A SURVEY ON META-LEARNING

Weisen JIANG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Humans can extract knowledge and experience from historical tasks to accelerate learning new tasks from few examples. However, deep networks are data-hungry, and a large number of labeled samples are required for training. In order to reduce the labor-intensive and time-consuming data labeling process, *meta-learning* (or *learning-to-learn*) aims at extracting meta-knowledge from seen tasks to accelerate learning on unseen tasks. This survey provides an overview of meta-learning. We review popular meta-learning algorithms, which are categorized into three groups: (i) optimization-based methods include meta-initialization and meta-regularization; (ii) metric-based methods developed for few-shot classification; and (iii) memory-based methods using a memory buffer or hypernetworks to store meta-knowledge. We propose MetaProx to introduce nonlinearity to meta-regularization by kernelized proximal regularization. We further discuss applications of meta-learning in natural language processing, including prompting and in-context learning. For prompting, we propose a novel algorithm MetaPrompter, consisting of a new verbalizer and meta-learning a prompt pool, to improve the effectiveness and parameter-efficiency of prompt tuning. Lastly, we present several directions for future research.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Motivation

Humans can easily learn new knowledge from a handful of examples and quickly adapt to unseen tasks. They leverage prior knowledge and experience from historical tasks to construct task-required knowledge when facing new tasks. Though deep networks have achieved great success [He et al., 2016, Russakovsky et al., 2015], they are data-hungry, thus, a large number of training samples are required for a new task. This challenge remains a crucial bottleneck in making progress in machine learning algorithms and many research efforts attempt to use knowledge learned from many tasks.

Multitask learning (MTL) [Zhang and Yang, 2021] learns common knowledge from several tasks by minimizing the weighted sum of losses on training data of each task. For the seen tasks, MTL has shown good performance on testing samples [Zhang and Yeung, 2010, Zhong and Kwok, 2012, Chen et al., 2018b, Javaloy and Valera, 2021, Liu et al., 2021a]. However, the learned MTL model is not guarantee to generalize better and fast learning on unseen tasks. Furthermore, MTL also suffers from scalability issue when we have many tasks, leading to a heavy burden on computation and memory. For example, in 5-way 1-shot classification on the *mini-ImageNet* data [Vinyals et al., 2016], there are $\binom{64}{5} \approx 7 \times 10^6$ tasks in total.

Transfer learning [Pan and Yang, 2009, Yang et al., 2020] finetunes a pretrained model on training data of a task to obtain a task-specific model. For example, for a *CIFAR-10* [Krizhevsky and Hinton, 2009] classification task, we first pretrain a network (e.g., *ResNet-18* [He et al., 2016]) on *ImageNet* dataset [Russakovsky et al., 2015], then use the pretrained network as an initialization for gradient descent algorithms for minimizing the training loss on *CIFAR-10*. Transfer learning has been successfully used in image classification [Quattoni et al., 2008, Guo et al., 2019], natural language processing [Blitzer et al., 2006, Fei and Li, 2020], and reinforcement learning [Gamrian and Goldberg, 2019, Zhu et al., 2023]. However, pretraining and finetuning are independent, thus, the pretrained model may not be suitable for finetuning on tasks with limited examples.

Recently, *meta-learning* (or *learning to learn*) [Bengio et al., 1991, Thrun and Pratt, 1998] provides a general framework to extract knowledge from historical tasks to accelerate learning unseen tasks for reducing the labor-intensive and time-consuming process of data. Figure 1.1 illustrates the procedure of meta-learning. Meta-learning algorithms usually operate in two levels. At each iteration, we sample a task; in the inner level, the base learner takes meta-knowledge (e.g., initialization, regularization, feature extractor) and training set to learn a task-specific model; in the outer level, the meta-learner takes the task-specific model and validation set to update the meta-knowledge. As the loss on validation set is a proxy measure to generalization ability, the meta-knowledge is explicitly tuned to learning new tasks with limit training data. At testing, the base learner uses the extracted meta-knowledge to achieve fast learning on unseen tasks. Meta-learning has been receiving increasing attention due to its diverse successful applications in few-shot learning [Wang et al., 2020d, Finn et al., 2017, Vinyals et al., 2016, Snell et al., 2017, Jiang et al., 2021b], hyperparameter optimization [Franceschi et al., 2018], neural architecture search [Liu et al., 2018, Zoph and Le, 2017], and reinforcement learning [Rakelly et al., 2019].

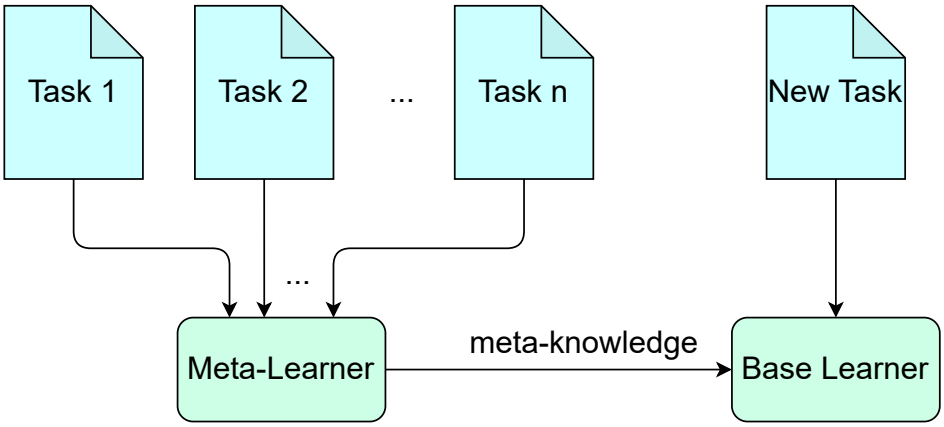


Figure 1.1: Illustration of meta-learning.

1.2 Formulation

In meta-learning, a collection $\mathcal{T}$ of tasks sampled from a task distribution $p(\tau)$ are used to learn a meta-parameter $\boldsymbol{\theta}$ and base learner's parameters $\{\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{T}|}\}$. Each task τ contains a support (also called training) set $\mathcal{S}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_s\}$ and a query (also called validation) set $\mathcal{Q}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_q\}$, where $\mathbf{x} \in \mathbb{R}^d$ are the features and y the labels. For classification task, $\mathcal{Y}_\tau$ is the label set of τ. Let $f(\cdot; \mathbf{w})$ be a model

parameterized by $\mathbf{w}$ and $\mathcal{L}(\mathcal{D}; \mathbf{w}) \equiv \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x},y)\in\mathcal{D}} \ell(f(\mathbf{x}; \mathbf{w}), y)$ be the supervised loss on data set $\mathcal{D}$, where $\ell(\cdot, \cdot)$ is a loss function (e.g., cross-entropy loss for classification, mean squared loss for regression). In each meta-training iteration, a mini-batch $\mathcal{B}$ of tasks are randomly sampled from $\mathcal{T}$. The base learner takes a task τ from $\mathcal{B}$ and the meta-parameter $\boldsymbol{\theta}$ to build the model $f(\cdot; \mathbf{w}_\tau)$. After all tasks in the min-batch are processed by the base learner, the meta-learner updates $\boldsymbol{\theta}$ by minimizing the loss $\sum_{\tau\in\mathcal{B}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau)$ w.r.t. $\boldsymbol{\theta}$, and the iteration repeats. During meta-testing, given an unseen task $\tau' \sim p(\tau)$, a model $f(\cdot; \mathbf{w}_{\tau'})$ is similarly learned from $\mathcal{S}_{\tau'}$ and $\boldsymbol{\theta}$. Finally, its performance is evaluated on $\mathcal{Q}_{\tau'}$.

1.3 Category of Meta-Learning Algorithms

Popular meta-learning algorithms can be categorized into three groups: optimization-based, metric-based, and memory-based.

For *optimization-based* methods, the base learner performs gradient descent to minimize a task-specific loss, where meta-parameters in the optimization algorithm are meta-knowledge learned by the meta-learner. The meta-parameters can be initialization, regularization, learning rate, preconditioning matrix, sample weights. Meta-initialization and meta-regularization are two representative methods. MAML [Finn et al., 2017] is a pioneering meta-initialization method: the base learner takes a meta-initialization and performs several gradient updates on the training set to obtain a task-specific model, while the meta-learner updates the meta-initialization by performing a gradient update on the validation set using the obtained task-specific model. As MAML is very general, many variants are proposed to improve its effectiveness (e.g., Meta-SGD[Li et al., 2017], T-Nets [Lee and Choi, 2018], Meta-Curvature [Park and Oliva, 2019], WarpGrad [Flennerhag et al., 2020]) and efficiency (e.g., FO-MAML [Finn et al., 2017], Reptile [Nichol et al., 2018]). The bilevel structure is complex and challenging to understand MAML from a theoretical view. Recently, many efforts have been devoted to study its convergence [Finn et al., 2019, Fallah et al., 2020, Wang et al., 2020a,b, Ji et al., 2020b,a] and generalization [Maurer and Jaakkola, 2005, Zhou et al., 2021, Denevi et al., 2019, 2020, Pentina and Lampert, 2014, 2015, Rothfuss et al., 2021a, Farid and Majumdar, 2021, Rajendran et al., 2020, Chen et al., 2021, Titsias et al., 2021]. In meta-regularization, the base learner minimizes a regularized loss on training data to obtain a task-specific model, while the meta-learner updates the learnable regularizer by minimizing the validation loss using the obtained model. Typical algorithms

include iMAML [Rajeswaran et al., 2019], Meta-MinibatchProx [Zhou et al., 2019], and regularization for linear models [Denevi et al., 2018, 2019, 2020]. As real-world tasks are usually complex, a single meta-initialization or meta-regularization may be insufficient for fast adaptation to all tasks. To deal with this issue, several structured meta-learning methods [Jerfel et al., 2019, Kong et al., 2020, Tripuraneni et al., 2021, Zhou et al., 2021] have been proposed.

Metric-based methods aim at meta-learning a good feature extractor to map inputs to an embedding space, where the base learner trains a simple but effective classifier with few samples. Convolutional neural network (e.g., VGG [Simonyan and Zisserman, 2015], ResNet [He et al., 2016]) and Vision Transformers [Dosovitskiy et al., 2021] are widely used in feature extraction. For classifier, we can use a nearest neighbor classifier (e.g., ProtoNet [Snell et al., 2017]), linear models (e.g., R2D2 [Bertinetto et al., 2018]), and kernal classifier (e.g., MetaOptNet [Lee et al., 2019b])).

Memory-based methods incorporate a memory structure to store meta-knowledge for accelerating learning future tasks. For example, an external memory (table or key-value pairs) is used in [Santoro et al., 2016, Ramalho and Garnelo, 2019, Munkhdalai et al., 2018, Babu et al., 2021]; hypernetworks (also called meta-networks) are external networks contain knowledge of how to generate task parameters [Munkhdalai and Yu, 2017, Rusu et al., 2019, Sitzmann et al., 2020, Ehret et al., 2021, Navon et al., 2021].

1.4 Organization

The rest of this survey is organized as follows and shown in Figure 1.2.

Chapter 2 provides an review of meta-learning, including *optimization-based*, *metric-based*, and *memory-based* methods. *Optimization-based* methods are general and can be used for classification, regression, and reinforcement learning. Meta-initialization and meta-regularization are two representative approaches. *Metric-based* methods are mainly designed for classification based on metric learning, i.e., they aim to learn a good feature extractor such that a simple classifier (e.g., linear classifier, kernel classifier, SVM) can be used for classifying samples. *Memory-based* meta-learning uses external memory to store meta-knowledge extracted from historical tasks for facilitating new tasks, where the memory is maintained by the meta-learner.

In Chapter 3, we propose a novel algorithm MetaProx to introduce nonlinearity to meta-

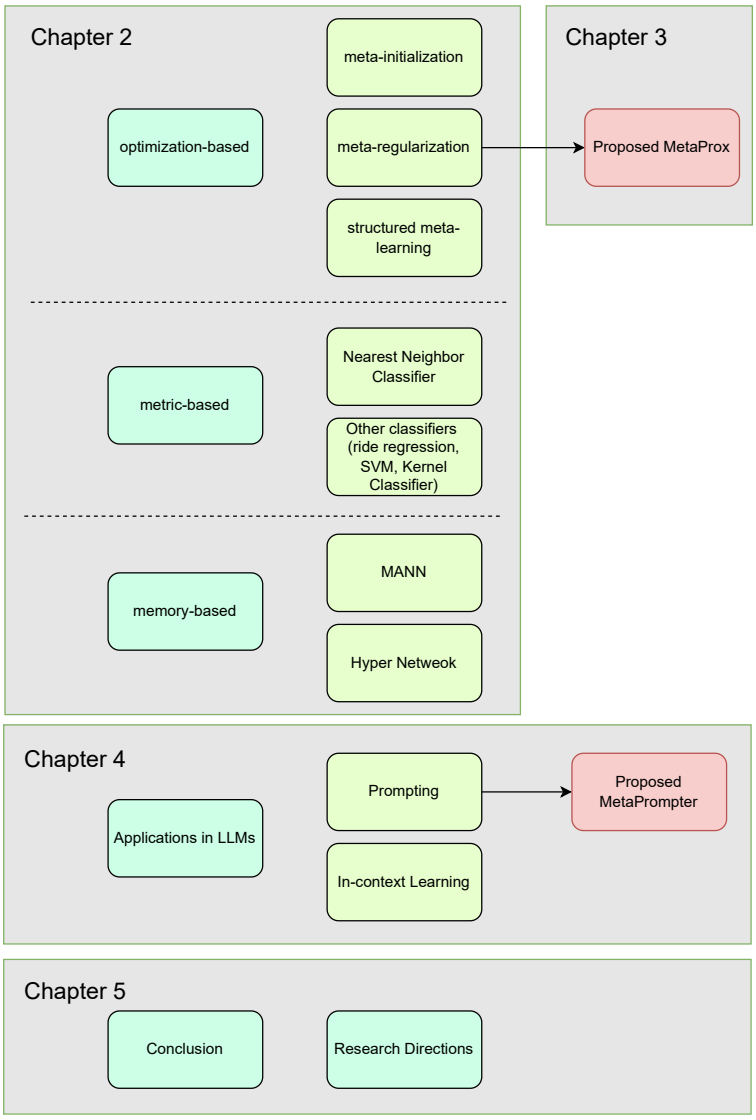


Figure 1.2: Outline.

regularization by kernelized proximal regularization.

Chapter 4 studies the application of meta-learning in large language models, including prompt tuning and in-context learning. We propose a novel algorithm MetaPrompter, consisting of a new verbalizer and meta-learning a prompt pool, to improve the effectiveness and parameter-efficiency of prompt tuning. We review exiting works combine meta-learning and in-context learning, then discuss the challenges of such combination.

Chapter 5 concludes the survey and proposes several possible questions for future research.

CHAPTER 2

A BRIEF REVIEW

In this chapter, we give an overview on existing meta-learning methods. We first review optimization-based methods, including meta-initialization, meta-regularization, and structured meta-learning methods. Next, we introduce metric-based methods, which consists of a feature extractor and a simple classifier, such as nearest neighbor classifier, linear classifier, SVM classifier. Last, we review memory-based meta-learning, including two representative methods Memory Augmented Neural Network and Hyper Network.

2.1 Optimization-Based Meta-Learning

Deep networks are usually trained from scratch by gradient descent algorithms on massive data. However, these algorithms are not designed to deal with limited examples or few gradient updates. Optimization-based meta-learning aims to adjust the optimization algorithms such that a good task model can be obtained with limited examples after few gradient updates. *Meta-initialization* and *meta-regularization* are two representative approaches.

2.1.1 Meta-Initialization

In conventional machine learning task, models are usually learned from a random initialization using a large amount of training data by gradient-based algorithms. However, when the training data is limited (e.g., few-shot classification), the initialization drastically affects generalization ability of the learned model. *Meta-initialization* methods aim to learn a good initialization from historical tasks such that a good model for an unseen task can be fine-tuned with limited samples by several gradient updates.

2.1.1.1 Model-Agnostic Meta-Learning (MAML)

A pioneer work for learning an initialization is Model-Agnostic Meta-Learning (MAML) proposed by Finn et al. [2017] . An illustration is shown in Figure 2.1.

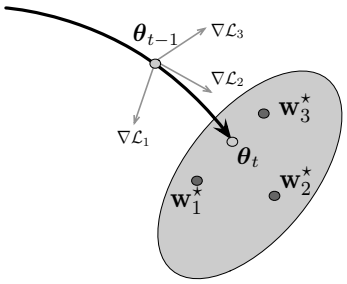


Figure 2.1: MAML.

Meta-Training. The procedure and its efficient variants are shown in Algorithm 1. MAML operates in two optimization levels (let $\mathcal{B}_t$ be a mini-batch of tasks at iteration t):

- *Inner Level* (steps 4-9): For each task $\tau \in \mathcal{B}_t$, the base learner takes its support set $\mathcal{S}_\tau$ and the meta-initialization $\boldsymbol{\theta}_{t-1}$ to build a task-specific model $\mathbf{w}_\tau^{(J)}$ by performing J gradient descent steps with initialization $\mathbf{w}_\tau^{(0)} = \boldsymbol{\theta}_{t-1}$ and step size $\alpha > 0$:

$$\mathbf{w}_\tau^{(j)} = \mathbf{w}_\tau^{(j-1)} - \alpha \nabla_{\mathbf{w}_\tau^{(j-1)}} \mathcal{L}\left(\mathcal{S}_\tau; \mathbf{w}_\tau^{(j-1)}\right),\ j = 1, \ldots, J. \tag{2.1}$$

  Note that $\mathbf{w}_\tau^{(J)}$ is a function of $\boldsymbol{\theta}_t$.

- *Outer Level* (steps 11-17): For each task $\tau \in \mathcal{B}_t$, the meta-learner takes its query set $\mathcal{Q}_\tau$ and the task-specific model $\mathbf{w}_\tau^{(J)}$ to compute the gradient of $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$ w.r.t. $\boldsymbol{\theta}_{t-1}$ (step 12), which is called meta-gradient. The meta-initialization is updated as (step 16):

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}\left(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}\right), \tag{2.2}$$

  where $\eta_t > 0$ is step size.

By the chain rule, the meta-gradient $\nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}) = \nabla_{\boldsymbol{\theta}_{t-1}} \mathbf{w}_\tau^{(J)} \nabla_{\mathbf{w}_\tau^{(J)}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$ (particularly, $\nabla_{\boldsymbol{\theta}_{t-1}} \mathbf{w}_\tau^{(J)}$) requires back-propagating through the entire inner optimization path, which incurs huge computations, especially for large models and a large J. To reduce computational cost, FO-MAML Finn et al. [2017] discard the second-order derivative and use the first-order approximation $\nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}) \approx \nabla_{\mathbf{w}_\tau^{(J)}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$ (step 13). Then,

Algorithm 1 MAML [Finn et al., 2017], FO-MAML [Finn et al., 2017], and Reptile [Nichol et al., 2018].

Require: stepsize η_t and α, meta-batch size b; number of inner gradient updates J.

1: **for** $t = 1, 2, \cdots, T$ **do**
2: sample a batch $\mathcal{B}_t$ of tasks from $\mathcal{T}$;
3: base learner:
4: **for** $\tau \in \mathcal{B}_t$ **do**
5: $\mathbf{w}_\tau^{(0)} = \boldsymbol{\theta}_{t-1}$;
6: **for** $j = 1, \cdots, J$ **do**
7: $\mathbf{w}_\tau^{(j)} = \mathbf{w}_\tau^{(j-1)} - \alpha \nabla_{\mathbf{w}_\tau^{(j-1)}} \mathcal{L}(\mathcal{S}_\tau; \mathbf{w}_\tau^{(j-1)})$;
8: **end for**
9: **end for**
10: meta-learner:
11: **for** $\tau \in \mathcal{B}_t$ **do**
12: if MAML: $\mathbf{g}_\tau = \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$;
13: if FO-MAML: $\mathbf{g}_\tau = \nabla_{\mathbf{w}_\tau^{(J)}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$;
14: if Reptile: $\mathbf{g}_\tau = \boldsymbol{\theta}_{t-1} - \mathbf{w}_\tau^{(J)}$;
15: **end for**
16: update meta-initialization: $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
17: **end for**
18: **return** $\boldsymbol{\theta}_T$.

the update (2.2) is approximated by

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \nabla_{\mathbf{w}_\tau^{(J)}} \mathcal{L}\left(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}\right). \tag{2.3}$$

To reduce computational cost, alternatively, Reptile [Nichol et al., 2018] approximates meta-gradient by the update direction from task model parameters to meta-initialization (step 14):

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \left(\boldsymbol{\theta}_{t-1} - \mathbf{w}_\tau^{(J)}\right). \tag{2.4}$$

As the update rules in (2.3) and (2.4) do not need to compute the derivative of $\mathbf{w}_\tau^{(J)}$ w.r.t. $\boldsymbol{\theta}_{t-1}$, it is flexible to choose the optimization algorithms used in the base learner, e.g., second-order algorithms (e.g., Gauss-Newton method [Wedderburn, 1974], L-BFGS [**?**]) or even non-differentiable methods like line search [**?**]. Besides efficiency, empirical results in [Finn et al., 2017, Nichol et al., 2018] demonstrate that the approximations in FO-MAML and Reptile do not hurt the performance of MAML.

Meta-Testing. Given an unseen task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$, the base learner takes $\mathcal{S}_{\tau'}$ and $\boldsymbol{\theta}_T$ to build a task-specific model $\mathbf{w}_{\tau'}^{(J)}$ as in steps 4-9 of Algorithm 1. $\mathbf{w}_{\tau'}^{(J)}$ is then used to predict the query sample $\mathcal{Q}_{\tau'}$ and evaluate performance.

2.1.1.2 Variants of MAML

Using preconditioning matrix. For deep networks $f(\cdot; \mathbf{w})$, the loss landscape is highly nonconvex and the update rule of base learner (2.1) based on first-order gradient may not be effective enough to capture the full geometry of loss landscape. To address this issue, several variants based on preconditioning gradients are proposed, e.g., MetaSGD [Li et al., 2017], Meta-Curvature [Park and Oliva, 2019], T-Nets [Lee and Choi, 2018], and WarpGrad [Flennerhag et al., 2020].

Instead of using the same learning rate for all parameters in the base learner, MetaSGD proposes to learn a learning rate for each parameter:

$$\mathbf{w}_{\tau}^{(j)} = \mathbf{w}_{\tau}^{(j-1)} - \alpha \, \mathsf{diag}(\boldsymbol{\phi}) \nabla_{\mathbf{w}_{\tau}^{(j-1)}} \mathcal{L}(\mathcal{S}_{\tau}; \mathbf{w}_{\tau}^{(j-1)}), \tag{2.5}$$

where $\boldsymbol{\phi}$ is meta-parameter. Meta-Curvature introduces a function $\mathbf{P}(\boldsymbol{\theta}_t; \boldsymbol{\phi})$ to generate block-diagonal preconditioning matrix for the base learner:

$$\mathbf{w}_{\tau}^{(j)} = \mathbf{w}_{\tau}^{(j-1)} - \alpha \mathbf{P}(\boldsymbol{\theta}_t; \boldsymbol{\phi}) \nabla_{\mathbf{w}_{\tau}^{(j-1)}} \mathcal{L}(\mathcal{S}_{\tau}; \mathbf{w}_{\tau}^{(j-1)}). \tag{2.6}$$

T-Nets inserts additional *linear* layers into the network to warp the gradients during back-propagation in the base-learner, while WrapGrad inserts *nonlinear* layers, which is more general. Parameters in the inserted layers are meta-parameters learned by the meta-learner. Hiller et al. [2022] introduces a constraint on condition number in the inner objective to learn a well-conditioned parameter space for the base learner.

Updating partial networks in the base learner. Unlike MAML that updates all network parameters in the base learner, recent work [Raghu et al., 2020, Oh et al., 2021, Shen et al., 2021] reveals that updating only part of network can improve both efficiency and effectiveness. Raghu et al. [2020] find that the success of MAML heavily rely on the high quality representations and propose ANIL (Almost No Inner Loop) algorithm, which only updates the last layer (i.e., classifier) and freezes the feature extractor in the inner level. In contrast, Oh et al. [2021] argue that representation change is necessary for unseen tasks (particularly for rare tasks), thus propose BOIL (Body Only update in Inner Loop) to update feature extractor in

the inner level while keep the last layer (i.e., classifier) frozen. CAVIA [Zintgraf et al., 2019] partitions the model parameters into context parameters and shared parameters, where only the context parameters are updated in the base learner. P-Transfer [Shen et al., 2021] proposes a layer-wise search method to determine which layers should be fine-tuned or kept frozen.

2.1.1.3 Meta-Learning by Target Models

MAML evaluates the task-specific model $\mathbf{w}_\tau^{(J)}$ on the query set $\mathcal{Q}_\tau$ to obtain the update direction for meta-initialization, which is also called $\mathcal{S}/\mathcal{Q}$ protocol. An alternative protocol $\mathcal{S}/\mathcal{T}$ is recently proposed in [Yoon et al., 2018, Lu et al., 2021, Flennerhag et al., 2022, Tack et al., 2022]. In $\mathcal{S}/\mathcal{T}$ protocol, the task model $\mathbf{w}_\tau^{(J)}$ is compared to the ground-truth $\mathbf{w}_\tau^\star$ in the parameter space or output space. As $\mathbf{w}_\tau^\star$ is unavailable in general, it is approximated by fine-tuning $\mathbf{w}_\tau^{(J)}$ on the query set for further L steps, i.e., $\mathbf{w}_\tau^{(J+L)}$. For parameter space, the update rule is

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_\tau} \nabla_{\boldsymbol{\theta}_{t-1}} \|\mathbf{w}_\tau^{(J)} - \mathsf{detach}(\mathbf{w}_\tau^{(J+L)})\|^2; \tag{2.7}$$

while for output space, the update rule is:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \frac{1}{|\mathcal{Q}_\tau|} \sum_{(\mathbf{x},y) \in \mathcal{Q}_\tau} \nabla_{\boldsymbol{\theta}_{t-1}} \mathsf{KL}\left(\mathsf{detach}(f(\mathbf{x}; \mathbf{w}_\tau^{(J+L)})), f(\mathbf{x}; \mathbf{w}_\tau^{(J)})\right), \tag{2.8}$$

where $\mathsf{KL}(\cdot,\cdot)$ is the Kullback-Leibler divergence [Kullback and Leibler, 1951].

Compared with $\mathcal{S}/\mathcal{Q}$ protocol, $\mathcal{S}/\mathcal{T}$ protocol has two advantages: (i) The meta-objective is not constrained to the same type of geometry in the base learner, which is particularly useful when the inner loss is ill-conditioning. By optimizing the meta-parameters in a well-behaved space (output space or parameter space), the learned meta-initialization can improve model generalization [Yoon et al., 2018, Lu et al., 2021, Flennerhag et al., 2020, Tack et al., 2022]. (ii) As J is usually small in MAML-like algorithms (e.g., $J = 1$ or 5 in MAML), the additional L steps can prevent meta-learning algorithms from a short-horizon bias.

2.1.1.4 Theoretical Analysis

Convergence. Due to the bilevel formulation, analyzing convergence for MAML-like algorithms is challenging. [Finn et al., 2019] study the convergence of MAML when the loss is convex, while [Fallah et al., 2020] establish convergence for one-step MAML and FO-MAML in a general non-convex case. For multi-step MAML, [Wang et al., 2020a,b] study its global convergence for over-parameterized networks, while [Ji et al., 2020b] analyze its local convergence. Besides, [Ji et al., 2020a] provide a convergence analysis for the multi-step ANIL algorithm.

Generalization of meta-learning is first analyzed in [Maurer and Jaakkola, 2005] by uniform algorithmic stability, and *stability-based bounds* are derived [Maurer and Jaakkola, 2005, Farid and Majumdar, 2021, Zhou et al., 2021, Denevi et al., 2019, 2020]. However, this requires stability assumption to be satisfied in both base and meta learner. *PAC-Bayes bound* [Pentina and Lampert, 2014, 2015, Rothfuss et al., 2021a, Farid and Majumdar, 2021] and *information-theoretic analysis* [Rajendran et al., 2020, Chen et al., 2021, Titsias et al., 2021] are two alternative tools to study the generalization error in meta-learning.

2.1.2 Meta-Regularization

2.1.2.1 iMAML

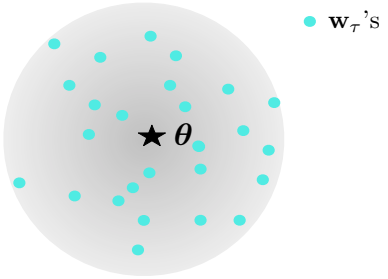


Figure 2.2: iMAML.

Meta-regularization [Denevi et al., 2018, 2019, Rajeswaran et al., 2019, Zhou et al., 2019, Denevi et al., 2020, Jiang et al., 2021a] is another optimization-based meta-learning method, which assumes that task models are close to a prior model. The base learner obtains task model by solving a regularized empirical risk minimization, while the prior model $\boldsymbol{\theta}$ in the regularization is learned by the meta-learner. Specifically, at iteration t, the base learner

Algorithm 2 iMAML [Rajeswaran et al., 2019].

Require: stepsize η_t and α, batch size b;
for $t = 1, 2, \cdots, T$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $\mathcal{T}$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\hat{\mathbf{w}}_\tau = \arg\min_{\mathbf{w}_\tau} \mathcal{L}(\mathcal{S}_\tau; \mathbf{w}_\tau) + \frac{\lambda}{2}\|\mathbf{w}_\tau - \boldsymbol{\theta}_{t-1}\|^2$;
  end for
  meta-learner:
  for $\tau \in \mathcal{B}_t$ **do**
    compute meta-gradient $\mathbf{g}_\tau = \left(\frac{1}{\lambda}\nabla^2_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) + \mathbf{I}\right)^{-1} \nabla_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau)$;
  end for
  $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{b}\sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for
return $\boldsymbol{\theta}_T$.

takes $\mathcal{S}_\tau$ and $\boldsymbol{\theta}_{t-1}$ to build task model $\hat{\mathbf{w}}_\tau$ by solving the regularized minimization:

$$\hat{\mathbf{w}}_\tau = \arg\min_{\mathbf{w}_\tau} \mathcal{L}(\mathcal{S}_\tau; \mathbf{w}_\tau) + \frac{\lambda}{2}\|\mathbf{w}_\tau - \boldsymbol{\theta}_{t-1}\|^2, \tag{2.9}$$

where $\lambda > 0$ is hyperparameter. Similar to meta-initialization, $\hat{\mathbf{w}}_\tau$ is a function of $\boldsymbol{\theta}_{t-1}$. As in MAML, meta-regularization methods (iMAML [Rajeswaran et al., 2019] and Denevi et al. [2018]) update the regularizer by the meta-learner by minimizing the loss

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|}\sum_{\tau \in \mathcal{B}_t} \nabla_{\boldsymbol{\theta}_{t-1}}\mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau). \tag{2.10}$$

By the chain rule, it follows that $\nabla_{\boldsymbol{\theta}_{t-1}}\mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau) = \nabla_{\boldsymbol{\theta}_{t-1}}\hat{\mathbf{w}}_\tau \nabla_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau)$. The second term $\nabla_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau)$ can be computed directly by auto-differentiation, but the first term $\nabla_{\boldsymbol{\theta}_{t-1}}\hat{\mathbf{w}}_\tau$ is more difficult as it is an implicit derivative. As $\hat{\mathbf{w}}_\tau$ is a minimizer to problem (2.9), it follows from the first-order optimal condition that

$$\nabla_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) + \lambda(\hat{\mathbf{w}}_\tau - \boldsymbol{\theta}_{t-1}) = \mathbf{0}. \tag{2.11}$$

By the implicit function theorem [Rudin, 1976], it follows that $\nabla^2_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau)\nabla_{\boldsymbol{\theta}_{t-1}}\hat{\mathbf{w}}_\tau + \lambda(\nabla_{\boldsymbol{\theta}_{t-1}}\hat{\mathbf{w}}_\tau - \mathbf{I}) = \mathbf{0}$, thus, the implicit derivative is

$$\nabla_{\boldsymbol{\theta}_{t-1}}\hat{\mathbf{w}}_\tau = \left(\frac{1}{\lambda}\nabla^2_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) + \mathbf{I}\right)^{-1}. \tag{2.12}$$

Compared with meta-initialization (e.g., MAML), iMAML has two advantages: (i) Computing the meta-gradients does not require to back-propagate through the inner optimization

path. Hence, we can use many gradient updates in the base learner to obtain an accurate solution, which can address the short-horizon bias issue in one-step MAML. (ii) The meta-gradient does not depends on the optimization path, thus, the chosen optimizer in the base learner is flexible, e.g., AdaDelta [Zeiler, 2012], Adam [Kingma and Ba, 2015], AdamW [Loshchilov and Hutter, 2019].

Though meta-gradients in step 9 can be obtained by conjugate gradient descent without computing the matrix inverse explicitly, iMAML still incurs huge computations in obtaining a good solution $\hat{\mathbf{w}}_\tau$ and performing conjugate gradient descent for computing meta-gradients. Several work [Denevi et al., 2018, 2019, Zhou et al., 2019, Jiang et al., 2021a] are proposed to reduce computational cost. For example, Meta-MinibatchProx [Zhou et al., 2019] and [Denevi et al., 2019] simply replace the loss in the meta-learner by $\mathbf{g}_\tau = \lambda(\boldsymbol{\theta}_{t-1} - \hat{\mathbf{w}}_\tau)$ (the loss in the base learner), and obtain meta-gradients $\mathbf{g}_\tau = \lambda(\boldsymbol{\theta}_{t-1} - \hat{\mathbf{w}}_\tau)$. Despite efficiency, overusing the support set in the meta-learner is not standard in meta-learning due to overfitting consideration. Using the query set in the meta-learner is more preferable [Finn et al., 2017, Rajeswaran et al., 2019]. In linear setting with the squared loss, both $\hat{\mathbf{w}}_\tau$ and meta-gradient have closed-form expressions, thus, can be calculated efficiently [Denevi et al., 2018]. However, linear models are *not expressive* enough for complex tasks.

2.1.3 Structured Meta-Learning

In real-world application, tasks are usually complex. Figure 2.3 shows some samples from a popular dataset *MetaDataset-BTAF* [Yao et al., 2019, 2020]. Each row shows one type of tasks (*bird*/*texture*/*aircraft*/*fungi* classification). In such complex environment, task model parameters are diverse and a single meta-model may be insufficient to capture all the meta-knowledge. To tackle this issue, a variety of methods have been proposed to learn

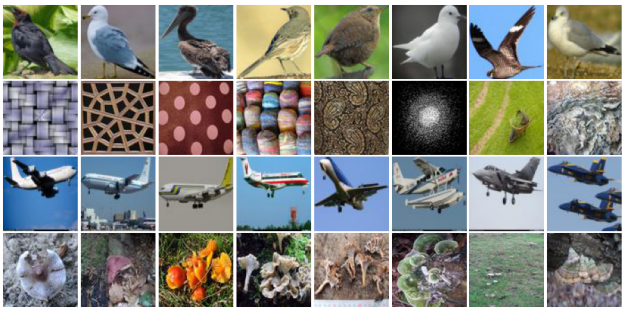
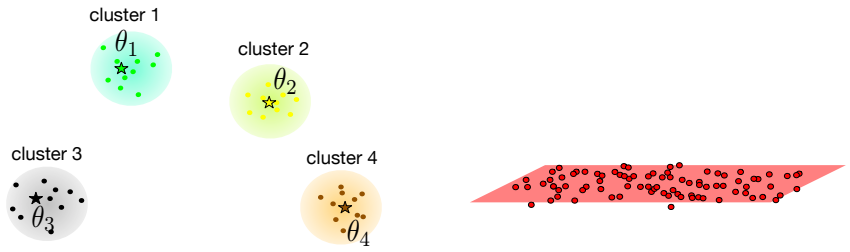


Figure 2.3: Some random images from *MetaDataset-BTAF* (Top to bottom: *Bird*, *Texture*, *Aircraft*, and *Fungi*).



(a) Centroid-based cluster. (b) Subspace.

Figure 2.4: Some possible structure.

structured meta-knowledge by exploring the task structure [Jerfel et al., 2019, Yao et al., 2019, 2020, Wang et al., 2020c, Zhou et al., 2021, Kong et al., 2020, Tripuraneni et al., 2021]. Some structures are shown in Figure 2.4.

One approach is to structure task model parameters into K centroid-based clusters, as shown in Figure 2.4(a). For example, Jerfel et al. [2019] formulate the task distribution as a mixture of hierarchical Bayesian models, and update the components (i.e., initializations) using an Expectation Maximization procedure. Another approach TSA-MAML [Zhou et al., 2021] uses a two-stage method. It first trains task models using vanilla MAML. Task-specific model parameters are then grouped into clusters by k-means clustering, and cluster centroids form group-specific initializations.

Alternatively, task model parameters can be formulated into a subspace (Figure 2.4(b)). In the linear regression setting where task model parameters are sampled from a low-dimensional subspace, recent work [Kong et al., 2020, Tripuraneni et al., 2021] use a moment-based estimator to recover the subspace using the property that the column space of the sample covariance matrix recovers the underlying subspace. However, for nonlinear models such as deep networks, this nice property no longer holds and the moment-based methods cannot be generalized.

Other structures are also possible. For example, HSML [Yao et al., 2019] formulates meta-knowledge into a hierarchical structure, which is learned by three stages: task representation learning is for extracting task embedding, hierarchical task clustering is for locating task's cluster, and knowledge adaptation is for constructing cluster-specific initialization. ARML [Yao et al., 2020] uses a graph structure to organize meta-knowledge, where each vertex represents one type of meta-knowledge. Given a task, ARML constructs a prototype-based relational graph to represents task-specific knowledge, which is used to

query the meta-knowledge graph for obtaining the most relevant knowledge. The obtained knowledge is fused into the meta-initialization to obtain a task-dependent initialization.

2.2 Metric-Based Meta-Learning

Metric-based methods are mainly developed for few-shot classification tasks. The meta-learner learns a good feature extractor to map inputs to an embedding space, while the base learner trains a classifier on the embedding space with few examples. Algorithms differ in which type of classifier is used: nearest neighbor classifier based on L_2 distance in ProtoNet [Snell et al., 2017, Allen et al., 2019], ridge regressor in R2D2 [Bertinetto et al., 2018], convex learner (e.g., MetaOptNet [Lee et al., 2019b]), kernel learner [Zhen et al., 2020, Xu et al., 2020, Patacchiola et al., 2020]. An illustration of metric-based methods is shown in Figure 2.5.

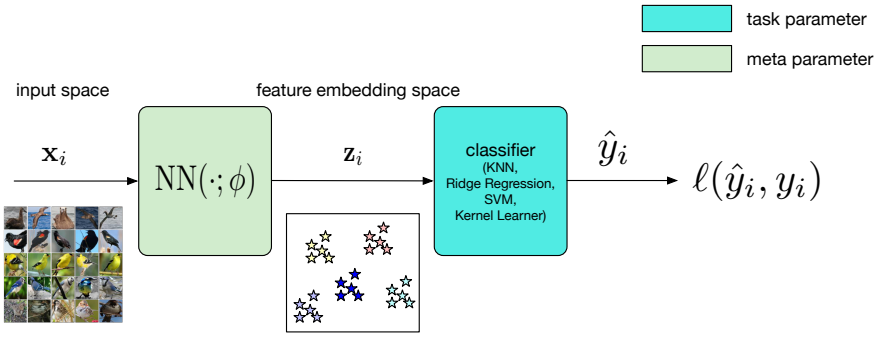


Figure 2.5: Metric-based meta-learning methods.

2.2.1 Nearest Neighbor Classifier

ProtoNet [Snell et al., 2017] is a representative metric-based method. The procedure is shown in Algorithm 3. It employs a neural network to map inputs to an embedding space, then represents each class's prototype by the mean embeddings of the corresponding samples in the support set. For each sample in the query set, its label prediction is based on the similarity between feature embedding and label embeddings. Let $\mathcal{S}_\tau^{(y)}$ be the subset of samples in $\mathcal{S}_\tau$ with label y, and $\mathsf{NN}(\cdot;\boldsymbol{\phi})$ be a feature extractor parameterized by $\boldsymbol{\phi}$. Specifically, the base learner builds class prototype as follows

$$\mathbf{p}_y = \frac{1}{|\mathcal{S}_\tau^{(y)}|} \sum_{(\mathbf{x}_i, y) \in \mathcal{S}_\tau^{(y)}} \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1}), \tag{2.13}$$

Algorithm 3 ProtoNet [Snell et al., 2017].

Require: stepsize η_t, batch size b; initialization $\boldsymbol{\phi}_0$; temperature κ;
for $t = 1, 2, \cdots, T$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $\mathcal{T}$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{p}_y = \frac{1}{|S_\tau^{(y)}|} \sum_{(\mathbf{x}_i, y_i) \in S_\tau^{(y)}} \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})$, for each $y \in \mathcal{Y}_\tau$;
    $\mathbb{P}(y_i|\mathbf{x}_i; \boldsymbol{\phi}_{t-1}) = \frac{\exp(-\kappa \operatorname{dist}(\mathbf{p}_{y_i}, \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})))}{\sum_{y' \in \mathcal{Y}_\tau} \exp(-\kappa \operatorname{dist}(\mathbf{p}_{y'}, \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})))}$ for $(\mathbf{x}_i, y_i) \in \mathcal{Q}_\tau$;
    $\mathcal{L}(\mathcal{Q}_\tau; \boldsymbol{\phi}_{t-1}) = -\frac{1}{|\mathcal{Q}_\tau|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{Q}_\tau} \log \mathbb{P}(y_i|\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$;
  end for
  meta-learner: $\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}_{t-1}} \mathcal{L}_\tau(\mathcal{Q}_\tau; \boldsymbol{\phi}_{t-1})$;
end for
return $\boldsymbol{\phi}_T$.

and the label prediction for $(\mathbf{x}, \cdot) \in \mathcal{Q}_\tau$ is ($y \in \mathcal{Y}_\tau$)

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\phi}_{t-1}) = \frac{\exp(-\kappa \operatorname{dist}(\mathbf{p}_y, \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})))}{\sum_{y' \in \mathcal{Y}_\tau} \exp(-\kappa \operatorname{dist}(\mathbf{p}_{y'}, \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})))}, \tag{2.14}$$

where κ is a temperature, and $\operatorname{dist}(\mathbf{z}_1, \mathbf{z}_2) = \frac{1}{2}\|\mathbf{z}_1 - \mathbf{z}_2\|^2$. The meta-learner updates meta-parameter as follows:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} + \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \frac{1}{|\mathcal{Q}_\tau|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{Q}_\tau} \nabla_{\boldsymbol{\phi}_{t-1}} \log \mathbb{P}(y_i|\mathbf{x}_i; \boldsymbol{\phi}_{t-1}), \tag{2.15}$$

The framework of ProtoNet is simple and general, and the distance metric is flexible. Other possible *metrics* are cosine similarity [Chen et al., 2018a, Gidaris and Komodakis, 2018, Liu et al., 2020], earth mover's distance [Zhang et al., 2020], a metric learned by deep networks [Sung et al., 2018], a metric based on graph convolution blocks [Garcia and Bruna, 2018].

Beside the distance metric, the *temperature* is also crucial to the performance, e.g., $\kappa = 2$ or 10 is a good setting [Liu et al., 2020, Oreshkin et al., 2018]. Theoretical result in [Oreshkin et al., 2018] reveals that the temperature balances two terms: (i) the similarity between query examples and their corresponding prototypes and (ii) the similarity between the samples and the prototypes of the non-belonging classes. Empirically, an optimal temperature exists and can be chosen by cross validation [Oreshkin et al., 2018]. [Chen et al., 2020a] propose to learn task-specific temperature by amortized variational inference.

For complex tasks, representing each class as a single prototype is not suitable when class is multi-modal. To address this issue, Allen et al. [2019] propose infinite mixture prototypes

to represent a class as a mixture of clusters, with the number of clusters derived from the training data.

2.2.2 Linear Classifier

Nearest neighbor classifier in ProtoNet [Snell et al., 2017] and its variants [Chen et al., 2018a, Liu et al., 2020] are simple and perform well in the few-shot classification tasks. However, when the embedding space is high-dimensional, a trainable linear classifier is more expressive and often performs better. R2D2 [Bertinetto et al., 2018] and MetaOptNet [Lee et al., 2019b] are two representative algorithms.

Let $\mathbf{z} = \mathtt{NN}(\mathbf{x}; \boldsymbol{\phi}) \in \mathbb{R}^e$ denote the feature embedding of $\mathbf{x}$, $\mathbf{Z}_\tau^{tr} \in \mathbb{R}^{|\mathcal{S}_\tau| \times e}$ is the embedding matrix of $\mathcal{S}_\tau$ (each row vector corresponds a feature embedding), and $\mathbf{Y}_\tau^{tr} \in \mathbb{R}^{|\mathcal{S}_\tau| \times C}$ is the label matrix (assume $|\mathcal{Y}_\tau| = C$).

The base learner takes $\mathcal{S}_\tau$ and $\boldsymbol{\phi}_{t-1}$ to construct task model $\mathbf{W}_\tau^\star$ by solving the following ridge regression problem:

$$\mathbf{W}_\tau^\star = \underset{\mathbf{W}_\tau \in \mathbb{R}^{e \times C}}{\arg\min} \frac{1}{2} \|\mathbf{Z}_\tau^{tr} \mathbf{W}_\tau - \mathbf{Y}_\tau^{tr}\|^2 + \frac{\lambda}{2} \|\mathbf{W}_\tau\|^2, \tag{2.16}$$

where $\lambda > 0$ is a hyperparameter. The above problem has a closed-form solution $\mathbf{W}_\tau^\star = \left(\mathbf{Z}_\tau^{tr\top} \mathbf{Z}_\tau^{tr} + \lambda \mathbf{I}\right)^{-1} \mathbf{Z}_\tau^{tr\top} \mathbf{Y}_\tau^\top$, which implicitly depends on $\boldsymbol{\phi}_{t-1}$ via $\mathbf{Z}_\tau^{tr}$. In the inner loop, different from MAML [Finn et al., 2017], R2D2 keeps the feature extractor frozen and only learns the linear classifier. The meta-learner takes $\mathbf{W}_\tau^\star$ to makes prediction on query samples $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{Q}_\tau$ as

$$\hat{\mathbf{y}}_i = a_{t-1}\, \mathtt{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})^\top \mathbf{W}_\tau^\star + c_{t-1}, \tag{2.17}$$

where a_{t-1} and c_{t-1} are meta-parameters for scaling. Meta-parameters are updated as

$$(\boldsymbol{\phi}_t, a_t, c_t) = (\boldsymbol{\phi}_{t-1}, a_{t-1}, c_{t-1}) - \eta \nabla_{(\boldsymbol{\phi}_{t-1}, a_{t-1}, c_{t-1})} \frac{1}{b} \sum_{\tau \in \mathcal{B}_t} \frac{1}{n_q} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{Q}_\tau} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i). \tag{2.18}$$

MetaOptNet extends the ridge regression in R2D2 to a general convex classifier, e.g., support vector machine (SVM). The base learner solves the dual problem:

$$\max_{\boldsymbol{\alpha}_{\tau,1}, \ldots, \boldsymbol{\alpha}_{\tau,C}} -\frac{1}{2} \sum_c \|\boldsymbol{\alpha}_{\tau,c}^\top \mathbf{Z}_\tau^{tr}\|^2 + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \alpha_{\tau, y_i, i} \tag{2.19}$$

$$\text{s.t.} \quad \alpha_{\tau, y_i, i} \le \gamma, \alpha_{\tau, c, i} \le 0 \text{ for } c \ne y_i, \mathbf{1}^\top \boldsymbol{\alpha}_{\tau, \cdot, i} = 0, \forall i. \tag{2.20}$$

Algorithm 4 MetaOptNet [Lee et al., 2019b].

Require: stepsize η_t, batch size b; hyperparameter γ; initialization $\boldsymbol{\phi}_0$;

for $t = 1, 2, \cdots, T$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $\mathcal{T}$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{z}_i = \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$ for each $(\mathbf{x}_i, y_i) \in S_\tau$;
    $\mathbf{Z}_\tau^{tr} = [\mathbf{z}_1, \ldots, \mathbf{z}_{n_s}]^\top$, $\mathbf{Y}_\tau^{tr} = [\mathbf{y}_1, \ldots, \mathbf{y}_{n_s}]^\top$;
    obtain dual solution $\boldsymbol{\alpha}_\tau^\star$ by

$$\max_{\boldsymbol{\alpha}_{\tau,1}, \ldots, \boldsymbol{\alpha}_{\tau,C}} -\frac{1}{2} \sum_c \|\boldsymbol{\alpha}_{\tau,c}^\top \mathbf{Z}_\tau^{tr}\|^2 + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \alpha_{\tau, y_i, i}$$
$$\text{s.t.} \quad \alpha_{\tau, y_i, i} \leq \gamma, \alpha_{\tau, c, i} \leq 0 \text{ for } c \neq y_i, \mathbf{1}^\top \boldsymbol{\alpha}_{\tau, \cdot, i} = 0, \forall i.$$

  end for
  meta-learner:
  for $\tau \in \mathcal{B}_t$ **do**
    for $(\mathbf{x}_i, y_i) \in \mathcal{Q}_\tau$ **do**
      $\hat{\mathbf{y}}_i = \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})^\top \mathbf{Z}_\tau^{tr\top} \boldsymbol{\alpha}_\tau^\star$,
    end for
    $\mathbf{g}_\tau = \frac{1}{n_q} \sum_{(\mathbf{x}_i, y_i) \in Q_\tau} \nabla_{\boldsymbol{\phi}_{t-1}} \ell(\hat{\mathbf{y}}_i, y_i)$;
  end for
  $\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for
return $\boldsymbol{\phi}_T$.

Unlike ridge regression, the above quadratic program (QP) has no closed-form solution. As the optimization variable is of size $n_s \times C$, which is independent of the embedding dimension, the dual variable can be obtain with low cost by an iterative solver, e.g., projected gradient methods [Calamai and Moré, 1987, Lin, 2007]. With the dual solution $\boldsymbol{\alpha}_\tau^\star \in \mathbb{R}^{n_s \times C}$, the prediction of a query example $(\mathbf{x}, y)$ is $\hat{\mathbf{y}} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})^\top \mathbf{Z}_\tau^{tr\top} \boldsymbol{\alpha}_\tau^\star$. The meta-learner updates the meta-parameters as:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \frac{1}{n_q} \sum_{(\mathbf{x}_i, y_i) \in Q_\tau} \nabla_{\boldsymbol{\phi}_{t-1}} \ell(\hat{\mathbf{y}}_i, y_i). \tag{2.21}$$

Note that $\boldsymbol{\alpha}_\tau^\star$ is a solution to the dual problem, implicitly depends on $\boldsymbol{\phi}_t$. Similar to iMAML [Rajeswaran et al., 2019], we can use implicit function theorem to compute the gradient of $\nabla_{\boldsymbol{\phi}_{t-1}} \boldsymbol{\alpha}_\tau^\star$. See the CVXPYLayers package [Agrawal et al., 2019] for an implementation to solve the dual problem and back-propagate gradients though the convex learner.

Kernel Learner. The dual formulation in MetaOptNet allows introduction of nonlinear-

ity with the kernel trick, e.g., radial basis function (RBF) kernel, cosine kernel, deep kernels [Wilson et al., 2016]. *Deep Kernels* are more preferable due to its effectiveness. MetaFun [Xu et al., 2020] learns multiple base functions for generating functional task representations by pooling or attention. MetaVRF [Zhen et al., 2020] uses Bochner's theorem [Rudin, 2017] to learn adaptive kernels with random Fourier feature for the base learner. Patacchiola et al. [2020] propose deep kernel transfer to learn a parameterized base kernel as well as the deep network. F-PACOH [Rothfuss et al., 2021b] builds on a PAC-Bayesian framework [Rothfuss et al., 2021a] to learn a hyper prior in the functional space.

2.2.3 Conditional Meta-Learning

Using the same embedding space may not be discriminative enough for all tasks, especially for complex tasks. To alleviate this issue, several metric learning algorithms with conditional feature extractor are proposed [Oreshkin et al., 2018, Jiang et al., 2018, Li et al., 2019, Denevi et al., 2022, Xing et al., 2019].

CAML Jiang et al. [2018] incorporates class representations into the embedding space to transform feature embedding of the base learner by conditional batch normalization. TADAM [Oreshkin et al., 2018] learns a task-dependent metric space by injecting task representation into feature extractor via a FiLM conditioning layer [Perez et al., 2018]. [Xing et al., 2019] propose AM3 to use auxiliary modalities such as semantic information of label name to rectify prototype representation of ProtoNet [Snell et al., 2017]. Recent visual language models [Radford et al., 2021, Tsimpoukelli et al., 2021, Alayrac et al., 2022] show that semantic feature is particularly useful for few-shot image classification tasks. Li et al. [2019] search for the most class-relevant features by a category traversal module, which consists of concentrator and projector learned by the meta-learner. Denevi et al. [2020] propose a conditional meta-learning approach to incorporate task information into feature representation, and theoretically analyze the effectiveness of conditional meta-learning in the linear regression setting. TASML [Wang et al., 2020c] learn the task model by weighing meta-training tasks on target tasks, where the weights are the similarity between target tasks and meta-training tasks. CNP [Garnelo et al., 2018] learns to map training data to a distribution over functions in RKHS. GPML [Nguyen et al., 2021] induces task information to construct a task-dependent kernel.

Conditional meta-learning also can be used for optimization-based methods. For example,

LEO [Rusu et al., 2019] encodes support samples into a task representation, which is decoded to obtain a task-specific initialization. Both encoder and decoder are learned in the outer loop. ModGrad [Simon et al., 2020] learns a task-dependent gradient modulation to generate preconditioning matrix for the base learner. ALFA [Baik et al., 2020] meta-learns a small hyper-network to map to generate hyperparameters in the base learner (e.g., learning rate, weight decay coefficients) from task embedding for fast adaptation. CxGrad [Lee et al., 2022] meta-learns a two-layer MLP to generate scaling factor in batch normalization layer based on task embedding.

2.3 Memory-based Meta-Learning

Memory-based meta-learning stores meta-knowledge extracted from historical tasks into memory, which is maintained by the meta-learner and used in future task learning. Typical algorithms are based on MANN [Santoro et al., 2016] and hypernetwork [Munkhdalai and Yu, 2017, Ha et al., 2017]

2.3.1 Memory Augmented Neural Network (MANN)

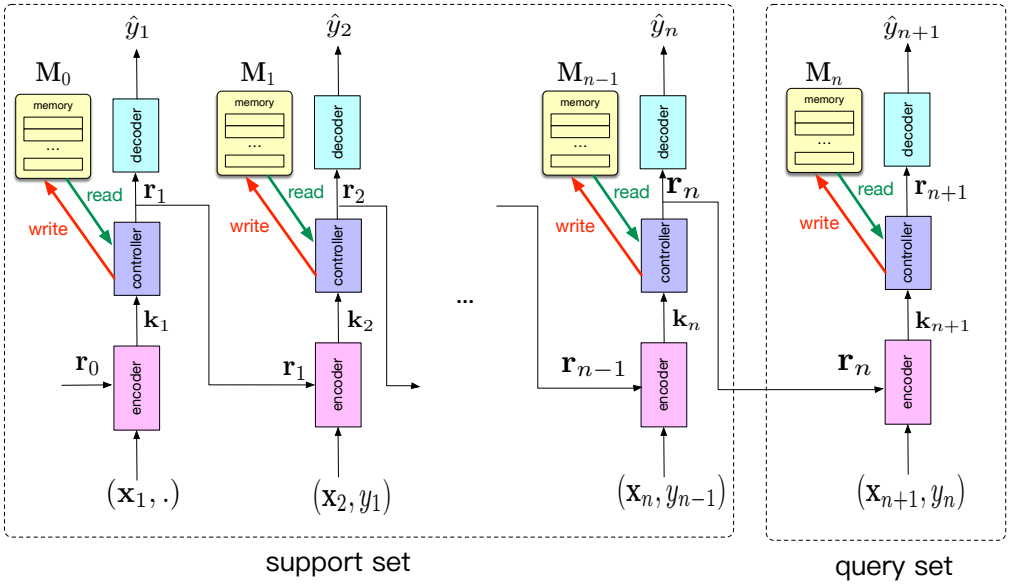


Figure 2.6: MANN.

MANN [Santoro et al., 2016] is a representative memory-based method, which is based on Neural Turing Machine. The architecture is shown in Figure 2.6, consists of four modules:

- *Encoder* takes input $\mathbf{x}_t$ and previous state $\mathbf{r}_{t-1}$ to produce an embedding $\mathbf{k}_t$. Encoder can be recurrent neural networks [Hochreiter and Schmidhuber, 1997, Cho et al., 2014, Ma and Hovy, 2016], convolutional neural networks [Krizhevsky et al., 2017, He et al., 2016], feedforward neural networks [Svozil et al., 1997].

- *Memory* $\mathbf{M}_t$ is used for storing knowledge from history data, containing m rows with each row vector being one type of knowledge. The memory can be read and written access by a controller.

- *Controller* is used to maintain the memory, including retrieving sample-specific knowledge and updating memory. For retrieval, sample-specific knowledge is a weighted combination of memory knowledge:

$$\mathbf{r}_t = \sum_{i=1}^{m} w_{t,i}^r \mathbf{M}_{t,i}, \tag{2.22}$$

  where the weight is similarity between $\mathbf{k}_t$ and knowledge vectors

$$\mathbf{w}_t^r = \mathsf{softmax}\left[\cos(\mathbf{k}_t, \mathbf{M}_{t,1}), \ldots, \cos(\mathbf{k}_t, \mathbf{M}_{t,m})\right]. \tag{2.23}$$

  To update memory, MANN uses a Least Recently Used Access (LRUA) module [Santoro et al., 2016], which writes memory to either the least used memory address or the most recently used memory address. Memory is updated as follows

$$\mathbf{M}_t = \mathbf{M}_{t-1} + \mathbf{w}_t^w \mathbf{k}_t^\top, \tag{2.24}$$

  where $\mathbf{w}_t^w$ is the writing weight produced by LRUA.

- *Decoder* takes $\mathbf{k}_t$ and feeds it to a predictor (e.g., a softmax output layer), outputs a prediction $\hat{y}_t$. At time $t+1$, the ground truth y_t is received, then both encoder and decoder are optimized by minimizing $\ell(\hat{y}_t, y_t)$ in an end-to-end manner.

APL [Ramalho and Garnelo, 2019] learns a surprised-based controller to maintain the memory, where surprise metric is defined on the label probability. As frequently reading and writing the memory module is computationally expensive, Rae et al. [2016] propose sparse reads and writes to improve the efficiency of memory access strategy.

In addition to Neural Turing Machine, other memory structures also can be applied in memory-based methods. For example, ARC uses an attention-based RNN to develop an adaptive representation space for classifying images. SNAIL [Mishra et al., 2018] combines

temporal convolutional networks and attention mechanism to retrieve history knowledge. CSN [Munkhdalai et al., 2018] is a conditional meta-learning approach, whose meta learner task-specific shifts from memory via a key-based attention mechanism and feeds them to the base learner to shift network's activation values. [Babu et al., 2021] distributes memory across layers to achieve fast adaptation in activation space. NEC [Pritzel et al., 2017] maintains a differentiable neural dictionary to store historical episode knowledge in reinforcement learning, whose keys are slow-changing state representations and values are fast-updated estimates of the value function. Wu et al. [2018] propose a memory mechanism called Kanerva Machine by combining a variational auto-encoder with a linear model, while Bartunov et al. [2020] use an energy model to store knowledge.

2.3.2 Hypernetworks

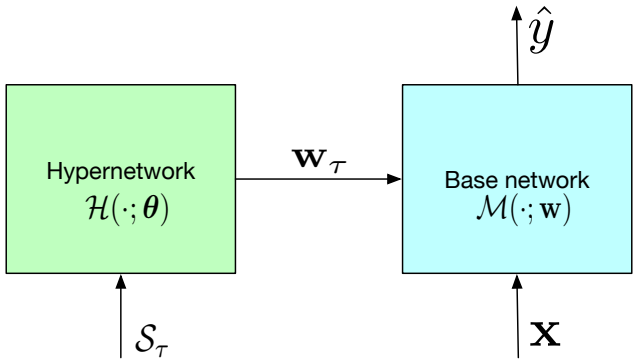


Figure 2.7: Hypernetwork.

Hypernetworks [Ha et al., 2017] (also called Meta-Networks [Munkhdalai and Yu, 2017]) are networks that can generate the weights for another deep networks conditioned on task information. Hypernetworks can be viewed as a memory contains knowledge of how to produce task model weights. Hence, meta-learning methods based on hypernetworks [Munkhdalai and Yu, 2017, Rusu et al., 2019, Sitzmann et al., 2020, Sendera et al., 2023, Galanti and Wolf, 2020, Ye and Ren, 2021, Beck et al., 2022, Knyazev et al., 2021, Qiao et al., 2018, Zhmoginov et al., 2022, Navon et al., 2021, Ehret et al., 2021] are categorized into memory-based.

An illustration is shown in Figure 2.7. Specifically, a hypernetwork $\mathcal{H}(\cdot;\boldsymbol{\theta})$, parameterized by $\boldsymbol{\theta}$, receives task context $\mathcal{S}_\tau$ and generates base network parameter $\mathbf{w}_\tau$. When the base network is deep network with millions of parameters, generating parameters for the whole model is infeasible. Some practical implementations include: (i) *generate parts of networks.* For example, HyperShot [Sendera et al., 2023], [Qiao et al., 2018], PFedHN [Shamsian

et al., 2021], HyperTransformer [Zhmoginov et al., 2022], and LEO [Rusu et al., 2019] generates parameters in the last layer or classifier; (ii) generate weights to combine model templates [Zhao et al., 2020, Suarez, 2017]. Specifically, $\{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\}$ are K templates. Hypernetwork maps $\mathcal{S}_\tau$ to a K-dimensional combination vector $\boldsymbol{\lambda}_\tau$, and the task model is

$$\mathbf{w}_\tau = \sum_{k=1}^{K} \boldsymbol{\lambda}_{\tau,k} \boldsymbol{\theta}_k. \tag{2.25}$$

(iii) generate parameters in feature transformation layers, such as normalization layer [Dosovitskiy and Djolonga, 2020], FiLM layer [Brockschmidt, 2020].

CHAPTER 3

PROPOSED METAPROX

3.1 Motivation

Meta-regularization [Rajeswaran et al., 2019, Zhou et al., 2019, Denevi et al., 2018, 2019, 2020] algorithms have shown promising performance, in which the base learner learns the task-specific model by minimizing the loss with a proximal regularizer (a biased regularizer from the meta-parameter). Denevi et al. [2018] uses a linear model with efficient closed-form solution for the base learner. As real-world tasks are usually complex, a linear model is not expressive enough. However, extending to nonlinear base learners requires computing the meta-gradient using matrix inversion, which can be infeasible for deep networks [Rajeswaran et al., 2019].

To introduce nonlinearity to the base learner, a recent approach is to make use of the kernel trick. For example, R2D2 [Bertinetto et al., 2018] and MetaOptNet [Lee et al., 2019b] use deep kernels [Wilson et al., 2016] in meta-learning for few-shot classification. Specifically, the deep network is learned in the meta-learner, while a base kernel is used in the base learner. Though they achieve state-of-the-art performance, their base learners use a Tikhonov regularizer rather than a learnable proximal regularizer as in meta-regularization methods.

As learning a meta-regularization has been shown to be effective in linear models for regression [Denevi et al., 2018] and classification [Denevi et al., 2019], in this section, we propose a kernel-based algorithm to meta-learn a proximal regularizer for a nonlinear base learner.

3.2 Method

To introduce nonlinearity in both models and losses, we propose MetaProx [Jiang et al., 2021a] by using deep kernels to learn a proximal regularizer for the base learner (Algorithm 5). Specifically, let $\mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})$ be a feature extractor parameterized by $\boldsymbol{\phi}$. An input $\mathbf{x}$ is mapped to $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})$ in an embedding space $\mathcal{E}$. $\mathbf{Z}_\tau \equiv [\mathbf{z}_1^\top; \ldots; \mathbf{z}_{n_s}^\top]$, where

$\mathbf{z}_i = \mathtt{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$ for $\mathbf{x}_i \in S_\tau$. $\mathcal{K}$ is a base kernel on $\mathcal{E} \times \mathcal{E}$, and $\mathcal{H}$ is the corresponding RKHS. The primal problem in the inner loop is:

$$\hat{f}_\tau = \arg\min_{f_\tau \in \mathcal{H}} \sum\nolimits_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i), y_i) + \frac{\lambda}{2} \|f_\tau - f_{\boldsymbol{\theta}_{t-1}}\|_{\mathcal{H}}^2.$$

By representer theorem, $\hat{f}_\tau(\cdot; \boldsymbol{\alpha}_\tau) = f_{\boldsymbol{\theta}_{t-1}}(\cdot) + \mathcal{K}(\mathbf{Z}_\tau, \cdot)^\top \boldsymbol{\alpha}_\tau$, where $\boldsymbol{\alpha}_\tau$ is solved in the dual space:

$$\boldsymbol{\alpha}_\tau = \min_{\boldsymbol{\alpha}} \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i; \boldsymbol{\alpha}), y_i) + \boldsymbol{\alpha}^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau) \boldsymbol{\alpha}. \tag{3.1}$$

The meta-learner then performs update as

$$(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) = (\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi}_{t-1}) - \eta \sum_{(\mathbf{x}, y) \in Q_\tau} \nabla_{(\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi}_{t-1})} \ell(f_\tau(\mathbf{z}; \boldsymbol{\alpha}_\tau), y). \tag{3.2}$$

MetaProx has several advantages:

1. After kernel extension, $f_{\boldsymbol{\theta}}$ is a function in $\mathcal{H}$. For nonlinear kernels (e.g., RBF kernel, cosine kernel), $f_{\boldsymbol{\theta}}$ is nonlinear, thus, MetaProx learns a meta-regularization for a *nonlinear* base learner.
2. $f_{\boldsymbol{\theta}}$ in the base learner is *learnable*. By setting $f_{\boldsymbol{\theta}} = \mathbf{0}$, MetaProx recovers the state-of-the-art MetaOptNet [Lee et al., 2019b].
3. For square loss, $\boldsymbol{\alpha}_\tau = (\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1}(\mathbf{y}_\tau - f_{\boldsymbol{\theta}_{t-1}}(\mathbf{Z}_\tau))$ has an efficient closed-form solution. For general losses, the dual problem is convex and can be solved efficiently, as the size of $\boldsymbol{\alpha}$ is very small (only n_s). Though MetaProx still requires matrix inversion in computing meta-gradients, the size is only $n_s \times n_s$, much smaller than $n_{\boldsymbol{\phi}} \times n_{\boldsymbol{\phi}}$ in iMAML [2].

3.3 Theoretical Analysis

Let $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \frac{1}{|\mathcal{T}|} \sum_{\tau \in \mathcal{T}} \sum_{(\mathbf{x}, y) \in Q_\tau} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\alpha}_\tau), y)$ be the empirical loss of generalization measure $\mathbb{E}_{\tau \sim p(\tau)} \sum_{(\mathbf{x}, y) \in Q_\tau} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\alpha}_\tau), y)$, where $\mathbf{z} = \mathtt{NN}(\mathbf{x}; \boldsymbol{\phi})$. With the linear kernel and square loss, the dual solution (3.1) is affine in the meta-parameter, and so is the primal

Algorithm 5 MetaProx [Jiang et al., 2021a].

Require: stepsize η_t, batch size b, feature extractor $\mathsf{NN}(\cdot;\boldsymbol{\phi})$, base kernel $\mathcal{K}$;
for $t = 1, 2, \cdots, T$ **do**
  sample a batch $\mathcal{B}_t$ of tasks from $\mathcal{T}$;
  base learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{z}_i = \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$ for each $(\mathbf{x}_i, y_i) \in S_\tau$;
    $f_\tau(\mathbf{z}; \boldsymbol{\alpha}) \equiv f_{\boldsymbol{\theta}_{t-1}}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top \boldsymbol{\alpha}$ denote the task model w.r.t. dual variables;
    $\boldsymbol{\alpha}_\tau = \arg\min_{\boldsymbol{\alpha}} \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \ell(f_\tau(\mathbf{z}_i; \boldsymbol{\alpha}), y_i) + \boldsymbol{\alpha}^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\boldsymbol{\alpha}$;
  end for
  meta-learner:
  for $\tau \in \mathcal{B}_t$ **do**
    $\mathbf{g}_\tau = \sum_{(\mathbf{x}, y) \in \mathcal{Q}_\tau} \nabla_{(\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi}_{t-1})} \ell(\hat{y}, y)$, where $\hat{y} = f_\tau(\mathbf{z}; \boldsymbol{\alpha}_\tau)$ and $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}_{t-1})$;
  end for
  $(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) = (\boldsymbol{\theta}_{t-1}, \boldsymbol{\phi}_{t-1}) - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for
return $(\boldsymbol{\theta}_T, \boldsymbol{\phi}_T)$.

solution $\mathbf{w}_\tau = \boldsymbol{\theta} + \lambda^{-1} \mathbf{X}_\tau^\top \boldsymbol{\alpha}_\tau$. Thus, the meta-loss $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is convex and convergence follows from convex optimization [**?**Denevi et al., 2018]. After introducing nonlinearity, the meta-loss is no longer convex. The following introduces Lipschitz-smoothness assumptions, which have been commonly used in stochastic non-convex optimization [Ghadimi and Lan, 2013, Reddi et al., 2016] and meta-learning in non-convex settings [Fallah et al., 2020, Zhou et al., 2019].

Assumption 3.3.1 (Smoothness)**.** (i) The deep network $\mathsf{NN}(\mathbf{x}; \boldsymbol{\phi})$ is Lipschitz-smooth, i.e., $\|\nabla_{\boldsymbol{\phi}} \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}) - \nabla_{\boldsymbol{\phi}} \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}')\| \leq \beta_1 \|\boldsymbol{\phi} - \boldsymbol{\phi}'\|$ with a Lipschitz constant $\beta_1 > 0$; (ii) the kernel $\mathcal{K}(\mathbf{z}, \mathbf{z}')$ is Lipschitz-smooth w.r.t. $(\mathbf{z}, \mathbf{z}')$; (iii) $f_{\boldsymbol{\theta}}(\mathbf{z})$ is Lipschitz-smooth w.r.t. $(\boldsymbol{\theta}, \mathbf{z})$; (iv) $\nabla_1^2 \ell(\hat{y}, y)$ is Lipschitz w.r.t. $\hat{y}$, i.e., $|\nabla_1^2 \ell(\hat{y}, y) - \nabla_1^2 \ell(\hat{y}', y)| \leq \beta_2 |\hat{y} - \hat{y}'|$ with a Lipschitz constant β_2; (v) $\mathbb{E}_{\tau \sim \mathcal{T}} \|\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \sum_{(\mathbf{x}, y) \in \mathcal{Q}_\tau} \ell(f_{\boldsymbol{\theta}}(\mathbf{z}; \boldsymbol{\alpha}_\tau), y) - \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})\|^2 = \sigma_{\mathbf{g}}^2$, where $\tau \sim \mathcal{T}$ denotes uniformly sample a task from $\mathcal{T}$.

The following Lemma guarantees smoothness of the meta-loss.

Lemma 3.3.2. *$\mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is Lipschitz-smooth w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi})$ with a Lipschitz constant β_{meta}.*

Theorem 3.3.1. *Let the step size be $\eta_t = \min(1/\sqrt{T}, 1/2\beta_{meta})$. Algorithm 5 satisfies*

$$\min_{1 \leq t \leq T} \mathbb{E} \|\nabla_{(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t)} \mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t)\|^2 = \mathcal{O}\left(\sigma_{\mathbf{g}}^2 / \sqrt{T}\right),$$

where the expectation is taken over the random training samples.

This rate is the same as MAML [Fallah et al., 2020, Ji et al., 2020b] and Meta-MinibatchProx [Zhou et al., 2019]. For MAML with $J > 1$ gradient steps, [Ji et al., 2020b] assumes that the step size in the inner loop is of the order $1/J$. This slows down inner loop learning when J is large. On the other hand, MetaProx does not have this restriction, as its meta-gradient depends only on the last iterate rather than all iterates along the trajectory.

Next, we study the global convergence of MetaProx. Prior work [Finn et al., 2019, Zhou et al., 2019] focus on the case where $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is strongly convex in $(\boldsymbol{\theta}, \boldsymbol{\phi})$. This can be restrictive in deep learning. We instead only require $\ell(\hat{y}, y)$ to be strongly convex in $\hat{y}$. This assumption is easily met by commonly-used loss functions such as the square loss and logistic loss with a compact domain. A recent work [Wang et al., 2020a] studies the global convergence of MAML in over-parameterized neural networks. Over-parameterization is closely related to the assumption of uniform conditioning [Jacot et al., 2018, Lee et al., 2019a, Liu et al., 2022a].

Assumption 3.3.3 (Uniform conditioning [Liu et al., 2022a])**.** A multivariable function $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is μ-uniformly conditioning if its tangent kernel [Jacot et al., 2018] satisfies $\min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \lambda_{\min}(\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi}) \nabla^{\top}_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})) \geq \mu > 0$, where $\lambda_{\min}(\cdot)$ is the smallest eigenvalue of the matrix argument.

Assume that the loss $\ell(\cdot, \cdot)$ is ρ-strongly convex w.r.t. the first argument and Assumption 3.3.1 holds. Let $\mathbf{x}_{\tau,j}$ be the jth query example of task τ, $\mathbf{z}_{\tau,j}$ be its embedding, and $\hat{y}_{\tau,j} = f_{\boldsymbol{\theta}}(\mathbf{z}_{\tau,j}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_{\tau,j})^\top \boldsymbol{\alpha}_\tau$ be its prediction, where $\boldsymbol{\alpha}_\tau$ is the dual solution. Let $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \left[\hat{y}_{\tau_1,1}; \ldots; \hat{y}_{\tau_1,n_q}; \ldots; \hat{y}_{\tau_{|\mathcal{T}|},1}; \ldots; \hat{y}_{\tau_{|\mathcal{T}|},n_q}\right]$ be an auxiliary function which maps the meta-parameter to predictions on all query examples. The following Theorem shows that the proposed algorithm converges to a global minimum of the empirical risk $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ at the rate of $\mathcal{O}(\sigma_{\mathbf{g}}^2/\sqrt{T})$. The rate is improved to exponential if the meta-learner adopts full gradient descent.

Theorem 3.3.2. *Assume that $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is uniform conditioning. (i) Let $\eta_t = \min(1/\sqrt{T}, 1/2\beta_{meta})$. Algorithm 5 satisfies $\min_{1\leq t\leq T} \mathbb{E}\mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) - \min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{O}\left(\sigma_{\mathbf{g}}^2/\sqrt{T}\right)$, where the expectation is taken over the random training samples. (ii) Let $\eta_t = \eta < \min(1/2\beta_{meta}, 4|\mathcal{T}|/\rho\mu)$ and $\mathbb{B}_t = \mathcal{T}$. Algorithm 5 satisfies $\mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) - \min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{O}((1 - \eta\rho\mu/4|\mathcal{T}|)^t)$.*

3.4 Experiments on Few-shot Regression

Data sets. Experiments are performed on three data sets.

(i) *Sine*. This is the sinusoid regression problem in Finn et al. [2017]. Samples x's are uniformly sampled from $[-5, 5]$. Each task τ learns a sine function $y = a_\tau \sin(x + b_\tau) + \xi$, where $a_\tau \in [0.1, 5]$, $b_\tau \in [0, \pi]$, and $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the label noise. We consider both $\sigma_\xi^2 = 0$ (noise-free) and $\sigma_\xi^2 = 1$. In addition to the 5-shot setting in Finn et al. [2017], we also evaluate on the more challenging 2-shot setting. We randomly generate a meta-training set of 8000 tasks, a meta-validation set of 1000 tasks for early stopping, and a meta-testing set of 2000 tasks for performance evaluation.

(ii) *Sale*. This is a real-world dataset from Tan and San Lau [2014], which contains weekly purchased quantities of 811 products over 52 weeks. For each product (task), a sample is to predict the sales quantity for the current week from sales quantities in the previous 5 weeks. Thus, each product contains 47 samples. We evaluate on the 5-shot and 1-shot settings. We randomly split the tasks into a meta-training set of 600 tasks, a meta-validation set of 100 tasks, and a meta-testing set of 111 tasks.

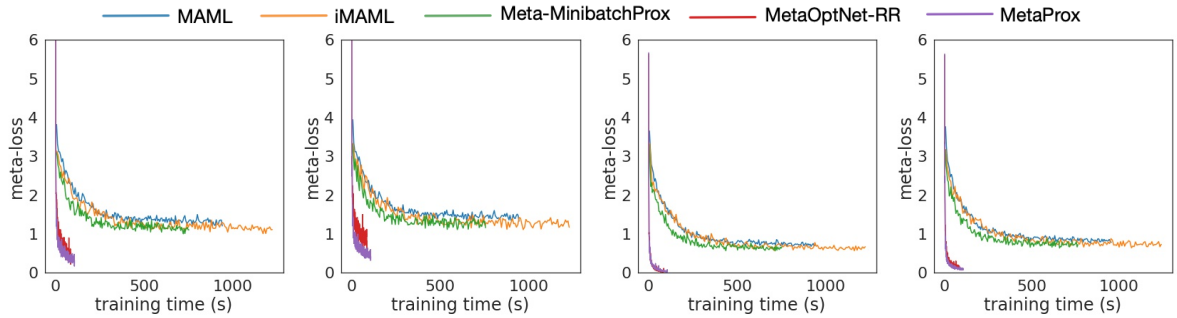
(iii) *QMUL*, which is a multiview face dataset [Gong et al., 1996] from Queen Mary University of London. This consists of grayscale face images of 37 people (32 for meta-training and 5 for meta-testing). We follow the setting in Patacchiola et al. [2020] and evaluate the model on 10-shot regression. Each person has 133 facial images covering a viewsphere of ± 90 in yaw and ± 30 in tilt at 10 increment. A task is a trajectory taken from the discrete manifold for images from the same person. The regression goal is to predict the tilt given an image. In the *in-range* setting, meta-training tasks are sampled from the entire manifold. In the more challenging *out-of-range* setting, meta-training tasks are sampled from the sub-manifold with yaw in $[-90, 0]$. In both settings, meta-testing tasks are sampled from the entire manifold. We randomly sample 2400 tasks for meta-training, and 500 tasks for meta-testing. As in Patacchiola et al. [2020], we do not use a meta-validation set since the dataset is small.

Network Architecture. For *Sine* and *Sale*, we use the network in Finn et al. [2017], which is a small multilayer perceptron with two hidden layers of size 40 and ReLU activation. For *QMUL*, we use the three-layer convolutional neural network in Patacchiola et al. [2020].

The embeddings are always from the last hidden layer. We use a simple linear kernel as base kernel, and $f_{\boldsymbol{\theta}}(\mathbf{z}) = \boldsymbol{\theta}^\top \mathbf{z}$.

Implementation Details. We use the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.001. Each mini-batch has 16 tasks. For *Sine* and *Sale*, the model ($\boldsymbol{\phi}$ and $f_{\boldsymbol{\theta}}$) is meta-trained for $40,000$ iterations. To prevent overfitting on the meta-training set, we evaluate the meta-validation performance every 500 iterations, and stop training when the loss on the meta-validation set has no significant improvement for 10 consecutive evaluations. For *QMUL*, we follow Patacchiola et al. [2020] and meta-train the model for 100 iterations. We repeat each experiment 30 times. For performance evaluation, we use the average mean squared error (MSE) on the meta-testing set.

Baselines. On *Sine* and *Sale*, we compare MetaProx with CommonMean [Denevi et al., 2018], MAML [Finn et al., 2017], MetaOptNet-RR [Lee et al., 2019b], Meta-MinibatchProx [Zhou et al., 2019], and iMAML [Rajeswaran et al., 2019]. CommonMean is a linear model, and MetaOptNet-RR is equivalent to MetaProx when $f_{\boldsymbol{\theta}} = 0$. Following [Finn et al., 2017], we set the number of inner gradient steps for MAML to 1 during meta-training, and 20 during meta-validation and meta-testing. Both Meta-MinibatchProx [Zhou et al., 2019] and iMAML [Rajeswaran et al., 2019] are meta-regularization approaches. For *QMUL*, we compare MetaProx with the baselines reported in [Patacchiola et al., 2020] (namely, DKT [Patacchiola et al., 2020], Feature Transfer [Donahue et al., 2014], and MAML). As further baselines, we also compare with Meta-MinibatchProx and MetaOptNet-RR to evaluate the improvement of MetaProx due to the learnable $f_{\boldsymbol{\theta}}$.



(a) 2-shot, $\sigma_\xi^2 = 0$. (b) 2-shot, $\sigma_\xi^2 = 1$. (c) 5-shot, $\sigma_\xi^2 = 0$. (d) 5-shot, $\sigma_\xi^2 = 1$.

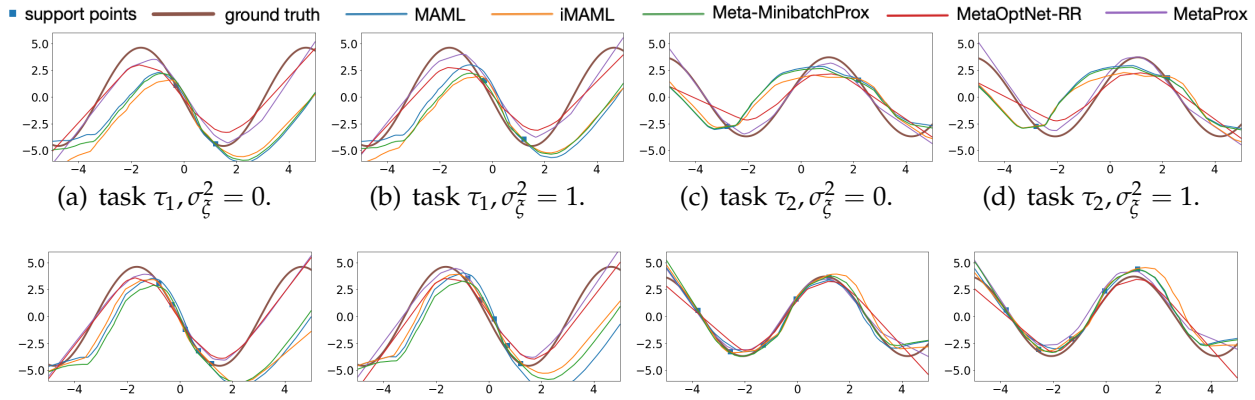
Figure 3.1: Convergence curves for few-shot sinusoid regression. Best viewed in color.

Results on Sine. Figure 3.1 shows the convergence curves of MetaProx and the baselines. We do not show the convergence of CommonMean, as it does not use a neural network backbone as the other methods. As can be seen, MetaProx converges much faster and better than the non-kernel-based methods (MAML, iMAML and Meta-MinibatchProx). In

the 2-shot settings, MetaProx converges to a loss smaller than that of MetaOptNet-RR.

Figure 3.2 shows the learned functions on 2 meta-testing tasks (τ_1 with ($a = 4.6$, $b = 3.2$) and τ_2 with ($a = 3.7$, $b = 0.5$)) in the 5-shot setting and more challenging 2-shot setting. As can be seen, MetaProx always fits the target curve well. Though MAML, iMAML and Meta-MinibatchProx can fit the support samples, it performs worse in regions far from the support samples. This is especially noticeable in the 2-shot setting.

Table 3.1 shows the MSE on the meta-testing set. Obviously, CommonMean (a linear model) fails in this nonlinear regression task. MetaProx and MetaOptNet-RR significantly outperform the other baselines. MetaProx (with the learned f_θ) performs better than MetaOptNet-RR, particularly when the data is noisy.



(a) task $\tau_1, \sigma_\xi^2 = 0$. (b) task $\tau_1, \sigma_\xi^2 = 1$. (c) task $\tau_2, \sigma_\xi^2 = 0$. (d) task $\tau_2, \sigma_\xi^2 = 1$.

(e) task $\tau_1, \sigma_\xi^2 = 0$. (f) task $\tau_1, \sigma_\xi^2 = 1$. (g) task $\tau_2, \sigma_\xi^2 = 0$. (h) task $\tau_2, \sigma_\xi^2 = 1$.

Figure 3.2: Sinusoid regression: Two meta-testing tasks τ_1 and τ_2 with different σ_ξ's in 2-shot ((a) –(d)) and 5-shot ((e)–(h)) settings. Best viewed in color.

Table 3.1: Average MSE (with 95% confidence intervals) of few-shot regression on the *Sine* and *Sale* datasets. (The confidence intervals in *Sale* experiments are ± 0.001 for all methods)

	Sine (2-shot)		*Sine* (5-shot)		*Sale*	
	noise-free	noisy	noise-free	noisy	1-shot	5-shot
CommonMean [Denevi et al., 2018]	4.58 ± 0.07	4.59 ± 0.07	4.29 ± 0.06	4.31 ± 0.06	0.090	0.074
MAML [Finn et al., 2017]	1.24 ± 0.12	1.91 ± 0.13	0.41 ± 0.03	1.15 ± 0.05	0.069	0.063
iMAML [Rajeswaran et al., 2019]	1.12 ± 0.11	1.84 ± 0.10	0.38 ± 0.02	1.02 ± 0.05	0.068	0.063
Meta-MinibatchProx [Zhou et al., 2019]	1.15 ± 0.08	1.87 ± 0.09	0.37 ± 0.02	1.01 ± 0.03	0.081	0.064
MetaOptNet-RR [Lee et al., 2019b]	0.18 ± 0.01	0.79 ± 0.01	0.01 ± 0.00	0.19 ± 0.01	0.088	0.068
MetaProx	$\mathbf{0.11 \pm 0.01}$	$\mathbf{0.43 \pm 0.01}$	$\mathbf{0.01 \pm 0.00}$	$\mathbf{0.13 \pm 0.01}$	**0.061**	**0.060**

Results on Sale. As can be seen from Table 3.1, the linear model (CommonMean) performs poorly as expected. MetaProx again outperforms the other baselines, particularly in the more challenging 1-shot setting.

Results on QMUL. Table 3.2 shows that MetaProx achieves the lowest MSE and the kernel methods (DKT+RBF, DKT+Spectral, MetaOptNet-RR, and MetaProx) perform better than

Table 3.2: Average MSE (with 95% confidence intervals) of few-shot regression on *QMUL* (10-shot). Results of the first four methods are from Patacchiola et al. [2020].

method	in-range	out-of-range
Feature Transfer [Donahue et al., 2014]	0.22 ± 0.03	0.18 ± 0.01
MAML [Finn et al., 2017]	0.21 ± 0.01	0.18 ± 0.02
DKT + RBF [Patacchiola et al., 2020]	0.12 ± 0.04	0.14 ± 0.03
DKT + Spectral [Patacchiola et al., 2020]	0.10 ± 0.02	0.11 ± 0.02
Meta-MinibatchProx [Zhou et al., 2019]	0.171 ± 0.022	0.193 ± 0.025
MetaOptNet-RR [Lee et al., 2019b]	0.021 ± 0.007	0.039 ± 0.009
MetaProx	$\mathbf{0.012} \pm \mathbf{0.003}$	$\mathbf{0.020} \pm \mathbf{0.005}$

non-kernel-based methods (Feature Transfer, MAML, and Meta-MinibatchProx). MetaProx with the learnable $f_{\boldsymbol{\theta}}$ reduces the errors of MetaOptNet-RR by half.

3.5 Experiments on Few-shot Classification

Table 3.3: Accuracies (with 95% confidence intervals) of 5-way few-shot classification on *mini-ImageNet* using *Conv4*. † means that the result is obtained by rerunning the code with our setup here. Other results from their original publications (Result on the 5-shot setting is not reported in iMAML Rajeswaran et al. [2019]).

method	1-shot	5-shot
MAML Finn et al. [2017]	48.7 ± 1.8	63.1 ± 0.9
FOMAML Finn et al. [2017]	48.1 ± 1.8	63.2 ± 0.9
REPTILE Nichol et al. [2018]	50.0 ± 0.3	66.0 ± 0.6
iMAML Rajeswaran et al. [2019]	49.0 ± 1.8	–
Meta-MinibatchProx Zhou et al. [2019]	50.8 ± 0.9	67.4 ± 0.9
ANIL Raghu et al. [2020]	46.7 ± 0.4	61.5 ± 0.5
R2D2 Bertinetto et al. [2018]	49.5 ± 0.2	65.4 ± 0.3
ProtoNet Snell et al. [2017]	49.4 ± 0.8	68.2 ± 0.7
MetaOptNet-SVM(lin)† Lee et al. [2019b]	49.8 ± 0.9	66.9 ± 0.7
MetaOptNet-SVM(cos)† Lee et al. [2019b]	50.1 ± 0.9	67.2 ± 0.6
MetaProx	$\mathbf{52.4} \pm 1.0$	$\mathbf{68.8} \pm 0.8$

Datasets. We use the standard 5-way K-shot setting ($K = 1$ or 5) on the *mini-ImageNet* [Vinyals et al., 2016] dataset, which consists of 100 randomly chosen classes from *ILSVRC*-2012 [Russakovsky et al., 2015]. Each class contains 600 84×84 images. We use the commonly-used split in Ravi and Larochelle [2017]: the 100 classes are randomly split into 64 for meta-training, 16 for meta-validation, and 20 for meta-testing.

Network Architecture. For the network backbone, we use the *Conv4* in [Finn et al., 2017, Vinyals et al., 2016] and *ResNet-12* in Lee et al. [2019b]. As the cosine similarity is more effective than ℓ_2 distance in few-shot classification Chen et al. [2018a], we adopt the

cosine kernel $\mathcal{K}(\mathbf{z}, \mathbf{z}') = \cos(\mathbf{z}, \mathbf{z}')$ as base kernel, where $\mathbf{z}$ is the embedding of sample $\mathbf{x}$ extracted from the last hidden layer as in regression. For each $c = 1, \ldots, 5$, $f_{\boldsymbol{\theta}^{(c)}} = [\mathcal{K}_{\mathbf{q}^{(1)}}; \ldots; \mathcal{K}_{\mathbf{q}^{(5)}}]^\top \boldsymbol{\theta}^{(c)}$ is a weighted prototype classifier on the embedding space, where $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(5)}$ are the class centroids computed from S_τ, and the weights $\{\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(5)}\}$ are meta-parameters.

Baselines. We compare MetaProx with the state-of-the-arts: (i) meta-initialization: MAML [Finn et al., 2017] and its variants FOMAML [Finn et al., 2017], and REPTILE [Nichol et al., 2018]; (ii) meta-regularization: iMAML **?** and Meta-MinibatchProx [Zhou et al., 2019]; and (iii) metric learning: ANIL [Raghu et al., 2020], R2D2 [Bertinetto et al., 2018], ProtoNet [Snell et al., 2017], and MetaOptNet [Lee et al., 2019b] with SVM using the linear kernel and cosine kernel. .

Table 3.4: Accuracies (with 95% confidence intervals) of 5-way few-shot classification on *mini-ImageNet* using *ResNet-12*. † means that the result is obtained by rerunning the code with our setup here.

method	1-shot	5-shot
FOMAML† Finn et al. [2017]	57.41 ± 0.71	72.12 ± 0.54
ANIL† Raghu et al. [2020]	59.66 ± 0.68	73.28 ± 0.49
ProtoNet Snell et al. [2017]	59.25 ± 0.64	75.60 ± 0.48
MetaOptNet-SVM(lin)† Lee et al. [2019b]	62.31 ± 0.64	78.21 ± 0.42
MetaOptNet-SVM(cos)† Lee et al. [2019b]	62.75 ± 0.42	78.68 ± 0.24
MetaProx	$\mathbf{63.82 \pm 0.23}$	$\mathbf{79.12 \pm 0.18}$

Implementation Details. The entire model is trained end-to-end. $\ell(\hat{\mathbf{y}}, y)$ is the cross-entropy loss. The CVXPYLayers package Agrawal et al. [2019] is used to solve the dual problem. We train the model for $80,000$ iterations, and each mini-batch has 4 tasks. We use the Adam optimizer Kingma and Ba [2015] with an initial learning rate of 0.001, which is then reduced by half every $2,500$ iterations. To prevent overfitting, we evaluate the meta-validation performance every 500 iterations, and stop training when the meta-validation accuracy has no significant improvement for 10 consecutive evaluations. We report the classification accuracy averaged over 600 tasks randomly sampled from the meta-testing set.

Results. Tables 3.3 and 3.4 show the results for *Conv4* and *ResNet-12*, respectively. As can be seen, MetaProx is always the best. Compared with MetaOptNet-SVM, MetaProx is better due to the learnable regularizer.

CHAPTER 4

APPLICATIONS IN LARGE LANGUAGE MODELS

Meta-learning is a general machine learning tool and has been successfully used in various applications, such as computer vision [Finn et al., 2017, Snell et al., 2017, Bertinetto et al., 2018, Perez-Rua et al., 2020, Kang et al., 2019, Yan et al., 2019, Wang et al., 2019] and natural language processing [Ohashi et al., 2021, Yao et al., 2021, Chen et al., 2022a, Han et al., 2021, Xiong et al., 2018, Ye and Ling, 2019, Li et al., 2022b]. We focus on its application in large language models, including prompting and in-context learning.

4.1 Application in Prompting

This section introduces an application of meta-learning in prompt tuning. We start with prompt tuning, and introduce recent state-of-the-art MetaPrompting, which learns a meta-initialized prompt for all task-specific prompts using the MAML algorithm. As a single prompt may be insufficient for complex tasks, we propose a meta structured-prompting method call **MetaPrompter** [Jiang et al., 2023], which learns a prompt pool for constructing instance-specific prompts and using a novel representative verbalizer (RepVerb).

4.1.1 Prompting

Recently, it is common to use a pre-trained MLM $\mathcal{M}(\cdot;\boldsymbol{\phi})$, with parameter $\boldsymbol{\phi}$, for various downstream tasks such as language understanding [Dong et al., 2019, Yang et al., 2019, Song et al., 2020], machine translation [Conneau and Lample, 2019, Guo et al., 2020], and text classification [Brown et al., 2020, Lester et al., 2021, Liu et al., 2022c]. Given a raw sentence represented as a sequence of n tokens $(x_1, \ldots, x_n)$, the MLM takes $\mathbf{x} = (\texttt{[CLS]}, x_1, \ldots, x_n, \texttt{[SEP]})$ as input (where `[CLS]` is the start token and `[SEP]` is the separator), and encodes it to a sequence of hidden representations $(\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \ldots, \mathbf{h}_n, \mathbf{h}_{\texttt{[SEP]}})$. In standard fine-tuning [Howard and Ruder, 2018, Devlin et al., 2019], an extra classifier (e.g., a fully connected layer with softmax normalization) is added on top of $\mathbf{h}_{\texttt{[CLS]}}$ to predict the label distribution. This classifier, together with $\boldsymbol{\phi}$, are tuned to maximize the probability of

correct labels. As language models are large (e.g., 175 billion parameters in GPT-3 [Brown et al., 2020]), fine-tuning all parameters can cause a heavy burden on computation and memory.

On the other hand, prompt learning [Brown et al., 2020, Shin et al., 2020, Ding et al., 2022] freezes the pre-trained model and formulates the downstream task as a cloze-style MLM problem. For example, in topic classification, "`Topic is [MASK]`" can be used as the prompt, where `[MASK]` is a special token for prediction. The *discrete* tokens "`Topic is`" are also called anchor tokens. An input text $\mathbf{x}$ is wrapped with the prompt and mapped to an input embedding sequence $(\mathcal{E}(\mathbf{x}), \mathcal{E}(\texttt{Topic}), \mathcal{E}(\texttt{is}), \mathcal{E}(\texttt{[MASK]}))$, where $\mathcal{E}(\cdot)$ denotes input embedding. Designing a suitable prompt requires domain expertise and a good understanding of the downstream tasks [Brown et al., 2020, Sanh et al., 2022]. Thus, manually-designed prompts are likely to be sub-optimal.

Unlike discrete prompts, prompt tuning [Lester et al., 2021, Liu et al., 2021b] uses a *continuous* prompt $\boldsymbol{\theta} \in \mathbb{R}^{L_p \times d_i}$ (of length L_p) to directly wrap the input embedding sequence as $(\mathcal{E}(\mathbf{x}), \boldsymbol{\theta}, \mathcal{E}(\texttt{[MASK]}))$. This can be further combined with anchor tokens to form a *template* [Liu et al., 2021b, Schick and Schütze, 2021, Ding et al., 2022]:

$$\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}) = (\mathcal{E}(\mathbf{x}), \boldsymbol{\theta}, \mathcal{E}(\texttt{Topic}), \mathcal{E}(\texttt{is}), \mathcal{E}(\texttt{[MASK]})).$$

The MLM then outputs the hidden embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}) \in \mathbb{R}^{d_o}$ of `[MASK]`, and infers the token to be filled at the `[MASK]` position.

A *verbalizer* [Lester et al., 2021, Ding et al., 2022, Hu et al., 2022] bridges the prediction at the `[MASK]` position and labels in prompt learning. Specifically, it is a *hand-crafted* mapping from each label y to a set of label-relevant tokens $\mathcal{V}_y$. For example, for $y = \texttt{SPORTS}$, we can have $\mathcal{V}_y = \{\texttt{sports}, \texttt{football}, \texttt{basketball}\}$. Prompt tuning then optimizes[1] $(\boldsymbol{\phi}, \boldsymbol{\theta})$ by maximizing the label probability:

$$\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{|\mathcal{V}_y|} \sum_{\texttt{w} \in \mathcal{V}_y} \mathbb{P}_{\mathcal{M}}(\texttt{[MASK]} = \texttt{w}|\mathbb{T}(\mathbf{x}; \boldsymbol{\theta})), \tag{4.1}$$

where $\mathbb{P}_{\mathcal{M}}(\texttt{[MASK]}|\mathbb{T}(\mathbf{x}; \boldsymbol{\theta}))$ is the probability distribution over vocabulary as predicted by the MLM at the `[MASK]` position.

The verbalizer is crucial to the performance of prompt learning [Lester et al., 2021, Ding et al., 2022]. However, selecting label-relevant tokens requires intensive human labor. Recent works [Hambardzumyan et al., 2021, Zhang et al., 2022a, Cui et al., 2022] propose *soft*

[1] $\boldsymbol{\phi}$ can be fixed for parameter-efficiency in prompt learning.

verbalizers, which map each label to a *continuous* embedding and predict label distribution based on the similarity between feature embedding and label embeddings. WARP [Hambardzumyan et al., 2021] and DART [Zhang et al., 2022a] obtain this label embedding by supervised learning, while ProtoVerb [Cui et al., 2022] uses contrastive learning [Chen et al., 2020b, Tian et al., 2020]. However, learning the embedding $\mathbf{v}_y \in \mathbb{R}^{d_o}$ for each label y can be challenging in the few-shot learning setting [Gao et al., 2019, Bao et al., 2020, Han et al., 2021, Chen et al., 2022a, Hou et al., 2022], as the number of samples per class is typically much smaller than d_o (e.g., $d_o = 768$ for BERT [Devlin et al., 2019]).

4.1.2 MetaPrompting

As prompt tuning is sensitive to prompt initialization in few-shot tasks [Lester et al., 2021], meta-learning can be used to search for a good initialization. MetaPrompting [Hou et al., 2022] uses MAML to learn a meta-initialization for the task-specific prompts. At iteration t, the base learner takes a task τ and meta-parameter $(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})$, and builds a task-specific model $(\boldsymbol{\phi}_{t,J}, \boldsymbol{\theta}_{t,J})$ by performing J gradient updates on the support set with step size $\alpha > 0$ and initialization $(\boldsymbol{\phi}_{t,0}, \boldsymbol{\theta}_{t,0}) \equiv (\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})$:

$$(\boldsymbol{\phi}_{t,j}, \boldsymbol{\theta}_{t,j}) = (\boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{t,j-1}) + \alpha \nabla_{(\boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{t,j-1})} \sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}_{t,j-1}, \boldsymbol{\theta}_{t,j-1}).$$

The meta-learner then updates the meta-initialization by maximizing the log-likelihood objective on the query set with step size $\eta > 0$:

$$(\boldsymbol{\phi}_t, \boldsymbol{\theta}_t) = (\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1}) + \eta \nabla_{(\boldsymbol{\phi}_{t-1}, \boldsymbol{\theta}_{t-1})} \sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}_{t,J}, \boldsymbol{\theta}_{t,J}).$$

Though MetaPrompting achieves state-of-the-art performance in the few-shot classification experiments [Hou et al., 2022], it suffers from three problems. (i) When the tasks are complex, it is challenging to obtain good prompts for all tasks and samples from a single meta-initialization. (ii) MetaPrompting uses a hand-crafted verbalizer. However, selecting good label tokens for the hand-crafted verbalizer is labor-intensive and not scalable for a large label set. (iii) MetaPrompting requires expensive tuning the whole MLM. Figure 4.1 shows the large gap in meta-testing accuracies with and without MLM tuning (Experimental details are in Section 4.1.4).

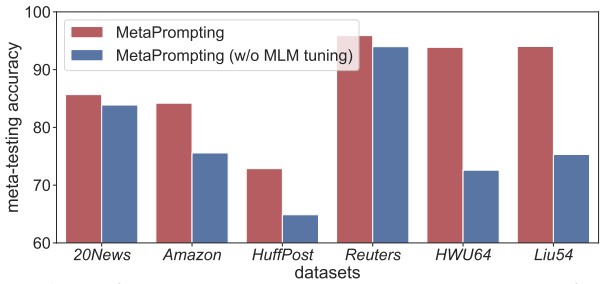


Figure 4.1: 5-way 5-shot classification meta-testing accuracy of MetaPrompting with or without MLM tuning on six data sets.

Algorithm 6 Representative Verbalizer (RepVerb).

procedure COMPUTE_LABEL_EMB($\mathcal{S}_\tau$):
    compute $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $(\mathbf{x}, \cdot) \in \mathcal{S}_\tau$;
    $\mathbf{v}_y = \frac{1}{|\mathcal{S}_{\tau,y}|} \sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}} \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $y \in \mathcal{Y}_\tau$;
end procedure
procedure PRED($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$)
    compute $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ for $\mathbf{x}$;
    $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{\exp(\rho \cos(\mathbf{v}_y, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}{\sum_{y'\in\mathcal{Y}_\tau} \exp(\rho \cos(\mathbf{v}_{y'}, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}$;
end procedure

4.1.3 The Proposed MetaPrompter

In this section, we propose a simple and effective soft verbalizer (representative verbalizer) without inducing additional parameters (Section 4.1.3.1). Moreover, while MetaPrompting uses a single meta-initialized prompt to build task-specific prompts, we propose in section 4.1.3.2 the extraction of task knowledge into a pool of multiple prompts, and constructs instance-dependent prompts by attention [Vaswani et al., 2017].

4.1.3.1 Representative Verbalizer (RepVerb)

Instead of explicitly learning an embedding $\mathbf{v}_y$ for each label y [Hambardzumyan et al., 2021, Cui et al., 2022, Zhang et al., 2022a], we propose the *Representative Verbalizer* (RepVerb), which constructs $\mathbf{v}_y$ from feature embeddings of the corresponding training samples (Algorithm 6). It does not require learning additional parameters, and is thus more effective on limited data as in few-shot learning.

Specifically, let $\mathcal{S}_{\tau,y}$ be the subset of samples in $\mathcal{S}_\tau$ with label y. For an input $\mathbf{x}$, we wrap it with the template and feed $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta})$ to the pre-trained MLM, and then obtain `[MASK]`'s embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ as its feature embedding. Similar to ProtoNet [Snell et al., 2017],

we propose to construct $\mathbf{v}_y$ for each y by averaging the corresponding samples' feature embeddings, as:

$$\mathbf{v}_y = \frac{1}{|\mathcal{S}_{\tau,y}|} \sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}} \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}). \tag{4.2}$$

To predict the label of a given $\mathbf{x}$, we measure the cosine similarity[2] between $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ and each $\mathbf{v}_y$ ($y \in \mathcal{Y}_\tau$):

$$\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta}) = \frac{\exp(\rho\cos(\mathbf{v}_y, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}{\sum_{y'\in\mathcal{Y}_\tau}\exp(\rho\cos(\mathbf{v}_{y'}, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}, \tag{4.3}$$

where $\rho > 0$ is the temperature. When $\rho \to \infty$, $\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta})$ becomes one-hot; whereas when $\rho \to 0$, $\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta})$ becomes uniform. In the experiments, we set $\rho = 10$ as in Oreshkin et al. [2018].

4.1.3.2 Meta Structured-Prompting

In the following, we propose the use of MAML and attention mechanism [Vaswani et al., 2017] to meta-learn a prompt pool. While MetaPrompting uses task-specific prompts [Hou et al., 2022], we propose the construction of instance-specific prompts, which allows more flexibility.

Meta-Learn a Prompt Pool

While MetaPrompting uses only a single initialization for the prompt, we propose to leverage a pool of prompts to extract more task knowledge, which is particularly effective when the tasks are complex. A prompt pool has K learnable prompts $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1,\ldots,K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p\times d_i}$ [Li et al., 2022a, Wang et al., 2022a,b]. Note that the size of the prompt pool is negligible compared with that of the MLM. For example, in our experiments, the MLM has 109.52×10^6 parameters, while the prompt pool has only $55,296$.

The prompt pool can be considered as shared meta-knowledge. Given an input $\mathbf{x}$, the attention weights between $\mathbf{x}$ and the K prompts are computed as $\mathbf{a} = \mathsf{softmax}(\frac{\mathbf{K}\mathbf{q}_\mathbf{x}}{\sqrt{d_o}})$, where $\mathbf{K} = [\mathbf{k}_1^\top;\ldots;\mathbf{k}_K^\top]$, and $\mathbf{q}_\mathbf{x} \in \mathbb{R}^{d_o}$ is the embedding of the `[MASK]` output by a pre-trained and

[2]Dissimilarity measures, such as the Euclidean distance, can also be used.

Algorithm 7 MetaPrompter.

Require: prompt length L_p; size of prompt pool K; $\lambda = 0.5$; step size α, η; meta-parameters $(\mathbf{K}, \mathbf{\Theta})$; query function $q(\cdot)$;

1: **for** $t = 1, \ldots, T$ **do**
2: sample a task $\tau = (\mathcal{S}_\tau, \mathcal{Q}_\tau) \in \mathcal{T}$;
3: *base learner:*
4: $(\mathbf{K}_{t,0}, \mathbf{\Theta}_{t,0}) \equiv (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})$;
5: **for** $j = 1, \ldots, J$ **do**
6: **for** $(\mathbf{x}, y) \in \mathcal{S}_\tau$ **do**
7: compute $\mathbf{q}_\mathbf{x}$ by $q(\cdot)$;
8: $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = \mathsf{softmax}(\mathbf{K}_{t,j-1}\mathbf{q}_\mathbf{x}/\sqrt{d_0})^\top \mathbf{\Theta}_{t,j-1}$;
9: feed $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ into $\mathcal{M}$, obtain $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$, and $\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ by (4.1);
10: **end for**
11: call COMPUTE_LABEL_EMB($\mathcal{S}_\tau$) of Algorithm 6 to obtain $\{\mathbf{v}_y : y \in \mathcal{Y}_\tau\}$;
12: for $(\mathbf{x}, y) \in \mathcal{S}_\tau$, call PRED($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$) of Algorithm 6 to obtain $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$, and compute $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$ by (4.5);
13: $\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = -\sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j})$;
14: $(\mathbf{K}_{t,j}, \mathbf{\Theta}_{t,j}) = (\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) - \alpha \nabla_{(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})} \mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})$;
15: **end for**
16: *meta-learner:*
17: **for** $(\mathbf{x}, y) \in \mathcal{Q}_\tau$ **do**
18: compute $\mathbf{q}_\mathbf{x}$ by $q(\cdot)$;
19: $\boldsymbol{\theta}_{\mathbf{x},J}(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = \mathsf{softmax}(\mathbf{K}_{t,J}\mathbf{q}_\mathbf{x}/\sqrt{d_0})^\top \mathbf{\Theta}_{t,J}$;
20: call PRED($\mathbf{x}; \mathbf{v}_y : y \in \mathcal{Y}_\tau$) of Algorithm 6 to obtain $\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$;
21: compute $\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ and $\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ by (4.1) and (4.5), respectively;
22: **end for**
23: $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = -\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$;
24: $(\mathbf{K}_t, \mathbf{\Theta}_t) = (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1}) - \eta \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$;
25: **end for**
26: **return** $(\mathbf{K}_T, \mathbf{\Theta}_T)$.

frozen MLM with the wrapped input (e.g., ($\mathbf{x}$. `Topic is [MASK]`)) [Wang et al., 2022a,b]. Such mapping from $\mathbf{x}$ to $\mathbf{q}_\mathbf{x}$ is called a query function $q(\cdot)$. An instance-dependent prompt is then generated by weighted averaging over all the values ($\boldsymbol{\theta}_i$'s):

$$\boldsymbol{\theta}_\mathbf{x}(\mathbf{K}, \mathbf{\Theta}) = \sum_{i=1}^{K} a_i \boldsymbol{\theta}_i, \tag{4.4}$$

where $\mathbf{\Theta} = [\boldsymbol{\theta}_1; \ldots; \boldsymbol{\theta}_K]$. While [Wang et al., 2022a,b] only selects the top-N most similar prompts from the pool, in (4.4) all the prompts are used and updated simultaneously.

The proposed procedure, which will be called MetaPrompter, is shown in Algorithm 7. At iteration t, the base learner takes $(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})$ and a task τ to optimize for a task-specific prompt pool by gradient descent (steps 4-15). $(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})$ is used as the initialization (step

4). For each inner iteration j, $(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})$ constructs the instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})$ in (4.4) (steps 7 and 8). Next, $\boldsymbol{\theta}_{\mathbf{x},j}$ is used to predict the label probability with a combination of the hand-crafted verbalizer (step 9) and soft verbalizer (steps 11 and 12):

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) = (1-\lambda)\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) + \lambda\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}), \tag{4.5}$$

where $\lambda \in [0,1]$. Let $\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) = -\sum_{(\mathbf{x},y)\in\mathcal{S}_\tau} \log \mathbb{P}\left(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}\right)$ be the loss on $\mathcal{S}_\tau$ (step 13). The base learner builds a task-specific prompt pool $(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ by taking J gradient updates ($j = 1, \ldots, J$) at step 14:

$$(\mathbf{K}_{t,j}, \mathbf{\Theta}_{t,j}) = (\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}) - \alpha\nabla_{(\mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1})}\mathcal{L}(\mathcal{S}_\tau; \mathbf{K}_{t,j-1}, \mathbf{\Theta}_{t,j-1}).$$

The meta-learner takes $(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ and $\mathcal{Q}_\tau$ to update the meta-parameters (steps 17-24). For $(\mathbf{x}, y) \in \mathcal{Q}_\tau$, we use $(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ to generate its prompt $\boldsymbol{\theta}_{\mathbf{x},J}(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ (steps 18 and 19), which is used for make prediction $\mathbb{P}\left(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J}\right)$ (steps 20 and 21). Let $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = -\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \log \mathbb{P}\left(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J}\right)$ be the negative log-likelihood loss on $\mathcal{Q}_\tau$ (step 23). The meta-learner updates meta-parameters by performing one gradient update on $\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ at step 24:

$$(\mathbf{K}_t, \mathbf{\Theta}_t) = (\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1}) - \eta\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})}\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}).$$

The meta-gradient

$$\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})}\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) = \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})}\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})}(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$$

requires back-propagating through the entire inner optimization path, which is computationally infeasible for large models and J is large. To reduce computation, we discard the second-order derivative and use the first-order approximation $\nabla_{(\mathbf{K}_{t-1}, \mathbf{\Theta}_{t-1})}\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J}) \approx \nabla_{(\mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})}\mathcal{L}(\mathcal{Q}_\tau; \mathbf{K}_{t,J}, \mathbf{\Theta}_{t,J})$ (step 24) as in [Finn et al., 2017, Hou et al., 2022].

Meta-Testing. Given an unseen task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$, the base learner takes $\mathcal{S}_{\tau'}$ and $(\mathbf{K}_T, \mathbf{\Theta}_T)$ to build a task-specific prompt pool $(\mathbf{K}_{T,J}, \mathbf{\Theta}_{T,J})$ as in steps 4-15. This pool is then used to construct instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},J}$ for each $(\mathbf{x}, \cdot) \in \mathcal{Q}_{\tau'}$. The MLM receives the wrapped input $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},J})$ and predicts the label probability by (4.5).

MetaPrompter is Parameter-Efficient. As MetaPrompter only tunes $(\mathbf{K}, \mathbf{\Theta})$, the total number of meta-parameters is $K(d_o + L_p d_i)$ (where d_i and d_o are the dimensions of the input

Table 4.1: Statistics of the data sets.

	#classes (meta-train/valid/test)	#samples	#tokens per sample (mean $\pm$ std)
20News	8/5/7	18,820	340 ± 151
Amazon	10/5/9	24,000	140 ± 32
HuffPost	20/5/16	36,900	11 ± 4
Reuters	15/5/11	620	168 ± 136
HWU64	23/16/25	11,036	7 ± 3
Liu54	18/18/18	25,478	8 ± 4

and feature embeddings, respectively). This is much smaller[3] than that of MetaPrompting (which is equal to $d_{\boldsymbol{\phi}} + L_p d_i$, where $d_{\boldsymbol{\phi}}$ is the size of $\boldsymbol{\phi}$), which requires tuning the whole MLM.

4.1.4 Experiments

4.1.4.1 Setup

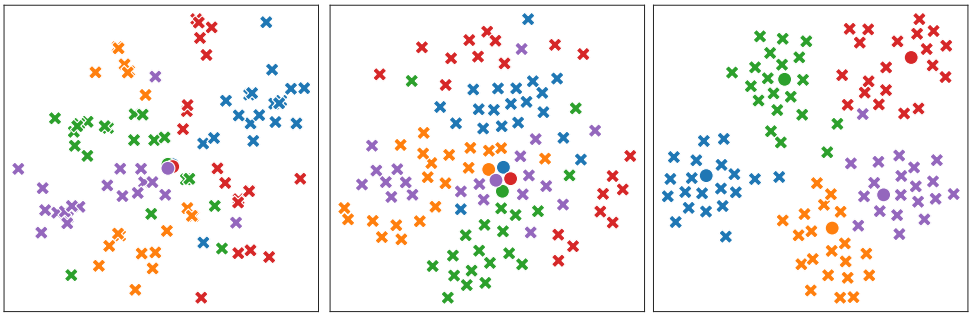
Data sets. Following [Chen et al., 2022a], we perform few-shot classification on six popularly used data sets: (i) *20News* [Lang, 1995], which contains informal discourses from news discussion forums of 20 topics; (ii) *Amazon* [He and McAuley, 2016] consists of customer reviews from 24 products. The task is to classify reviews into product categories; (iii) *HuffPost* [Misra, 2022], which contains news headlines of 41 topics published in the HuffPost between 2012 and 2018. These headlines are shorter and less grammatical than formal sentences, thus are more challenging for classification; (iv) *Reuters* [Lewis, 1997] is a collection of Reuters newswire articles of 31 topics from 1996 to 1997; (v) *HWU64* [Liu et al., 2019] is an intent classification data set, containing user utterances of 64 intents; (vi) *Liu54* [Liu et al., 2019] is an imbalanced intent classification data set of 54 classes collected on Amazon Mechanical Turk. We use the meta-training/meta-validation/meta-testing splits provided in [Chen et al., 2022a]. A summary of data sets is in Table 4.1.

Following [Bao et al., 2020, Han et al., 2021, Chen et al., 2022a, Hou et al., 2022], we perform experiments in the 5-way 1-shot and 5-way 5-shot settings with 15 query samples per class. The pre-trained BERT (*bert-base-uncased*) from HuggingFaces [Wolf et al., 2019] is used as the pre-trained MLM as [Chen et al., 2022a, Hou et al., 2022]. Experiments are run on a DGX station with 8 V100 32GB GPUs. The experiment is repeated three times with different random seeds.

[3]For example, $d_o = d_i = 768$, $d_{\boldsymbol{\phi}} = 109 \times 10^6$ in BERT. Moreover, Both K and L_p are 8 in the experiments.

Table 4.2: Meta-testing accuracy of 5-way few-shot classification.

		20News	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
5-shot	WARP [Hambardzumyan et al., 2021]	61.43 ± 0.15	59.53 ± 0.20	46.31 ± 0.31	68.67 ± 0.71	68.60 ± 0.40	73.11 ± 0.26
	ProtoVerb [Cui et al., 2022]	71.33 ± 0.11	71.74 ± 0.21	57.93 ± 0.17	80.93 ± 0.54	73.43 ± 0.51	76.19 ± 0.33
	RepVerb	$\mathbf{78.81} \pm 0.08$	$\mathbf{77.56} \pm 0.16$	$\mathbf{61.90} \pm 0.08$	$\mathbf{88.33} \pm 0.40$	$\mathbf{78.37} \pm 0.49$	$\mathbf{82.14} \pm 0.23$
1-shot	WARP [Hambardzumyan et al., 2021]	49.87 ± 0.63	48.94 ± 0.34	38.21 ± 0.35	52.88 ± 0.67	53.20 ± 0.76	58.68 ± 0.64
	ProtoVerb [Cui et al., 2022]	54.13 ± 0.46	55.07 ± 0.27	41.40 ± 0.21	57.27 ± 0.73	55.17 ± 0.81	60.16 ± 0.37
	RepVerb	$\mathbf{59.86} \pm 0.38$	$\mathbf{59.18} \pm 0.31$	$\mathbf{44.65} \pm 0.20$	$\mathbf{63.63} \pm 0.41$	$\mathbf{59.83} \pm 0.71$	$\mathbf{66.17} \pm 0.40$



(a) WARP. (b) ProtoVerb. (c) RepVerb.

Figure 4.2: t-SNE visualization of `[MASK]`'s embeddings (crosses) and label embeddings (circles) for a 5-way 5-shot task randomly sampled from *Reuters*.

4.1.4.2 Evaluation on RepVerb

First, we compare the performance of the proposed RepVerb with the state-of-the-art soft verbalizers of: (i) WARP [Hambardzumyan et al., 2021][4], and (ii) ProtoVerb [Cui et al., 2022]. As the focus is on evaluating verbalizers, all methods use the same discrete prompt "`Topic is [MASK]`", and fine-tune all parameters for 5 steps with a learning rate of 0.00005 as in [Cui et al., 2022].

Results. Table 4.2 reports the meta-testing accuracies. As can be seen, RepVerb outperforms WARP and ProtoVerb on both the 1-shot and 5-shot settings.

For a 5-way 5-shot task randomly from *Reuters*, Figure 4.2 shows the t-SNE visualization of the embeddings ($\mathbf{h}_{\texttt{[MASK]}}(\mathbf{x})$'s) of 100 samples ($\mathbf{x}$'s)[5] and learned label embeddings ($\mathbf{v}_y$'s). As can be seen, the RepVerb embedding is more discriminative and compact than WARP and ProtoVerb. Moreover, by design, RepVerb's label embedding is consistent with the samples' feature embeddings, while those of WARP and ProtoVerb are not.

[4]Note that the vebalizer of WARP is the same as that of DART [Zhang et al., 2022a].

[5]5-way $\times$ (5 support samples + 15 query samples) = 100.

4.1.4.3 Evaluation on MetaPrompter

For MetaPrompter, hyperparameters K and L_p are chosen from $\{1, 2, 4, 8, 16, 32, 64\}$ using the meta-validation set. For the base learner, $\alpha = 0.1$, and $J = 5$ (resp. 15) at meta-training (resp. meta-validation or meta-testing). We train the prompt pool for $T = 3,000$ iterations using the Adam optimizer [Kingma and Ba, 2015] with a learning rate of 0.001. To prevent overfitting, we evaluate the meta-validation performance every 100 iterations and choose the checkpoint with the best meta-validation performance for meta-testing. For the hard verbalizer, label tokens are obtained by tokenizing the class name and its synonyms as in [Hou et al., 2022, Hu et al., 2022]. Following [Lester et al., 2021], prompts are initialized from input embeddings of randomly sampled label tokens for both MetaPrompting and MetaPrompter.

We compare with a variety of baselines. These include state-of-the-art prompt-based methods of (i) MetaPrompting [Hou et al., 2022], and its variants (ii) MetaPrompting+WARP / MetaPrompting+ProtoVerb / MetaPrompting+RepVerb, which combine meta-prompting with the the soft verbalizer of WARP / ProtoVerb / RepVerb, respectively. Moreover, we also compare with the non-prompt-based methods of: (iii) HATT [Gao et al., 2019], which meta-learns a prototypical network [Snell et al., 2017] with a hybrid attention mechanism; (iv) DS [Bao et al., 2020], which learns attention scores based on word frequency; (v) MLADA [Han et al., 2021], which uses an adversarial domain adaptation network to extract domain-invariant features during meta-training; and (vi) ConstrastNet [Chen et al., 2022a], which performs feature extraction by contrastive learning.

Results. Table 4.3 shows the number of parameters and meta-testing accuracy in the 5-shot setting. As can be seen, MetaPrompter is more accurate than both prompt-based and non-prompt-based baselines. Moreover, since MetaPrompter only tunes the prompt pool and

Table 4.3: 5-way 5-shot classification meta-testing accuracy. Results marked with † are from [Chen et al., 2022a]. "–" indicates that the corresponding result is not reported in [Chen et al., 2022a].

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† [Gao et al., 2019]	0.07	55.00	66.00	56.30	56.20	-	-
DS† [Bao et al., 2020]	1.73	68.30	81.10	63.50	96.00	-	-
MLADA† [Han et al., 2021]	0.73	77.80	86.00	64.90	96.70	-	-
ConstrastNet† [Chen et al., 2022a]	109.52	71.74	85.17	65.32	95.33	92.57	93.72
MetaPrompting [Hou et al., 2022]	109.52	85.67 ± 0.44	84.19 ± 0.30	72.85 ± 1.01	95.89 ± 0.23	93.86 ± 0.97	94.01 ± 0.26
MetaPrompting+WARP	109.52	85.81 ± 0.48	85.54 ± 0.20	71.71 ± 0.72	97.28 ± 0.30	93.99 ± 0.76	94.33 ± 0.27
MetaPrompting+ProtoVerb	109.52	86.18 ± 0.51	84.91 ± 0.38	73.11 ± 0.80	97.24 ± 0.25	93.81 ± 0.81	94.38 ± 0.18
MetaPrompting+RepVerb	109.52	86.89 ± 0.39	85.98 ± 0.28	74.62 ± 0.88	97.32 ± 0.31	94.23 ± 0.67	94.45 ± 0.33
MetaPrompter	0.06	$\mathbf{88.57} \pm 0.38$	$\mathbf{86.36} \pm 0.24$	$\mathbf{74.89} \pm 0.75$	$\mathbf{97.63} \pm 0.22$	$\mathbf{95.30} \pm 0.51$	$\mathbf{95.47} \pm 0.21$

Table 4.4: 5-way 1-shot classification meta-testing accuracy. Results marked with † are from [Chen et al., 2022a]. "–" indicates that the corresponding result is not reported in [Chen et al., 2022a].

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† [Gao et al., 2019]	0.07	44.20	49.10	41.10	43.20	-	-
DS† [Bao et al., 2020]	1.73	52.10	62.60	43.00	81.80	-	-
MLADA† [Han et al., 2021]	0.73	59.60	68.40	64.90	82.30	-	-
ConstrastNet† [Chen et al., 2022a]	109.52	71.74	76.13	53.06	86.42	86.56	85.89
MetaPrompting [Hou et al., 2022]	109.52	82.46 ± 0.50	76.92 ± 0.77	68.62 ± 0.56	92.56 ± 0.77	91.06 ± 0.41	87.79 ± 0.29
MetaPrompting +WARP	109.52	82.93 ± 0.39	78.27 ± 0.72	67.78 ± 0.41	94.74 ± 0.56	91.30 ± 0.35	88.69 ± 0.26
MetaPrompting+ProtoVerb	109.52	83.15 ± 0.41	78.19 ± 0.65	68.96 ± 0.52	95.26 ± 0.40	91.27 ± 0.63	90.05 ± 0.15
MetaPrompting+RepVerb	109.52	84.13 ± 0.30	78.59 ± 0.43	$\mathbf{69.02} \pm 0.51$	95.78 ± 0.33	91.32 ± 0.44	90.13 ± 0.20
MetaPrompter-PE	0.06	$\mathbf{84.62} \pm 0.29$	$\mathbf{79.05} \pm 0.21$	67.12 ± 0.23	$\mathbf{96.34} \pm 0.20$	$\mathbf{92.11} \pm 0.30$	$\mathbf{93.72} \pm 0.18$

keeps the language model frozen, it has much fewer meta-parameters than MetaPrompting and ConstrastNet.

Furthermore, MetaPrompting+RepVerb performs better than MetaPrompting+WARP and MetaPrompting+ProtoVerb, demonstrating that the proposed RepVerb is beneficial to MetaPrompting.

Table 4.4 shows the number of parameters and meta-testing accuracy in 5-way 1-shot setting. As can be seen, the state-of-the-art prompt-based methods always achieve higher accuracy than the non-prompt-based. Furthermore, MetaPrompter performs the best on 5 of the 6 data sets. Besides, RepVerb is again useful to MetaPrompting on all six data sets.

4.1.4.4 Visualization

In this section, we visualize the meta-knowledge in the prompt pool learned from the 5-way 5-shot classification task on *Reuters*. Table 4.5 shows the nearest tokens to each of the K ($= 8$) learned prompts. Figure 4.3 shows the average attention weights between the

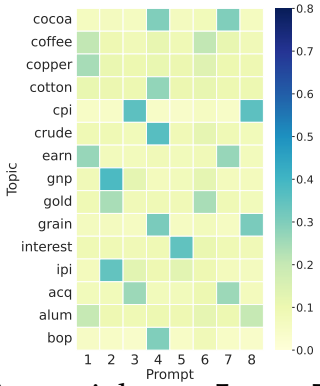


Figure 4.3: Distribution of attention weights on 5-way 5-shot classification of *Reuters* (15 topics).

Table 4.5: Nearest tokens to the learned prompts for *Reuters*.

prompt id	nearest tokens
1	copper, steel, trading, gas, fx, aluminum, earn, coffee
2	gross, ship, index, money, gold, tin, iron, retail
3	product, cpi, industrial, acquisitions, jobs, supplying, orange, sugar
4	cocoa, production, grain, livestock, wholesale, cotton, bop, crude
5	oil, national, rubber, nat, interest, price, reserves, regional
6	nat, wholesale, sugar, golden, reserves, drinks, production, product
7	chocolate, sugar, cheat, orange, trade, fx, cash, acquiring
8	aluminum, livestock, cpc, tin, shops, wheat, petrol, supply

K prompts and meta-training samples belonging to class (topic) y:

$$\frac{1}{|\mathcal{T}_y|}\sum_{\tau\in\mathcal{T}_y}\frac{1}{|\mathcal{S}_{\tau,y}|}\sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}}\mathsf{softmax}\left(\frac{\mathbf{K}_{T,J}\mathbf{q}_{\mathbf{x}}}{\sqrt{d_o}}\right),$$

where $\mathcal{T}_y$ is the subset of tasks in $\mathcal{T}$ having class y. As can be seen, samples from each target class prefer prompts whose tokens are related to that class. For example, samples from the topic *cocoa* tend to use the 4th and 7th prompts (whose tokens are close to words like *cocoa*, *chocolate* as can be seen from Table 4.5), while samples from the topic *coffee* tend to use the 1st and 6th prompts (whose tokens are close to words like *coffee* and *sugar*.

Recall that the prompt pool has K learnable prompts $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1, \ldots, K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p \times d_i}$. Let $\boldsymbol{\theta}_i^{(j)}$ be the jth row of $\boldsymbol{\theta}_i$. Moreover, let $\frac{1}{|\mathcal{V}_y|}\sum_{\mathtt{w}\in\mathcal{V}_y}\mathcal{E}(\mathtt{w})$ be the embedding of topic (class) y, where $\mathcal{V}_y$ is a set of tokens relevant to label y (obtained from Hou et al. [2022]), and $\mathcal{E}(\cdot)$ is the input embedding. Figure 4.4 shows the cosine similarities between the learned prompt tokens $\{\boldsymbol{\theta}_i^{(j)} : i = 1, \ldots, K, j = 1 \ldots, L_p\}$ and topic embeddings. As can be seen, embedding of *cocoa* is close to $\boldsymbol{\theta}_4^{(1)}$ and $\boldsymbol{\theta}_7^{(1)}$. Thus, samples from *cocoa* prefer the 4th and 7th prompts (Figure 4.3). Similarly, embedding of *coffee* is close to $\boldsymbol{\theta}_1^{(8)}$ and $\boldsymbol{\theta}_6^{(6)}$. Thus, samples from *coffee* prefer the 1st and 6th prompts (Figure 4.3).

4.2 Application in In-Context Learning

4.2.1 In-Context Learning

In-Context Learning (ICL) [Min et al., 2022, Chen et al., 2022b, Dong et al., 2022] uses a pre-trained MLM to learn a new task by formatting training examples as demonstration.

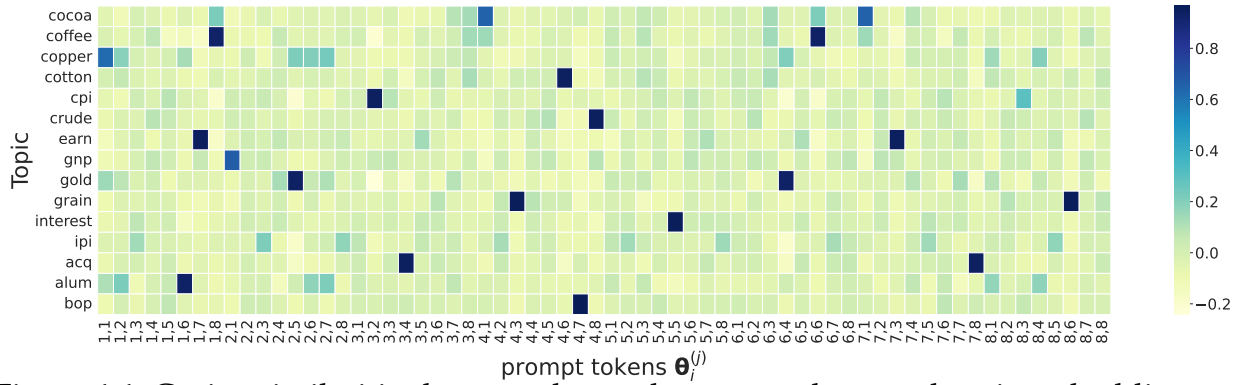


Figure 4.4: Cosine similarities between learned prompt tokens and topic embeddings on 5-way 5-shot classification of *Reuters*. In the x-axis, (i, j) stands for the jth row of $\boldsymbol{\theta}_i$ (i.e., $\boldsymbol{\theta}_i^{(j)}$)

For example, given a query input $\mathbf{x}$ and a training set $\mathcal{S}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n_s\}$, ICL concentrates all training samples as:

$$\begin{aligned}
\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \mathcal{S}_\tau) = \text{``}&\texttt{Question:} \quad \mathbf{x}_1? \\
&\texttt{Answer:} \quad y_1. \\
&\quad\vdots \qquad\quad \vdots \\
&\texttt{Question:} \quad \mathbf{x}_{n_s}? \\
&\texttt{Answer:} \quad y_{n_s}. \\
&\texttt{Question:} \quad \mathbf{x}? \\
&\texttt{Answer:} \quad \text{''}
\end{aligned} \tag{4.6}$$

A pre-trained LLM $\mathcal{M}(\cdot; \boldsymbol{\phi})$ receives $\mathbb{T}(\mathbf{x}; \mathcal{S}_\tau)$ and generates an answer $\hat{y}$ for the testing question $\mathbf{x}$. The task objective is

$$\mathcal{L}(\mathcal{Q}_\tau; \mathcal{S}_\tau, \boldsymbol{\phi}) = \sum_{(\mathbf{x}, y) \in \mathcal{Q}_\tau} \mathbb{P}_{\mathcal{M}(\cdot; \boldsymbol{\phi})}(y | \mathbb{T}(\mathbf{x}; \mathcal{S}_\tau)). \tag{4.7}$$

Though LLMs have shown the ability of in-context learning, their pretraining procedure is not designed for in-context learning. Recently, MetaICL [Min et al., 2022] aims to finetune LLM such that it can be more suitable for ICL tasks, i.e.,

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \frac{\eta}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \nabla_{\boldsymbol{\phi}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \mathcal{S}_\tau, \boldsymbol{\phi}_{t-1}), \; t = 1, \ldots, T, \tag{4.8}$$

where $\mathcal{B}_t$ is a batch of tasks at iteration t. At meta-testing, given an unseen task $\tau' = (\mathcal{S}_{\tau'}, \mathcal{Q}_{\tau'})$, MetaICL takes $\boldsymbol{\phi}_T$ and $\mathcal{S}_{\tau'}$ to generate the answer for the query question $\mathbf{x}$ as

$$\hat{y} = \arg\max_y \mathbb{P}_{\mathcal{M}(\cdot; \boldsymbol{\phi}_T)}(y | \mathbb{T}(\mathbf{x}; \mathcal{S}_{\tau'})). \tag{4.9}$$

Though ICL has shown promising performance, it faces two challenges. (i) Organizing demonstration is crucial, but manually-designing requires expertise knowledge and intensive labor. (ii) At meta-testing, MetaICL does not require to adapt the MLM on support set, which saves computation but significantly sacrifices performance (Table 5 in [Min et al., 2022]).

4.2.2 Demonstration Organization

Demonstration organization [Dong et al., 2022] consists of three components: (i) demonstration selection h_{sel}; (ii) demonstration ordering h_{ord}; (iii) demonstration instruction h_{ins}. In the above example (4.6), all examples are selected, ordered by the sample index with the instruction "`Question:` $\mathbf{x}$`?` `Answer:` y". Let $\boldsymbol{\Theta} \equiv (\boldsymbol{\theta}_s, \boldsymbol{\theta}_o, \boldsymbol{\theta}_i)$ denote parameters in $(h_{\text{sel}}, h_{\text{ord}}, h_{\text{ins}})$. The wrapped input is $\tilde{\mathbf{x}} \equiv \mathbb{T}(\mathbf{x}; \mathcal{S}_\tau, \boldsymbol{\Theta})$ parameterized by $\boldsymbol{\Theta}$.

Demonstration selection aims to seek good samples for ICL. For example, unsupervised methods select the closest neighbors [Liu et al., 2022b] or diverse samples [Levy et al., 2022] for demonstrating. Supervised methods are based on a two-stage (recall-select) method [Rubin et al., 2022, Ye et al., 2023, Li et al., 2023] or reinforcement learning (action is to select a sample and reward is validation performance) [Zhang et al., 2022b].

Demonstration ordering is sensitive to the performance of ICL [Lu et al., 2022]. To deal with this problem, recent studies sort examples by their L_2 distance [Liu et al., 2022b] or entropy [Lu et al., 2022] to the query input.

Demonstration instruction also plays an important role in ICL. A well-designed instruction can be more effective than simply concatenating examples, and, various instructions have been designed by experts [Sanh et al., 2022].

4.2.3 Meta-learning for Organizing Demonstrations

We introduce some research directions to enhance ICL by using meta-learning for better demonstration organization. Different from MetaICL [Min et al., 2022] and ICT [Chen et al., 2022b], which focus on fine-tuning the MLM, we aim to meta-learn how to organize demonstrations. The optimization problem is as follows:

$$(\boldsymbol{\phi}_t, \boldsymbol{\Theta}_t) = (\boldsymbol{\phi}_{t-1}, \boldsymbol{\Theta}_{t-1}) - \frac{\eta}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \nabla_{(\boldsymbol{\phi}_{t-1}, \boldsymbol{\Theta}_{t-1})} \mathcal{L}(\mathcal{Q}_\tau; \boldsymbol{\phi}_{t-1}, \mathbb{T}(\cdot; \mathcal{S}_\tau, \boldsymbol{\Theta}_{t-1})). \tag{4.10}$$

Using meta-learning in demonstration organization faces new challenges.

(i) Demonstration selection/ordering/instruction are *discrete* operators. Hence, the gradients in (4.10) can not be computed directly by auto-differentiation. Existing available tools are reparameterization [Goodfellow et al., 2014, Jang et al., 2017] and score function [Sutton et al., 1999, Sutton and Barto, 2018], but they are *data-inefficient*.

(ii) Selection/ordering/instruction operators are applied in sequel, and affect each other. Hence, optimizing all operators together is challenging.

(iii) Current SOTA ICL methods [Rubin et al., 2022, Ye et al., 2023, Li et al., 2023] require a large amount of labeled samples for training the retriever, which is labor-intensive.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTION

5.1 Conclusion

This survey provides an overview of meta-learning, including three categories. Chapter 2 reviews (i) *Optimization-based* methods (meta-initialization and meta-regularization), which are preferable due to their simplicity and effectiveness; (ii) *Metric-based* methods that designed for few-shot classification; and (iii) *Memory-based* methods including MANN and HyperNetworks. In Chapter 3, we propose a novel algorithm MetaProx to introduce non-linearity to meta-regularization by kernelized proximal regularization, which is empirically shown to be efficient and effectively. We discuss the applications of meta-learning in large language models in Chapter 4, including prompting and in-context learning. For prompting, we propose MetaPrompter to improve the effectiveness and parameter-efficiency of prompt tuning.

5.2 Future Directions

Meta-learning for prompting and in-context learning (ICL) in LLMs. In Section 4.1, we proposed MetaPrompter to improve MetaPrompting by learning a prompt pool. As topics (labels) usually have *a hierarchical structure* [Yang et al., 2016, Angelov, 2020], one possible direction is to leverage this structure in the prompt pool. Section 4.2 introduces the recent state-of-the-art ICL for NLP. Demonstration organization is crucial for ICL, which consists of demonstration selection/ordering/instruction. Using meta-learning to improve ICL is a prospective direction, but also challenging as all three operators are *discrete* and gradient-based algorithms can not be applied directly. In meta-learning, this issue is more severe due to the bilevel structure.

Editing meta-knowledge. LLMs (e.g., LLAMA [Touvron et al., 2023], GPT-3.5 [OpenAI, 2022], GPT-4 [OpenAI, 2023]) achieve great success in many unseen tasks, which suggests that LLMs have a lot of meta-knowledge. However, as some training data include toxic language and historical biases along race and gender, LLMs may contain harmful

meta-knowledge leading to discriminatory, exclusionary, hateful outputs. Furthermore, LLMs may learn social biases along race, gender, sexual orientation, language, cultural. One future research direction is finding, editing, and removing harmful meta-knowledge (hidden embeddings or parameters) in LLMs.

Bibliography

A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter. Differentiable convex optimization layers. In *Neural Information Processing Systems*, 2019.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning. In *Neural Information Processing Systems*, 2022.

Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. Infinite mixture prototypes for few-shot learning. In *International Conference on Machine Learning*, 2019.

Dimo Angelov. Top2vec: Distributed representations of topics. Preprint arXiv:2008.09470, 2020.

Sudarshan Babu, Pedro Savarese, and Michael Maire. Online meta-learning via learning with layer-distributed memory. In *Neural Information Processing Systems*, 2021.

Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. In *Neural Information Processing Systems*, 2020.

Yujia Bao, Menghua Wu, Shiyu Chang, and Regina Barzilay. Few-shot text classification with distributional signatures. In *International Conference on Learning Representations*, 2020.

Sergey Bartunov, Jack Rae, Simon Osindero, and Timothy Lillicrap. Meta-learning deep energy-based memory models. In *International Conference on Learning Representations*, 2020.

Jacob Beck, Matthew Thomas Jackson, Risto Vuorio, and Shimon Whiteson. Hypernetworks in meta-reinforcement learning. In *Annual Conference on Robot Learning*, 2022.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. Learning a synaptic learning rule. In *International Joint Conference on Neural Networks*, 1991.

Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, 2006.

Marc Brockschmidt. GNN-FiLM: Graph neural networks with feature-wise linear modulation. In *International Conference on Machine Learning*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Neural Information Processing Systems*, 2020.

Paul H Calamai and Jorge J Moré. Projected gradient methods for linearly constrained problems. *Mathematical Programming*, 1987.

Jiaxin Chen, Li-Ming Zhan, Xiao-Ming Wu, and Fu-lai Chung. Variational metric scaling for metric-based meta-learning. In *AAAI Conference on Artificial Intelligence*, 2020a.

Junfan Chen, Richong Zhang, Yongyi Mao, and Jie Xu. ContrastNet: A contrastive learning framework for few-shot text classification. In *AAAI Conference on Artificial Intelligence*, 2022a.

Qi Chen, Changjian Shui, and Mario Marchand. Generalization bounds for meta-learning: An information-theoretic analysis. In *Neural Information Processing Systems*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, 2020b.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2018a.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. Meta-learning via language model in-context tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2022b.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, 2018b.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Workshop of Syntax, Semantics and Structure in Statistical Translation*, 2014.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Neural Information Processing Systems*, 2019.

Ganqu Cui, Shengding Hu, Ning Ding, Longtao Huang, and Zhiyuan Liu. Prototypical verbalizer for prompt-based few-shot tuning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In *Neural Information Processing Systems*, 2018.

Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *International Conference on Machine Learning*, 2019.

Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. The advantage of conditional meta-learning for biased regularization and fine tuning. In *Neural Information Processing Systems*, 2020.

Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. Conditional meta-learning of linear representations. In *Neural Information Processing Systems*, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. OpenPrompt: An open-source framework for prompt-learning. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*, 2014.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Neural Information Processing Systems*, 2019.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. A survey for in-context learning. Preprint arXiv:2301.00234, 2022.

Alexey Dosovitskiy and Josip Djolonga. You only train once: Loss-conditional training of deep networks. In *International Conference on Learning Representations*, 2020.

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. In *International Conference on Learning Representations*, 2021.

Benjamin Ehret, Christian Henning, Maria Cervera, Alexander Meulemans, Johannes Von Oswald, and Benjamin F Grewe. Continual learning in recurrent neural networks. In *International Conference on Learning Representations*, 2021.

Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2020.

Alec Farid and Anirudha Majumdar. Generalization bounds for meta-learning via PAC-Bayes and uniform stability. In *Neural Information Processing Systems*, 2021.

Hongliang Fei and Ping Li. Cross-lingual unsupervised sentiment classification with multi-view transfer learning. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 2017.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International Conference on Machine Learning*, 2019.

Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. In *International Conference on Learning Representations*, 2020.

Sebastian Flennerhag, Yannick Schroecker, Tom Zahavy, Hado van Hasselt, David Silver, and Satinder Singh. Bootstrapped meta-learning. In *International Conference on Learning Representations*, 2022.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, 2018.

Tomer Galanti and Lior Wolf. On the modularity of hypernetworks. In *Neural Information Processing Systems*, 2020.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. In *International Conference on Machine Learning*, 2019.

Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *AAAI Conference on Artificial Intelligence*, 2019.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *International Conference on Machine Learning*, 2018.

Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Shaogang Gong, Stephen McKenna, and John J Collins. An investigation into face pose distributions. In *International Conference on Automatic Face and Gesture Recognition*, 1996.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014.

Junliang Guo, Linli Xu, and Enhong Chen. Jointly masked sequence-to-sequence model for non-autoregressive neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, 2020.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. SpotTune: transfer learning through adaptive fine-tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level adversarial reprogramming. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Chengcheng Han, Zeqiu Fan, Dongxiang Zhang, Minghui Qiu, Ming Gao, and Aoying Zhou. Meta-learning adversarial domain adaptation network for few-shot text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

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

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *International Conference on World Wide Web*, 2016.

Markus Hiller, Mehrtash Harandi, and Tom Drummond. On enforcing better conditioned meta-learning for rapid few-shot adaptation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Neural Information Processing Systems*, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

Yutai Hou, Hongyuan Dong, Xinghao Wang, Bohan Li, and Wanxiang Che. MetaPrompting: Learning to learn better prompts. In *International Conference on Computational Linguistics*, 2022.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2018.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Neural Information Processing Systems*, 2018.

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

Adrián Javaloy and Isabel Valera. RotoGrad: Gradient homogenization in multitask learning. In *International Conference on Learning Representations*, 2021.

Ghassen Jerfel, Erin Grant, Tom Griffiths, and Katherine A Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. In *Neural Information Processing Systems*, 2019.

Kaiyi Ji, Jason D Lee, Yingbin Liang, and H Vincent Poor. Convergence of meta-learning with task-specific adaptation over partial parameters. In *Neural Information Processing Systems*, 2020a.

Kaiyi Ji, Junjie Yang, and Yingbin Liang. Multi-step model-agnostic meta-learning: Convergence and improved algorithms. Preprint arXiv:2002.07836, 2020b.

Weisen Jiang, James Kwok, and Yu Zhang. Effective meta-regularization by kernelized proximal regularization. In *Neural Information Processing Systems*, 2021a.

Weisen Jiang, Yu Zhang, and James T Kwok. SEEN: Few-shot classification with self-ensemble. In *International Joint Conference on Neural Networks*, 2021b.

Weisen Jiang, Yu Zhang, and James Kwok. Effective structured-prompting by meta-learning and representitive verbalizer. In *International Conference on Machine Learning*, 2023.

Xiang Jiang, Mohammad Havaei, Farshid Varno, Gabriel Chartrand, Nicolas Chapados, and Stan Matwin. Learning to learn with conditional class dependencies. In *International Conference on Learning Representations*, 2018.

Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *IEEE International Conference on Computer Vision*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Boris Knyazev, Michal Drozdzal, Graham W Taylor, and Adriana Romero Soriano. Parameter prediction for unseen deep architectures. In *Neural Information Processing Systems*, 2021.

Weihao Kong, Raghav Somani, Zhao Song, Sham Kakade, and Sewoong Oh. Meta-learning for mixed linear regression. In *International Conference on Machine Learning*, 2020.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 2017.

Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 1951.

Ken Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of International Machine Learning Conference*, 1995.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Neural Information Processing Systems*, 2019a.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019b.

Sanghyuk Lee, Seunghyun Lee, and Byung Cheol Song. Contextual gradient scaling for few-shot learning. In *IEEE Winter Conference on Applications of Computer Vision*, 2022.

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, 2018.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Empirical Methods in Natural Language Processing*, 2021.

Itay Levy, Ben Bogin, and Jonathan Berant. Diverse demonstrations improve in-context compositional generalization. Preprint arXiv:2212.06800, 2022.

D. Lewis. Reuters-21578 text categorization test collection. *Distribution 1.0, AT&T Labs-Research*, 1997.

Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Xin Zhao. Learning to transfer prompts for text generation. In *North American Chapter of the Association for Computational Linguistics*, 2022a.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. Unified demonstration retriever for in-context learning. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

Yuyang Li, Zhenzhenand Zhang, Jian-Yun Nie, and Dongsheng Li. Improving few-shot relation classification by prototypical representation learning with definition text. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2022b.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. Preprint arXiv:1707.09835, 2017.

Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 2007.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Neural Information Processing Systems*, 2021a.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022a.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out*, 2022b.

Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. In *European Conference on Computer Vision*, 2020.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. GPT understands, too. Preprint arXiv:2103.10385, 2021b.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*, 2022c.

Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. Benchmarking natural language understanding services for building conversational agents. In *International Workshop on Spoken Dialogue Systems Technology*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Su Lu, Han-Jia Ye, Le Gan, and De-Chuan Zhan. Towards enabling meta-learning from target models. In *Neural Information Processing Systems*, 2021.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. In *Annual Meeting of the Association for Computational Linguistics*, 2016.

Andreas Maurer and Tommi Jaakkola. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 2005.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In *North American Chapter of the Association for Computational Linguistics*, 2022.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

Rishabh Misra. News category dataset. Preprint arXiv:2209.11429, 2022.

Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, 2017.

Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *International conference on machine learning*, 2018.

Aviv Navon, Aviv Shamsian, Ethan Fetaya, and Gal Chechik. Learning the pareto front with hypernetworks. In *International Conference on Learning Representations*, 2021.

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. Learning to learn with gaussian processes. In *Uncertainty in Artificial Intelligence*, 2021.

Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. Preprint arXiv:1803.02999, 2018.

Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. BOIL: Towards representation change for few-shot learning. In *International Conference on Learning Representations*, 2021.

Sora Ohashi, Junya Takayama, Tomoyuki Kajiwara, and Yuki Arase. Distinct label representations for few-shot text classification. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

OpenAI. GPT-3.5. Technical Report, 2022.

OpenAI. GPT-4. Technical Report, 2023.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. TADAM: Task dependent adaptive metric for improved few-shot learning. In *Neural Information Processing Systems*, 2018.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 2009.

Eunbyung Park and Junier B Oliva. Meta-Curvature. In *Neural Information Processing Systems*, 2019.

Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O'Boyle, and Amos J Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In *Neural Information Processing Systems*, 2020.

Anastasia Pentina and Christoph Lampert. A PAC-Bayesian bound for lifelong learning. In *International Conference on Machine Learning*, 2014.

Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *Neural Information Processing Systems*, 2015.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.

Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. Incremental few-shot object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning*, 2017.

Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Transfer learning for image classification with sparse prototype representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.

Jack Rae, Jonathan J Hunt, Ivo Danihelka, Timothy Harley, Andrew W Senior, Gregory Wayne, Alex Graves, and Timothy Lillicrap. Scaling memory-augmented neural networks with sparse reads and writes. In *Neural Information Processing Systems*, 2016.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

Janarthanan Rajendran, Alexander Irpan, and Eric Jang. Meta-learning requires meta-augmentation. In *Neural Information Processing Systems*, 2020.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Neural Information Processing Systems*, 2019.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, 2019.

Tiago Ramalho and Marta Garnelo. Adaptive posterior learning: few-shot learning with a surprise-based memory module. In *International Conference on Learning Representations*, 2019.

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

Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International Conference on Machine Learning*, 2016.

Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. PACOH: Bayes-optimal meta-learning with pac-guarantees. In *International Conference on Machine Learning*, 2021a.

Jonas Rothfuss, Dominique Heyn, Andreas Krause, et al. Meta-learning reliable priors in the function space. In *Neural Information Processing Systems*, 2021b.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning. In *North American Chapter of the Association for Computational Linguistics*, 2022.

Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976.

Walter Rudin. *Fourier Analysis on Groups*. Courier Dover Publications, 2017.

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*, 115(3):211–252, 2015.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *International Conference on Learning Representations*, 2019.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*, 2022.

Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning*, 2016.

Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *European Chapter of the Association for Computational Linguistics*, 2021.

Marcin Sendera, Marcin Przewięźlikowski, Konrad Karanowski, Maciej Zięba, Jacek Tabor, and Przemysław Spurek. Hypershot: Few-shot learning by kernel hypernetworks. In *IEEE Winter Conference on Applications of Computer Vision*, 2023.

Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*, 2021.

Zhiqiang Shen, Zechun Liu, Jie Qin, Marios Savvides, and Kwang-Ting Cheng. Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In *AAAI Conference on Artificial Intelligence*, 2021.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Empirical Methods in Natural Language Processing*, 2020.

Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. On modulating the gradient for meta-learning. In *European Conference on Computer Vision*, 2020.

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

Vincent Sitzmann, Eric Chan, Richard Tucker, Noah Snavely, and Gordon Wetzstein. MetaSDF: Meta-learning signed distance functions. In *Neural Information Processing Systems*, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Neural Information Processing Systems*, 2017.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and permuted pre-training for language understanding. In *Neural Information Processing Systems*, 2020.

Joseph Suarez. Language modeling with recurrent highway hypernetworks. In *Neural Information Processing Systems*, 2017.

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 *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Neural Information Processing Systems*, 1999.

Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 1997.

Jihoon Tack, Jongjin Park, Hankook Lee, Jaeho Lee, and Jinwoo Shin. Meta-learning with self-improving momentum target. In *Neural Information Processing Systems*, 2022.

Swee Chuan Tan and Jess Pei San Lau. Time series clustering: A superior alternative for market basket analysis. In *International Conference on Advanced Data and Information Engineering*, 2014.

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*. 1998.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *Neural Information Processing Systems*, 2020.

Michalis K Titsias, Francisco JR Ruiz, Sotirios Nikoloutsopoulos, and Alexandre Galashov. Information theoretic meta learning with gaussian processes. In *Uncertainty in Artificial Intelligence*, 2021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. Preprint arXiv:2307.09288, 2023.

Nilesh Tripuraneni, Chi Jin, and Michael Jordan. Provable meta-learning of linear representations. In *International Conference on Machine Learning*, 2021.

Maria Tsimpoukelli, Jacob L Menick, Serkan Cabi, SM Eslami, Oriol Vinyals, and Felix Hill. Multimodal few-shot learning with frozen language models. *Neural Information Processing Systems*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, and Daan Wierstra. Matching networks for one shot learning. In *Neural Information Processing Systems*, 2016.

Haoxiang Wang, Ruoyu Sun, and Bo Li. Global convergence and induced kernels of gradient-based meta-learning with neural nets. Preprint arXiv:2006.14606, 2020a.

Lingxiao Wang, Qi Cai, Zhuoran Yang, and Zhaoran Wang. On the global optimality of model-agnostic meta-learning. In *International Conference on Machine Learning*, 2020b.

Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. Structured prediction for conditional meta-learning. In *Neural Information Processing Systems*, 2020c.

Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM Computing Surveys*, 53(3):1–34, 2020d.

Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *IEEE International Conference on Computer Vision*, 2019.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. DualPrompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2022b.

Robert WM Wedderburn. Quasi-likelihood functions, generalized linear models, and the gauss—newton method. *Biometrika*, 1974.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *International Conference on Artificial Intelligence and Statistics*, 2016.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. Preprint arXiv:1910.03771, 2019.

Yan Wu, Gregory Wayne, Karol Gregor, and Timothy Lillicrap. Learning attractor dynamics for generative memory. In *Neural Information Processing Systems*, 2018.

Chen Xing, Negar Rostamzadeh, Boris N. Oreshkin, and Pedro O. Pinheiro. Adaptive cross-modal few-shot learning. In *Neural Information Processing Systems*, 2019.

Wenhan Xiong, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. One-shot relational learning for knowledge graphs. In *Empirical Methods in Natural Language Processing*, 2018.

Jin Xu, Jean-Francois Ton, Hyunjik Kim, Adam Kosiorek, and Yee Whye Teh. MetaFun: Meta-learning with iterative functional updates. In *International Conference on Machine Learning*, 2020.

Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta R-CNN: Towards general solver for instance-level low-shot learning. In *IEEE International Conference on Computer Vision*, 2019.

Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer learning*. Cambridge University Press, 2020.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*, 2019.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *North American Chapter of the Association for Computational Linguistics*, 2016.

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *International Conference on Machine Learning*, 2019.

Huaxiu Yao, Xian Wu, Zhiqiang Tao, Yaliang Li, Bolin Ding, Ruirui Li, and Zhenhui Li. Automated relational meta-learning. In *International Conference on Learning Representations*, 2020.

Huaxiu Yao, Ying-xin Wu, Maruan Al-Shedivat, and Eric Xing. Knowledge-aware meta-learning for low-resource text classification. In *Empirical Methods in Natural Language Processing*, 2021.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. Compositional exemplars for in-context learning. In *International Conference on Machine Learning*, 2023.

Qinyuan Ye and Xiang Ren. Learning to generate task-specific adapters from task description. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

Zhi-Xiu Ye and Zhen-Hua Ling. Multi-level matching and aggregation network for few-shot relation classification. In *Annual Meeting of the Association for Computational Linguistics*, 2019.

Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Neural Information Processing Systems*, 2018.

Matthew D Zeiler. AdaDelta: an adaptive learning rate method. Preprint arXiv:1212.5701, 2012.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. Differentiable prompt makes pre-trained language models better few-shot learners. In *International Conference on Learning Representations*, 2022a.

Yiming Zhang, Shi Feng, and Chenhao Tan. Active example selection for in-context learning. In *Empirical Methods in Natural Language Processing*, 2022b.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Uncertainty in Artificial Intelligence*, 2010.

Dominic Zhao, Johannes von Oswald, Seijin Kobayashi, João Sacramento, and Benjamin F Grewe. Meta-learning via hypernetworks. In *Neural Information Processing Systems Workshop*, 2020.

Xiantong Zhen, Haoliang Sun, Yingjun Du, Jun Xu, Yilong Yin, Ling Shao, and Cees Snoek. Learning to learn kernels with variational random features. In *International Conference on Machine Learning*, 2020.

Andrey Zhmoginov, Mark Sandler, and Maksym Vladymyrov. HyperTransformer: Model generation for supervised and semi-supervised few-shot learning. In *International Conference on Machine Learning*, 2022.

Wenliang Zhong and James Tin Yau Kwok. Convex multitask learning with flexible task clusters. In *International Conference on Machine Learning*, 2012.

Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *Neural Information Processing Systems*, 2019.

Pan Zhou, Yingtian Zou, X Yuan, Jiashi Feng, Caiming Xiong, and SC Hoi. Task similarity aware meta learning: Theory-inspired improvement on MAML. In *Conference on Uncertainty in Artificial Intelligence*, 2021.

Zhuangdi Zhu, Kaixiang Lin, Anil K Jain, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, 2019.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017.