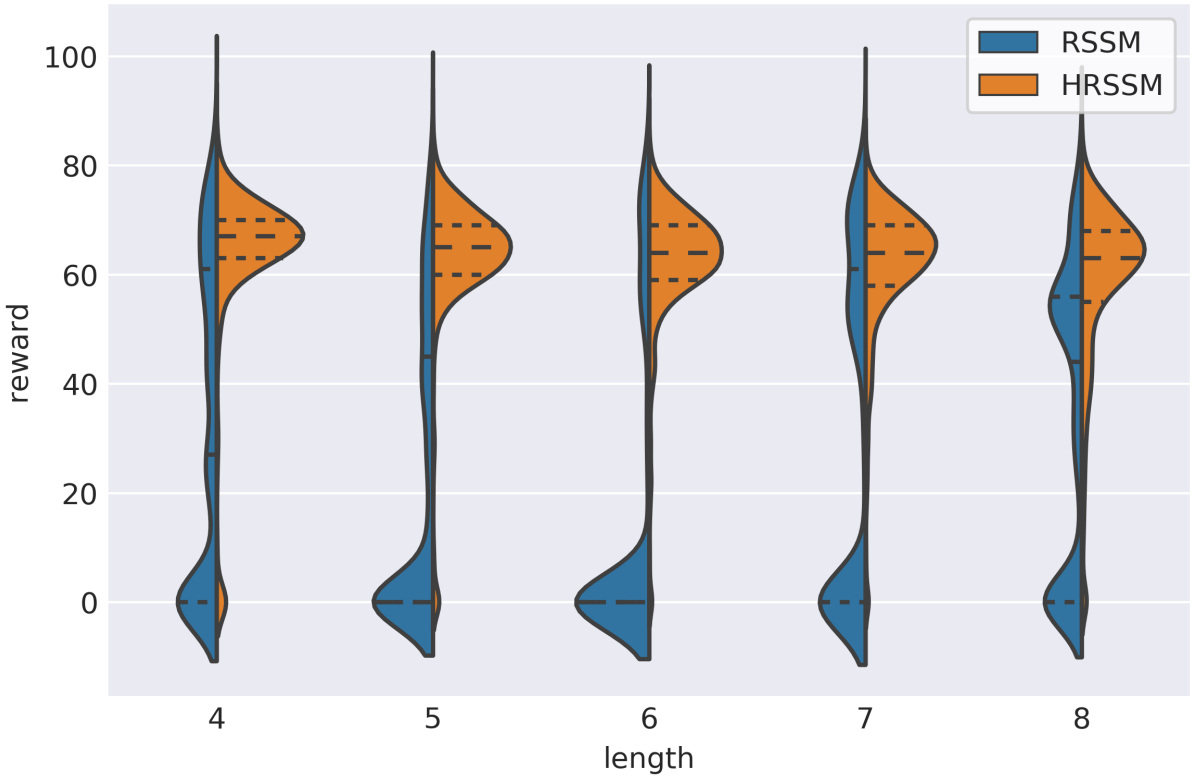


Fig. 4.8. Goal-oriented navigation with different lengths of imagined trajectories.

4.6. Conclusion

In this paper, we introduce the *Variational Temporal Abstraction* (VTA), a generic generative temporal model that can discover the hierarchical temporal structure and its stochastic hierarchical state transitions. We also propose to use this temporal abstraction for temporally-extended future imagination in imagination-augmented agent-learning. Experiment results shows that in general sequential data modeling, the proposed model discovers plausible latent temporal structures and perform hierarchical stochastic state transitions. Also, in connection to the model-based imagination-augmented agent for a 3D navigation task, we demonstrate the potential of the proposed model in improving the efficiency of agent-learning.

4.7. Appendix

4.7.1. Goal-Oriented Navigation

The Algorithm 4.1 describe the procedure of goal-oriented navigation with imagination. This is based on the imagination-based planner that is shown in Algorithm 4.2.

Algorithm 4.1 Goal-oriented navigation with imagination (single episode)

Input: environment `env`, environment model $\mathcal{E}$, max length of imagined trajectories l_{img}
Output: reward r

Initialize reward $r \leftarrow 100$
Initialize environment $x \leftarrow$ `env.reset()`
Initialize context $X_{\text{ctx}} \leftarrow [x]$
Sample goal position and extract goal map feature g
while $r > 0$ **do**
  Sample action a from imagination-based planner given $\mathcal{E}, X_{\text{ctx}}, g, l_{\text{img}}$ (Algorithm 4.2)
  Do action $x \leftarrow$ `env.step`(a)
  Update context $X_{\text{ctx}} \leftarrow X_{\text{ctx}} + [x]$
  Update reward $r \leftarrow r - 1$
  if current position is at the goal position **then**
    break
  end if
end while
return reward r

Algorithm 4.2 Imagination-based planner

Input: environment model $\mathcal{E}$, previously observed sequence (context) X_{ctx}, maximum length of imagined trajectories l_{img}, goal map feature g
Output: action $a_{\max}$

Initialize model $\mathcal{E}$ with $c_0 = f_{\text{ctx}}(X_{\text{ctx}})$
Initialize $d_{\max} \leftarrow -\infty$
Initialize $a_{\max} \leftarrow$ None
Set a list $\mathcal{A}$ of all possible action sequences based on l_{img}
for each action seuqence A in $\mathcal{A}$ **do**
  Get a sequence of states $\mathcal{S} \in \mathbb{R}^{l_{\text{img}} \times d}$ by doing imagination with model $\mathcal{E}$ conditioned on A
  Compute cosine-similarity $D \in \mathbb{R}^{l_{\text{img}}}$ between all states $\mathcal{S}$ and goal map feature g
  if $\max(D) > d_{\max}$ **then**
    Update $d_{\max} \leftarrow \max(D)$
    Update $a_{\max} \leftarrow A[0]$
  end if
end for
return action $a_{\max}$

4.7.2. Implementation Details

For bouncing balls, we define the reconstruction loss (data likelihood) by using binary cross-entropy. The images from 3D maze are pre-processed by reducing the bit depth to 5 bits (Kingma and Dhariwal, 2018) and therefore the reconstruction loss is computed by using Gaussian distribution. We used the AMSGrad (Reddi et al., 2018), a variant of Adam, optimizer with learning rate $5e-4$ and all mini-batchs are with 64 sequences with length $T=20$. Both CNN-based encoder and decoder are composed of 4 convolution layers with ELU activations (Clevert et al., 2016). A GRU (Cho et al., 2014) is used for all RNNs with 128 hidden units. The state representations of temporal abstraction and observation abstraction are sampled from 8-dimensional diagional Gaussian distributions.

4.7.3. Action-Conditioned Temporal Abstraction State Transition

We implement action-conditioned state transition as:

$$p\left(z_t|a_t, z_{<t}, m_{<t}\right) = \begin{cases} \delta(z_t = z_{t-1}) & \text{if } m_{t-1}=0 \text{ (COPY)}, \\ \tilde{p}(z_t|c_t) & \text{otherwise (UPDATE)} \end{cases}$$

where the action input a_t is only affecting the UPDATE operation and we feed it into the deterministic path as the following:

$$c_t = \begin{cases} c_{t-1} & \text{if } m_{t-1}=0 \text{ (COPY)}, \\ f_z\left(z_{t-1} \| a_t, c_{t-1}\right) & \text{otherwise (UPDATE)}. \end{cases}$$

4.7.4. Evidence Lower Bound (ELBO)

Log-likelihood $\log p\left(X\right)$**.**

$$\begin{aligned} \log p\left(X\right) &= \log \sum_M \int_{Z,S} p\left(X,Z,S,M\right) \\ &\geq \sum_M \int_{Z,S} q\left(Z,S,M|X\right) \log \frac{p\left(X,Z,S,M\right)}{q\left(Z,S,M|X\right)} \\ &= \sum_M \int_{Z,S} q\left(Z,S,M|X\right) \log \frac{p\left(X|Z,S\right) p\left(Z,S,M\right)}{q\left(Z,S,M|X\right)} \\ &= \sum_M \int_{Z,S} q\left(Z,S,M|X\right) \left[\log p\left(X|Z,S\right) + \log \frac{p\left(Z,S,M\right)}{q\left(Z,S,M|X\right)}\right] \\ &= \underbrace{\mathbb{E}_{q(Z,S,M|X)}\left[\log p\left(X|Z,S\right)\right]}_{\text{reconstruction}} - \underbrace{\mathrm{KL}\left[q\left(Z,S,M|X\right)||p\left(Z,S,M\right)\right]}_{\text{KL divergence}} \end{aligned}$$

Decomposing $p\left(X|Z,S\right)$ and $p\left(Z,S,M\right)$.

$$p\left(X|Z,S\right)=\prod_t \underbrace{p\left(x_t|s_t\right)}_{\text{decoder}}$$

$$p\left(Z,S,M\right)=\prod_t \underbrace{p\left(z_t|z_{t-1},m_{t-1}\right)}_{\text{temporal abstract transition}} \underbrace{p\left(s_t|s_{t-1},z_t,m_{t-1}\right)}_{\text{observation abstract transition}} \underbrace{p\left(m_t|s_t\right)}_{\text{boundary prior}}$$

Decomposing $q\left(Z,S,M|X\right)$.

$$\begin{aligned}
q\left(Z,S,M|X\right)&=q\left(M|X\right)q\left(Z,S|M,X\right)\\
&=q\left(M|X\right)q\left(Z|M,X\right)q\left(S|Z,M,X\right)\\
&=q\left(M|X\right)\prod_t q\left(z_t|M,X\right)q\left(s_t|z_t,M,X\right)\\
q(M|X)&=\prod_t q(m_t|X)=\prod_t \text{Bern}\left(m_t|\sigma(\varphi(X))\right)
\end{aligned}$$

Reconstruction Term in ELBO.

$$\begin{aligned}
&\sum_M \int_{Z,S} q\left(Z,S,M|X\right)\log p\left(X|Z,S\right)\\
&=\underbrace{\sum_M q\left(M|X\right)}_{\text{sample } M}\int_{Z,S} q\left(Z,S|M,X\right)\log p\left(X|Z,S\right)\\
&\approx \int_{Z,S} q\left(Z,S|M,X\right)\sum_t \log p\left(x_t|s_t\right)\\
&=\int_{Z,S}\sum_t q\left(z_t,s_t|M,X\right)\log p\left(x_t|s_t\right)\\
&=\int_{Z,S}\sum_t \underbrace{q\left(z_t|M,X\right)q\left(s_t|z_t,M,X\right)}_{\text{sampling } z_t \text{ and } s_t}\log p\left(x_t|z_t,s_t\right)\\
&\approx \sum_t \log p\left(x_t|s_t\right)
\end{aligned}$$

KL Term in ELBO.

$$\begin{aligned}
&\log q\left(Z,S,M|X\right)-\log p\left(Z,S,M\right)\\
&=\log q\left(M|X\right)+\log q\left(Z,S|M,X\right)-\log p\left(Z,S,M\right)\\
&=\log q\left(M|X\right)+\log q\left(Z|M,X\right)+\log q\left(S|Z,M,X\right)-\log p\left(Z,S,M\right)\\
&=\log q\left(M|X\right)+\sum_t \log\frac{q\left(z_t|M,X\right)q\left(s_t|z_t,M,X\right)}{p\left(z_t|z_{t-1},m_{t-1}\right)p\left(s_t|s_{t-1},z_t,m_{t-1}\right)p\left(m_t|s_t\right)}\\
&=\log q\left(M|X\right)+\sum_t \log\frac{q\left(z_t|M,X\right)}{p\left(z_t|z_{t-1},m_{t-1}\right)}+\log\frac{q\left(s_t|z_t,M,X\right)}{p\left(s_t|s_{t-1},z_t,m_{t-1}\right)}+\log\frac{1}{p\left(m_t|s_t\right)}
\end{aligned}$$

$$
\begin{aligned}
&\sum_{M} \int_{Z,S} q\left(Z,S,M|X\right) \left[\log q\left(Z,S,M|X\right) - \log p\left(Z,S,M\right)\right] \\
&= \sum_{M} q\left(M|X\right) \int_{Z,S} q\left(Z,S|M,X\right) \left[\log q\left(Z,S,M|X\right) - \log p\left(Z,S,M\right)\right] \\
&= \sum_{M} q\left(M|X\right) \int_{Z,S} q\left(Z,S|M,X\right) \left[\log q\left(M|X\right) + \sum_{t} \log \frac{q\left(z_t|M,X\right) q\left(s_t|z_t,M,X\right)}{p\left(z_t|z_{t-1},m_{t-1}\right) p\left(s_t|s_{t-1},z_t,m_{t-1}\right) p\left(m_t|s_t\right)}\right] \\
&= \sum_{M} q\left(M|X\right) \left[\log q\left(M|X\right) + \sum_{t} \int_{Z,S} q\left(Z,S|M,X\right) \log \frac{q\left(z_t|M,X\right) q\left(s_t|z_t,M,X\right)}{p\left(z_t|z_{t-1},m_{t-1}\right) p\left(s_t|s_{t-1},z_t,m_{t-1}\right) p\left(m_t|s_t\right)}\right] \\
&= \sum_{M} q\left(M|X\right) \left[\log q\left(M|X\right) + \sum_{t} \int_{z_t,s_t} q\left(z_t,s_t|M,X\right) \log \frac{q\left(z_t|M,X\right) q\left(s_t|z_t,M,X\right)}{p\left(z_t|z_{t-1},m_{t-1}\right) p\left(s_t|s_{t-1},z_t,m_{t-1}\right) p\left(m_t|s_t\right)}\right] \\
&= \sum_{M} q\left(M|X\right) \left[\log q\left(M|X\right) + \sum_{t} \mathrm{KL}\left(q'(z_t)||p'(z_t)\right) + \int_{z_t,s_t} q\left(z_t,s_t|M,X\right) \log \frac{q\left(s_t|z_t,M,X\right)}{p\left(s_t|s_{t-1},z_t,m_{t-1}\right) p\left(m_t|s_t\right)}\right] \\
&= \sum_{M} q\left(M|X\right) \left[\log q\left(M|X\right) + \sum_{t} \mathrm{KL}\left(q'(z_t)||p'(z_t)\right) + \int_{z_t} \underbrace{q'\left(z_t\right)}_{\text{sample } z_t} \left[\mathrm{KL}\left(q'(s_t)||p'(s_t)\right) - \log p\left(m_t|s_t\right)\right]\right] \\
&\approx \sum_{M} q\left(M|X\right) \left[\log q\left(M|X\right) + \sum_{t} \mathrm{KL}\left(q'(z_t)||p'(z_t)\right) + \mathrm{KL}\left(q'(s_t)||p'(s_t)\right) - \log p\left(m_t|s_t\right)\right] \\
&= \sum_{M} q\left(M|X\right) \left[\log \frac{\prod_t q\left(m_t|X\right)}{\prod_t p\left(m_t|s_t\right)} + \sum_{t} \mathrm{KL}\left(q'(z_t)||p'(z_t)\right) + \mathrm{KL}\left(q'(s_t)||p'(s_t)\right)\right] \\
&= \sum_{t'} \mathrm{KL}\left(q'\left(m_{t'}\right)||p'\left(m_{t'}\right)\right) + \sum_{M} \underbrace{q\left(M|X\right)}_{\text{sample } M} \left[\sum_{t} \mathrm{KL}\left(q'\left(z_t\right)||p'\left(z_t\right)\right) + \mathrm{KL}\left(q'\left(s_t\right)||p'\left(s_t\right)\right)\right] \\
&\approx \sum_{t} \underbrace{\mathrm{KL}\left(q'\left(m_t\right)||p'\left(m_t\right)\right)}_{\text{sequence decomposer}} + \underbrace{\mathrm{KL}\left(q'\left(z_t\right)||p'\left(z_t\right)\right)}_{\text{temporal abstraction}} + \underbrace{\mathrm{KL}\left(q'\left(s_t\right)||p'\left(s_t\right)\right)}_{\text{observation abstraction}}
\end{aligned}
$$

where $p'(m_t) = p(m_t|s_t), p'(z_t) = p\left(z_t|z_{t-1},m_{t-1}\right), p'(s_t) = p\left(s_t|s_{t-1},z_t,m_{t-1}\right)$

$$q'(m_t) = q(m_t|X), q'(z_t) = q\left(z_t,s_t|M,X\right), q'(s_t) = q\left(s_t|z_t,M,X\right)$$

Chapter 5

Visual Concept Reasoning Networks

Taesup Kim, Sungwoong Kim and Yoshua Bengio
Appeared in:
- *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI 2021)*

Personal Contribution. Based on Yoshua Bengio's consciousness prior (Bengio, 2017), I came up with an idea to apply its concept to convolutional neural networks for visual recognition tasks. In particular, Sungwoong Kim and I were also interested in the topic of global reasoning for visual recognition tasks that we had heavily discussed for a while. Then, I discovered the fact that ResNeXt (Xie et al., 2017) had an interesting architecture that could handle multiple visual concepts with its multiple branches and be implemented in a compact way. I tried to modify the ResNeXt by introducing some additional modules to enable high-level global reasoning. Kakao Brain members also supported us by providing some implementations to run a large number of experiments on visual recognition tasks. Yoshua Bengio advised me by providing some meaningful comments to improve the performance. I was the main writer for this article, and Sungwoong Kim and Yoshua Bengio did the final editing.

Context. A split-transform-merge strategy has been broadly used as an architectural constraint in convolutional neural networks for visual recognition tasks. It approximates sparsely connected networks by explicitly defining multiple branches to simultaneously learn representations with different visual concepts or properties. However, dependencies or interactions between these representations are typically implemented by dense and local operations, and also any adaptiveness or high-level reasoning is lacking. In this paper, we therefore introduced the concept of *context-level adaptation* to improve over existing strategies. We combined it with our *Visual Concept Reasoning Networks* (VCRNet) to enable reasoning between high-level visual concepts. We softly associated each branch with a visual concept and derived a compact concept state by selecting a few local descriptors through

an attention module. We described our proposed model in terms of split-transform-*attend-interact-modulate*-merge stages, which are implemented by opting for a highly modularized architecture. Extensive experiments on visual recognition tasks such as image classification, semantic segmentation, object detection, scene recognition, and action recognition showed that our proposed model, VCRNet, can consistently improve performance by increasing the number of parameters by less than 1%.

Keywords: visual concepts, multi-branch architecture, visual recognition, attention, reasoning, global context, feature-wise modulation

5.1. Introduction

Convolutional neural networks have shown notable success in visual recognition tasks by learning hierarchical representations. The main properties of convolutional operations, which are local connectivity and weight sharing, are the key factors that make it more efficient than fully-connected networks for processing images. The local connectivity particularly comes up with a fundamental concept, receptive field, that defines how far the local descriptor can capture the context in the input image. In principle, the receptive field can be expanded by stacking multiple convolutional layers or increasing their kernel size. However, it is known that the effective receptive field only covers a fraction of the theoretical size of it (Luo et al., 2016). This eventually prevents convolutional neural network from capturing the global context based on long-range dependencies. On the other hand, most convolutional neural networks are characterized by dense and local operations that take advantage of the weight sharing property. Hence, it hence typically lacks internal mechanism for high-level reasoning based on abstract semantic concepts such as those humans manipulate with natural language and inspiring inductive biases based on modern theories of consciousness (Bengio, 2017; Goyal et al., 2020). It is related to system 2 cognitive abilities, which include things like reasoning, planning, and imagination, that are assumed to capture the global context from interactions each involving a few abstract factors and accordingly give feedback to the local descriptor for decision-making.

There have been approaches to enhance capturing long-range dependencies such as non-local networks (Wang et al., 2018). The main concept of it, which is related to self-attention (Vaswani et al., 2017), is to compute a local descriptor by adaptively aggregating other descriptors from all positions, regardless of relative spatial distance. In this setting, the image feature map is plugged into a fully-connected graph neural network, where all local positions are fully connected to all others. It is able to capture long-range dependencies and extract the global context, but it still works with dense operations and lacks high-level reasoning. Both LatentGNN (Zhang et al., 2019) and GloRe (Chen et al., 2019) alleviate these issues by introducing compact graph neural networks with some latent nodes designed to aggregate local descriptors.

In this work, we propose *Visual Concept Reasoning Networks* (VCRNet) to enable reasoning between high-level visual concepts. We exploit a modularized multi-branch architecture that follows a split-transform-merge strategy (Xie et al., 2017). While it explicitly has multiple branches to simultaneously learn multiple visual concepts or properties, it only considers the dependencies or interactions between them by using dense and local operations. We extend the architecture by split-transform-*attend-interact-modulate*-merge stages, and this allows the model to capture the global context by reasoning with sparse interactions between high-level visual concepts from different branches.

The main contributions of the paper are as follows:

- We propose Visual Concept Reasoning Networks (VCRNet) that efficiently capture the global context by reasoning over high-level visual concepts.
- We compactly implement our proposed model by exploiting a modularized multi-branch architecture composed of split-transform-attend-interact-modulate-merge stages.
- We show that our proposed model improves the performance more than other models while increasing the number of parameters by less than 1% on multiple visual recognition tasks.

5.2. Related Works

Multi-branch architectures are carefully designed with multiple branches characterized by different dense operations, and split-transform-merge stages are used as the building blocks. The Inception models (Szegedy et al., 2015) are one of the successful multi-branch architectures that define branches with different scales to handle multiple scales. ResNeXt (Xie et al., 2017) is another version of ResNet (He et al., 2016) having multiple branches with the same topology in residual blocks, and it is efficiently implemented by grouped convolutions. In this work, we utilize this residual block and associate each branch of it with a visual concept.

There have been several works to adaptively modulate the feature maps based on the external context or the global context of input data. Squeeze-and-Excitation networks (SE, Hu et al. (2018)) use a gating mechanism to do channel-wise re-scaling in accordance with the channel dependencies based on the global context. Gather-Excite networks (GE, Hu et al. (2018)) further re-scale locally and are able to finely redistribute the global context to the local descriptors. Convolutional block attention module (CBAM, Woo et al. (2018)) independently and sequentially applies channel-wise and spatial-wise gating networks to modulate the feature maps. All these approaches extract the global context by using the global average pooling, which equally attends all local positions. Dynamic layer normalization (DLN, Kim et al. (2017)) and Feature-wise Linear Modulation (FiLM, Perez et al. (2018)) present a method of feature modulation on normalization layers by conditioning on the global context and the external context, respectively.

Content-based soft-attention mechanisms (Bahdanau et al., 2015) have been broadly used on neural networks to operate on a set of interchangeable objects and aggregate it. Particularly, Transformer models (Vaswani et al., 2017) have shown impressive results by using multi-head self-attention modules to improve the ability of capturing long-range dependencies. Non-local networks (NL, Wang et al. (2018)) use this framework in pixel-level self-attention blocks to implement non-local operations. There are some additional related works,

including one that augments the self-attention modules with convolutional operations (Bello et al., 2019), and another that replaces all of them with a form of self-attention (Ramachandran et al., 2019). Global-context networks (GC, Cao et al. (2019)) simplify the non-local networks by replacing the pixel-level self-attention with an attention module having a single fixed query that is globally shared and learned. Attention-augmented convolutional networks (Bello et al., 2019) similarly augment convolutional operators with self-attention modules as the non-local networks, but concatenate feature maps from the convolution path and self-attention path. LatentGNN (Zhang et al., 2019) and the global reasoning module (GloRe, Chen et al. (2019)) differently simplify the non-local networks: they first map local descriptors into latent nodes, where the number of nodes is smaller than the number of local positions, and they capture long-range dependencies from interactions between the latent nodes. Our proposed model is similar to these two models, but we take the advantage of the multi-branch architecture and the attention mechanism to efficiently extract a set of distinct visual concept states from the input data.

5.3. Methods

In this section, we introduce our proposed model, Visual Concept Reasoning Networks (VCRNet), and describe the overall architecture and its components in detail. The proposed model is designed to reason over high-level visual concepts and accordingly modulate feature maps based on its result. In the following, we assume the input data $X \in \mathbb{R}^{HW \times d}$ is a 2D tensor as an image feature map, where H, W, and d refer to the height, width, and feature size of X, respectively. Moreover, for simplicity, we denote all modules by a function $F_{\text{func}}(\cdot; \theta)$, where θ is a learnable parameter and the subscript 'func' briefly explains the functionality of the module.

5.3.1. Modularized Multi-Branch Residual Block

Residual blocks are composed of a skip connection and multiple convolutional layers (He et al., 2016). We especially take advantage of using a residual block of ResNeXt (Xie et al., 2017) that operates by grouped convolutions. This block is explicable by a *split-transform-merge* strategy and a highly modularized multi-branch architecture. It has an additional dimension of *cardinality* to define the number of branches used in the block. The branches are defined by separate networks, which are based on the same topology and implemented by grouped convolutions, processing non-overlapping low-dimensional feature maps. In this work, we use this block by regarding each branch as a network learning a representation of a specific *visual concept* and, therefore, refer to the cardinality as the number of visuals concepts C.

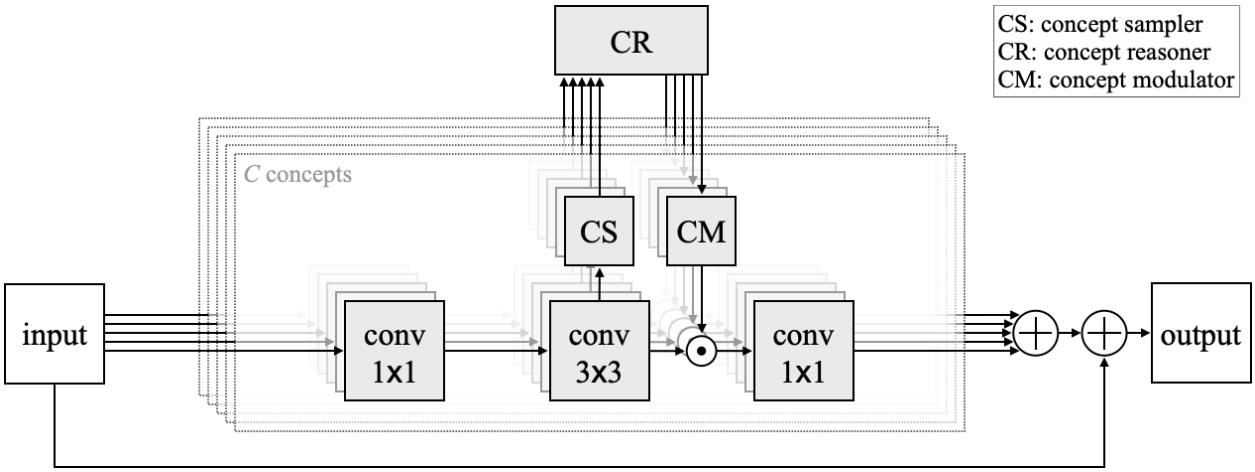


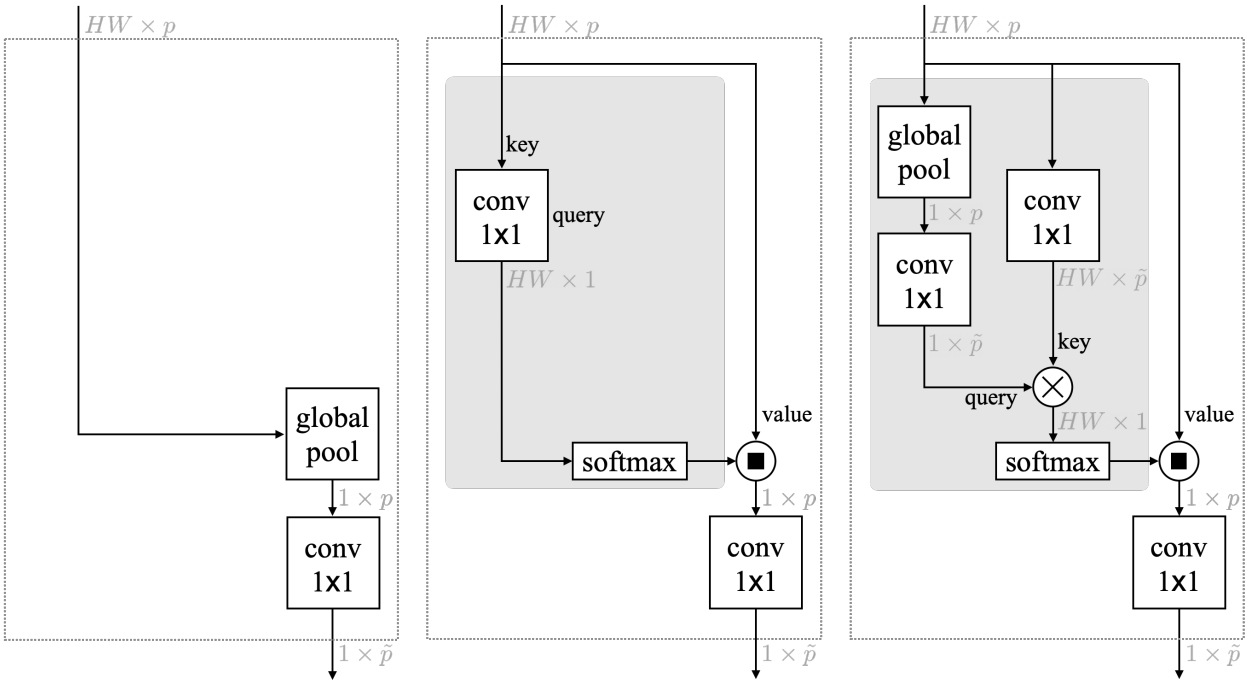
Fig. 5.1. A residual block with visual concept reasoning modules: (1) concept sampler, (2) concept reasoner, and (3) concept modulator. Multiple concepts are processed in parallel by implementing each modules with group convolutions.

The split-transform-merge strategy can be described by visual concept processing as follows. Each concept c has a compact concept-wise feature map $Z_c \in \mathbb{R}^{HW \times p}$, where p is a lot smaller than d. It is initially extracted from the input data X by *splitting* it into a low-dimensional feature map $\tilde{X}_c \in \mathbb{R}^{HW \times p}$ with a 1×1 convolution $F_{\text{split}}(X; \theta_c^{\text{split}})$. Afterward, it is followed by a *concept-wise transformation* based on a 3×3 convolution $F_{\text{trans}}(\tilde{X}_c; \theta_c^{\text{trans}})$ while keeping the feature size compact. The extracted concept-wise feature maps $\{Z_c\}_{c=1}^{C}$ are then projected back into the input space to be *merged* as $Y = X + \sum_{c=1}^{C} F_{\text{merge}}(Z_c; \theta_c^{\text{merge}})$. This overall multi-branch procedure interestingly can be highly modularized and parallelized by grouped convolutions. However, it lacks the ability of reasoning over the high-level visual concepts that capture both local and global contexts.

We propose to extend this approach by introducing additional modules to enable visual concept reasoning. Our proposed model is based on a new strategy with split-transform-*attend-interact-modulate*-merge stages. The new stages completely work inside the residual block with the following modules: (a) concept sampler, (b) concept reasoner, and (c) concept modulator. The overall architecture is depicted in Figure 5.1 showing how it is highly modularized, sharing the topology across different concepts. We refer to networks having residual blocks with these modules, as Visual Concept Reasoning Networks (VCRNet).

5.3.2. Concept Sampler

The concept-wise feature maps $\{Z_c\}_{c=1}^{C}$ are composed of all possible pixel-level local descriptors, which contain spatially local feature information, as sets of vectors. To do efficient reasoning over the visual concepts, it first requires a set of abstract feature vectors representing the visual concepts. Therefore, a form of aggregation mechanism is necessary to derive



(a) Pooling-based concept sampler (b) Attention-based concept sampler (static/shared query) (c) Attention-based concept sampler (dynamic/adaptive query)

Fig. 5.2. Concept samplers with different approaches ($\otimes$ is a weighted-sum operation).

a set of visual concept states, where each state is a vector, from the concept-wise feature maps. We implement this by presenting a *concept sampler* (CS) module. Each concept c has a separate concept sampler $F_{\text{CS}}(Z_c; \theta_c^{\text{CS}})$ that aggregates the set of local descriptors in Z_c and converts it into a concept state $h_c \in \mathbb{R}^{1\times\tilde{p}}$, where we set $\tilde{p} = \min(p/4, 4)$. We introduce two types of concept samplers that are based on pooling and attention operations, respectively.

Pooling-based Sampler. Global average pooling is one of the simplest ways to extract the global context from a feature map without explicitly capturing long-range dependencies. It equally and densely attends all local positions to aggregate the local descriptors. Our pooling-based sampler adopts this operation to compute the concept state h_c as shown in Figure 5.2.a, and it is formulated as:

$$h_c = F_{\text{GAP}}(Z_c)W_c^{\text{v}} = \left(\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W} Z_c[i,j]\right) W_c^{\text{v}}, \tag{5.3.1}$$

where $Z_c[i,j] \in \mathbb{R}^{1\times p}$ is a local descriptor at position (i,j), and $W_c^{\text{v}} \in \mathbb{R}^{p\times\tilde{p}}$ is a learnable projection weight. In comparison with the attention-based sampler, it is simple and compact having a small number of parameters, but there is no data-adaptive process. Due to its simplicity, similar approaches have been broadly used in previous works such as SENet (Hu et al., 2018) and CBAM (Woo et al., 2018).

Attention-based Sampler. The attention mechanism operates by mapping a query vector and a set of interchangeable key-value vector pairs into a single vector, which is a weighted sum of value vectors. It allows us to aggregate a set of local descriptors by sparsely and adaptively selecting them. We hence apply this approach to our concept sampler. For each concept c, the query vector $q_c \in \mathbb{R}^{1\times\tilde{p}}$ describes what to focus on during aggregation. The concept-wise feature map Z_c converts into a set of key-value vector pairs that we separately project into a key map $K_c = Z_c W_c^{\mathrm{k}}$ and a value map $V_c = Z_c W_c^{\mathrm{v}}$, where $W_c^{\mathrm{k}}, W_c^{\mathrm{v}} \in \mathbb{R}^{p\times\tilde{p}}$ are learnable projection weights. The concept state h_c is derived by computing the dot products of the query vector q_c with the key map K_c and subsequently applying a softmax function to obtain the attention weights over the value map V_c as:

$$h_c = \mathrm{softmax}\left(\frac{q_c\left(Z_c W_c^{\mathrm{k}}\right)^\top}{\sqrt{\tilde{p}}}\right)\left(Z_c W_c^{\mathrm{v}}\right). \tag{5.3.2}$$

The query vector q_c can be either learned as a model parameter or computed by a function of the feature map Z_c. The former approach defines a static query that is shared globally over all data. GCNet (Cao et al., 2019) uses this approach, instead of global average pooling, to extract the global context. It can be simplified and implemented by replacing the term $q_c\left(Z_c W_c^{\mathrm{k}}\right)^\top$ in Equation 5.3.2 with a 1×1 convolution as depicted in Figure 5.2.b. The latter approach, in contrast, uses a dynamic query that varies according to Z_c. We set the query as an output of a function as $q_c = F_{\mathrm{GAP}}(Z_c)W_c^{\mathrm{q}}$, where F_{GAP} is equal to the pool-based sampler, as shown in Equation 5.3.1.

Difference with Multi-head Self-attention. The concept samplers can be viewed as multi-head attention modules in Transformer models (Vaswani et al., 2017), where we set each concept to be operated by a single-head attention module. However, our concept samplers don't process the same input feature map as Transformers do. Each concept is only accessible to its corresponding feature map, and this encourages the concept samplers to attend and process different features. Moreover, we explicitly define concept-wise queries to aggregate pixel-wise (low-level) descriptors and obtain global descriptors (high-level concepts) rather than only capturing long-range dependencies as non-local networks (Wang et al., 2018), which work with pixel-level dense self-attention operations.

5.3.3. Concept Reasoner

The visual concept states are extracted independently from separate branches among which no communication exists. Therefore, we introduce a reasoning module, the *Concept Reasoner* (CR), to make the visual concept states interact with each other and accordingly update them. We opt for using a graph-based method by defining a fully-connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V}$ and directional edges $\mathcal{E}$. The node $v_c \in \mathcal{V}$ corresponds to a single

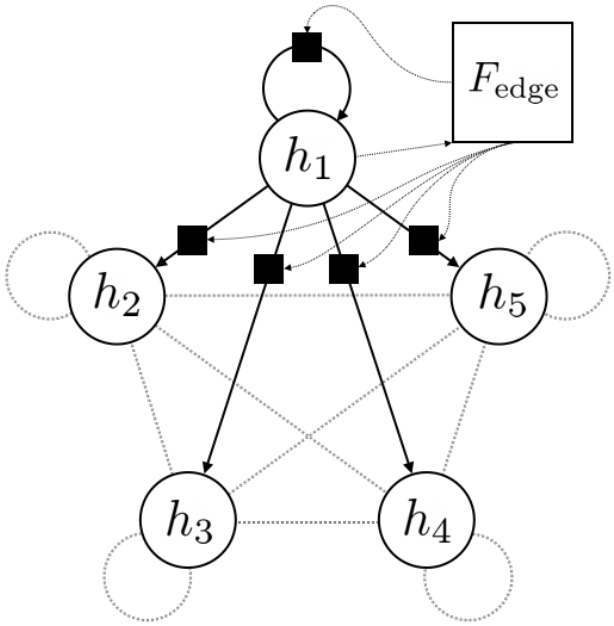


Fig. 5.3. Concept reasoner.

visual concept c and is described by the visual concept state h_c. The edge $e_{cc'} \in \mathcal{E}$ defines the relationship or dependency between visual concepts c and c'. It is further specified by an adjacency matrix $A \in \mathbb{R}^{C \times C}$ to represent edge weight values in a matrix form. Based on this setting, we describe the update rule of the visual concept states as:

$$\tilde{h}_c = \text{ReLU}\left(\text{BN}\left(h_c + \sum_{c'=1}^{C} A[c, c'] h_{c'}\right)\right), \tag{5.3.3}$$

where $A[c, c'] \in \mathbb{R}$ is a edge weight value, and batch normalization (BN) and ReLU activation are used. This can also be implemented in a matrix form as $\tilde{H} = \text{ReLU}(\text{BN}(H + AH))$, where $H = [h_1; h_2; ...; h_C] \in \mathbb{R}^{C \times \tilde{p}}$ is vertically stacked concept states. The adjacency matrix A can be treated as a module parameter that is learned during training. This sets the edges to be static so that all relationships between visual concepts are consistently applied to all data. However, we relax this constraint by dynamically computing the edge weights based on the concept states. A function $A[c, :] = F_{\text{edge}}(h_c; W^{\text{edge}}) = \tanh(h_c W^{\text{edge}})$, where $W^{\text{edge}} \in \mathbb{R}^{\tilde{p} \times C}$ is a learnable projection weight, is used to get all edge weights $A[c, :]$ related to the concept c as shown in Figure 5.3. This function learns how each concept adaptively relates to the others based on its state.

5.3.4. Concept Modulator

The updated concept states are regarding not only a single concept, but also the others as a result of reasoning based on interactions. This information has to be further propagated to local concept features, which are extracted from the main stream of the network. However,

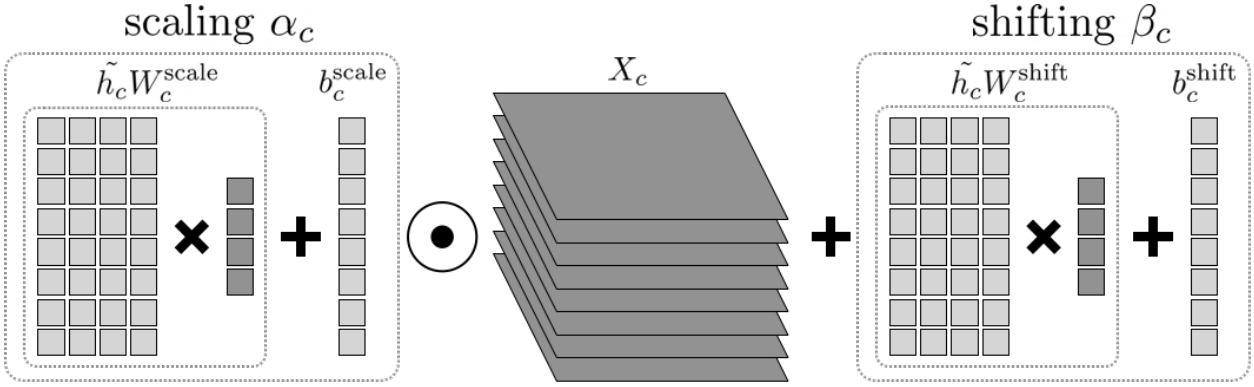


Fig. 5.4. Concept modulator.

this is a non-trivial problem due to dimensional mismatch, since the concept states are vectors not containing any explicit spatial information. We alleviate this issue by implementing a module, the *Concept Modulator* (CM), which is based on a feature modulation approach. It modulates the concept-wise feature maps by channel-wise scaling and shifting operations. These operations are conditioned on the updated concept states to fine-tune the feature maps based on the result of reasoning. We design this module based on DLN (Kim et al., 2017) and FiLM (Perez et al., 2018). Both models use feature-wise affine transformations on normalization layers by dynamically generating the affine parameters instead of learning them. In this way, we define separate modules for the visual concepts as shown in Figure 5.4. Each concept-wise feature map X_c is modulated as:

$$\begin{aligned} \tilde{X}_c = F_{\text{CM}}(\tilde{h}_c, X_c; \theta_c^{\text{CM}}) = \text{ReLU}\left(\alpha_c \odot X_c + \beta_c\right), \\ \alpha_c = \tilde{h}_c W_c^{\text{scale}} + b_c^{\text{scale}},\ \beta_c = \tilde{h}_c W_c^{\text{shift}} + b_c^{\text{shift}}, \end{aligned} \tag{5.3.4}$$

where $\odot$ indicates channel-wise multiplication. $\alpha_c, \beta_c \in \mathbb{R}^{1\times p}$ are scaling and shifting parameters, respectively, which are adaptively computed by linearly mapping the updated concept state $\tilde{h}_c$.

Pixel-level Modulator. We further implement pixel-level concept modulators to propagate the global context adaptively and differently into local descriptors. Each concept state h_c is derived by computing the attention map $M_c \in \mathbb{R}^{HW\times 1}$ from the concept sampler as shown in Equation 5.3.2, and we assume it contains the spatial information related to the concept c. Therefore, we utilize this attention map for the pixel-level concept modulator. We first re-normalize the attention map by its maximum value:

$$\begin{aligned} \tilde{M}_c = \frac{M_c}{\max(M_c)}, \\ \text{where } M_c = \text{softmax}\left(\frac{q_c\left(Z_c W_c^{\text{k}}\right)^\top}{\sqrt{\tilde{p}}}\right). \end{aligned} \tag{5.3.5}$$

Table 5.1. Comprehensive results of image classification on the ImageNet validation set

Model	Error (%)		# of Params	GFLOPs
	Top-1	Top-5		
ResNeXt-50 (Xie et al., 2017)	21.10	5.59	25.03M	4.24
ResNeXt-50 + SE (Hu et al., 2018)	20.79	5.38	27.56M	4.25
ResNeXt-50 + CBAM (Woo et al., 2018)	20.73	5.36	27.56M	4.25
ResNeXt-50 + GC (Cao et al., 2019)	20.44	5.34	27.58M	4.25
ResNeXt-50 + GloRe (Chen et al., 2019)	20.15	5.14	30.79M	5.86
ResNeXt-50 + VCR (ours)	19.97	**5.03**	25.26M	4.26
ResNeXt-50 + VCR (ours, pixel-level)	**19.94**	5.18	25.26M	4.29
ResNeXt-101 (Xie et al., 2017)	19.82	4.96	44.18M	7.99
ResNeXt-101 + SE (Hu et al., 2018)	19.39	4.73	48.96M	8.00
ResNeXt-101 + CBAM (Woo et al., 2018)	19.60	4.87	48.96M	8.00
ResNeXt-101 + GC (Cao et al., 2019)	19.52	5.03	48.99M	8.00
ResNeXt-101 + GloRe (Chen et al., 2019)	19.56	4.85	49.93M	9.61
ResNeXt-101 + VCR (ours)	**18.84**	**4.48**	44.60M	8.01

Without this re-normalization, the learning doesn't work properly. The re-normalized attention map $\tilde{M}_c$ is used to project the updated concept state $\tilde{h}_c$ into all local positions by projection $\tilde{M}_c\tilde{h}_c \in \mathbb{R}^{HW\times\tilde{p}}$. Based on this projection, we are able to do pixel-level feature modulation as:

$$\begin{aligned} \tilde{X}_c &= F_{\text{CM}}(\tilde{h}_c, \tilde{M}_c, X_c; \theta_c^{\text{CM}}) = \text{ReLU}\left(\alpha_c \cdot X_c + \beta_c\right) \\ \alpha_c &= \left(\tilde{M}_c\tilde{h}_c\right) W_c^{\text{scale}} + b_c^{\text{scale}},\ \beta_c = \left(\tilde{M}_c\tilde{h}_c\right) W_c^{\text{shift}} + b_c^{\text{shift}}, \end{aligned} \tag{5.3.6}$$

where $\cdot$ is an element-wise multiplication. Both α_c and β_c have the same size as the feature map X_c so that all local positions have separate scaling and shifting parameters.

5.4. Experiments

In this section, we describe our experimental settings on visual recognition tasks such as image classification, object detection/segmentation, scene recognition, and action recognition with large-scale datasets. In all experiments, we use ResNeXt (Xie et al., 2017), which performs better than ResNet (He et al., 2016) with less parameters, as a base architecture with cardinality = 32 and base width = 4d. As our main contribution is to exploit the *multi-branch* architecture to enable high-level concept reasoning by implementing split-transform-*attend-interact-modulate*-merge stages, we only use the ResNeXt as a backbone network, because it is the only one already having the split-transform-merge stages allowing us to seamlessly implement our VCRNet. Furthermore, our proposed model, VCRNet, is also defined by $C = 32$ concepts in all residual blocks. We also compare VCRNet against other networks (modules), which have a form of attention or reasoning modules, such as Squeeze-and-Excitation (SE, Hu et al. (2018)), Convolutional Block Attention Module (CBAM, Woo

Table 5.2. Comprehensive results of object detection and instance segmentation on the COCO 2017 validation set

Backbone Network	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	# Params
ResNeXt-50 (Xie et al., 2017)	40.70	62.02	44.49	36.75	58.89	39.03	43.94M
ResNeXt-50 + SE (Hu et al., 2018)	41.04	62.61	44.45	37.13	59.53	39.79	46.47M
ResNeXt-50 + CBAM (Woo et al., 2018)	41.69	63.54	45.17	37.48	60.27	39.71	46.47M
ResNeXt-50 + GC (Cao et al., 2019)	41.66	63.76	45.29	37.58	60.36	39.92	46.48M
ResNeXt-50 + GloRe (Chen et al., 2019)	**42.31**	**64.18**	**46.13**	**37.83**	**60.63**	40.17	49.71M
ResNeXt-50 + VCR (ours)	41.81	63.93	45.67	37.71	60.36	**40.25**	44.18M
ResNeXt-50 + VCR (ours, pixel-level)	42.02	64.15	45.87	37.75	60.62	40.22	44.18M

et al. (2018)), Global Context block (GC, Cao et al. (2019)), and Global Reasoning unit (GloRe, Chen et al. (2019)). All networks are implemented in all residual blocks in the ResNeXt except GloRe, which is partially adopted in the second and third residual stages. In all experiments, we mainly set VCRNet with using (1) attention-based concept samplers with dynamic queries, (2) concept reasoners with dynamic edge weights, and (3) concept modulators with channel-level feature map scaling and shifting.

5.4.1. Image Classification

We conduct experiments on the large-scale image classification task of the ImageNet dataset (Russakovsky et al., 2015). The dataset consists of 1.28M training images and 50K validation images from 1000 different classes. All networks are trained on the training set and evaluated on the validation set by reporting the top-1 and top-5 errors with single center-cropping. Our training setting is explained in detail in Appendix. The overall experimental results are shown in Table 5.1, where all results are reproduced by our training setting for a fair comparison. For evaluation, we always take the final model, which is obtained by exponential moving average (EMA) with the decay value 0.9999. VCRNet consistently outperforms other networks in both ResNeXt-50 and ResNeXt-101 settings. Moreover, it is more compact than the others as it only increases the number of parameters by less than $1\%(\approx 0.95\%)$. In contrast, GloRe (Chen et al., 2019), which also does high-level reasoning, requires more parameters than ours, although it is partially applied in the selected blocks of ResNeXt. In addition, we test the pixel-level concept modulators to reuse the attention maps extracted from the concept samplers to modulate local descriptors at *pixel-level* as GloRe has a pixel-level re-projection mechanism. The modification slightly improves the top-1 performance by using the same number of parameters, but it increases the computational cost (GFLOPs).

5.4.2. Object Detection and Segmentation

We further do some experiments on object detection and instance segmentation on the MSCOCO 2017 dataset (Lin et al., 2014). The MSCOCO dataset contains 115K images

Table 5.3. Results of scene recognition on Places-365 validation set

Model	Error (%)		# of
	Top-1	Top-5	Params
ResNeXt-50 (Xie et al., 2017)	43.49	13.54	23.73M
ResNeXt-50 + SE (Hu et al., 2018)	43.18	13.41	26.26M
ResNeXt-50 + CBAM (Woo et al., 2018)	43.18	13.45	26.26M
ResNeXt-50 + GC (Cao et al., 2019)	43.07	13.34	26.28M
ResNeXt-50 + GloRe (Chen et al., 2019)	42.94	13.22	29.48M
ResNeXt-50 + VCR (ours)	**42.92**	**12.96**	23.96M

Table 5.4. Results of action recognition on Kinetics-400 validation set

Backbone network (Slow-only pathway)	Error (%)		# of
	Top-1	Top-5	Params
ResNeXt-50 (Xie et al., 2017)	26.41	9.43	40.07M
ResNeXt-50 + SE (Hu et al., 2018)	25.06	8.70	42.58M
ResNeXt-50 + CBAM (Woo et al., 2018)	24.87	8.81	42.59M
ResNeXt-50 + GC (Cao et al., 2019)	25.31	9.32	42.60M
ResNeXt-50 + GloRe (Chen et al., 2019)	25.52	9.23	45.81M
ResNeXt-50 + VCR(ours)	**24.73**	**8.39**	40.28M

over 80 categories for training, and 5K images for validation. Our experiments are based on the Detectron2 (Wu et al., 2019). All backbone networks are based on the ResNeXt-50 and pre-trained on the ImageNet dataset by default. We employ and train the Mask R-CNN with FPN (He et al., 2017). We follow the training procedure of the Detectron2 and use the 1× schedule setting. Furthermore, synchronized batch normalization is used instead of freezing all related parameters. For evaluation, we use the standard setting of evaluating object detection and instance segmentation via the standard mean average-precision scores at different boxes and the mask IoUs, respectively. Table 5.2 is the list of results by only varying the backbone network. It shows similar tendencies to the results of ImageNet. However, GloRe (Chen et al., 2019) is showing the best performance, and our pixel-level model comes as the second best. We assume that this result is from two factors. One is the additional capacity, which is relatively larger than other models, used by Glore. The other is that GloRe uses a pixel-level re-projection mechanism that applies the result of reasoning by re-computing all local descriptors. Especially this task requires to do prediction on pixel-level so that it would be beneficial to do so. Therefore, we also make our model to use pixel-level feature modulation. It further improves the performance without requiring additional parameters.

Table 5.5. Ablation study on VCRNet

(a) Concept Sampler

Model (ResNeXt-50)	Top-1 Error (%)	# of Params
pool	20.21	25.17M
static attn	20.18	25.17M
dynamic attn	19.97	25.26M

(b) Concept Reasoner

Model (ResNeXt-50)	Top-1 Error (%)	# of Params
no edge	20.23	25.26M
static edge	20.02	25.28M
dynamic edge	19.97	25.26M

(c) Concept Modulator

Model (ResNeXt-50)	Top-1 Error (%)	# of Params
only scale	20.13	25.22M
only shift	20.05	25.22M
scale + shift	19.97	25.26M

5.4.3. Scene and Action Recognition

Places365 (Zhou et al., 2017) is a dataset labeled with scene semantic categories for the scene recognition task. This task is challenging due to the ambiguity between classes, and several scene classes may share some similar objects causing confusion among classes. We specifically use the Places365-Standard setting that the train set has up to 1.8M images from 365 scene classes, and the validation set has 50 images per class. All networks are trained from random initialization and evaluated on the validation set by following the setting used in our ImageNet experiments. Additionally, we insert Dropout (Srivastava et al., 2014) layers in residual blocks with $p = 0.02$ to avoid some over-fitting. The human action recognition task is another task appropriate to demonstrate how the network can generalize well not only to 2D image data, but also to 3D video data. We use the Kinetics-400 dataset (Kay et al., 2017) including 400 human action categories with 235K training videos and 20K validation videos. We follow the slow-only experiment setting used in Feichtenhofer et al. (2019) that simply takes the ImageNet pre-trained model with a parameter inflating approach (Carreira and Zisserman, 2017). Both tasks are classification tasks similar to the ImageNet image classification, and the results shown in Table 5.3 and 5.4 explain that our approach are generally performing better than other baselines in various visual classification tasks. Moreover, action recognition results suggest that our model can be generally applied to all types of data.

5.4.4. Ablation Study

Concept Sampler. We have proposed different approaches for the concept sampler (pooling-based and attention based samplers). To compare these approaches, we train our proposed networks (ResNeXt-50 + VCR) by having different concept samplers and keeping all other modules fixed. Table 5.5.(a) compares the performance of these approaches on the ImageNet image classification task. The attention-based approach with dynamic queries (dynamic attn) outperforms the others, and we assume that this is due to having more adaptive power. Furthermore, the results interestingly show that our models consistently perform better than other baseline networks except a network with GloRe, which are shown in Table 5.1, regardless of the type of concept sampler.

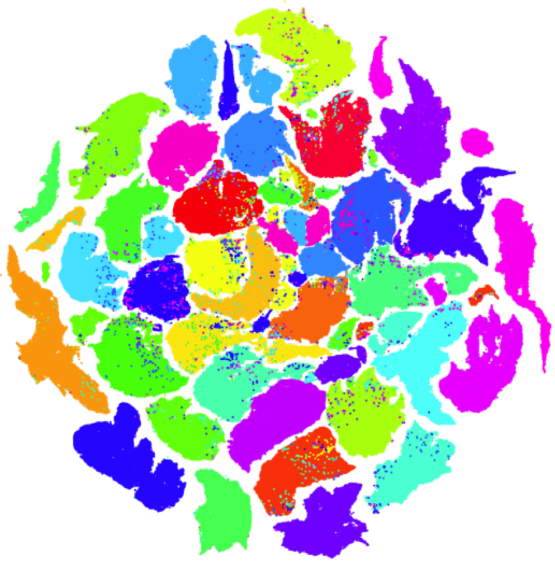


Fig. 5.5. t-SNE plots of visual concept states. $C = 32$ concepts are distinguished by 32 colors.

Concept Reasoner. To investigate the effectiveness of reasoning based on interactions between concepts, we conduct some experiments by modifying the concept reasoner. We first remove the concept interaction term in Equation 5.3.3 and evaluate it to measure the effectiveness of reasoning. Moreover, we also compare the performance between learned static edges and computed dynamic edges. In Table 5.5.(b), the results show that the reasoning module is beneficial in terms of the performance. Notably, it also reveals that using dynamic edges can improve the reasoning and reduce the number of parameters.

Concept Modulator. Our feature modulation consists of both channel-wise scaling and shifting operations. Previous works have used only scaling (gating) (Hu et al., 2018; Woo et al., 2018; Hu et al., 2018) or only shifting (Cao et al., 2019). We compare different settings of the feature modulation as shown in Table 5.5.(c). Using only shifting performs better than using only scaling, and combining both operations is recommended as the best option.

5.4.5. Visualization

We use t-SNE (van der Maaten and Hinton, 2008; Chan et al., 2019) to visualize how visual concept states are represented in the feature space. We collect a set of concept states, which are all extracted from the same concept sampler, by doing inference with the ImageNet validation set. In Figure 5.5, it is shown that the concept states are clustered and separated by concepts.

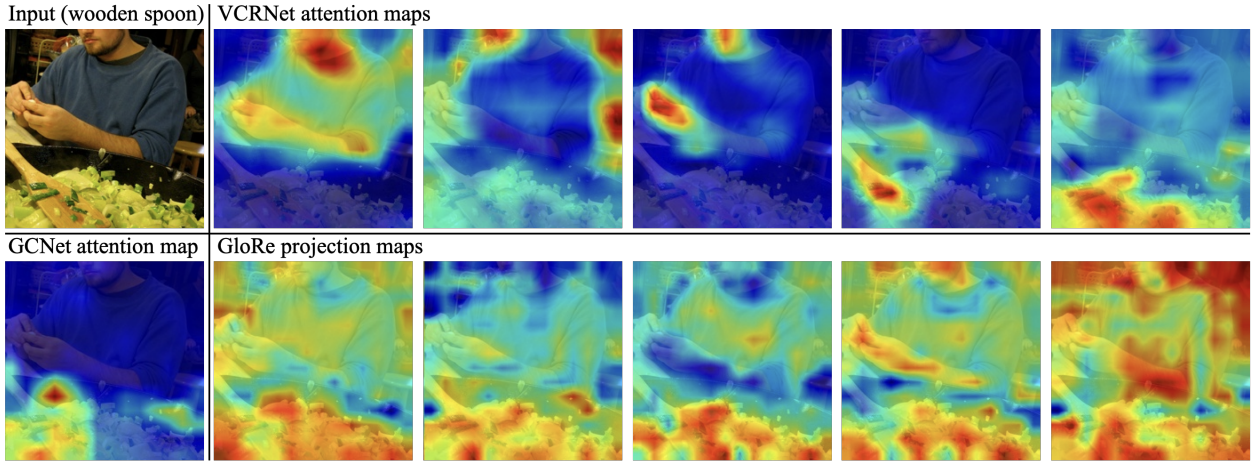


Fig. 5.6. Visualization of attention (projection) maps from VCRNet, GCNet, and GloRe.

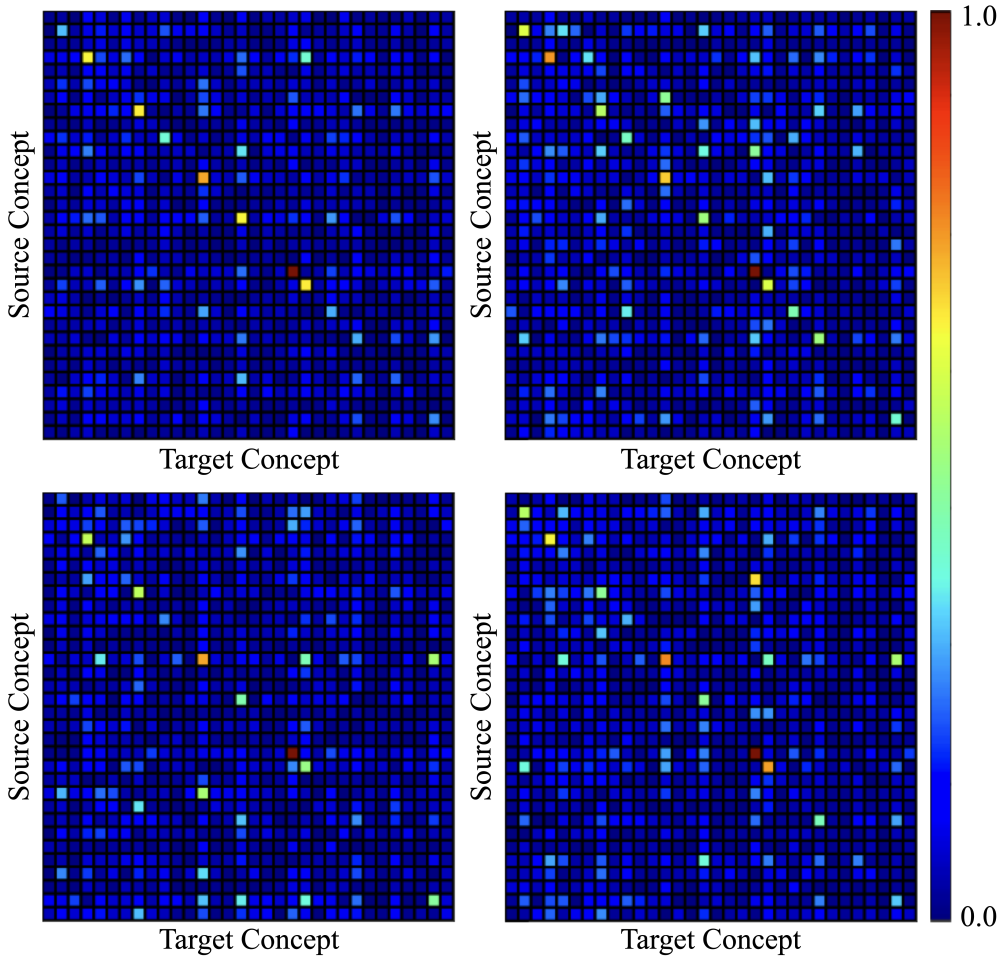


Fig. 5.7. Visualization of interactions between concepts (the adjacency matrix A with source concept c and target concept c' in Equation 5.3.3). Each figure is generated from a different image in the ImageNet validation set.

This result can be further explained by observing the attention maps computed from the concept samplers. Interestingly, they reveal the fact that the concept samplers sparsely attend different regions or objects, and this would result in clustered concept states. This also suggests that our proposed architecture is able to learn distinct concepts without any supervision that explicitly associates branches with certain labeled concepts. We also visualize attention (projection) maps from other networks such as GCNet (Cao et al., 2019) and GloRe (Chen et al., 2019) in Figure 5.6. GCNet only produces a single attention map, and it tends to sparsely attend foreground objects. GloRe similarly computes projection maps as our approach, but the maps are densely attending regions with some redundancies between them.

We furthermore extract edge absolute values (interactions) between concepts from different images and visualize them in Figure 5.7. It shows that each image has different interactions between concepts, and concepts are interacting sparsely so that most edge values are near zero.

5.5. Conclusion

In this work, we propose Visual Concept Reasoning Networks (VCRNet) that efficiently capture the global context by reasoning over high-level visual concepts. Our proposed model precisely fits into a modularized multi-branch architecture by having split-transform-attend-interact-modulate-merge stages. The experimental results shows that it consistently outperforms other baseline models on multiple visual recognition tasks and only increases the number of parameters by less than 1%. We strongly believe research in these approaches will provide notable improvements on more difficult visual recognition tasks in the future. As future works, we are looking forward to remove dense interactions between branches to encourage more specialized concept-wise representation learning and improve the reasoning process. Moreover, we expect to have consistent and specialized visual concepts that are shared and updated over all stages in the network by removing dense interactions between different branches as possible. Explicitly associating branches in the networks with labeled concepts, such as human languages, may also improve the learning of tasks requiring high-level reasoning.

5.6. Appendix

5.6.1. Implementation Details

Our proposed model is based on a modularized multi-branch architecture. Each branch can be described as a separate network that is associated to a single concept as shown in Figure 5.8. The concept reasoner is only shared between branches to make different concepts to interact. At the end, all branches are merged into a single output by summation. By using grouped convolutional operations, the overall architecture can be viewed as a single branch, but internally operates by separate networks in parallel.

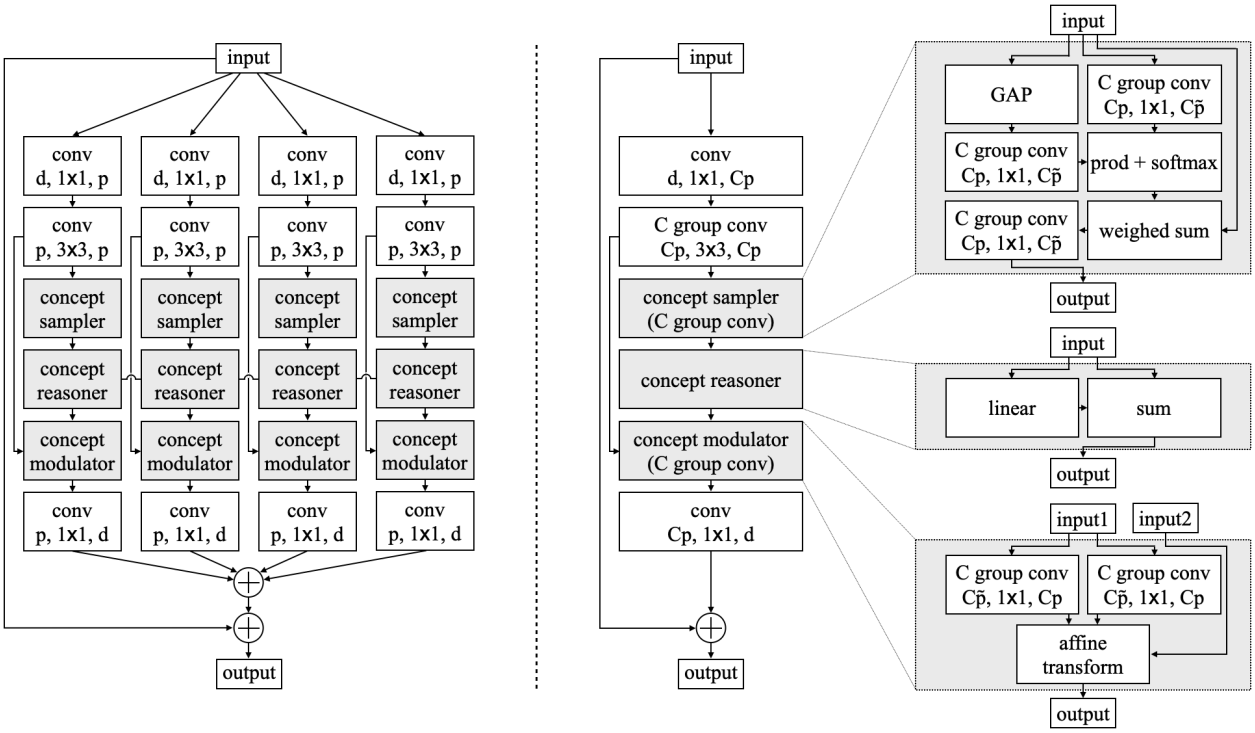


Fig. 5.8. Both architectures are equivalent. **Left:** Each branch is associated to a single concept c, and the concept reasoner is shared between branches. **Right :** By using grouped convolutions, the architecture can be implemented as a single-branch network. Reshaping operations are omitted in this figure.

5.6.2. Image Classification Training Setting

For training the networks, we follow the standard data loader that uses data augmentation with resizing and random flipping and cropping to get images with the size of 224×224. All images are also normalized into $[0, 1]$ by using the RGB-wise mean and standard deviation. We train all networks from scratch on distributed learning system using synchronous SGD optimizer with weight decay 0.0001 and momentum 0.9. The learning rate is warmed up

for the initial 5 epochs that is linearly increased from 0.0 to 1.6 (Loshchilov and Hutter, 2017). This is the setting when the mini-batch size is 4096 with 32 GPUs (128 images per each GPU) for both ResNeXt-50 and ResNeXt-101. Afterward, we use the cosine learning rate scheduler for 115 epochs that decays the learning rate from 1.6 to 0.0001. Moreover, we apply label-smoothing regularization (Szegedy et al., 2016) during training.

5.6.3. Additional Experimental Results

We do some additional experiments to investigate the effectiveness of normalization layers in our proposed modules. In our main implementation, we use batch normalization (BN) layers in concept samplers and reasoners. However, the effectiveness of batch normalization layers highly depends on the mini-batch size or the image size. In image classification experiments, we set the mini-batch size significantly large, which is 4096, and the batch normalization layers seem to properly improve the performance as shown in Table 5.6.

Table 5.6. The effectiveness of BN on image classification

Setting (Model: ResNeXt-50 + VCR)	Error (%)		# of Params	GFLOPs
	Top-1	Top-5		
remove BN (channel-level)	20.15	5.12	25.25M	4.26
remove BN (pixel-level)	20.17	5.06	25.25M	4.29
use BN (channel-level)	19.97	5.03	25.26M	4.26
use BN (pixel-level)	19.94	5.18	25.26M	4.29

We further run experiments on object detection and instance segmentation on the MSCOCO dataset. All experiments are initialized by using the pre-trained models and trained with the mini-batch size 16 that is relatively smaller than the setting of image classification. The results show that the batch normalization layers conversely degrade the performance in some cases (see Table 5.7).

Table 5.7. The effectiveness of BN on object detection and instance segmentation

Setting	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	# Params
remove BN (channel-level)	41.68	63.63	45.45	37.46	60.20	39.94	44.15M
remove BN (pixel-level)	42.11	64.24	45.89	37.80	60.72	40.12	44.15M
use BN (channel-level)	41.81	63.93	45.67	37.71	60.36	40.25	44.18M
use BN (pixel-level)	42.02	64.15	45.87	37.75	60.62	40.22	44.18M

5.6.4. Visualization: Class Activation Mapping

We visualize class activation mapping (CAM, Zhou et al. (2016)) to localize class-specific regions in images. The visualization results are shown in Figure 5.9.

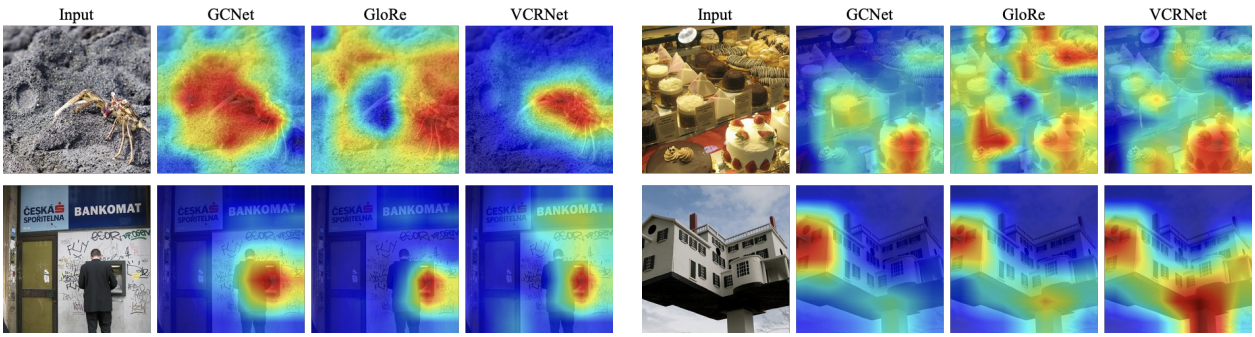


Fig. 5.9. Visualizations of class activation mapping (CAM) from different networks

Chapter 6

Conclusion

In this thesis, we have proposed a series of methods to make deep neural networks adaptive to different circumstances in computationally efficient ways. We first categorized the adaption in deep learning into 3 different levels as explained in Chapter 2. Based on different perspectives, we defined them as (1) task-level, (2) temporal-level, and (3) context-level adaptations and explained each of them in detail by providing some motivations and related works.

The first category is a *task-level adaptation*, where deep neural networks are expected to learn a distribution over tasks to quickly adapt to new tasks. This task-level adaptation can be explained by a meta-learning framework, and in Chapter 3, we proposed a Bayesian meta-learning framework to model the uncertainty induced during fast task adaptation with a few examples. It is based on an optimization-based meta-learning framework, where we combined efficient gradient-based meta-learning with nonparametric variational inference in a principled probabilistic framework. That way, it is able to approximate the complex multimodal task posterior in a flexible way that uses a number of particles. Moreover, as it is broadly applicable, we showed our method in various types of learning tasks including supervised learning, active learning, and reinforcement learning.

In the second category, we explained *temporal-level adaptation*, which is also referred to as temporal abstraction. Specifically, we focused on models that are able to do jumpy future predictions to improve long-term predictions of the future with less computational complexity. In Chapter 4, we introduced a variational approach to learning and inference of temporally hierarchical structure and representation for sequential data. This allows the model to learn to decompose sequential data into some subsequences in an unsupervised manner. Here, time steps included in the same subsequence are assumed to share a high-level (i.e. global) representation, and each time step is further described in detail with a local representation. That way, we proposed to implement the jumpy future predictions by state transitions between subsequences. We also showed the potential of our proposed model

in improving the efficiency of agent learning, which can be characterized by a model-based imagination-augmented agent.

Lastly, we investigated *context-level adaptation*, where deep neural networks can adaptively change or modify some weight variables according to some conditional context. This can be implemented by attention mechanisms and feature-wise transformations, and in Chapter 5, we proposed a convolutional neural network that is based on both methods. It can extract the global context from image data in an adaptive way and also propagate that information to local descriptors (e.g. feature maps) by modulating them. Moreover, these features can be compactly implemented in a modularized multi-branch architecture, where group convolutional operations are mainly used.

Along with our contributions in this thesis, we further expect some future works to improve the adaptation power in deep neural networks as described in the following:

- **Conditional Computation.** This can be achieved by modular approaches, where models are composed of a set of modules. Here, the main goal is to learn not only these modules, but also how to combine them to improve the generalization on unseen concepts or tasks. In other words, this can be referred to as modular meta-learning.
- **Neural Architecture Search (NAS).** Recent works in NAS are mostly focusing on typical problem settings, which do not require any adaptation. By introducing some components with adaptive computations such as attention mechanisms or feature-wise transformations, it would allow the discovered model to better handle unexpected circumstances. Additionally, a meta-learning framework can be adopted to enable task-level adaptation.
- **Ensemble Modeling.** Ensemble modeling can be interpreted as a mixture of experts (MoE), where each base model is an expert with its own specialty. Rather than simply averaging the results from all base models, we can define an auxiliary model that learns to adaptively combine the base models. That way, the final results are differently computed by conditioning on the context extracted from input data.
- **Transformers.** Transformers with multi-head self-attention modules are now broadly used as fundamental building blocks in deep neural networks. Based on its adaptability and flexibility, RNNs and CNNs are replaced by Transformers or hybrid models are used. In this regard, it would be important to reformulate the existing adaption methods to the setting of Transformers.

As we discussed and highlighted at the beginning of this thesis, the ability to quickly learn or adapt to new concepts, tasks, and environments is a critical feature for deep neural networks being deployed in real-world complex applications. For this reason, we strongly believe that our proposed approaches could be an important step towards computationally efficient adaptable deep neural networks.

References

Sungjin Ahn, Anoop Korattikara, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. *arXiv preprint arXiv:1206.6380*, 2012.

Marie Auger-Méthé, Chris Field, Christoffer M Albertsen, Andrew E Derocher, Mark A Lewis, Ian D Jonsen, and Joanna Mills Flemming. State-space models' dirty little secrets: even simple linear gaussian models can have estimation problems. *Scientific reports*, 6: 26677, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Anoop Korattikara Balan, Vivek Rathod, Kevin Murphy, and Max Welling. Bayesian dark knowledge. *CoRR*, abs/1506.04416, 2015.

Matthias Bauer, Mateo Rojas-Carulla, Jakub Bartłomiej Świątkowski, Bernhard Schölkopf, and Richard E Turner. Discriminative k-shot learning using probabilistic models. *arXiv preprint arXiv:1706.00326*, 2017.

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. abs/1511.06297, 2015a.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, 2015b.

Y. Bengio, S. Bengio, and J. Cloutier. Learning a synaptic learning rule. In *IJCNN-91-Seattle International Joint Conference on Neural Networks*, volume ii, pages 969 vol.2–, 1991. doi: 10.1109/IJCNN.1991.155621.

Yoshua Bengio. The consciousness prior. *CoRR*, abs/1709.08568, 2017.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche opérationnelle, 1990.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432, 2013.

John B Biggs. The role of metalearning in study processes. *British journal of educational psychology*, 55(3):185–212, 1985.

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

Jane Bromley, I. Guyon, Yann Lecun, Eduard Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. In J. Cowan and G. Tesauro, editors, *Advances in neural information processing systems (NIPS 1993)*, volume 6. Morgan Kaufmann, 1993.

Randy L Buckner. The role of the hippocampus in prediction and imagination. *Annual review of psychology*, 61:27–48, 2010.

Lars Buesing, Theophane Weber, Sebastien Racaniere, SM Eslami, Danilo Rezende, David P Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, et al. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 1971–1980, 2019.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

João Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4724–4733, 2017.

David M Chan, Roshan Rao, Forrest Huang, and John F Canny. Gpu accelerated t-distributed stochastic neighbor embedding. *Journal of Parallel and Distributed Computing*, 131:1–13, 2019.

Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988, 2015.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Hanjun Dai, Bo Dai, Yan-Ming Zhang, Shuang Li, and Le Song. Recurrent hidden semi-markov model. 2016.

Hal Daumé III. Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 135–142. AUAI Press, 2009.

Harm de Vries, Florian Strub, Jeremie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *ICLR*, 2017.

Vincent Dumoulin, Ethan Perez, Nathan Schucher, Florian Strub, Harm de Vries, Aaron Courville, and Yoshua Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

Harrison Edwards and Amos Storkey. Towards a neural statistician. *arXiv preprint arXiv:1606.02185*, 2016.

Li Fe-Fei et al. A bayesian approach to unsupervised one-shot learning of object categories. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1134–1141. IEEE, 2003.

Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. Slowfast networks for video recognition. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6201–6210, 2019.

Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. *arXiv preprint arXiv:1710.11622*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Bayesian Deep Learning workshop, NIPS*, 2016.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

Marta Garnelo, Dan Rosenbaum, Chris J Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J Rezende, and SM Eslami. Conditional neural processes. *arXiv preprint arXiv:1807.01613*, 2018.

Zoubin Ghahramani and Geoffrey E Hinton. Variational learning for switching state-space models. *Neural computation*, 12(4):831–864, 2000.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Linda S. Gottfredson. Mainstream science on intelligence: An editorial with 52 signatories, history, and bibliography. *Intelligence*, 24(1):13–23, 1997. ISSN 0160-2896. Special Issue Intelligence and Social Policy.

Anirudh Goyal, Alex Lamb, Shagun Sodhani, Jordan Hoffmann, Sergey Levine, Yoshua Bengio, and Bernhard Scholkopf. Recurrent independent mechanisms, 2020. URL `https://openreview.net/forum?id=BylaUTNtPS`.

Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *arXiv preprint arXiv:1801.08930*, 2018.

Karol Gregor, George Papamakarios, Frederic Besse, Lars Buesing, and Theophane Weber. Temporal difference variational auto-encoder. In *International Conference on Learning Representations*, 2019.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Michael Hahn and Frank Keller. Modeling human reading with neural attention. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 85–95, Austin, Texas, November 2016. Association for Computational Linguistics.

Jessica B Hamrick. Analogues of mental simulation and imagination in deep learning. *Current Opinion in Behavioral Sciences*, 29:8 – 16, 2019. ISSN 2352-1546. SI: 29: Artificial Intelligence (2019).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey, 2020.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9401–9411. Curran Associates, Inc., 2018.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017.

Norbert Jankowski, Wlodzislaw Duch, and Krzysztof Grabczewski. *Meta-Learning in Computational Intelligence*. Springer Publishing Company, Incorporated, 2013. ISBN 3642268587.

Dinesh Jayaraman, Frederik Ebert, Alexei Efros, and Sergey Levine. Time-agnostic prediction: Predicting predictable video frames. In *International Conference on Learning Representations*, 2019.

Alexander R. Johansen and Richard Socher. Learning when to skim and when to read. In *The 2nd Workshop on Representation Learning for NLP*. 2017.

Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *CoRR*, abs/1705.06950, 2017.

Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Taesup Kim, Inchul Song, and Yoshua Bengio. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. In Francisco Lacerda, editor, *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2411–2415. ISCA, 2017.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10215–10224. Curran Associates, Inc., 2018.

Thomas Kipf, Yujia Li, Hanjun Dai, Vinícius Flores Zambaldi, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compositional imitation learning: Explaining and executing one task at a time. *CoRR*, abs/1812.01483, 2018.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning workshop*, 2015.

Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Alexandre Lacoste, Thomas Boquet, Negar Rostamzadeh, Boris Oreshki, Wonchang Chung, and David Krueger. Deep prior. *arXiv preprint arXiv:1712.05016*, 2017.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Alex M Lamb, Anirudh Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, 2016.

Barbara Landau, Linda B Smith, and Susan S Jones. The importance of shape in early lexical learning. *Cognitive development*, 3(3):299–321, 1988.

Neil D Lawrence and John C Platt. Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*, page 65. ACM, 2004.

L.M. Le Cam. *Asymptotic methods in statistical decision theory*. Springer, 1986.

Shane Legg and Marcus Hutter. Universal intelligence: A definition of machine intelligence. 2007.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. Rationalizing neural predictions. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Austin, Texas, November 2016. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *European Conference Computer Vision ECCV*, 2014.

Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, pages 2378–2386, 2016.

Yang Liu, Prajit Ramachandran, Qiang Liu, and Jian Peng. Stein variational policy gradient. *arXiv preprint arXiv:1704.02399*, 2017.

Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations, ICLR*, 2017.

Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4898–4906. Curran Associates, Inc., 2016.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417, 2015.

Yajie Miao, Jinyu Li, Yongqiang Wang, Shi-Xiong Zhang, and Yifan Gong. Simplifying long short-term memory acoustic models for fast training and decoding. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 2284–2288. IEEE, 2016.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Meta-learning with temporal convolutions. *arXiv preprint arXiv:1707.03141*, 2017.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

Sinéad L Mullally and Eleanor A Maguire. Memory, imagination, and predicting the future: a common brain mechanism? *The Neuroscientist*, 20(3):220–234, 2014.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11):2, 2011.

Alexander Neitz, Giambattista Parascandolo, Stefan Bauer, and Bernhard Schölkopf. Adaptive skip intervals: Temporal abstraction for recurrent dynamical models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9816–9826. 2018.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms. *ArXiv e-prints*, March 2018.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Karl Pertsch, Oleh Rybkin, Jingyun Yang, Konstantinos G. Derpanis, Joseph Lim, Kostas Daniilidis, and Andrew Jaegle. Keyin: Discovering subgoal structure with keyframe-based video prediction. *CoRR*, abs/1904.05869, 2019.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 68–80. Curran Associates, Inc., 2019.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *In International Conference on Learning Representations (ICLR)*, 2017.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations*, 2018.

Ronald A. Rensink. The dynamic representation of scenes. *Visual Cognition*, 7(1-3):17–42, 2000.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850, 2016a.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016b.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Jürgen Schmidhuber. A 'self-referential' weight matrix. In Stan Gielen and Bert Kappen, editors, *ICANN '93*, pages 446–450, London, 1993. Springer London.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. 2017.

Gordon L Shulman and Maurizio Corbetta. Control of goal-directed and stimulus-driven attention in the brain. *Nature reviews. Neuroscience*, 3(3):201–215, 2002. ISSN 1471-003X.

David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc

Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529: 484–503, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Linda B Smith and Lauren K Slone. A developmental approach to machine learning? *Frontiers in psychology*, 8:2124, 2017.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4080–4090, 2017.

Inchul Song, Junyoung Chung, Taesup Kim, and Yoshua Bengio. Dynamic frame skipping for fast speech recognition in recurrent neural network based acoustic models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4984–4988, 2018.

Aravind Srinivas, Sahil Sharma, and Balaraman Ravindran. Dynamic Frame skip Deep Q Network. *arXiv e-prints*, art. arXiv:1605.05365, 2016.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Kenneth O. Stanley, David B. D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artif. Life*, 15(2):185–212, April 2009. ISSN 1064-5462.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

Joshua Brett Tenenbaum. *A Bayesian framework for concept learning.* PhD thesis, Massachusetts Institute of Technology, 1999.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Vincent Vanhoucke, Matthieu Devin, and Georg Heigold. Multiframe deep neural networks for acoustic modeling. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 7582–7585. IEEE, 2013.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

Xiaolong Wang, Ross B. Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

Xue-Xin Wei, Jason Prentice, and Vijay Balasubramanian. A principle of economy predicts the functional architecture of grid cells. *Elife*, 4:e08362, 2015.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM: convolutional block attention module. In *European Conference Computer Vision ECCV*, 2018.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2048–2057, Lille, France, 07–09 Jul 2015. PMLR.

Flood Sung Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

Shun-Zheng Yu. Hidden semi-markov models. *Artif. Intell.*, 174(2):215–243, February 2010. ISSN 0004-3702.

Songyang Zhang, Xuming He, and Shipeng Yan. LatentGNN: Learning efficient non-local relations for visual recognition. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7374–7383, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Xun Zheng, Manzil Zaheer, Amr Ahmed, Yuan Wang, Eric P Xing, and Alexander J Smola. State space LSTM models with particle MCMC inference. *arXiv preprint arXiv:1711.11179*, 2017.

Bolei Zhou, Aditya. Khosla, Lapedriza. Agata., Aude. Oliva, and Antonio. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.