Meta-Learning with Complex Tasks

by

WEISEN JIANG

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science and Engineering

July 2024, Hong Kong

Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

WEISEN JIANG

16 July 2024

Meta-Learning with Complex Tasks

by

WEISEN JIANG

This is to certify that I have examined the above Ph.D. thesis
and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by
the thesis examination committee have been made.

Prof. James T. Kwok, Thesis Supervisor

Prof. Xiaofang Zhou, Head of Department

Department of Computer Science and Engineering

15 July 2024

Acknowledgments

I would not complete this work without the help of many people.

First and foremost, I would like to express my deepest gratitude to my Ph.D. advisors, Prof. James T. Kwok and Prof. Yu Zhang. Their guidance and encouragement helped me in all the time of research and writing of this thesis.

I would like to extend my sincere thanks to the committee members of PQE, proposal, and defense: Professors Brian Mak, Minhao Cheng, Dan Xu, Junxian He, Yangqiu Song, Rong Tang, and Sinno Jialin Pan. Their insightful suggestions and valuable comments have undoubtedly improved the quality of my thesis.

Many thanks are given to my collaborators: Hansi Yang, Longhui Yu, Han Shi, Jincheng Yu, Zhengying Liu, Zhenguo Li, Weiyang Liu, Baijiong Lin, Feiyang Ye, Yulong Zhang, Shuhao Chen, Yanbin Wei, Shuai Fu, and Xuehao Wang. It was my honor to work together with them and I really enjoyed these collaborations.

I also thank my teammates in Prof. Kwok's and Prof. Zhang's groups. Without them, my university life would not be so colorful.

Last but not least, I'm deeply indebted to my family members for their unconditional love, support, and encouragement during my Ph.D. journey.

Table of Contents

List of Figures

List of Tables

List of Notations

Notation	Description	Notation	Description
$\mathbb{R}^d$	d-dimensional vector space	$\mathbf{I}$	identity matrix
$\mathbf{x}, \mathbf{z}, \mathbf{a}$	vectors	$\otimes$	Kronecker product
$\mathbf{A}, \mathbf{B}, \mathbf{C}$	matrices	$\sigma_{\min}(\mathbf{A})$	smallest singular value of $\mathbf{A}$
$\mathbf{x}^\top, \mathbf{A}^\top$	transpose of $\mathbf{x}, \mathbf{A}$	$\lambda_{\min}(\mathbf{A})$	smallest eigenvalue value of $\mathbf{A}$
$\mathbf{A}^\perp$	$\mathbf{A}$'s orthogonal complement	$\|\cdot\|, \|\cdot\|_F$	ℓ_2 norm, Frobenius norm
τ	task	$\mathcal{B}$	mini-batch
$p(\tau)$	task distribution	$\mathcal{S}_\tau$	task τ's support (training) set
$\mathcal{T}$	set of meta-training tasks	$\mathcal{Q}_\tau$	task τ's query (validation) set
$(\mathbf{x}, y)$	sample: feature $\mathbf{x}$ and its label y	$\mathcal{S}_\tau^{(y)}$	samples in $\mathcal{S}_\tau$ with label y
$\mathcal{D}$	data set of samples	$\mathcal{Y}_\tau$	task τ's label set
$\|\mathcal{D}\|$	number of elements in $\mathcal{D}$	$\mathcal{V}_y$	set of tokens relevant to label y
$\mathbf{w}_\tau$	task-specific parameter	$\ell(\hat{y}, y)$	loss function
$\boldsymbol{\theta}$	meta-parameter	$\mathcal{L}(\mathcal{D}; \mathbf{w})$	loss on $\mathcal{D}$ using model $\mathbf{w}$
$f(\mathbf{x}; \mathbf{w})$	prediction of $\mathbf{x}$ using model $\mathbf{w}$	$\boldsymbol{\phi}, \boldsymbol{\Phi}$	parameters
t	iteration counter	b	mini-batch size
J, T	number of total iterations	ρ, λ	hyperparameter
η, η_t	learning rate	$\mathbf{h}_{[\text{MASK}]}$	embedding of the [MASK] token
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2	$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	multivariate normal distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\mathcal{K}(\cdot, \cdot)$	kernel	$\mathcal{H}$	Hilbert space
$\mathbb{T}(\cdot; \boldsymbol{\theta})$	template with soft prompt $\boldsymbol{\theta}$	$\mathbb{I}(\cdot)$	indicator function
$\text{softmax}(\cdot)$	softmax function	$\mathcal{U}(a, b)$	uniform distribution over $[a, b]$
Q, R, A	question, reasoning chain, answer		

Meta-Learning with Complex Tasks

WEISEN JIANG

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology

ABSTRACT

Meta-Learning aims at extracting shared knowledge (meta-knowledge) from historical tasks to accelerate learning on new tasks. It has achieved promising performance in various applications and many meta-learning algorithms have been developed to learn a meta-model that contains meta-knowledge (e.g., meta-initialization/meta-regularization) for task-specific learning procedures. In this thesis, we focus on meta-learning with *complex* tasks, thus, task-specific knowledge is diverse and various meta-knowledge is required.

First, we extend learning an efficient meta-regularization for *linear* models to *nonlinear* models by kernelized proximal regularization, allowing more powerful models like deep networks to deal with complex tasks. Second, we formulate the task-specific model parameters into *a subspace mixture* and propose a model-agnostic meta-learning algorithm to learn the subspace bases. Each subspace represents one type of meta-knowledge and *structured meta-knowledge* accelerates learning complex tasks more effectively than a simple meta-model. Third, we propose an effective and parameter-efficient meta-learning algorithm for prompt tuning on natural language processing tasks. The proposed algorithm learns *a pool of multiple meta-prompts* to extract meta-knowledge from meta-training tasks and then constructs *instance-dependent prompts* as weighted combinations of all the meta-prompts by attention. Instance-dependent prompts are flexible and powerful for prompting complex tasks.

Next, we study mathematical reasoning tasks using large language models (LLMs). To verify the candidate answers generated by LLMs, we propose combining the meta-knowledge of forward and backward reasoning. Lastly, we propose question augmentation to enlarge the question set for training LLMs to *enhance the LLMs' mathematical reasoning meta-knowledge*. The original questions are augmented in two directions: in the forward direction, we rephrase the questions by few-shot prompting; in the backward direction, we mask a number in the question and create a backward question to predict the masked number when the answer is provided.

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 in various applications [73, 199], they are data-hungry. Hence, a large number of training samples are required to learn a new task. This challenge remains a crucial bottleneck in making progress in machine learning algorithms and many research efforts attempt to use historical knowledge to improve data-efficiency in learning tasks.

Multitask learning (MTL) [276] 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 [277, 282, 27, 92, 133]. However, the learned MTL model is not guaranteed to generalize better and achieve faster learning on *unseen tasks*. Furthermore, MTL also suffers from the 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 [231], there are $\binom{64}{5} \approx 7 \times 10^6$ tasks in total.

Transfer learning [166, 257] finetunes a pretrained model on training data of a task to obtain a task-specific model. For example, for a *CIFAR-10* [112] classification task, we first pretrain a network (e.g., *ResNet-18* [73]) on *ImageNet* dataset [199], then use the pretrained network as initialization for gradient descent algorithms to minimize the training loss on *CIFAR-10*. Transfer learning has been successfully used in image classification [181, 70], natural language processing [13, 49], and reinforcement learning [58, 287]. However, pretraining and finetuning are independent, thus, *the pretrained model may not be suitable for learning new tasks with limited samples*.

Recently, *meta-learning* (or *learning to learn*) [10, 223] provides a general framework to extract meta-knowledge from historical tasks to accelerate learning unseen tasks for

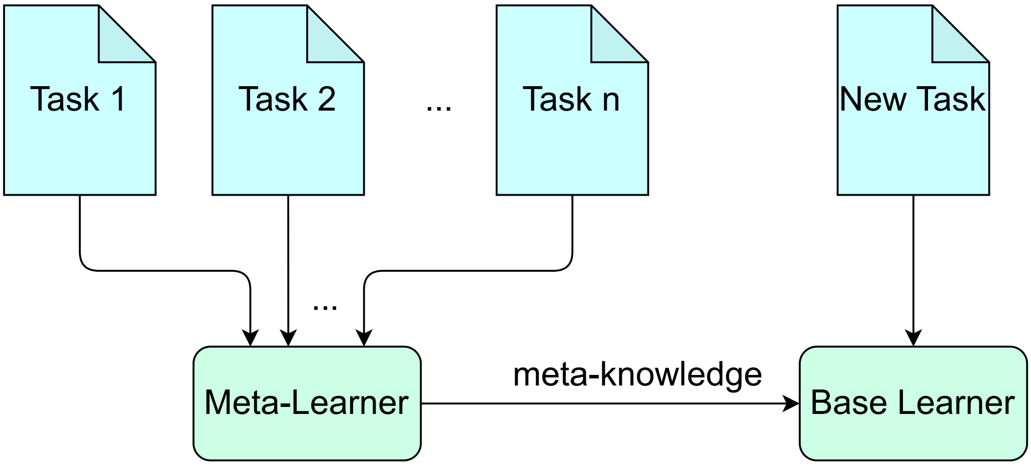


Figure 1.1: Illustration for meta-learning.

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 the validation set is a proxy measure of generalization ability, the meta-knowledge is explicitly tuned to learning new tasks with limited 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 successful applications in few-shot learning [238, 51, 231, 216, 98], hyperparameter optimization [55], neural architecture search [135, 288], and reinforcement learning [189].

1.2 Category of Meta-Learning Algorithms

Based on the type of base learner, popular meta-learning algorithms can be categorized into four groups: optimization-based, metric-based, memory-based, and in-context learning.

For *optimization-based* methods, the base learner performs gradient descent to minimize a task-specific training 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, and sample weights. Meta-initialization and meta-regularization are two representative methods. MAML [51] 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[127], T-Nets [118], Meta-Curvature [167], WarpGrad [53]) and efficiency (e.g., FO-MAML [51], Reptile [157]). The bilevel structure is complex and challenging to understand MAML from a theoretical view. Recently, many efforts have been devoted to study its convergence [52, 47, 234, 235, 95, 94] and generalization [149, 285, 35, 36, 173, 174, 195, 48, 187, 20, 225]. In meta-regularization, the base learner minimizes a regularized loss of 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 [188], Meta-MinibatchProx [284], and regularization for linear models [34, 35, 36]. As real-world tasks are usually complex, *non-structured meta-learning* methods learn a single meta-initialization or meta-regularization may be insufficient for capturing meta-knowledge of all tasks. To deal with this issue, *structured meta-learning* methods [93, 111, 229, 285] propose to formulate meta-knowledge into structures like clusters, such that each task can choose a suitable cluster center as initialization.

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 networks (e.g., VGG [214], ResNet [73]) and Vision Transformers [43] are widely used in feature extraction. For classifier, some popular candidates are the nearest neighbor classifier (e.g., ProtoNet [216]), linear models (e.g., R2D2 [11]), and kernel classifier (e.g., MetaOptNet [117])).

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 [203, 190, 155, 5]; hypernetworks (also called meta-networks) are external networks that contain knowledge of how to generate task parameters [154, 201, 215, 45, 156].

In-Context Learning (ICL) (or *Few-Shot Prompting*) [16, 150, 25, 136, 197, 138] can also be viewed as a meta-learning algorithm. It uses large language models (LLMs) to solve a task by feeding K examples as part of the input. The K examples are concatenated as a prompt

$$\mathbf{P}_{\text{ICL}} = \text{“Question: } Q^{(1)} \texttt{\textbackslash n} \text{ Answer: } A^{\star(1)} \ldots \text{ Question: } Q^{(K)} \texttt{\textbackslash n} \text{ Answer: } A^{\star(K)}\text{”},$$

where $Q^{(i)}$ and $A^{\star(i)}$ are the question and answer, respectively. In inference, a new question Q is appended to the prompt as "$\mathbf{P}_{\text{ICL}}$ \n Question: Q \n Answer:" and fed to the LLM for generating output sequences. An answer extractor is used to extract the prediction $\hat{A}$ from the output (e.g., the number after the last "Answer:" [16]). Compared with optimization-based algorithms, ICL is more efficient in computation and memory, as the LLM is fixed and shared across tasks. This can be crucial as LLMs are usually very large (e.g., *GPT-3* [16] has 175 billion parameters). ICL has demonstrated promising performance on a variety of tasks [16, 197, 136, 263, 252].

1.3 Applications of Meta-Learning

1.3.1 Computer Vision Tasks

Meta-Learning has a wide variety of applications in computer vision regimes, including few-shot classification [231, 191], object detection [175], landmark prediction [67], few-shot object segmentation [207], few-shot image generation [270], and density estimation [193].

Few-shot classification (FSC) is the most common application of meta-learning and is used in the thesis to evaluate the performance of meta-learning algorithms. The task is to classify classes with limited samples per class, which is very challenging, and many meta-learning methods have been proposed to improve their performance. For example, metric-based meta-learning algorithms like ProtoNet are specialized to FSC.

The benchmark of FSC is more complex than that of traditional machine learning benchmark, which evaluates the model from *seen instances to unseen instances*. However, in meta-learning, the FSC benchmark evaluates the generalization ability of the learned model from *seen classes to unseen classes*. Popular FSC benchmarks are *miniImageNet* [231, 191], *CIFAR-FS* [11], *Omniglot* [113], *Meta-Dataset* [259, 228].

1.3.2 Natural Language Processing Tasks

Large language models have achieved great success recently and many pretrained models are released for downstream tasks such as language understanding [42, 258, 217], machine translation [32, 69], and text classification [16, 119]. As finetuning the large models causes a heavy burden on computations, many parameter-efficient (PE) learning methods are proposed, e.g., prompt tuning and in-context learning. Prompt learning [16, 211, 40] freezes the pretrained model and formulates the downstream task as a cloze-style masked language model (MLM) problem [38]. In-Context Learning (ICL) [150, 25] uses a pretrained MLM to learn a new task by formatting training examples as a demonstration.

Similar to computer vision, collecting or designing many samples for training is infeasible. To deal with this issue, meta-learning is used to improve the data-efficiency of PE learning. For example, MetaPrompting [81] proposes to learn a good meta-initialization for the prompt vector, while MetaICL [150] finetunes the language model to make it more suitable for in-context learning.

1.4 Thesis Contributions and Organization

In this thesis, we study meta-learning with **complex** tasks, which are challenging for existing meta-learning algorithms. Figure 1.2 shows an overview of the organization of the thesis. In Chapter 2, we introduce the background of meta-learning as well as several representative algorithms, and prompting learning in NLP. The main contributions of other chapters are summarized as follows.

1. Chapter 3 learns a meta-regularization for **nonlinear** models to deal with complex tasks. The content of this Chapter is mainly based on Jiang et al. [97].

   - We introduce nonlinearity to **meta-regularization** by kernelized proximal regularization.
   - We propose a novel meta-learning algorithm called MetaProx for learning the meta-regularization. For regression tasks, the base learner has an efficient closed-form solution.
   - We establish local and global convergence of the proposed algorithm.
   - Experiments on a variety of benchmark regression and classification datasets demonstrate that MetaProx is better than the state-of-the-art.

2. Chapter 4 formulates meta-knowledge into **a subspace mixture** to handle complex

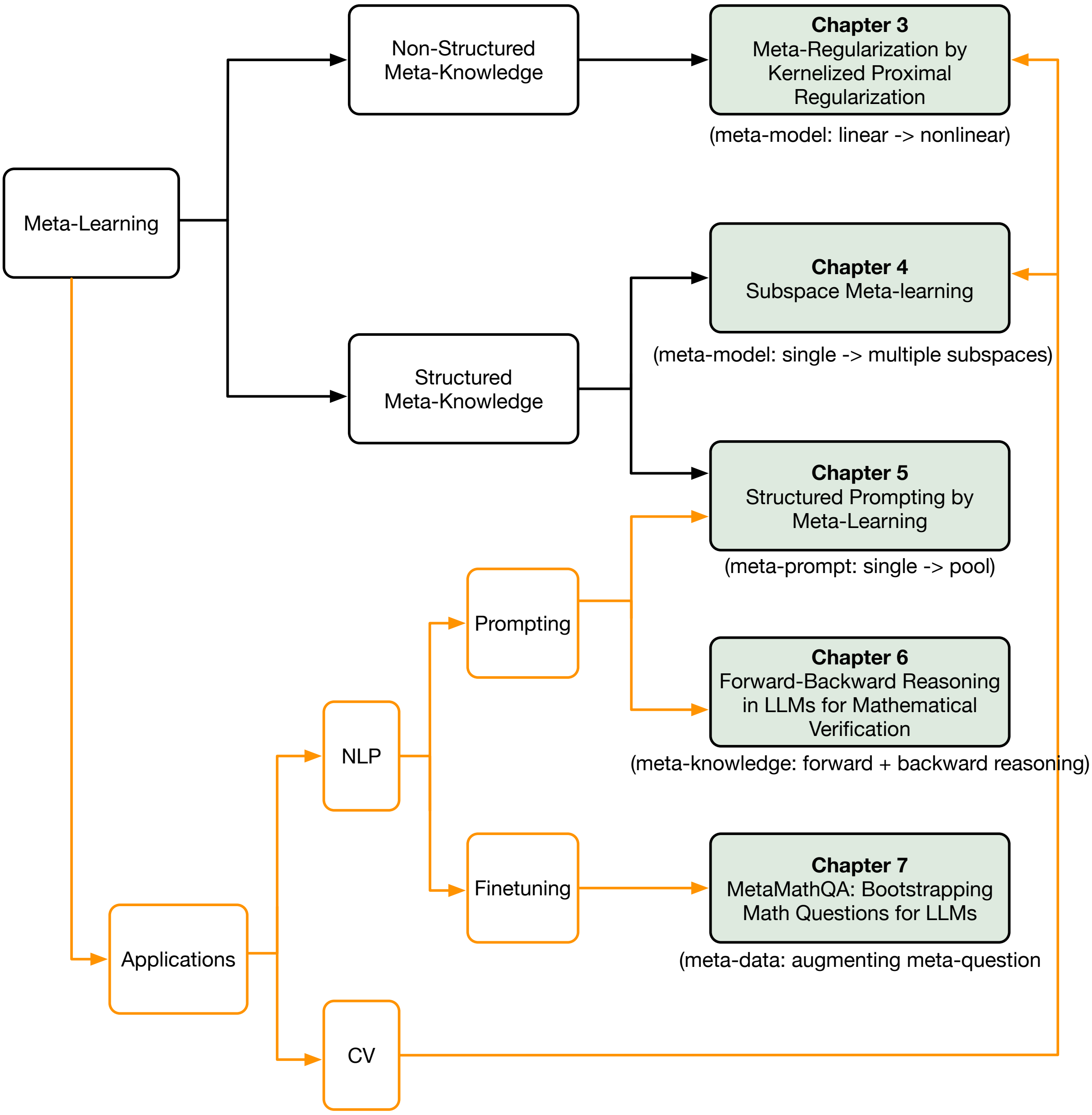

Meta-Learning
Non-Structured Meta-Knowledge
Structured Meta-Knowledge
Chapter 3
Meta-Regularization by Kernelized Proximal Regularization
(meta-model: linear -> nonlinear)
Chapter 4
Subspace Meta-learning
(meta-model: single -> multiple subspaces)
Chapter 5
Structured Prompting by Meta-Learning
(meta-prompt: single -> pool)
Prompting
Chapter 6
Forward-Backward Reasoning in LLMs for Mathematical Verification
(meta-knowledge: forward + backward reasoning)
NLP
Finetuning
Chapter 7
MetaMathQA: Bootstrapping Math Questions for LLMs
(meta-data: augmenting meta-question
Applications
CV

Figure 1.2: Outline of the thesis.

tasks that have diverse model weights. The content of this Chapter is mainly based on Jiang et al. [99].

- We formulate task model parameters into multiple subspaces (each subspace represents one type of meta-knowledge) and propose a model-agnostic algorithm MUSML to learn the subspace bases.
- We provide a theoretical analysis of the population risk, empirical risk, and generalization gap.
- Extensive experiments on regression and classification datasets demonstrate the effectiveness of learning a subspace mixture.

3. Chapter 5 studies applications of meta-learning in language models and learns **a prompt pool** for constructing instance-dependent prompts for complex NLP tasks. The content of this Chapter is mainly based on Jiang et al. [101].

    - We use a prompt pool to extract meta-knowledge and construct instance-dependent prompts by attention.
    - We design a novel soft verbalizer called representative verbalizer, which builds label embeddings by averaging feature embeddings.
    - Combining meta-learning a prompt pool with a novel soft verbalizer, we propose a novel parameter-efficient meta-learning algorithm MetaPrompter.
    - Experimental results demonstrate the usefulness and parameter-efficiency of MetaPrompter. Moreover, the superiority of the proposed representative verbalizer over existing verbalizers is verified by empirical evaluations.

4. Chapter 6 studies the problem of **verifying candidate answers of mathematical problems** by leveraging LLMs' **forward and backward reasoning meta-knowledge**. The content of this Chapter is mainly based on Jiang et al. [103].

    - We use the meta-knowledge of backward reasoning for mathematical verification, i.e., masking a number in the original question and asking the LLM to predict the masked number when a candidate answer is provided.
    - We propose FOBAR to combine FOrward and BAckward Reasoning meta-knowledge for verification.
    - Experimental results on six standard mathematical benchmarks and three LLMs show that FOBAR achieves SOTA performance. In particular, FOBAR outperforms Self-Consistency which uses forward reasoning alone, demonstrating that combining forward and backward reasoning together is better. Additionally, FOBAR outperforms Self-Verification, confirming that using the simple template and the proposed combination is more effective.
    - Empirical results on two non-mathematical reasoning tasks show that FOBAR also performs well.

5. Chapter 7 studies **data augmentation** in finetuning open-source LLMs to enhance the **meta-knowledge of solving mathematical problems**. The content of this Chapter is mainly based on Yu, Jiang, et al. [267].

- We propose a novel question bootstrapping method to augment the training dataset, resulting in *MetaMathQA*. Question bootstrapping rewrites questions with both forward and backward reasoning paths.
- Based on the *MetaMathQA* dataset, MetaMath is finetuned from state-of-the-art open-source LLMs (e.g., *LLaMA-2*), showing excellent forward and backward reasoning ability on mathematical tasks.
- Our work studies data augmentation for improving the mathematical problem-solving meta-knowledge of LLMs. Despite being simple, our method significantly outperforms many intricate methods. Our results highlight the importance of data augmentation and also shed light on other reasoning tasks.

Other publications [98, 100, 102, 129, 278, 244].

CHAPTER 2

Background

2.1 Formulation of Meta-Learning

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 $\{\mathbf{w}_1, \dots, \mathbf{w}_{|\mathcal{T}|}\}$. Each task τ contains a support (also called training) set $\mathcal{S}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \dots, n_s\}$ and a query (also called validation) set $\mathcal{Q}_\tau = \{(\mathbf{x}_i, y_i) : i = 1, \dots, n_q\}$, where $\mathbf{x} \in \mathbb{R}^d$ are the features and y the labels. For the 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, squared loss for regression). In each meta-training iteration, a mini-batch $\mathcal{B}$ of tasks is 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 $\mathcal{Q}_{\tau'}$.

2.2 Representative Meta-Learning Algorithms

2.2.1 MAML

A pioneer work for learning a *meta-initialization* is Model-Agnostic Meta-Learning (MAML) proposed by Finn et al. [51]. An illustration is shown in Figure 2.1.

Meta-Training. MAML operates in two optimization levels (let $\mathcal{B}_t$ be a mini-batch of tasks at iteration t):

- *Inner Level*: 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

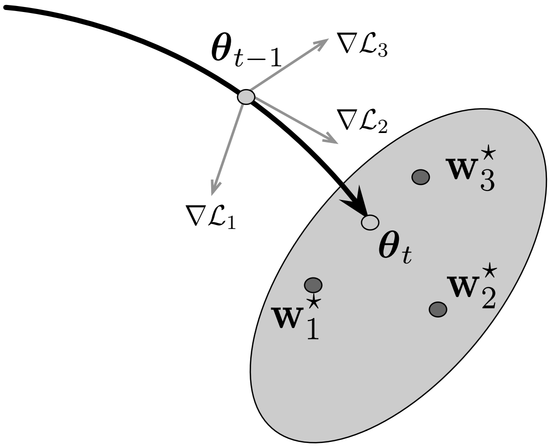


Figure 2.1: Illustration for MAML.

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* : 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. $\boldsymbol{\theta}_{t-1}$, i.e., meta-gradient. The meta-initialization is updated as:

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

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, incurring huge computations, especially for large models and a large J. To reduce the computational cost, FO-MAML [51] discard the second-order derivative and use the first-order approximation $\nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)}) \approx \nabla_{\mathbf{w}_\tau^{(J)}} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{w}_\tau^{(J)})$. Alternatively, Reptile [157] approximates meta-gradient by the update direction from task model parameters to meta-initialization, i.e., $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} \left(\boldsymbol{\theta}_{t-1} - \mathbf{w}_\tau^{(J)}\right)$. As this update rules 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 [242]) or even non-differentiable methods like line search [15].

Besides efficiency, empirical results in [51, 157] demonstrate that the approximations in FOMAML 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 $\mathbf{w}_{\tau'}^{(J)}$, which is then used to predict the query sample in $\mathcal{Q}_{\tau'}$ and evaluate performance.

Several works improve MAML using preconditioning gradients in the inner loop, e.g., MetaSGD [127], Meta-Curvature [167], T-Nets [118], and WarpGrad [53]. Unlike MAML that updates all network parameters in the base learner, recent work [186, 159, 210] reveals that updating only part of the network can improve both efficiency and effectiveness.

2.2.2 iMAML

Meta-regularization [34, 35, 188, 284, 36, 97] is another optimization-based meta-learning method, which assumes that task models are close to a prior model. A representative algorithm is iMAML [188]. The base learner obtains the 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 each meta-iteration t, the base learner takes $\mathcal{S}_\tau$ and $\boldsymbol{\theta}_{t-1}$ to build task model $\hat{\mathbf{w}}_\tau$ by solving the regularized minimization problem:

$$\hat{\mathbf{w}}_\tau = \arg\min_{\mathbf{w}_\tau} \mathcal{L}(\mathcal{S}_\tau; \mathbf{w}_\tau) + \frac{\lambda}{2}\|\mathbf{w}_\tau - \boldsymbol{\theta}_{t-1}\|^2, \tag{2.3}$$

where $\lambda > 0$ is hyperparameter. Similar to meta-initialization, $\hat{\mathbf{w}}_\tau$ is a function of $\boldsymbol{\theta}_{t-1}$. The meta-learner in meta-regularization algorithms (iMAML [188] and Denevi et al. [34]) update the regularizer by performing gradient descent steps on the validation loss

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \frac{\eta_t}{|\mathcal{B}_t|}\sum_{\tau\in\mathcal{B}_t} \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(\mathcal{Q}_\tau; \hat{\mathbf{w}}_\tau). \tag{2.4}$$

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 (2.3), it follows from the first-order optimal condition that

$$\nabla_{\hat{\mathbf{w}}_\tau} \mathcal{L}(\mathcal{S}_\tau; \hat{\mathbf{w}}_\tau) + \lambda(\hat{\mathbf{w}}_\tau - \boldsymbol{\theta}_{t-1}) = \mathbf{0}. \tag{2.5}$$

By implicit function theorem [198], 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

$$\nabla_{\boldsymbol{\theta}_{t-1}}\hat{\mathbf{w}}_\tau = \left(\frac{1}{\lambda}\nabla^2_{\hat{\mathbf{w}}_\tau}\mathcal{L}(\mathcal{S}_\tau;\hat{\mathbf{w}}_\tau) + \mathbf{I}\right)^{-1}. \tag{2.6}$$

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 depend on the optimization path, thus, the chosen optimizer in the base learner is flexible, e.g., AdaDelta [271], Adam [108], AdamW [143].

2.2.3 Prototypical Networks

ProtoNet [216] is a representative metric-based method. 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 embedding. 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

$$\mathbf{p}_y = \frac{1}{|\mathcal{S}_\tau^{(y)}|}\sum_{(\mathbf{x}_i,y)\in\mathcal{S}_\tau^{(y)}}\mathsf{NN}(\mathbf{x}_i;\boldsymbol{\phi}_{t-1}), \tag{2.7}$$

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

$$\mathbb{P}(y|\mathbf{x};\boldsymbol{\phi}_{t-1}) = \frac{\exp(-\kappa\,\mathrm{dist}(\mathbf{p}_y,\mathsf{NN}(\mathbf{x};\boldsymbol{\phi}_{t-1})))}{\sum_{y'\in\mathcal{Y}_\tau}\exp(-\kappa\,\mathrm{dist}(\mathbf{p}_{y'},\mathsf{NN}(\mathbf{x};\boldsymbol{\phi}_{t-1}))}, \tag{2.8}$$

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

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} + \frac{\eta_t}{|\mathcal{B}_t|}\sum_{\tau\in\mathcal{B}_t}\frac{1}{|\mathcal{Q}_\tau|}\sum_{(\mathbf{x}_i,y_i)\in\mathcal{Q}_\tau}\nabla_{\boldsymbol{\phi}_{t-1}}\log\mathbb{P}(y_i|\mathbf{x}_i;\boldsymbol{\phi}_{t-1}), \tag{2.9}$$

The framework of ProtoNet is simple and general, and the distance metric is flexible. Other possible *metrics* are cosine similarity [22, 63, 137], earth mover's distance [273], a metric learned by deep networks [219], a metric based on graph convolution blocks [60].

2.2.4 MetaOptNet

When the embedding space is high-dimensional, a trainable linear classifier is more expressive than the nearest neighbor learner ProtoNet [216]. MetaOptNet [117] is a representative algorithm. Let $\mathbf{z} = \mathsf{NN}(\mathbf{x}; \boldsymbol{\phi}) \in \mathbb{R}^e$ 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 to 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:

$$\mathbf{W}_\tau^\star = \arg\min_{\mathbf{W}_\tau \in \mathbb{R}^{e \times C}} \frac{1}{2} \|\mathbf{Z}_\tau^{tr} \mathbf{W}_\tau - \mathbf{Y}_\tau^{tr}\|^2 + \frac{\lambda}{2} \|\mathbf{W}_\tau\|^2, \tag{2.10}$$

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 [51], 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

$$\hat{\mathbf{y}}_i = a_{t-1} \, \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})^\top \mathbf{W}_\tau^\star + c_{t-1}, \tag{2.11}$$

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

$$(\boldsymbol{\phi}_t, a_t, c_t) = (\boldsymbol{\phi}_{t-1}, a_{t-1}, c_{t-1}) - \eta \nabla_{(\boldsymbol{\phi}_{t-1}, a_{t-1}, c_{t-1})} \frac{1}{b} \sum_{\tau \in \mathcal{B}_t} \frac{1}{n_q} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{Q}_\tau} \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i). \tag{2.12}$$

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:

$$\max_{\boldsymbol{\alpha}_{\tau,1}, \dots, \boldsymbol{\alpha}_{\tau,C}} -\frac{1}{2} \sum_c \|\boldsymbol{\alpha}_{\tau,c}^\top \mathbf{Z}_\tau^{tr}\|^2 + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \alpha_{\tau, y_i, i} \tag{2.13}$$

$$\text{s.t.} \quad \alpha_{\tau, y_i, i} \le \gamma, \alpha_{\tau, c, i} \le 0 \text{ for } c \ne y_i, \mathbf{1}^\top \boldsymbol{\alpha}_{\tau, \cdot, i} = 0, \forall i. \tag{2.14}$$

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 obtained with low cost by an iterative solver, e.g., projected gradient methods [17, 130]. 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:

$$\boldsymbol{\phi}_t = \boldsymbol{\phi}_{t-1} - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \frac{1}{n_q} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{Q}_\tau} \nabla_{\boldsymbol{\phi}_{t-1}} \ell(\hat{\mathbf{y}}_i, y_i). \tag{2.15}$$

Note that $\boldsymbol{\alpha}_\tau^\star$ is a solution to the dual problem, implicitly depends on $\boldsymbol{\phi}_t$. Similar to iMAML [188], we can use implicit function theorem to compute the gradient of $\nabla_{\boldsymbol{\phi}_{t-1}} \boldsymbol{\alpha}_\tau^\star$. See the CVXPYLayers package [2] for an implementation to solve the dual problem and back-propagate gradients through the convex learner.

2.3 Prompt Learning for Language Models

2.3.1 Prompt Tuning

Recently, it is common to use a pretrained MLM $\mathcal{M}(\cdot; \boldsymbol{\phi})$, with parameter $\boldsymbol{\phi}$, for various downstream tasks such as language understanding [42, 258, 217], machine translation [32, 69], and text classification [16, 119, 140]. Given a raw sentence represented as a sequence of n tokens $(x_1, \dots, x_n)$, the MLM takes $\mathbf{x} = (\texttt{[CLS]}, x_1, \dots, x_n, \texttt{[SEP]})$ as input (where [CLS] is the start token and [SEP] is the separator), and encodes it into a sequence of hidden representations $(\mathbf{h}_{\texttt{[CLS]}}, \mathbf{h}_1, \dots, \mathbf{h}_n, \mathbf{h}_{\texttt{[SEP]}})$. In standard finetuning [83, 38], 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 [16]), finetuning all parameters can cause a heavy burden on computation and memory.

On the other hand, prompt learning [16, 211, 40] freezes the pretrained 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 [16, 202]. Thus, manually-designed prompts are likely to be sub-optimal.

Unlike discrete prompts, prompt tuning [119, 139] 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* [139, 204, 40]:

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

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* [119, 40, 86] 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 $y =$ ‘‘SPORTS’’, we can have $\mathcal{V}_y =$ {‘‘sports’’, ‘‘football’’, ‘‘basketball’’}. Prompt tuning then optimizes[1] $(\boldsymbol{\phi}, \boldsymbol{\theta})$ by maximizing the label probability:

$$\hat{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta}) = \frac{1}{|\mathcal{V}_y|} \sum_{\texttt{w} \in \mathcal{V}_y} \mathbb{P}_{\mathcal{M}}(\texttt{[MASK]} = \texttt{w}|\mathbb{T}(\mathbf{x};\boldsymbol{\theta})), \tag{2.16}$$

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 [119, 40]. However, selecting label-relevant tokens requires intensive human labor. Recent works [71, 275, 33] propose *soft* verbalizers, which map each label to a *continuous* embedding and predict label distribution based on the similarity between feature embedding and label embeddings. WARP [71] and DART [275] obtain this label embedding by supervised learning, while ProtoVerb [33] uses contrastive learning [21, 224]. 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 [59, 9, 72, 18, 81], as the number of samples per class is typically much smaller than d_o (e.g., $d_o = 768$ for BERT [38]).

2.3.2 MetaPrompting

As prompt tuning is sensitive to prompt initialization in few-shot tasks [119], meta-learning can be used to search for a good initialization. MetaPrompting [81] 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

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

$\alpha > 0$ 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 [81], 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.

2.4 CoT Prompting for Mathematical Reasoning Tasks

2.4.1 Chain-of-Thought Prompting

CoT Prompting. Wei et al. (2022) propose augmenting question-answer pairs with intermediate steps such that the LLM can solve questions step-by-step. Specifically, each in-context example is a triplet $(Q^{(i)}, R^{(i)}, A^{\star(i)})$, where $R^{(i)}$ is a reasoning chain with natural language descriptions of steps leading from the question $Q^{(i)}$ to the ground-truth answer $A^{\star(i)}$. In inference, a new question Q is appended to the prompt:

$$\mathbf{P}_{\text{CoT}} = \text{“Question: } Q^{(1)}\text{; Answer: } R^{(1)}, A^{\star(1)} \ldots \text{ Question: } Q^{(K)}\text{; Answer: } R^{(K)}, A^{\star(K)}\text{”}$$

and “$\mathbf{P}_{\text{CoT}}$ `\n` Question: {Q} `\n` Answer:” is fed to the LLM for generating both its reasoning chain R and answer A. CoT prompting has achieved SOTA performance on a wide variety of tasks [243, 109, 57, 279, 237, 281, 283, 280].

Recently, many works [57, 281, 147, 171, 212, 245, 283, 24, 274] have been proposed to improve the quality of reasoning chains in CoT prompting. ComplexCoT [57] selects examples with more steps as in-context examples, while PHP [281] iteratively uses the previous answers as hints in prompting. These aforementioned works can be viewed as forward reasoning, which starts from the question and generates a reasoning chain to reach the answer [246, 208]. Instead of taking a single reasoning chain by greedy decoding, Self-Consistency [237] samples a diverse set of chains and obtains a set of candidate answers. The final answer is then selected by majority voting.

Backward Reasoning (a.k.a. backward chaining) [176, 200, 107, 128, 266] starts with an answer and works backward to verify the sequence of steps or conditions necessary to reach this answer. Backward reasoning is particularly useful in domains when the answer is known, e.g., in automated theorem provers [200, 194, 236, 106, 178]. Recently, Self-Verification [246] rewrites the question with an answer into a declarative statement and then asks the LLM to predict a number in the question. RCoT [254] regenerates a sentence (a sequence of tokens) in the question conditioning on the answer and detects whether there is factual inconsistency in the constructed question by three complicated steps. Self-Verification and RCoT need additional rewriting and reconstruction for creating backward questions. Moreover, Self-Verification and RCoT use backward reasoning alone for verifying candidate answers.

2.4.2 Mathematical Reasoning

Solving mathematical reasoning tasks like *GSM8K* [30] and *MATH* [77] is one of the most challenging problem in LLMs. Many CoT methods [243, 57, 237, 283] have been proposed to design powerful prompts for activating the *close-source* LLMs' mathematical reasoning ability. Self-Consistency [237], which is the SOTA method, samples multiple reasoning paths and selects the final answer by majority voting.

Another category of work is finetuning-based methods, which finetunes *open-source* models (e.g., *LLaMA* [226], *Mistral* [96]) with the knowledge from some advanced closed-source LLMs [161, 163]. Magister et al. [148] investigates the transfer of reasoning capabilities via knowledge distillation. Yuan et al. [268] proposes to apply rejection sampling finetuning (RFT) to improve mathematical reasoning performance. WizardMath [145] proposes a reinforced evol-instruct method to enhance reasoning abilities by supervised finetuning and PPO training [206]. MAmmoTH [269] combines CoT and Program-of-Thought [23] rationales for teaching LLMs to use external tools (e.g., Python interpreter) for solving mathematical problems.

Knowledge Distillation [78, 65] transfers knowledge from a larger teacher model to a smaller student model, achieving promising performance in many applications [209, 168, 75, 151], Recently, [124, 87, 79, 148, 84, 56, 213] propose to transfer reasoning abilities from LLMs (e.g., *GPT-3.5-Turbo* [161], *PaLM* [29]) to small language models (e.g., *T5* [184], *GPT-2* [182]). For example, Finetune-CoT [79] samples multiple reasoning paths from LLMs and finetunes the student model with the correct ones, while Self-Improve [87] chooses the one with the highest confidence. Li et al. [124] further feeds the question and ground-truth label to LLMs for prompting its reasoning path. Shridhar et al. [213] proposes to generate sub-questions and solution pairs for training. Small models finetuned by knowledge distillation can achieve similar performance to LLMs [148, 79] on both common sense reasoning (e.g., *CommonSenseQA* [220]) and symbol reasoning (e.g., *CoinFlip* [243]). However, for solving challenging mathematical problems (e.g., *GSM8K* [30]), there is still a large performance gap [79, 56, 148].

CHAPTER 3

Meta-Regularization by Kernelized Proximal Regularization

3.1 Introduction

Humans can easily learn new tasks from a handful of examples using prior knowledge and experience. In contrast, deep networks are data-hungry, and a large number of training samples are required to mitigate overfitting. To reduce the labor-intensive and time-consuming process of data labeling, *meta-learning* (or *learning to learn*) [10, 223] aims to exact meta-knowledge from seen tasks to accelerate learning on unseen tasks. Recently, meta-learning has been receiving increasing attention due to its diverse successful applications in few-shot learning [238, 51, 231, 216], hyperparameter optimization [55], neural architecture search [135, 288], and reinforcement learning [189].

Many meta-learning algorithms operate on two levels. A base learner learns task-specific models in the inner loop, and a meta-learner learns the meta-parameter in the outer loop. A popular class of algorithms is based on meta-initialization [51, 157, 47, 228], such as the well-known MAML [51]. It learns a model initialization such that a good model for an unseen task can be learned from limited samples by a few gradient updates. However, computing the meta-gradient requires back-propagating through the entire inner optimization path, which is infeasible for large models and/or many gradient steps. During testing, it is common for MAML's base learner to perform many gradient steps to seek a more accurate solution [51]. However, for regression using a linear base learner and square loss, we will show that though the meta-learner can converge to the optimal meta-initialization, the base learner may overfit the training data at meta-testing.

Another class of meta-learning algorithms is based on meta-regularization [188, 284, 34, 35, 36], 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. [34] uses a linear model with an efficient closed-form solution for the base learner.

However, extending to nonlinear base learners requires computing the meta-gradient using matrix inversion, which can be infeasible for deep networks [188].

To introduce nonlinearity to the base learner, a recent approach is to make use of the kernel trick. For example, R2D2 [11] and MetaOptNet [117] use deep kernels [247] 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 [34] and classification [35], in this chapter, we propose a kernel-based algorithm to meta-learn a proximal regularizer for a nonlinear base learner. By kernel extension, the learnable function in the proximal regularizer is a function in the reproducing kernel Hilbert space (RKHS) induced by the base kernel. The proposed algorithm is guaranteed to converge to a critical point of the meta-loss and its global convergence is also established. Experiments on various benchmark regression and classification datasets demonstrate the superiority of the proposed algorithm over the state-of-the-arts.

Our contributions are summarized as follows. (i) By kernelizing proximal regularization, we introduce nonlinearity to meta-regularization. (i) We propose a novel meta-learning algorithm called MetaProx for learning the nonlinear meta-regularization. For regression tasks, the base learner has an efficient closed-form solution. For classification tasks, as the dimension of dual variables is low, the computation cost is also low. (i) We establish the local and global convergence of the proposed algorithm. (i) Experiments on a variety of benchmark regression and classification datasets demonstrate that MetaProx performs better than the state-of-the-arts.

3.2 Meta-Initialization versus Meta-Regularization

In this section, we consider a simple regression setting with a linear model and square loss. Each task τ is a linear regressor with parameter $\mathbf{w}_\tau^\star \in \mathbb{R}^d$. We assume that each input $\mathbf{x}$ is sampled from $\mathcal{N}(\mathbf{0}, \sigma_{\mathbf{x}}^2 \mathbf{I})$ and the output y is obtained as $\mathbf{x}^\top \mathbf{w}_\tau^\star + \xi$, where $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the random noise. We compare two representative meta-learning algorithms: (i) MAML [51], which is based on meta-initialization and performs one

Algorithm 1 MAML [51].

```
Require: step size $\gamma$ and $\eta_t$, batch size $b$;
for $t = 1, 2, 3, \ldots$ do
    sample a batch $\mathcal{B}_t$ of tasks from $p(\tau)$;
    base learner:
    for $\tau \in \mathcal{B}_t$ do
        $\mathbf{w}_\tau^{(\mathrm{gd})}(\boldsymbol{\psi}_t) = \boldsymbol{\psi}_t - \gamma \mathbf{X}_\tau^\top (\mathbf{X}_\tau \boldsymbol{\psi}_t - \mathbf{y}_\tau)$;
        $\mathbf{g}_\tau = \frac{1}{2} \sum_{(\mathbf{x},y) \in Q_\tau} \nabla_{\boldsymbol{\psi}_t} (\mathbf{x}^\top \mathbf{w}_\tau^{(\mathrm{gd})}(\boldsymbol{\psi}_t) - y)^2$;
    end for
    meta-learner: $\boldsymbol{\psi}_{t+1} = \boldsymbol{\psi}_t - \frac{\eta_t}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$;
end for
```

gradient descent step in the inner loop of the bilevel optimization problem; and (ii) learning around a common mean (denoted CommonMean) [34], which is based on meta-regularization. It learns the model parameters for task τ by minimizing the loss with a proximal regularizer around the meta-parameter $\boldsymbol{\theta}$:

$$\begin{aligned} \mathbf{w}_\tau^{(\mathrm{prox})}(\boldsymbol{\theta}) &= \arg\min_{\mathbf{w}} \sum\nolimits_{(\mathbf{x}_i, y_i) \in S_\tau} \frac{1}{2} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 + \frac{\lambda}{2} \|\mathbf{w} - \boldsymbol{\theta}\|^2 \\ &= (\lambda \mathbf{I} + \mathbf{X}_\tau^\top \mathbf{X}_\tau)^{-1} (\lambda \boldsymbol{\theta} + \mathbf{X}_\tau^\top \mathbf{y}_\tau), \end{aligned} \tag{3.1}$$

where $\mathbf{X}_\tau = [\mathbf{x}_1^\top; \ldots; \mathbf{x}_{n_s}^\top]$ is the sample matrix from S_τ (each column of $\mathbf{X}_\tau$ is an input vector), and $\mathbf{y}_\tau = [y_1; \ldots; y_{n_s}]$ is the corresponding label vector. Note that $\mathbf{w}_\tau^{(\mathrm{prox})}$ is a function of $\boldsymbol{\theta}$. Algorithms 1 and 2 show MAML and CommonMean, respectively, for this problem.

Recently, Balcan et al. [7] study the convex online meta-learning setting and show that both approaches achieve the same average task regret. Here, we consider the offline setting. First, the following Proposition shows that both MAML and CommonMean converge to the same meta-parameter. All proofs for theoretical results in this chapter are in the Appendix.

Proposition 3.2.1. *Let $\eta_t = 1/t$. Assume that $\gamma < 1/\sigma_{\mathbf{x}}^2$. Both $\boldsymbol{\psi}_t$ in MAML (with one inner gradient step) and $\boldsymbol{\theta}_t$ in CommonMean converge to $\bar{\mathbf{w}} = \mathbb{E}_\tau \mathbf{w}_\tau^\star$.*

The following Proposition shows that $\bar{\mathbf{w}}$ in Proposition 3.2.1 is also the best $\boldsymbol{\psi}$ (for MAML) or $\boldsymbol{\theta}$ (for CommonMean) with the lowest population risk for this meta-learning problem.

Proposition 3.2.2. *Assume $\gamma < 1/\sigma_{\mathbf{x}}^2$. We have $\bar{\mathbf{w}} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y) \in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(prox)} - y)^2 = \arg\min_{\boldsymbol{\psi}} \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y) \in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(gd)} - y)^2$.*

Algorithm 2 CommonMean [34].

```
Require: hyperparameter $\lambda$, step size $\eta_t$, batch size $b$;
for $t = 1, 2, 3, \dots$ do
    sample a batch $\mathcal{B}_t$ of tasks from $p(\tau)$;
    base learner:
    for $\tau \in \mathcal{B}_t$ do
        $\mathbf{w}_\tau^{(\text{prox})} = \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{X}_\tau \mathbf{w} - \mathbf{y}_\tau\|^2 + \frac{\lambda}{2}\|\mathbf{w} - \boldsymbol{\theta}_t\|^2$;
        $\mathbf{g}_\tau = \frac{1}{2}\sum_{(\mathbf{x},y)\in\mathcal{Q}_\tau} \nabla_{\boldsymbol{\theta}_t}(\mathbf{x}^\top \mathbf{w}_\tau^{(\text{prox})} - y)^2$;
    end for
    meta-learner: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta_t}{b}\sum_{\tau\in\mathcal{B}_t} \mathbf{g}_\tau$;
end for
```

During meta-testing, we sample a task $\tau' \sim p(\tau)$ with parameter $\mathbf{w}^\star_{\tau'}$. Let $\mathbf{X}_{\tau'}$ be the sample matrix from $S_{\tau'}$, and $\mathbf{y}_{\tau'}$ be the corresponding label vector. We assume that $\mathbf{X}_{\tau'}$ is full rank and $n_s < d$. To simplify notations, we drop the subscript τ' in the following. Let the singular value decomposition of $\mathbf{X}$ be $\mathbf{U\Sigma V}^\top$ (where $\mathbf{\Sigma} = \text{diag}([\nu_1, \dots, \nu_{n_s}])$), and $\mathbf{V}^\perp$ be $\mathbf{V}$'s orthogonal complement.

As only forward passes are needed, it is common for the base learner in MAML to perform multiple gradient steps [51]. With the convex loss and linear model here, the base learner can obtain a globally optimal solution $\mathbf{w}^{(\text{gd}\infty)}$ directly (which is equivalent to taking infinite gradient steps). As is common in few-shot learning, the number of support samples is much smaller than feature dimensionality. Hence, $\mathbf{w}^{(\text{gd}\infty)}$ is not unique but depends on the learned initialization. Let $\mathbf{w}^{(\text{gd}\infty)}$ be written as $\mathbf{w}^{(\text{gd}\infty)} = \mathbf{V}\mathbf{a}^{(\text{gd}\infty)} + \mathbf{V}^\perp \mathbf{b}^{(\text{gd}\infty)}$. For gradient descent, its update direction $\mathbf{X}^\top(\mathbf{Xw} - \mathbf{y}) = \mathbf{V\Sigma U}^\top(\mathbf{Xw} - \mathbf{y})$ is always in the span of $\mathbf{V}$ and so $\mathbf{b}^{(\text{gd}\infty)}$ **remains unchanged**.

Let $\mathbf{w}^\star$ and $\boldsymbol{\theta}$ be written as $\mathbf{w}^\star = \mathbf{V}\mathbf{a}^\star + \mathbf{V}^\perp\mathbf{b}^\star$ and $\boldsymbol{\theta} = \mathbf{V}\mathbf{a}_0 + \mathbf{V}^\perp \mathbf{b}_0$. Moreover, let $\tilde{\mathbf{a}} = \mathbf{a}_0 - \mathbf{a}^\star$ and $\tilde{\mathbf{b}} = \mathbf{b}_0 - \mathbf{b}^\star$.

Proposition 3.2.3 ([68]). *Assume that $\gamma < \min_{1\le j\le n_s} 1/\nu_j^2$. We have $\mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{w}^{(gd\infty)} - \mathbf{w}^\star\|^2 = \|\mathbf{b}^{(gd\infty)} - \mathbf{b}^\star\|^2 + \sum_{j=1}^{n_s}\left(\frac{\sigma_\xi}{\nu_j}\right)^2$, where the expectation is over the label noise vector $\boldsymbol{\xi}$.*

For CommonMean, using the Woodbury matrix identity, we have $\mathbf{w}^{(\text{prox})} = \boldsymbol{\theta} + \mathbf{X}^\top(\lambda\mathbf{I} + \mathbf{XX}^\top)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) = \mathbf{V}(\mathbf{a}_0 + \mathbf{\Sigma U}^\top(\lambda\mathbf{I} + \mathbf{XX}^\top)^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta})) + \mathbf{V}^\perp\mathbf{b}_0 \equiv \mathbf{V}\mathbf{a}^{(\text{prox})} + \mathbf{V}^\perp\mathbf{b}^{(\text{prox})}$, where $\mathbf{b}^{(\text{prox})} = \mathbf{b}_0$. Assume that $\mathbf{w}^{(\text{gd}\infty)}$ is initialized with $\boldsymbol{\theta}$. Since $\mathbf{b}^{(\text{gd}\infty)}$ remains unchanged, $\mathbf{w}^{(\text{prox})}$ and $\mathbf{w}^{(\text{gd}\infty)}$ **only differ in the components lying in the column space of V.**

Proposition 3.2.4. $\mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{w}^{(prox)} - \mathbf{w}^\star\|^2 = \|\tilde{\mathbf{b}}\|^2 + \sum_{j=1}^{n_s}\left(\frac{\lambda \tilde{a}_j}{\lambda+\nu_j^2}\right)^2 + \sum_{j=1}^{n_s}\left(\frac{\sigma_\xi}{(\lambda/\nu_j)+\nu_j}\right)^2$*, where the expectation is over the label noise vector* $\boldsymbol{\xi}$*.*

As can be seen, when the labels are noise-free ($\sigma_\xi^2 = 0$), $\mathbf{w}^{(\text{gd}\infty)}$ performs better than $\mathbf{w}^{(\text{prox})}$. However, when the labels are noisy, as $n_s < d$, gradient descent always converges to zero training error and overfits the noisy labels. On the other hand, the estimation error of $\mathbf{w}^{(\text{prox})}$ equals to that of $\mathbf{w}^{(\text{gd}\infty)}$ when $\lambda = 0$. For $\lambda > 0$, it trades off between fitting the noisy labels (the last term in Proposition 3.2.4) and introducing an estimation bias of $\mathbf{a}^\star$ (the second term in Proposition 3.2.4).

3.3 The Proposed MetaProx

It is straightforward to use the dual formulation for the CommonMean algorithm. When the square loss is used as $\ell(\cdot,\cdot)$, it is easy to see that the dual variable has the closed-form solution

$$\boldsymbol{\alpha}_\tau = (\mathbf{I} + \lambda^{-1}\mathbf{X}_\tau\mathbf{X}_\tau^\top)^{-1}(\mathbf{y}_\tau - \mathbf{X}_\tau\boldsymbol{\theta}). \tag{3.2}$$

Compared with the primal formulation, we only need to invert a $n_s \times n_s$ matrix (instead of the $d \times d$ matrix $\lambda\mathbf{I} + \mathbf{X}_\tau^\top\mathbf{X}_\tau$). In meta-learning, usually $n_s \ll d$ (e.g., $n_s = 5$). From the dual solution $\boldsymbol{\alpha}_\tau$, the primal solution can be recovered as $\mathbf{w}_\tau = \boldsymbol{\theta} + \lambda^{-1}\mathbf{X}_\tau^\top\boldsymbol{\alpha}_\tau$. Given a query example $(\mathbf{x}, y) \in \mathcal{Q}_\tau$, the model predicts $\hat{y} = \mathbf{x}^\top\mathbf{w}_\tau = \mathbf{x}^\top\boldsymbol{\theta} + \lambda^{-1}\mathbf{x}^\top\mathbf{X}_\tau^\top\boldsymbol{\alpha}_\tau$. The loss gradient is $\nabla_{\boldsymbol{\theta}}\ell(\hat{y}, y) = \nabla_1\ell(\hat{y}, y)\nabla_{\boldsymbol{\theta}}\hat{y}$, where $\nabla_1\ell(\hat{y}, y)$ denotes the gradient w.r.t. the first argument, and $\nabla_{\boldsymbol{\theta}}\hat{y} = \mathbf{x} + \lambda^{-1}(\nabla_{\boldsymbol{\theta}}\boldsymbol{\alpha}_\tau)^\top\mathbf{X}_\tau\mathbf{x}$, $\nabla_{\boldsymbol{\theta}}\boldsymbol{\alpha}_\tau = -(\mathbf{I} + \lambda^{-1}\mathbf{X}_\tau\mathbf{X}_\tau^\top)^{-1}\mathbf{X}_\tau$. The complexity of computing $\nabla_{\boldsymbol{\theta}}\ell(\hat{y}, y)$ is thus very low ($\mathcal{O}(n_s^3 + n_s^2 d)$).

The dual formulation also allows the introduction of nonlinearity with the kernel trick. Based on deep kernels [247], recent state-of-the-arts (R2D2 [11], MetaOptNet [117], and DKT [169]) propose to use a base kernel in the base learner and update the deep network in the meta-learner. However, their regularizers are not learnable. In this work, we propose learning a proximal regularizer for the base learner (Algorithm 3). 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 $\mathbf{z}_i = \mathsf{NN}(\mathbf{x}_i; \boldsymbol{\phi}_{t-1})$ for $\mathbf{x}_i \in \mathcal{S}_\tau$. $\mathcal{K}$ 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:

$$\hat{f}_\tau = \arg\min_{f_\tau \in \mathcal{H}} \sum\nolimits_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \ell(f_\tau(\mathbf{z}_i), y_i) + \frac{\lambda}{2} \| f_\tau - f_{\boldsymbol{\theta}_{t-1}} \|^2_{\mathcal{H}}. \tag{3.3}$$

By the representer theorem [205], the solution of (3.3) is $f_\tau = f_{\boldsymbol{\theta}} + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \alpha_{\tau,i} \mathcal{K}_{\mathbf{z}_i}$, where $\mathcal{K}_{\mathbf{z}_i} = \mathcal{K}(\mathbf{z}_i, \cdot) \in \mathcal{K}$, and $\boldsymbol{\alpha}_\tau = [\alpha_{\tau,1}; \dots; \alpha_{\tau,n_s}]$ is obtained from the convex program

$$\min_{\boldsymbol{\alpha}_\tau} \sum\nolimits_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \ell(f_\tau(\mathbf{z}_i), y_i) + \boldsymbol{\alpha}_\tau^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau) \boldsymbol{\alpha}_\tau, \tag{3.4}$$

where $\mathbf{Z}_\tau = [\mathbf{z}_1^\top; \dots; \mathbf{z}_{n_s}^\top]$, and $\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)$ is the kernel matrix. Note that the hyperparameter λ in (3.3) is absorbed into $\mathbf{z}$ as the network is learnable. With the square loss, the dual solution of (3.4) is $\boldsymbol{\alpha}_\tau = (\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1}(\mathbf{y}_\tau - f_{\boldsymbol{\theta}}(\mathbf{Z}_\tau))$, where $f_{\boldsymbol{\theta}}(\mathbf{Z}_\tau) = [f_{\boldsymbol{\theta}}(\mathbf{z}_1); \dots; f_{\boldsymbol{\theta}}(\mathbf{z}_{n_s})]$. For general loss functions, the dual problem has no closed-form solution, but this has only n_s variables (which is usually small) and can be solved efficiently.

After the base learner has obtained the dual solution $\boldsymbol{\alpha}_\tau$, the meta-learner updates $f_{\boldsymbol{\theta}}$ and network parameter $\boldsymbol{\phi}$ by one gradient descent step on the validation loss $\sum_{(\mathbf{x}, y) \in \mathcal{Q}_\tau} \ell(\hat{y}, y)$, where $\hat{y} \equiv f_\tau(\mathbf{z}) = f_{\boldsymbol{\theta}}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top \boldsymbol{\alpha}_\tau$. By the chain rule, $\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \ell(\hat{y}, y) = \nabla_1 \ell(\hat{y}, y) \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \hat{y}$. The first component $\nabla_1 \ell(\hat{y}, y)$ can be computed directly and the second component is

$$\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \hat{y} = \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} f_{\boldsymbol{\theta}}(\mathbf{z}) + (\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}))^\top \boldsymbol{\alpha}_\tau + (\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \boldsymbol{\alpha}_\tau)^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}). \tag{3.5}$$

Both $\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} f_{\boldsymbol{\theta}}(\mathbf{z})$ and $\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})$ can be obtained by direct differentiation. By the chain rule, we have $\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \boldsymbol{\alpha}_\tau = \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau \nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \mathbf{p}$, where $\mathbf{p} = [f_{\boldsymbol{\theta}}(\mathbf{z}_1); \dots; f_{\boldsymbol{\theta}}(\mathbf{z}_{n_s}); \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_1); \dots; \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_{n_s})] \in \mathbb{R}^{n_s + n_s^2}$ is the input to the dual problem. $\nabla_{(\boldsymbol{\theta}, \boldsymbol{\phi})} \mathbf{p}$ can be directly computed. When the square loss is used, $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau = -(\mathbf{I} + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau))^{-1} \left[\mathbf{I} \;\middle|\; \mathbf{I} \otimes \boldsymbol{\alpha}_\tau^\top \right]$. For a general loss, $\boldsymbol{\alpha}_\tau$ is obtained by solving the convex program. Hence, $\boldsymbol{\alpha}_\tau$ depends implicitly on $\mathbf{p}$ and $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau$ can be obtained by implicit differentiation. Denote the dual objective in (3.4) by $g(\mathbf{p}, \boldsymbol{\alpha})$. By the implicit function theorem [198], $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau = -\left(\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)\right)^{-1} \frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)$, where $\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \nabla_1^2 \ell(f_\tau(\mathbf{z}_i), y_i) \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_i) \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_i)^\top + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)$, $\frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) = [\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\mathbf{D} \mid (\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)\mathbf{D}) \otimes \boldsymbol{\alpha}_\tau^\top + \mathbf{v}^\top \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\alpha}_\tau^\top]$, $\mathbf{D} = \operatorname{diag}([\nabla_1^2 \ell(f_\tau(\mathbf{z}_1), y_1); \dots; \nabla_1^2 \ell(f_\tau(\mathbf{z}_{n_s}), y_{n_s})])$, $\mathbf{v} = [\nabla_1 \ell(f_\tau(\mathbf{z}_1), y_1); \dots; \nabla_1 \ell(f_\tau(\mathbf{z}_{n_s}), y_{n_s})]$, where $\otimes$ is the Kronecker product. The whole procedure, called MetaProx, is shown in Algorithm 3. Let $n_{\boldsymbol{\phi}}$ and $n_{\boldsymbol{\theta}}$ be the numbers of parameters in $\boldsymbol{\phi}$

Algorithm 3 MetaProx.

```
Require: stepsize $\eta_t$, 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 $(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau$;
        $f_\tau(\mathbf{z};\boldsymbol{\alpha}) \equiv f_{\boldsymbol{\theta}_{t-1}}(\mathbf{z}) + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})^\top \boldsymbol{\alpha}$ denote the task model w.r.t. dual variables;
        $\boldsymbol{\alpha}_\tau = \arg\min_{\boldsymbol{\alpha}} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}_\tau} \ell(f_\tau(\mathbf{z}_i; \boldsymbol{\alpha}), y_i) + \boldsymbol{\alpha}^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau) \boldsymbol{\alpha}$;
    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 $(\boldsymbol{\theta}_T, \boldsymbol{\phi}_T)$.
```

and $\boldsymbol{\theta}$, respectively. Computing $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\ell(\hat{y}, y)$ takes $\mathcal{O}(n_s^3 + n_s^2(n_{\boldsymbol{\theta}} + n_{\boldsymbol{\phi}}))$ time, which is linear in the number of meta-parameters. This is lower than the other meta-learning algorithms (e.g., MAML [51] with single step takes $\mathcal{O}(n_\phi^2)$ time, iMAML [188]: $\mathcal{O}(n_\phi^3)$, CommonMean [34]: $\mathcal{O}(d^3)$).

The proposed 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 [117].

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 [188].

3.4 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 $\mathbf{E}_{\tau\sim p(\tau)}\sum_{(\mathbf{x},y)\in Q_\tau}\ell(f_{\boldsymbol{\theta}}(\mathbf{z};\boldsymbol{\alpha}_\tau), y)$, where $\mathbf{z} = \mathsf{NN}(\mathbf{x};\boldsymbol{\phi})$. With the linear kernel and square loss, the dual solution (3.2) is affine in the meta-parameter, and so is the primal 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 [15, 34]. 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 [62, 192] and meta-learning in non-convex settings [47, 284].

Assumption 3.4.1 (Smoothness)**.** (i) The deep network $\mathsf{NN}(\mathbf{x};\boldsymbol{\phi})$ is Lipschitz-smooth, i.e., $\|\nabla_{\boldsymbol{\phi}}\mathsf{NN}(\mathbf{x};\boldsymbol{\phi}) - \nabla_{\boldsymbol{\phi}}\mathsf{NN}(\mathbf{x};\boldsymbol{\phi}')\| \leq \eta_1\|\boldsymbol{\phi} - \boldsymbol{\phi}'\|$ with a Lipschitz constant $\eta_1 > 0$; (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) $\mathbb{E}_{\tau\sim\mathcal{T}}\|\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\sum_{(\mathbf{x},y)\in Q_\tau}\ell(f_{\boldsymbol{\theta}}(\mathbf{z};\boldsymbol{\alpha}_\tau), y) - \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})\|^2 = \sigma_{\mathbf{g}}^2$, where $\tau \sim \mathcal{T}$ denotes uniformly sample a task from $\mathcal{T}$; (v) $\nabla_1^2\ell(\hat{y}, y)$ is Lipschitz w.r.t. $\hat{y}$, i.e., $|\nabla_1^2\ell(\hat{y}, y) - \nabla_1^2\ell(\hat{y}', y)| \leq \eta_2|\hat{y} - \hat{y}'|$ with a Lipschitz constant $\eta_2 > 0$.

The following Lemma guarantees the smoothness of the meta-loss.

Lemma 3.4.2. *$\mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is Lipschitz-smooth w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi})$ with a Lipschitz constant η_{meta}.*

Theorem 3.4.1. *Let the step size be $\eta_t = \min(1/\sqrt{T}, 1/2\eta_{meta})$. Algorithm 3 satisfies*

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

where the expectation is taken over the random training samples.

This rate is the same as MAML [47, 95] and Meta-MinibatchProx [284]. For MAML with $J > 1$ gradient steps, [95] 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 [52, 284] 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 [234] studies the global convergence of MAML in over-parameterized neural networks. Over-parameterization is closely related to the assumption of uniform conditioning [90, 116, 134].

Assumption 3.4.3 (Uniform conditioning [134])**.** A multivariable function $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is μ-uniformly conditioning if its tangent kernel [90] satisfies

$$\min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \lambda_{\min}(\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})}^{\top}\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})) \geq \mu > 0,$$

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 3.4.1 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 $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \left[\hat{y}_{\tau_1,1}; \dots; \hat{y}_{\tau_1,n_q}; \dots; \hat{y}_{\tau_{|\mathcal{T}|},1}; \dots; \hat{y}_{\tau_{|\mathcal{T}|},n_q}\right]$ 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 3.4.2. *Assume $\mathcal{M}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is uniform conditioning. (i) Let $\eta_t = \min(1/\sqrt{T}, 1/2\eta_{meta})$. Algorithm 3 satisfies $\min_{1\le t\le T} \mathbb{E}\mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) - \min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{O}\left(\sigma_{\mathbf{g}}^2/\sqrt{T}\right)$, where the expectation is taken over the random training samples. (ii) Let $\eta_t = \eta < \min(1/2\eta_{meta}, 4|\mathcal{T}|/\rho\mu)$ and $\mathcal{B}_t = \mathcal{T}$. Algorithm 3 satisfies $\mathcal{L}_{meta}(\boldsymbol{\theta}_t, \boldsymbol{\phi}_t) - \min_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{L}_{meta}(\boldsymbol{\theta}, \boldsymbol{\phi}) = \mathcal{O}((1-\eta\rho\mu/4|\mathcal{T}|)^t)$.*

3.5 Experiments on Few-shot Regression

Data sets. Experiments are performed on three data sets.

(i) *Sine*. This is the sinusoid regression problem in [51]. 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 [51], 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 [221], 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 [64] 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 [169] 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 [169], we do not use a meta-validation set since the dataset is small.

Network Architecture. For *Sine* and *Sale*, we use the network in [51], 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 [169]. 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 [108] 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 [169] 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.

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)

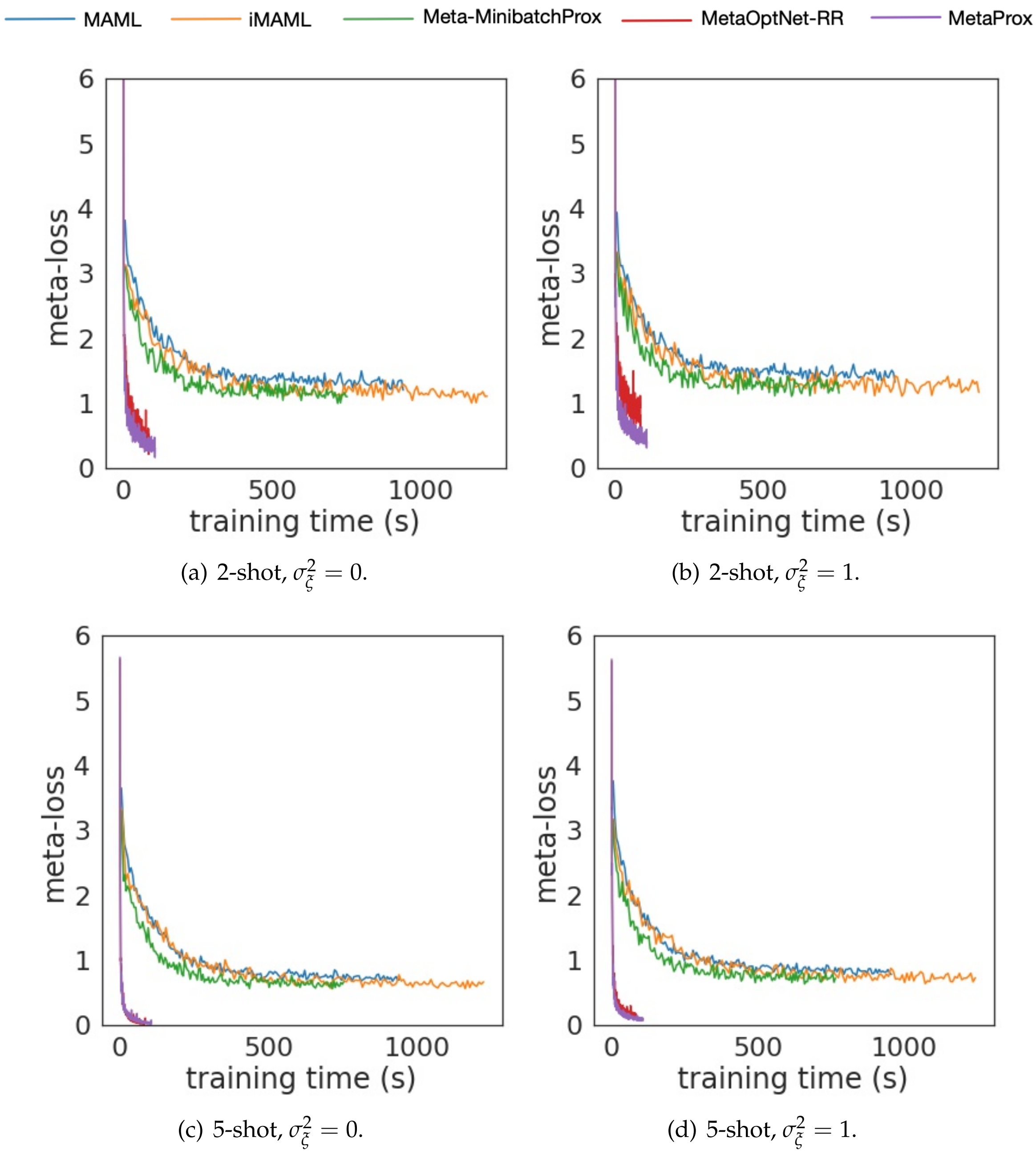
	Sine (2-shot)		*Sine* (5-shot)		*Sale*	
	noise-free	noisy	noise-free	noisy	1-shot	5-shot
CommonMean [34]	4.58 ± 0.07	4.59 ± 0.07	4.29 ± 0.06	4.31 ± 0.06	0.090	0.074
MAML [51]	1.24 ± 0.12	1.91 ± 0.13	0.41 ± 0.03	1.15 ± 0.05	0.069	0.063
iMAML [188]	1.12 ± 0.11	1.84 ± 0.10	0.38 ± 0.02	1.02 ± 0.05	0.068	0.063
Meta-MinibatchProx [284]	1.15 ± 0.08	1.87 ± 0.09	0.37 ± 0.02	1.01 ± 0.03	0.081	0.064
MetaOptNet-RR [117]	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**

Baselines. On *Sine* and *Sale*, we compare MetaProx with CommonMean [34], MAML [51], MetaOptNet-RR [117], Meta-MinibatchProx [284], and iMAML [188]. CommonMean is a linear model, and MetaOptNet-RR is equivalent to MetaProx when $f_{\boldsymbol{\theta}} = 0$. Following [51], 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 [284] and iMAML [188] are meta-regularization approaches. For *QMUL*, we compare MetaProx with the baselines reported in [169] (i.e., DKT [169], Feature Transfer [41], and MAML). As further baselines, we also compare with Meta-MinibatchProx and MetaOptNet-RR to evaluate the improvement of MetaProx due to the learnable $f_{\boldsymbol{\theta}}$.

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 outperforms the other baselines. MetaProx (with the learned $f_{\boldsymbol{\theta}}$) performs better than MetaOptNet-RR, particularly when the data is noisy.



(a) 2-shot, $\sigma_\xi^2 = 0$.

(b) 2-shot, $\sigma_\xi^2 = 1$.

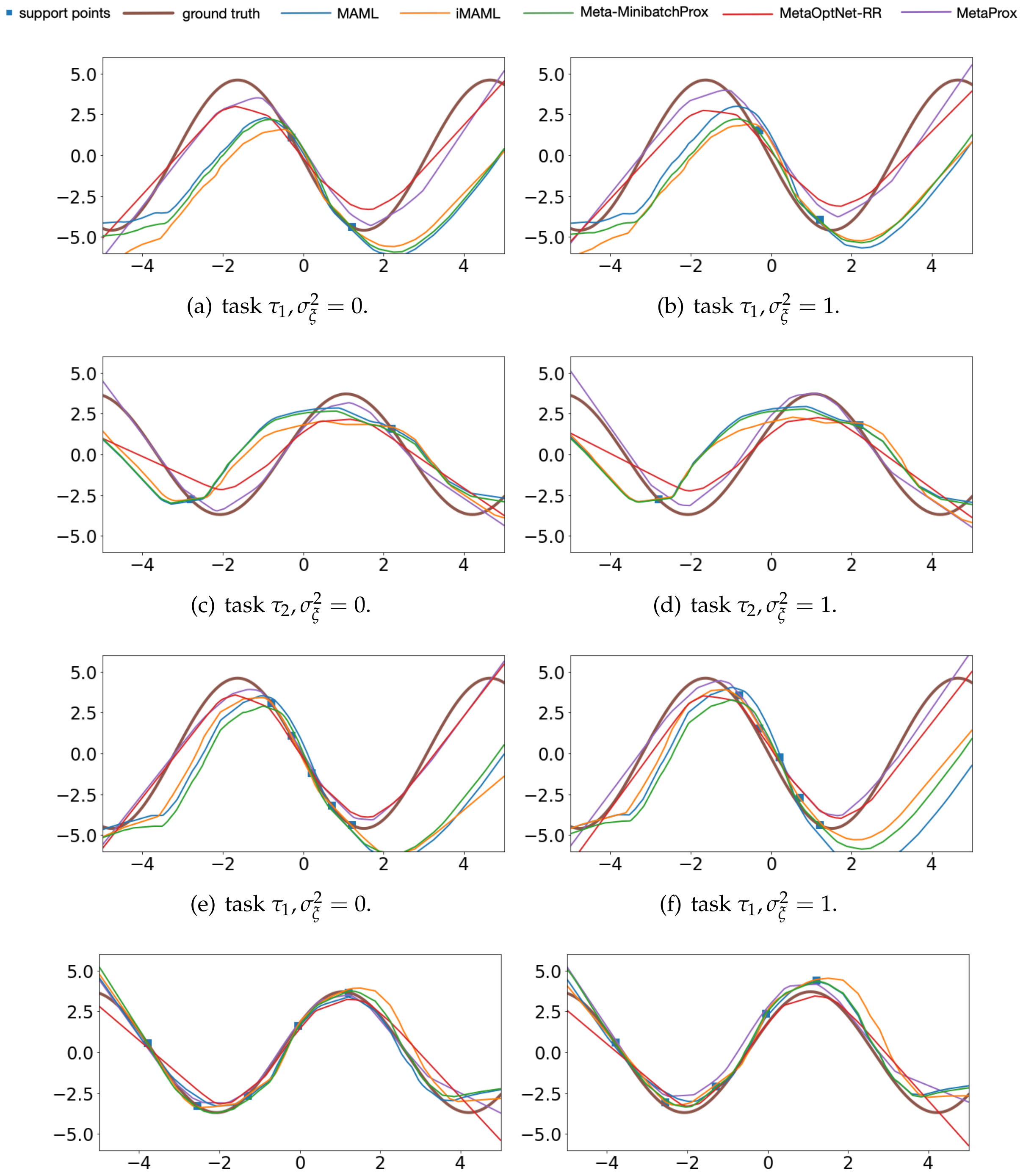
(c) 5-shot, $\sigma_\xi^2 = 0$.

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

Figure 3.1: Convergence curves for few-shot sinusoid regression.

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 non-kernel-based methods (Feature Transfer, MAML, and Meta-MinibatchProx). MetaProx with the learnable f_θ reduces the errors of MetaOptNet-RR by half.



(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.

3.6 Experiments on Few-shot Classification

Datasets. We use the standard 5-way K-shot setting ($K = 1$ or 5) on *mini-ImageNet* [231], which consists of 100 randomly chosen classes from *ILSVRC*-2012 [199]. Each class contains 600 84×84 images. We use the commonly-used split in [191]: the 100 classes are

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 [169].

	in-range	out-of-range
Feature Transfer [41]	0.22 ± 0.03	0.18 ± 0.01
MAML [51]	0.21 ± 0.01	0.18 ± 0.02
DKT + RBF [169]	0.12 ± 0.04	0.14 ± 0.03
DKT + Spectral [169]	0.10 ± 0.02	0.11 ± 0.02
Meta-MinibatchProx [284]	0.171 ± 0.022	0.193 ± 0.025
MetaOptNet-RR [117]	0.021 ± 0.007	0.039 ± 0.009
MetaProx	$\mathbf{0.012 \pm 0.003}$	$\mathbf{0.020 \pm 0.005}$

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 [51, 231] and *ResNet-12* in [117]. As the cosine similarity is more effective than ℓ_2 distance in few-shot classification [22], 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 $\mathcal{S}_\tau$, 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 [51] and its variants FOMAML [51], and REPTILE [157]; (ii) meta-regularization: iMAML [188] and Meta-MinibatchProx [284]; and (iii) metric learning: ANIL [186], R2D2 [11], ProtoNet [216], and MetaOptNet [117] with SVM using the linear kernel and cosine kernel.

Implementation Details. The entire model is trained end-to-end. $\ell(\hat{\mathbf{y}}, y)$ is the cross-entropy loss. The CVXPYLayers package [2] 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 [108] 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

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 [188]).

	1-shot	5-shot
MAML [51]	48.7 ± 1.8	63.1 ± 0.9
FOMAML [51]	48.1 ± 1.8	63.2 ± 0.9
REPTILE [157]	50.0 ± 0.3	66.0 ± 0.6
iMAML [188]	49.0 ± 1.8	–
Meta-MinibatchProx [284]	50.8 ± 0.9	67.4 ± 0.9
ANIL [186]	46.7 ± 0.4	61.5 ± 0.5
R2D2 [11]	49.5 ± 0.2	65.4 ± 0.3
ProtoNet [216]	49.4 ± 0.8	68.2 ± 0.7
MetaOptNet-SVM(lin)† [117]	49.8 ± 0.9	66.9 ± 0.7
MetaOptNet-SVM(cos)† [117]	50.1 ± 0.9	67.2 ± 0.6
MetaProx	$\mathbf{52.4} \pm 1.0$	$\mathbf{68.8} \pm 0.8$

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.

	1-shot	5-shot
FOMAML† [51]	57.41 ± 0.71	72.12 ± 0.54
ANIL† [186]	59.66 ± 0.68	73.28 ± 0.49
ProtoNet [216]	59.25 ± 0.64	75.60 ± 0.48
MetaOptNet-SVM(lin)† [117]	62.31 ± 0.64	78.21 ± 0.42
MetaOptNet-SVM(cos)† [117]	62.75 ± 0.42	78.68 ± 0.24
MetaProx	$\mathbf{63.82} \pm \mathbf{0.23}$	$\mathbf{79.12} \pm \mathbf{0.18}$

is better due to the learnable regularizer.

3.7 Conclusion

In this chapter, we propose MetaProx, an effective meta-regularization algorithm for meta-learning. MetaProx combines deep kernel and meta-regularization. By reformulating the inner problem in the dual space, a learnable proximal regularizer is introduced to the base learner. The meta-parameters in the regularizer and network are updated by the meta-learner. We also establish convergence of MetaProx. Extensive experiments on standard datasets for regression and classification verify the effectiveness of learning a proximal regularizer. Furthermore, MetaProx is more computationally efficient than existing non-kernel-based methods.

CHAPTER 4

Subspace Meta-Learning

4.1 Introduction

Typical meta-learning algorithms [51, 34, 188, 284] learn a globally-shared meta-model for all tasks. For example, the Model-Agnostic Meta-Learning (MAML) algorithm [51] learns a meta-initialization such that a good model for an unseen task can be finetuned from limited samples by a few gradient updates. MetaProx proposed in Chapter 3 seeks a common meta-regularization for all task models. However, when the task environments are complex and heterogeneous, task model parameters are diverse and a single meta-model may not be sufficient to capture all the meta-knowledge.

To tackle this issue, a variety of methods have been proposed to learn structured meta-knowledge by exploring the task structure [93, 259, 260, 285, 229]. For example, [93] formulate the task distribution as a mixture of hierarchical Bayesian models, and update the components (i.e., initializations) using an Expectation Maximization procedure. TSA-MAML [285] first trains task models using vanilla MAML. Tasks are grouped into clusters by k-means clustering, and cluster centroids form group-specific initializations.

Alternatively, task model parameters can be formulated into a subspace. In the linear regression setting where task model vectors are sampled from a low-dimensional subspace, recent attempts [111, 229] use a moment-based estimator to recover the subspace based on 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.

In this chapter, we propose a model-agnostic algorithm called MUSML (MUltiple Subspaces for Meta-Learning). Each subspace represents one type of meta-knowledge, and subspace bases are treated as meta-parameters. For each task, the base learner builds a task model from each subspace. The meta-learner then updates the subspace bases by minimizing a weighted validation loss of the task models. We theoretically establish upper bounds on the population risk, empirical risk and generalization gap. All these bounds depend on the complexity of the subspace mixture (number of component subspaces and subspace dimensionality). Experiments on various datasets verify the effectiveness of the proposed MUSML.

Our major contributions are four-fold: (i) We formulate task model parameters into a subspace mixture and propose a novel algorithm to learn the subspace bases. (ii) The proposed MUSML is model-agnostic and can be used on linear and nonlinear models. (iii) We theoretically study the generalization properties of the learned subspaces. (iv) We perform extensive experiments on synthetic and real-world datasets. Results on the synthetic dataset confirm that MUSML is able to discover the underlying subspaces of task model parameters. Results on the real-world datasets demonstrate superiority of MUSML over the state-of-the-arts.

4.2 Learning Multiple Subspaces for Meta-Learning

4.2.1 Linear Regression Tasks

We first focus on the linear setting, where the task is linear regression and all task parameters lie in one single (linear) subspace. The following proposition shows that the underlying subspace can be recovered using a moment-based estimator. All proofs of theoretical results in this chapter are in the Appendix.

Proposition 4.2.1. *[111, 229]. Assume that $p(\tau)$ is a distribution of linear regression tasks. Each task τ is associated with a $\mathbf{w}_\tau^\star \in \mathbb{R}^d$, and its samples are generated as $y = \mathbf{x}^\top \mathbf{w}_\tau^\star + \xi$, where $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\xi \sim \mathcal{N}(0, \sigma_\xi^2)$ is the noise. Then, $\mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{(\mathbf{x},y)\sim\tau,(\mathbf{x}',y')\sim\tau} y y' \mathbf{x}\mathbf{x}'^\top = \mathbb{E}_{\tau\sim p(\tau)} \mathbf{w}_\tau^\star \mathbf{w}_\tau^{\star\top}$.*

Proposition 4.2.1 shows that $\hat{\mathbf{S}} \equiv \frac{1}{|\mathcal{B}|}\sum_{\tau\in\mathcal{B}} y_\tau y_\tau' \mathbf{x}_\tau \mathbf{x}_\tau'^\top$ is an unbiased estimator of $\mathbb{E}_{\tau\sim p(\tau)} \mathbf{w}_\tau^\star \mathbf{w}_\tau^{\star\top}$, where $(\mathbf{x}_\tau, y_\tau)$ and $(\mathbf{x}_\tau', y_\tau')$ are two samples drawn from τ, and $\mathcal{B}$ is a collection of tasks. Hence, the column space of $\hat{\mathbf{S}}$ recovers the column space of $\mathbb{E}_{\tau\sim p(\tau)} \mathbf{w}_\tau^\star \mathbf{w}_\tau^{\star\top}$ (i.e., the underlying subspace) when the number of tasks is sufficient.

4.2.2 The Proposed MUSML

While Proposition 4.2.1 can be used to recover the column space in a linear meta-learning setting, extension to the nonlinear setting (such as deep networks) is difficult. To address this problem, we propose a model-agnostic algorithm called MUSML (MUltiple Subspaces for Meta-Learning). We assume that the model parameters $\mathbf{w}_\tau$'s lie in K subspaces $\{\mathbb{S}_1, \dots, \mathbb{S}_K\}$, which can be seen as an approximation to a nonlinear manifold. For simplicity, we assume that all K subspaces have the same dimensionality m (this can be easily extended to the case where the subspaces have different dimensionalities). Let $\mathbf{S}_k \in \mathbb{R}^{d \times m}$ be a basis of $\mathbb{S}_k$. $\{\mathbf{S}_1, \dots, \mathbf{S}_K\}$ are then the meta-parameters to be learned.

The proposed procedure is shown in Algorithm 4. Given a task τ, the base learner searches for the model parameter $\mathbf{w}_\tau$ over all subspaces with fixed $\mathbf{S}_k$ (steps 4-11). In each subspace $\mathbb{S}_k$, we search for the best linear combination $\mathbf{v}^\star_{\tau,k}$ of the subspace's basis to form $\mathbf{w}_\tau$:

$$\mathbf{v}^\star_{\tau,k} = \arg\min_{\mathbf{v}_\tau \in \mathbb{R}^m} \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}_\tau). \tag{4.1}$$

$\mathbf{S}_k \mathbf{v}^\star_{\tau,k}$ is then the task model parameter corresponding to the kth subspace. When $\ell(f(\mathbf{x}; \mathbf{w}), y)$ is convex in $\mathbf{w}$, it is easy to verify that $\mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}_\tau)$ is also convex in $\mathbf{v}_\tau$. Hence, problem (4.1) can be solved as a convex program [15]. However, for nonlinear models such as deep networks, the loss function in (4.1) is nonconvex, and thus finding $\mathbf{v}^\star_{\tau,k}$ is computationally intractable. Instead, we seek an approximate minimizer $\mathbf{v}_{\tau,k}$ by performing J gradient descent steps from an initialization $\mathbf{v}^{(0)}_{\tau,k}$, i.e.,

$$\mathbf{v}^{(t'+1)}_{\tau,k} = \mathbf{v}^{(t')}_{\tau,k} - \alpha \nabla_{\mathbf{v}^{(t')}_{\tau,k}} \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}^{(t')}_{\tau,k}),$$

where $\alpha > 0$ is the step size and $\mathbf{v}_{\tau,k} \equiv \mathbf{v}^{(J)}_{\tau,k}$.

At meta-training, one can assign τ to the subspace with the best training set performance. However, this is inefficient for learning meta-parameters since only one subspace is updated at each step. Similar to DARTS [135], we relax the categorical choice to a softmax selection over all candidate subspaces. The relaxed operation is differentiable and all subspace bases can then be updated simultaneously, which accelerates learning. Let $o_{\tau,k} = \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_k \mathbf{v}_{\tau,k})$ be the training loss for task τ when the kth subspace (where $k = 1, \dots, K$) is used to construct its task model. The meta-learner updates $\{\mathbf{S}_1, \dots, \mathbf{S}_K\}$ by performing one gradient update on the weighted validation loss (steps 13-14):

$$\sum_{k=1}^{K} \frac{\exp(-o_{\tau,k}/\gamma)}{\sum_{k'=1}^{K} \exp(-o_{\tau,k'}/\gamma)} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{S}_k \mathbf{v}_{\tau,k}), \tag{4.2}$$

where $\gamma > 0$ is the temperature. When γ is close to 0, the softmax selection becomes one-hot; whereas when γ increases to ∞, the selection becomes uniform. In practice,

Algorithm 4 MUltiple Subspaces for Meta-Learning (MUSML).

```
Require: stepsize $\alpha$, $\{\eta_t\}$; number of inner gradient steps $J$, number of subspaces
    $K$, subspace dimension $m$, temperature $\{\gamma_t\}$; initialization $\mathbf{v}^{(0)}$;
for $t = 0, 1, \ldots, T-1$ do
    sample a task $\tau$ with $\mathcal{S}_\tau$ and $\mathcal{Q}_\tau$;
    base learner:
    for $k = 1, \ldots, K$ do
        initialize $\mathbf{v}_{\tau,k}^{(0)} = \mathbf{v}^{(0)}$;
        for $t' = 0, 1, \ldots, J-1$ do
            $\mathbf{v}_{\tau,k}^{(t'+1)} = \mathbf{v}_{\tau,k}^{(t')} - \alpha \nabla_{\mathbf{v}_{\tau,k}^{(t')}} \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_{k,t}\mathbf{v}_{\tau,k}^{(t')})$;
        end for
        $\mathbf{v}_{\tau,k} \equiv \mathbf{v}_{\tau,k}^{(J)}$;
        $o_{\tau,k} = \mathcal{L}(\mathcal{S}_\tau; \mathbf{S}_{k,t}\mathbf{v}_{\tau,k})$;
    end for
    meta-learner:
    $\mathcal{L}_{vl} = \sum_{k=1}^{K} \frac{\exp(-o_{\tau,k}/\gamma_t)}{\sum_{k'=1}^{K} \exp(-o_{\tau,k'}/\gamma_t)} \mathcal{L}(\mathcal{Q}_\tau; \mathbf{S}_{k,t}\mathbf{v}_{\tau,k})$;
    $\{\mathbf{S}_{1,t+1}, \ldots, \mathbf{S}_{k,t+1}\} = \{\mathbf{S}_{1,t}, \ldots, \mathbf{S}_{k,t}\} - \eta_t \nabla_{\{\mathbf{S}_{1,t}, \ldots, \mathbf{S}_{k,t}\}} \mathcal{L}_{vl}$;
end for
Return $\mathbf{S}_{1,T}, \ldots, \mathbf{S}_{K,T}$.
```

we start at a high temperature and anneal to a small but nonzero temperature as in [91, 26, 286]. Note that $\{o_{\tau,k} : k = 1, \ldots, K\}$ depend on the bases and $\nabla_{\{\mathbf{S}_1, \ldots, \mathbf{S}_k\}} o_{\tau,k}$ can be computed by auto-differentiation.

At meta-testing, for each testing task τ', we assign τ' to the subspace with the lowest training loss, i.e., $\mathbf{w}_{\tau'} = \mathbf{S}_{k_{\tau'}} \mathbf{v}_{\tau', k_{\tau'}}$, where $k_{\tau'} \equiv \arg\min_{1 \le k \le K} \mathcal{L}(\mathcal{S}_{\tau'}; \mathbf{S}_k \mathbf{v}_{\tau',k})$ is the chosen subspace index.

4.3 Theoretical Analysis

In this section, we study the generalization performance of the learned subspace bases $\mathcal{S} \equiv \{\mathbf{S}_1, \ldots, \mathbf{S}_K\}$ at meta-testing. The following assumptions on smoothness and compactness are standard in meta-learning [8, 66, 47] and bilevel optimization [55, 8]. The boundedness assumption on the loss function is widely used in analyzing the generalization of meta-learning algorithms [149, 173, 3] and traditional machine learning algorithms [14].

Assumption 4.3.1. (i) $\ell(f(\mathbf{x}; \mathbf{w}), y)$ and $\nabla_{\mathbf{w}} \ell(f(\mathbf{x}; \mathbf{w}), y)$ are ϱ-Lipschitz and β-Lipschitz in $\mathbf{w}$, respectively;[1] (ii) $\{\mathbf{v}_{\tau,k} : \tau \sim p(\tau), k = 1, \ldots, K\}$ and column vectors of $\mathbf{S}_k$

[1] $\|\ell(f(\mathbf{x}; \mathbf{w}), y) - \ell(f(\mathbf{x}; \mathbf{w}'), y)\| \le \varrho \|\mathbf{w} - \mathbf{w}'\|$ and $\|\nabla_{\mathbf{w}} \ell(f(\mathbf{x}; \mathbf{w}), y) - \nabla_{\mathbf{w}} \ell(f(\mathbf{x}; \mathbf{w}'), y)\| \le \beta \|\mathbf{w} - \mathbf{w}'\|$.

($k = 1, \dots, K$) are in a compact set, and their ℓ_2-norms are upper bounded by a constant $\rho > 0$. (iii) $\ell(\cdot, \cdot)$ is upper bounded by a constant $\nu > 0$.

Let $\mathcal{R}(\mathcal{S}) \equiv \mathbb{E}_{\tau'} \mathbb{E}_{\mathcal{D}^{tr}_{\tau'}} \mathbb{E}_{(\mathbf{x},y)\sim\tau'} \ell(f(\mathbf{x}; \mathbf{S}_{k_{\tau'}} \mathbf{v}_{\tau',k_{\tau'}}), y)$ be the expected population risk, and $\hat{\mathcal{R}}(\mathcal{S}) \equiv \mathbb{E}_{\tau'} \mathbb{E}_{\mathcal{D}^{tr}_{\tau'}} \mathcal{L}(\mathcal{D}^{tr}_{\tau'}; \mathbf{S}_{k_{\tau'}} \mathbf{v}_{\tau',k_{\tau'}})$ be the expected empirical risk averaged over all tasks. The following Theorem characterizes the generalization gap, i.e., the gap between the population and empirical risks.

Theorem 4.3.1. *With Assumption 4.3.1, we have*

$$\mathcal{R}(\mathcal{S}) \le \hat{\mathcal{R}}(\mathcal{S}) + K\sqrt{\frac{\nu^2 + 12\varrho\nu(1 + m\alpha\delta)^J}{2n_s}}, \tag{4.3}$$

where $\delta = \beta\rho^2 > 0$.

The proof is based on the connection between generalization and stability [14]. The dependence on n_s in (4.3) agrees with their Theorem 11, as the stability constant in Algorithm 6 is of the order $\mathcal{O}(1/n_s)$. From (4.3), one can observe that increasing the subspace complexity (i.e., m or K) increases the upper bound of $\mathcal{R}(\mathcal{S}) - \hat{\mathcal{R}}(\mathcal{S})$.

For notation simplicity, throughout this section, we omit the superscript τ'. Let $\mathbf{z} \equiv (\mathbf{x}, y)$ be the samples, $\ell(\mathbf{z}; \mathbf{w}) \equiv \ell(f(\mathbf{x}; \mathbf{w}), y)$, and $\nabla_{\mathbf{w}} \ell(f(\mathbf{z}; \mathbf{S}\mathbf{v})) \equiv \nabla_{\mathbf{w}} \ell(f(\mathbf{z}; \mathbf{w})) \mid_{\mathbf{w}=\mathbf{S}\mathbf{v}}$.

We first show the stability constant (in Lemma 9 of [14]) in Algorithm 4 is of the order $\mathcal{O}(1/n_s)$.

Let $\{\mathbf{z}'_i : i = 1, \dots, n_s\}$ be another n_s samples from τ. Let $\mathcal{S}^{(i)}_\tau$ be another training set which differs from $\mathcal{S}_\tau$ only in the ith sample (i.e., $\mathcal{S}^{(i)}_\tau \equiv (\mathcal{S}_\tau - \{\mathbf{z}_i\}) \cup \{\mathbf{z}'_i\}$). We let $\mathbf{v}_{\tau,k,i}$ (resp. $\mathbf{v}_{\tau,k}$) be the task model obtained from the base learner when using the training set $\mathcal{S}^{(i)}_\tau$ (resp. $\mathcal{S}_\tau$).

Lemma 4.3.2 (Lemma 9 of [14]). *Let $\mathcal{D} = \{\mathbf{z}_1, \dots, \mathbf{z}_m\}$ be a dataset containing m samples. For any learning algorithm $\mathcal{A}$ (receives a training set and outputs a learned model) and loss function ℓ such that $0 \le \ell(\cdot) \le M$, we have*

$$\mathbb{E}_{\mathcal{D}} \left[\mathbb{E}_{\mathbf{z}} \ell(\mathbf{z}; \mathcal{A}(\mathcal{D})) - \mathcal{R}(\mathcal{D}; \mathcal{A}(\mathcal{D}))\right]^2 \le \frac{M^2}{2m} + 3M\mathbb{E}_{\mathcal{D},\mathbf{z}'_i} |\ell(\mathbf{z}_i; \mathcal{A}(\mathcal{D})) - \ell(\mathbf{z}_i; \mathcal{A}(\mathcal{D}^{(i)}))|$$

for any $i \in \{1, \dots, m\}$, where $\mathcal{D}^{(i)}$ is a dataset obtained by replacing $\mathbf{z}_i$ with $\mathbf{z}'_i$, and $\mathcal{R}(\mathcal{D}; \mathcal{A}(\mathcal{D}))$ is the empirical risk.

Lemma 4.3.3. *For the base learner in Algorithm 4, we have*

$$\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}_i'\sim\tau}\left|\ell(f(\mathbf{x}_i;\mathbf{S}_k\mathbf{v}_{\tau,k}),y_i)-\ell(f(\mathbf{x}_i;\mathbf{S}_k\mathbf{v}_{\tau,k,i}),y_i)\right|\le\frac{2\varrho(1+\alpha\beta\rho^2m)^J}{n_s}.$$

Let $\mathbf{w}^\star_{\tau'}\equiv\arg\min_{\mathbf{w}_{\tau'}}\mathbb{E}_{(\mathbf{x},y)\sim\tau'}\ell(f(\mathbf{x};\mathbf{w}_{\tau'}),y)$ be the optimal task model for task τ', and $\mathcal{R}^\star\equiv\mathbb{E}_{\tau'}\mathbb{E}_{(\mathbf{x},y)\sim\tau'}\ell(f(\mathbf{x};\mathbf{w}^\star_{\tau'}),y)$ be the minimum expected loss averaged over all tasks. The following Theorem provides an upper bound on the expected excess risk [285] $\mathcal{R}(\mathcal{S})-\mathcal{R}^\star$, which compares the performance of the learned task model with that of the optimal model.

Theorem 4.3.2. *With Assumption 4.3.1, we have*

$$\mathcal{R}(\mathcal{S})-\mathcal{R}^\star\le\rho\sqrt{m}\,\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}}\|\mathbf{v}_{\tau',k_{\tau'}}-\mathbf{v}^\star_{\tau',k_{\tau'}}\|$$
$$+\varrho\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}}\mathrm{dist}(\mathbf{w}^\star_{\tau'},\mathbb{S}_{k_{\tau'}})+K\sqrt{\frac{\nu^2+12\varrho\nu(1+m\alpha\delta)^J}{2n_s}},$$

where $\mathrm{dist}(\mathbf{w}^\star_{\tau'},\mathbb{S}_{k_{\tau'}})\equiv\min_{\mathbf{w}\in\mathbb{S}_{k_{\tau'}}}\|\mathbf{w}-\mathbf{w}^\star_{\tau'}\|$ *is the distance between* $\mathbf{w}^\star_{\tau'}$ *and* $\mathbb{S}_{k_{\tau'}}$.

From Theorem 4.3.2, $\mathcal{R}(\mathcal{S})-\mathcal{R}^\star$ is upper-bounded by three terms: (i) The first term measures the distance between the approximate minimizer $\mathbf{v}_{\tau',k_{\tau'}}$ and exact minimizer $\mathbf{v}^\star_{\tau',k_{\tau'}}$; (ii) The second term arises from the approximation error of $\mathbf{w}^\star_{\tau'}$ using the learned subspaces; (iii) The third term depends on the complexity of subspaces (i.e., m and K). For the centroid-based clustering method in [285], the upper bound of its expected excess risk contains a term $\mathbb{E}_{\tau'}\mathbb{E}_{\mathcal{D}^{tr}_{\tau'}}\|\boldsymbol{\omega}_{k^\star_{\tau'}}-\mathbf{w}^\star_{\tau'}\|^2$, where $\boldsymbol{\omega}_{k^\star_{\tau'}}$ is the centroid of the cluster that τ' is assigned to. The distance $\|\boldsymbol{\omega}_{k^\star_{\tau'}}-\mathbf{w}^\star_{\tau'}\|^2$ plays the same role as the term $\mathrm{dist}(\mathbf{w}^\star_{\tau'},\mathbb{S}_{k_{\tau'}})$ in Theorem 4.3.2, which measures how far the optimal model $\mathbf{w}^\star_{\tau'}$ is away from the subspaces or clusters.

4.4 Experiments on Few-shot Regression

4.4.1 Synthetic Data

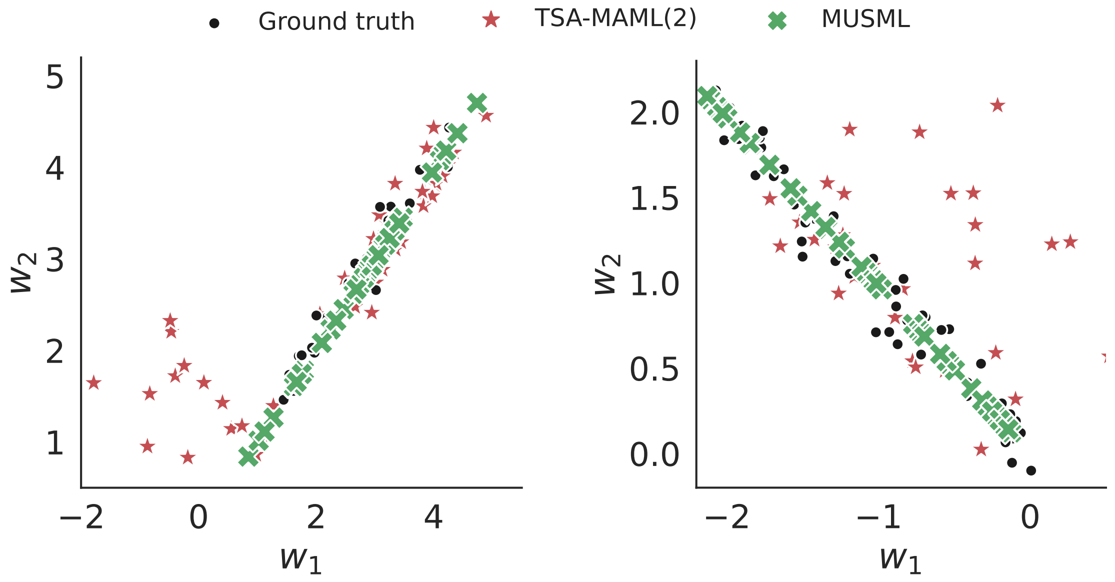
In this experiment, we use a synthetic 1-dimensional data set to examine whether MUSML can discover subspaces in which the task model parameters lie. We use a 5-shot regression setting, with 14,000 meta-training, 2,000 meta-validation, and 6,000 meta-testing tasks. The model for task τ is $f(x;\mathbf{w}_\tau) = \exp(0.1 w_{\tau,1} x) + w_{\tau,2} |\sin(x)|$, in which $\mathbf{w}_\tau = [w_{\tau,1}; w_{\tau,2}]$ is randomly sampled from one of the two subspaces: (i) *Line-A*: $\mathbf{w}_\tau = \mathbf{S}_1 a_\tau + 0.1\boldsymbol{\xi}_\tau$, where $\mathbf{S}_1 = [1;1]$, $a_\tau \sim \mathcal{U}(1,5)$, and $\boldsymbol{\xi}_\tau \sim \mathcal{N}(\mathbf{0};\mathbf{I})$; and (ii) *Line-B*: $\mathbf{w}_\tau = \mathbf{S}_2 a_\tau + 0.1\boldsymbol{\xi}_\tau$, where $\mathbf{S}_2 = [-1;1]$, $a_\tau \sim \mathcal{U}(0,2)$, and $\boldsymbol{\xi}_\tau \sim \mathcal{N}(\mathbf{0};\mathbf{I})$. The samples of task τ are generated as $y = f(x;\mathbf{w}_\tau) + 0.05\xi$, where $x \sim \mathcal{U}(-5,5)$ and $\xi \sim \mathcal{N}(0,1)$. The experiment is repeated 10 times with different seeds.

The subspace bases are trained for $T = 30,000$ iterations using the Adam optimizer [108]. For the meta-learner, the initial learning rate is 0.001, which is then reduced by half every 5,000 iterations. The base learner uses a learning rate of $\alpha = 0.05$, $\mathbf{v}^{(0)} = \frac{1}{m}\mathbf{1}$, and J is 5 (resp. 20) at meta-training (resp. meta-testing). The temperature is $\gamma_t = \max(10^{-5}, 0.5 - t/T)$, a linear annealing schedule as in [26, 286]. To prevent overfitting, we evaluate the meta-validation performance every 2,000 iterations, and stop training when the meta-validation accuracy has no significant improvement for 5 consecutive evaluations.

The proposed MUSML (with $K = 2, m = 1$) is compared with the following meta-learning baselines: (i) MAML [51], (ii) BMG [54], which uses target bootstrap, and structured meta-learning algorithms, including (iii) Dirichlet process mixture model (DPMM) [93], (iv) hierarchically structured meta-learning (HSML) [259], (v) automated relational meta-learning (ARML) [260] using a graph structure, and (vi) task similarity aware meta-learning (TSA-MAML) [285] with different numbers of clusters. We use these baselines' official implementations (except for DPMM and BMG whose imple-

Table 4.1: Meta-testing MSE (with standard deviation) of 5-shot regression on synthetic data. For TSA-MAML, the number in brackets is the number of clusters used.

MAML [51]	0.74 ± 0.03
BMG [54]	0.67 ± 0.03
DPMM [93]	0.56 ± 0.09
HSML [259]	0.49 ± 0.10
ARML [260]	0.60 ± 0.07
TSA-MAML(2) [285]	0.58 ± 0.10
TSA-MAML(10) [285]	0.24 ± 0.09
TSA-MAML(20) [285]	0.12 ± 0.10
TSA-MAML(40) [285]	0.14 ± 0.09
TSA-MAML(80) [285]	0.13 ± 0.08
MUSML	$\mathbf{0.07} \pm 0.01$



(a) Tasks from *Line-A*.

(b) Tasks from *Line-B*.

Figure 4.1: Visualization of task model parameters.

Table 4.2: Average Euclidean distance (with standard deviation) between the estimated task model parameters and ground-truth in 5-shot setting on synthetic data. For TSA-MAML, the number in brackets is the number of clusters used.

MAML [51]	1.69 ± 0.02
BMG [54]	1.55 ± 0.03
DPMM [93]	0.85 ± 0.10
HSML [259]	0.80 ± 0.09
ARML [260]	0.91 ± 0.11
TSA-MAML(2) [285]	0.88 ± 0.12
TSA-MAML(10) [285]	0.47 ± 0.19
TSA-MAML(20) [285]	0.33 ± 0.18
TSA-MAML(40) [285]	0.36 ± 0.19
TSA-MAML(80) [285]	0.36 ± 0.18
MUSML	$\mathbf{0.17} \pm 0.01$

mentations are not publicly available). For performance evaluation, the mean squared error (MSE) on the meta-testing set is used.

Results. Table 4.1 shows the meta-testing MSE. As can be seen, structured meta-learning methods (DPMM, HSML, ARML, TSA-MAML, and MUSML) are significantly better than methods with a globally-shared meta-model (MAML and BMG). In particular, MUSML performs the best. Furthermore, simply increasing the number of clusters in TSA-MAML fails to beat MUSML.

Table 4.3: Meta-testing MSE (with standard deviation) of 15-shot regression on *Pose*. Results on MAML and MR-MAML are from [265].

MAML [51]	5.39 ± 1.31
MR-MAML [265]	2.26 ± 0.09
BMG [54]	2.16 ± 0.15
DPMM [93]	1.99 ± 0.08
HSML [259]	2.04 ± 0.13
ARML [260]	2.21 ± 0.15
TSA-MAML [285]	1.96 ± 0.07
MUSML	$\mathbf{1.83} \pm 0.05$

Figure 4.1 visualizes the task model parameters obtained by TSA-MAML(2) and MUSML on 100 randomly sampled meta-testing tasks (50 per subspace). As can be seen, MUSML successfully discovers the underlying subspaces, while the centroid-based clustering method TSA-MAML does not. Table 4.2 shows the average Euclidean distance between the estimated task model parameters and the ground truth. As can be seen, MUSML is more accurate in estimating the task models, confirming the effectiveness of the learned subspaces.

4.4.2 *Pose* Data

While the synthetic data used in the previous experiment is tailored for the proposed subspace model, in this section, we perform experiments on a real-world pose prediction dataset from [265]. This is created based on the Pascal 3D data [250]. Each object contains 100 samples, where input $\mathbf{x}$ is a 128×128 grey-scale image and output y is its orientation relative to a fixed canonical pose. Following [265], we adopt a 15-shot regression setting and randomly select 50 objects for meta-training, 15 for meta-validation, and 15 for meta-testing. The experiment is repeated 15 times with different seeds.

The MR-MAML regularization [265] is used on all the methods except the vanilla MAML. MUSML uses the same encoder-decoder network in [265] as the model $f(\mathbf{x}; \mathbf{w})$. Hyperparameters K and m, as well as the number of clusters in TSA-MAML, are chosen from 1 to 5 using the meta-validation set. For performance evaluation, the MSE on the meta-testing set is used.

Table 4.3 shows the meta-testing MSE. As can be seen, MUSML is again better than the other baselines, confirming the effectiveness of the learned subspaces.

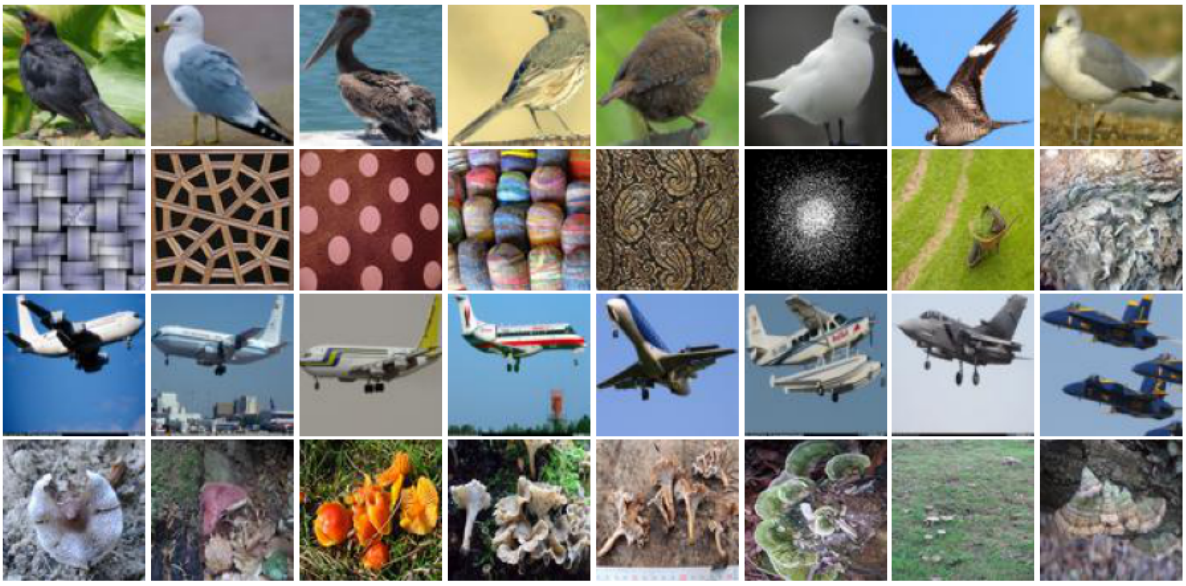


Figure 4.2: Some random images from the meta-testing set of *Meta-Dataset-BTAF* (Top to bottom: *Bird*, *Texture*, *Aircraft*, and *Fungi*).

Table 4.4: Statistics of the datasets.

	#classes (meta-training/meta-validation/meta-testing)
Bird	64/16/20
Texture	30/7/10
Aircraft	64/16/20
Fungi	64/16/20
CIFAR-FS	64/16/20
mini-ImageNet	64/16/20
Omniglot	71/15/16

4.5 Experiments on Few-shot Classification

4.5.1 Experimental Setup

Setup. In this experiment, we use three meta-datasets: (i) *Meta-Dataset-BTAF*, proposed in [259], which consists of four image classification datasets: (a) *Bird*; (b) *Texture*; (c) *Aircraft*; and (d) *Fungi*. Sample images are shown in Figure 4.2. (ii) *Meta-Dataset-ABF*, proposed in [285], which consists of *Aircraft*, *Bird*, and *Fungi*. (iii) *Meta-Dataset-CIO*, which consists of three widely-used few-shot datasets: *CIFAR-FS* [11], *mini-ImageNet* [231], and *Omniglot* [113]. We use the meta-training/meta-validation/meta-testing splits in [260, 285, 113]. A summary of the datasets is in Table 4.4.

As for the network architecture, we use the standard *Conv4* backbone [51, 260, 285], and a simple prototype classifier with cosine similarity on top [216, 63] as $f(\mathbf{x}; \mathbf{w})$. Hyperparameters K and m are chosen from 1 to 5 on the meta-validation set.

We use the cross-entropy loss for $\ell(\cdot, \cdot)$. The number of parameters in Conv4 is $113,088$. For the base learner, $\alpha = 0.01$, $\mathbf{v}^{(0)} = \frac{1}{m}\mathbf{1}$, and J is set to 5 (resp. 15) at meta-training (resp. meta-validation or meta-testing). We train the subspace bases for $T = 100,000$ iterations using the Adam optimizer [108] with an initial learning rate of 0.001, which is then reduced by half every $5,000$ iterations. The temperature is set to $\gamma_t = \max(10^{-5}, 0.8 - t/T)$, which is again a linear annealing schedule [26, 286]. To prevent overfitting, we evaluate the meta-validation performance every $2,000$ iteration and stop training when the meta-validation accuracy has no significant improvement for 5 consecutive evaluations. Hyperparameters K and m are chosen from 1 to 5 on the meta-validation set. In practice, as shown in Section 4.5.5, m can simply be fixed at 2, and K can be chosen from 3 to 4. As the search space of K is small, the additional cost of tuning K and m is small.

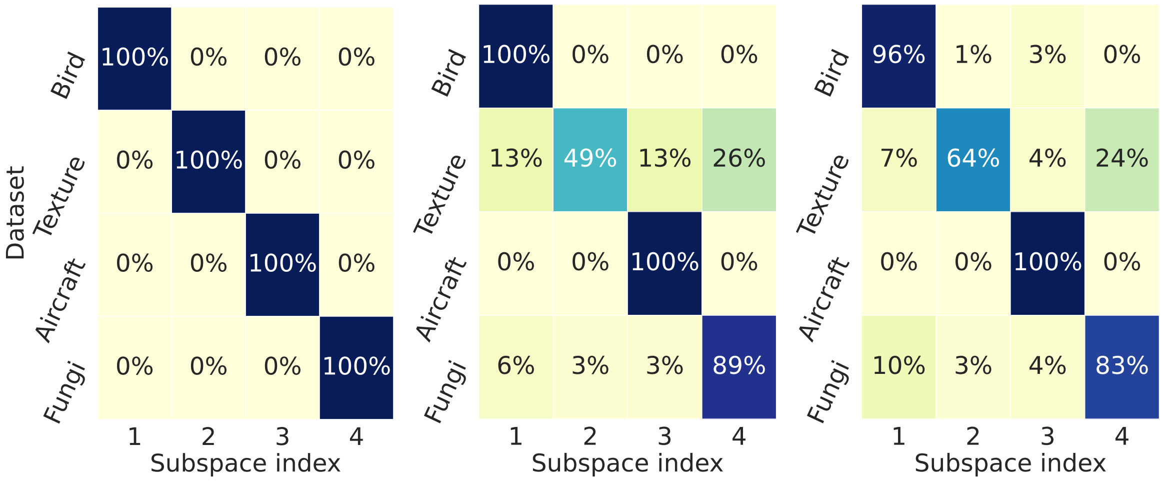
MUSML is compared with the following state-of-the-arts in the 5-way 5-shot and 5-way 1-shot settings: (i) meta-learning algorithms with a globally-shared meta-model, including MAML, ProtoNet, ANIL [186], and BMG; (ii) structured meta-learning algorithms, including DPMM, HSML, ARML, TSA-MAML and its variant using ProtoNet as the base learner (denoted TSA-ProtoNet). The number of clusters in TSA-MAML and TSA-ProtoNet are tuned from 1 to 5 on the meta-validation set. For performance evaluation, the classification accuracy on the meta-testing set is used. The experiment is repeated 5 times with different seeds.

4.5.2 *Meta-Dataset-BTAF*

Table 4.5 shows the 5-shot results. As can be seen, MUSML is more accurate than both structured and unstructured meta-learning methods, demonstrating the benefit of structuring task model parameters into subspaces. Figure 4.3 shows the assignment of tasks to the learned subspaces in MUSML. As can be seen, meta-training tasks from the same dataset are always assigned to the same subspace, demonstrating that MUSML can discover the task structure from meta-training tasks. Though the meta-validation and meta-testing classes are not seen during meta-training, most of the corresponding tasks are still assigned to the correct subspaces. The assignment for *Texture* is slightly

Table 4.5: 5-way 5-shot accuracy (with 95% confidence interval) on *Meta-Dataset-BTAF*. Results marked with † are from [260].

	Bird	*Texture*	*Aircraft*	*Fungi*	average
MAML† [51]	68.52 ± 0.73	44.56 ± 0.68	66.18 ± 0.71	51.85 ± 0.85	57.78
ProtoNet [216]	71.48 ± 0.72	50.36 ± 0.67	71.67 ± 0.69	55.68 ± 0.82	62.29
ANIL [186]	70.67 ± 0.72	44.67 ± 0.95	66.05 ± 1.07	52.89 ± 0.30	58.57
BMG [54]	71.56 ± 0.76	49.44 ± 0.73	66.83 ± 0.79	52.56 ± 0.89	60.10
DPMM [93]	72.22 ± 0.70	49.32 ± 0.68	73.55 ± 0.69	56.82 ± 0.81	63.00
TSA-MAML [285]	72.31 ± 0.71	49.50 ± 0.68	74.01 ± 0.70	56.95 ± 0.80	63.20
HSML† [259]	71.68 ± 0.73	48.08 ± 0.69	73.49 ± 0.68	56.32 ± 0.80	62.39
ARML† [260]	73.68 ± 0.70	49.67 ± 0.67	74.88 ± 0.64	57.55 ± 0.82	63.95
TSA-ProtoNet [285]	73.70 ± 0.73	50.91 ± 0.74	73.55 ± 0.78	56.11 ± 0.82	63.57
MUSML	$\mathbf{76.79} \pm 0.72$	$\mathbf{52.41} \pm 0.75$	$\mathbf{77.76} \pm 0.82$	$\mathbf{57.74} \pm 0.81$	**66.18**



(a) Meta-training. (b) Meta-validation. (c) Meta-testing.

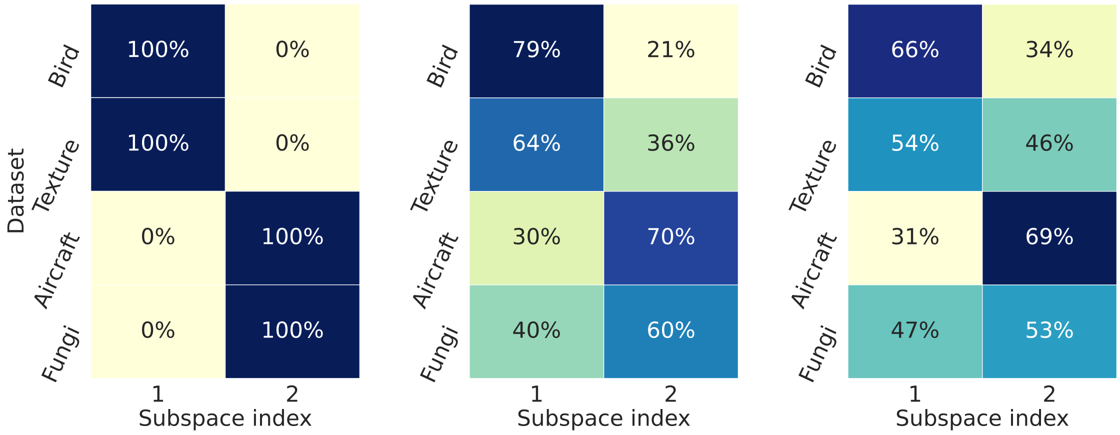
Figure 4.3: Task assignment to the learned subspaces in 5-way 5-shot setting on *Meta-Dataset-BTAF* (the number of subspaces K selected by the meta-validation set is 4). Darker color indicates higher percentage.

worse, as the *Texture* and *Fungi* images are more similar to each other (Figure 4.2).

Table 4.6 shows the 1-shot results. MUSML, while still the best overall, has a smaller improvement than in the 5-shot setting. This suggests that having more training samples is beneficial for the base learner to choose a proper subspace. The assignment of tasks to the learned subspaces is shown in Figure 4.4.

Table 4.6: 5-way 1-shot accuracy (with 95% confidence interval) on *Meta-Dataset-BTAF*. Results marked with † are from [260].

	Bird	*Texture*	*Aircraft*	*Fungi*	average
MAML† [51]	53.94 ± 1.45	31.66 ± 1.31	51.37 ± 1.38	42.12 ± 1.36	44.77
ProtoNet [216]	60.37 ± 1.31	40.57 ± 0.78	52.83 ± 0.93	44.10 ± 1.36	49.50
ANIL [186]	53.36 ± 1.42	31.91 ± 1.25	52.87 ± 1.34	42.30 ± 1.28	45.11
BMG [54]	54.12 ± 1.46	32.19 ± 1.21	52.09 ± 1.35	43.00 ± 1.37	45.35
DPMM [93]	61.30 ± 1.47	35.21 ± 1.35	57.88 ± 1.37	43.81 ± 1.45	49.55
TSA-MAML [285]	61.37 ± 1.42	35.41 ± 1.39	**58.78** ± 1.37	44.17 ± 1.25	49.93
HSML† [259]	60.98 ± 1.50	35.01 ± 1.36	57.38 ± 1.40	44.02 ± 1.39	49.35
ARML† [260]	**62.33** ± 1.47	35.65 ± 1.40	58.56 ± 1.41	44.82 ± 1.38	50.34
TSA-ProtoNet [285]	60.41 ± 1.02	40.98 ± 1.20	53.29 ± 0.89	43.91 ± 1.31	49.64
MUSML	60.52 ± 0.33	**41.33** ± 1.30	54.69 ± 0.69	**45.60** ± 0.43	**50.53**



(a) Meta-training. (b) Meta-validation. (c) Meta-testing.

Figure 4.4: Task assignment to the learned subspaces in 5-way 1-shot on *Meta-Dataset-BTAF* (K selected by meta-validation set is 2).

4.5.3 *Meta-Dataset-ABF* and *Meta-Dataset-CIO*

Tables 4.7 and 4.8 show the results on *Meta-Dataset-ABF* and *Meta-Dataset-CIO*, respectively. Here, we only consider the 5-shot setting, which is more useful for subspace learning. As can be seen, MUSML consistently outperforms centroid-based clustering methods (DPMM, TSA-MAML, TSA-ProtoNet) and structured meta-learning methods (HSML, ARML). MUSML again outperforms methods with a globally-shared meta-model (MAML, ProtoNet, ANIL, BMG), confirming the effectiveness of using a subspace mixture. The performance of MUSML on *Omniglot* is slightly worse in Table 4.8. This may be due to the fact that *Omniglot* is a simple dataset and a single meta-model is good

Table 4.7: Accuracy (with 95% confidence interval) of 5-way 5-shot classification on *Meta-Dataset-ABF*. Results marked with † are from [285].

	Aircraft	*Bird*	*Fungi*	average
MAML† [51]	67.82 ± 0.65	70.55 ± 0.77	53.20 ± 0.82	63.86
ProtoNet[216]	69.74 ± 0.64	71.46 ± 0.69	55.66 ± 0.68	65.62
ANIL [186]	69.24 ± 0.87	70.34 ± 1.20	53.71 ± 0.67	64.43
BMG [54]	69.75 ± 0.72	73.04 ± 0.77	54.61 ± 0.84	65.80
DPMM [93]	70.22 ± 0.69	73.28 ± 1.33	54.28 ± 1.01	66.26
TSA-MAML† [285]	72.84 ± 0.63	74.80 ± 0.76	56.86 ± 0.67	68.17
HSML† [259]	69.89 ± 0.90	68.99 ± 1.01	53.63 ± 1.03	64.17
ARML [260]	70.20 ± 0.91	69.12 ± 1.01	54.23 ± 1.07	64.52
TSA-ProtoNet [285]	74.42 ± 0.62	75.11 ± 0.72	56.77 ± 0.69	68.77
MUSML	$\mathbf{79.88} \pm 0.61$	$\mathbf{75.63} \pm 0.73$	$\mathbf{57.80} \pm 0.80$	**71.10**

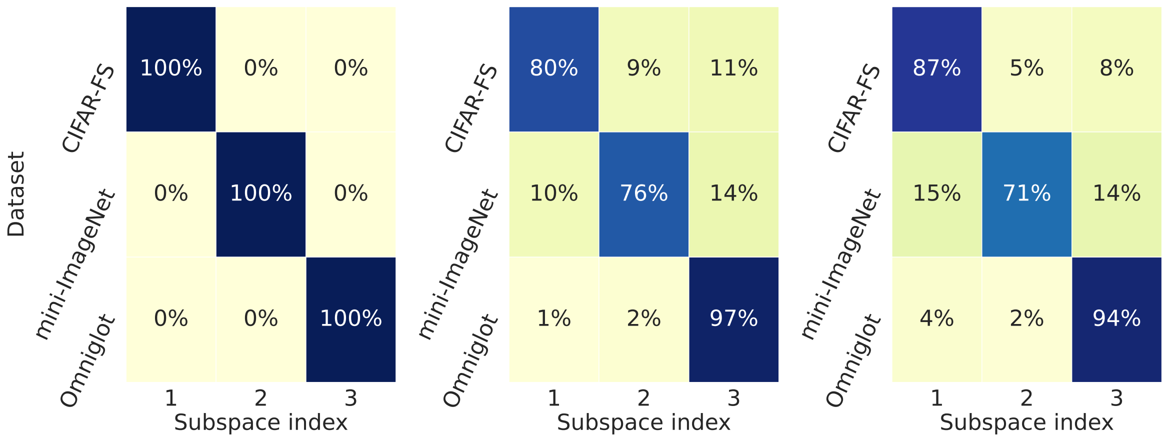
Table 4.8: Accuracy (with 95% confidence interval) of 5-way 5-shot classification on *Meta-Dataset-CIO*.

	CIFAR-FS	*mini-ImageNet*	*Omniglot*	average
MAML [51]	66.28 ± 1.61	60.20 ± 1.20	96.91 ± 0.39	74.46
ProtoNet [216]	71.32 ± 1.54	62.90 ± 1.07	95.32 ± 0.25	76.51
ANIL [186]	66.08 ± 0.90	60.62 ± 0.94	97.13 ± 0.13	74.61
BMG [54]	70.49 ± 1.22	63.97 ± 1.19	$\mathbf{97.92} \pm 0.42$	77.46
DPMM [93]	69.84 ± 1.42	62.92 ± 1.28	97.14 ± 0.28	76.63
TSA-MAML [285]	71.11 ± 1.55	62.57 ± 1.31	96.99 ± 0.31	76.89
HSML [259]	69.24 ± 1.57	62.28 ± 1.23	95.10 ± 0.32	75.54
ARML [260]	68.88 ± 1.91	63.26 ± 1.33	96.23 ± 0.31	76.12
TSA-ProtoNet [285]	72.37 ± 1.46	63.23 ± 1.52	96.21 ± 0.33	77.27
MUSML	$\mathbf{73.25} \pm 1.42$	$\mathbf{65.12} \pm 1.48$	95.13 ± 0.28	**77.83**

enough. As shown in Figure 4.5, its meta-validation and meta-testing tasks are often assigned to the same subspace.

4.5.4 Cross-Domain Few-Shot Classification

We examine the effectiveness of MUSML on cross-domain few-shot classification, which is more challenging as the testing domain is unseen at meta-training. We perform 5-way 5-shot classification, where *Meta-Dataset-BTAF* is used for meta-training, and *Meta-Dataset-CIO* for meta-testing. Table 4.9 shows the meta-testing accuracy. As can be



(a) Meta-training. (b) Meta-validation. (c) Meta-testing.

Figure 4.5: Task assignment to the learned subspaces in 5-way 5-shot setting on *Meta-Dataset-CIO* (K selected by the meta-validation set is 3). Darker color indicates higher percentage.

Table 4.9: Accuracy of cross-domain 5-way 5-shot classification (*Meta-Dataset-BTAF* $\rightarrow$ *Meta-Dataset-CIO*).

MAML	ProtoNet	ANIL	BMG	DPMM	TSA-MAML	HSML	ARML	TSA-ProtoNet	MUSML
64.25	66.13	65.19	66.98	66.73	66.85	65.18	65.37	66.92	**67.41**

seen, MUSML is also effective in unseen domains.

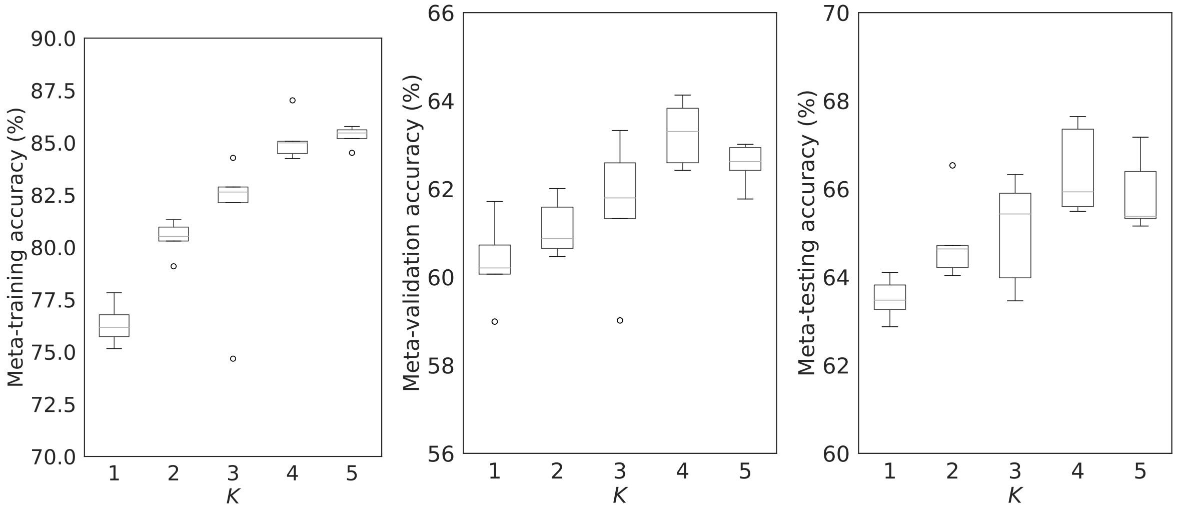
4.5.5 Effects of K and m

In this experiment, we study the effects of K and m on the 5-shot performance of MUSML on *Meta-Dataset-BTAF*. Figure 4.6(a) shows that the meta-training accuracy increases with K. However, a large $K = 5$ is not advantageous at meta-validation (Figure 4.6(b)) and meta-testing (Figure 4.6(c)).

Figures 4.7(b) and 4.7(c) show that the meta-validation and meta-testing accuracies of MUSML increase when m increases from 1 to 2, but larger m's ($m = 3, 4, 5$) lead to worse performance. This is because the obtained task model parameters ($\mathbf{W}$) lie close to the union of 2-dimensional subspaces[2], and so a larger m does not improve performance. Figure 4.8 also shows that for the 4 subspaces, the first 2 singular values of $\mathbf{W}$ are dominant.

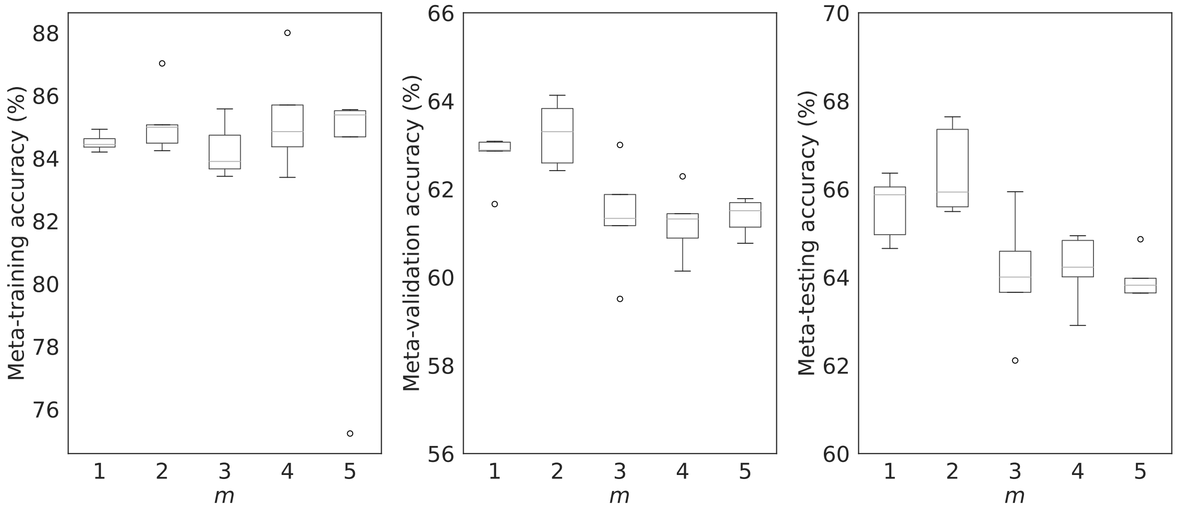
To demonstrate the theoretical results in Section 4.3, we further study the effects of K and m on the meta-testing loss. The average training (resp. testing) loss of meta-testing

[2] For example, for the $\mathbf{W}$ solution obtained with $m = 5$ on *Meta-Dataset-BTAF* (under 5-way 5-shot setting), approximation by a rank-2 matrix $\hat{\mathbf{W}}$ leads to a relative error ($\|\mathbf{W} - \hat{\mathbf{W}}\|_F / \|\mathbf{W}\|_F$) of only 4.1%.



(a) Meta-training. (b) Meta-validation. (c) Meta-testing.

Figure 4.6: 5-way 5-shot classification accuracy on *Meta-Dataset-BTAF* with varying K (m is fixed at 2).



(a) Meta-training. (b) Meta-validation. (c) Meta-testing.

Figure 4.7: 5-way 5-shot classification accuracy on *Meta-Dataset-BTAF* with varying m (K is fixed at 4).

tasks is an empirical estimate of $\hat{\mathcal{R}}(\mathcal{S})$ (resp. $\mathcal{R}(\mathcal{S})$), while their gap measures the generalization performance.

Figure 4.9(a) shows that, for $m \geq 2$, increasing K leads to a reduction in the training loss. However, the testing loss does not always decrease when K increases (Figure 4.9(b)). Figure 4.9(c) shows that a large K or m may enlarge the generalization gap, which justifies Theorem 4.3.1. As shown in Figure 4.9(c), the generalization gap is approximately linear with K, which agrees with the relationship between the upper bound of $\mathcal{R}(\mathcal{S}) - \hat{\mathcal{R}}(\mathcal{S})$ and K in Theorem 4.3.1.

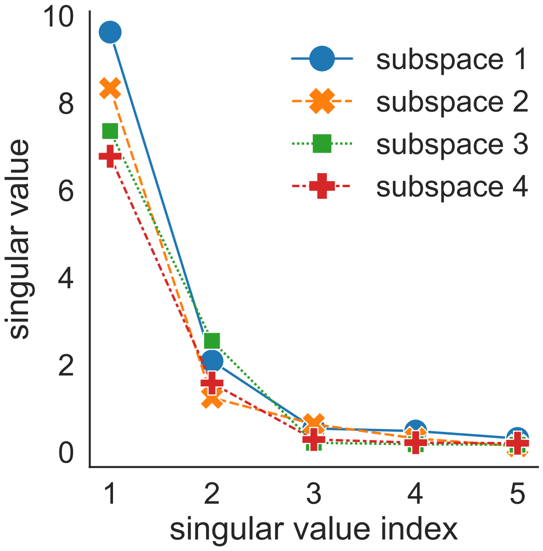
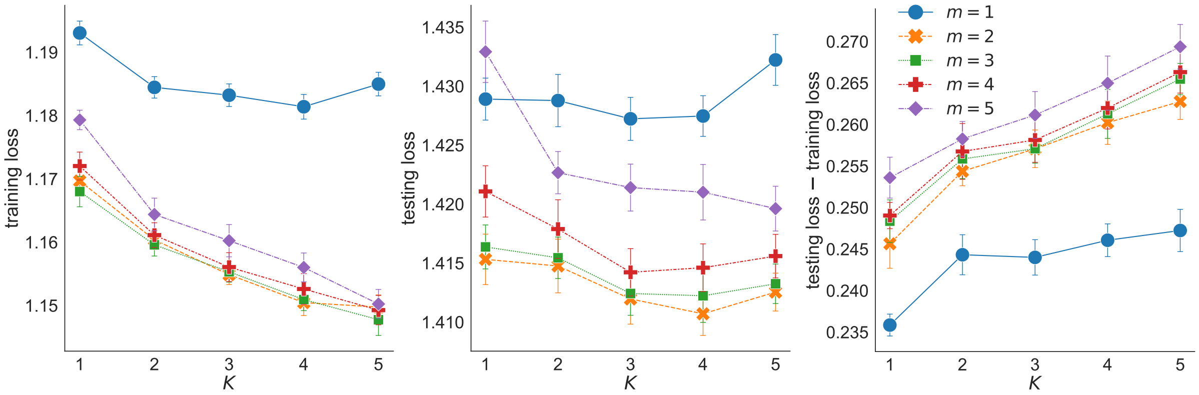


Figure 4.8: Singular values of model parameters of meta-testing tasks under the 5-way 5-shot setting on *Meta-Dataset-BTAF* ($K = 4$ and $m = 5$).



(a) Training loss. (b) Testing loss. (c) Generalization gap.

Figure 4.9: Effects of K and m on the training loss, testing loss, and generalization gap (with 95% confidence interval) of meta-testing tasks under the 5-way 5-shot setting on *Meta-Dataset-BTAF*.

Table 4.10: Accuracy of 5-way 5-shot classification on *Meta-Dataset-BTAF*.

γ	0.0001	0.001	0.01	0.1	1.0	2.0	MUSML
accuracy	51.22	60.12	61.15	63.16	62.11	62.02	**66.18**

4.5.6 Effects of Temperature Scaling Schedule

The temperature schedule used is linear annealing as in DynamicConvolution [26] and ProbMask [286]. We conduct a 5-way 5-shot experiment on *Meta-Dataset-BTAF* to evaluate MUSML with a constant temperature. Table 4.10 reports the meta-testing accuracy. We can see that using a constant γ is inferior.

Table 4.11: Accuracy of 5-way 5-shot classification on meta-datasets.

	Meta-Dataset-BTAF	*Meta-Dataset-ABF*	*Meta-Dataset-CIO*
Meta-SGD [127]	58.93	64.19	75.95
MUSML-SGD	65.72	69.15	77.48
Meta-Curvature [167]	60.02	64.51	76.13
MUSML-Curvature	**66.10**	**69.23**	**77.96**

4.5.7 Improving Existing Meta-Learning Approaches

As the proposed MUSML is general, a subspace mixture is also useful for other meta-learning approaches. In this experiment, we combine MUSML with Meta-Curvature [167] and Meta-SGD [127]. Table 4.11 reports 5-way 5-shot accuracies on meta-datasets. As can be seen, MUSML is beneficial for both Meta-Curvature and Meta-SGD.

4.6 Conclusion

In this chapter, we formulate task model parameters into a subspace mixture and propose a model-agnostic meta-learning algorithm with subspace learning called MUSML. For each task, the base learner builds a task model from each subspace, while the meta-learner updates the meta-parameters by minimizing a weighted validation loss. The generalization performance is theoretically studied. Experimental results on benchmark datasets for classification and regression validate the effectiveness of the proposed MUSML.

CHAPTER 5

Structured Prompting by Meta-Learning

Meta-learning is a general machine learning tool and has been successfully used in various applications, such as computer vision [51, 216, 11, 175, 104, 255, 239] and natural language processing [160, 261, 18, 72, 251, 264, 126]. This chapter introduces an application of meta-learning to prompt tuning.

5.1 Introduction

In recent years, large pretrained language models have achieved great success in solving a variety of downstream tasks [83, 38, 258, 32, 217, 69, 185, 16, 119, 33]. Though finetuning the whole model [83, 38] is effective and widely-used, optimizing and storing all the task-specific parameters can be computation- and memory-expensive when the model is large (e.g., GPT-3 [16] contains 100+ billion parameters). To alleviate this issue, many parameter-efficient finetuning approaches have been proposed. Examples include adapter tuning [82, 131, 85] and prompt learning [183, 211, 16, 119, 139, 125, 140, 180, 136]. Prompt learning is preferable due to its effectiveness and also that it can be easily plugged into a pretrained MLM without invasive modification [125, 71, 76, 218].

Prompt learning formulates the downstream task as a cloze-style MLM problem. It is useful for few-shot tasks due to its effectiveness, parameter-efficiency, and plug-and-play nature [183, 16, 136]. Specifically, prompt learning wraps an input text with a discrete *prompt* (e.g., "`Topic is [MASK]`") and feeds it to the MLM to predict a token at the `[MASK]` position. A *verbalizer* [119, 40, 86] then maps the predicted token to the label. However, designing an effective prompt requires a good understanding of the downstream tasks.

Recently, *prompt tuning* [119, 139, 275] proposes to wrap the input embedding with a *continuous* prompt. To reduce the number of parameters to be learned, the MLM is kept frozen. The continuous prompt can be further combined with discrete tokens to form a *template* [139, 204, 40].

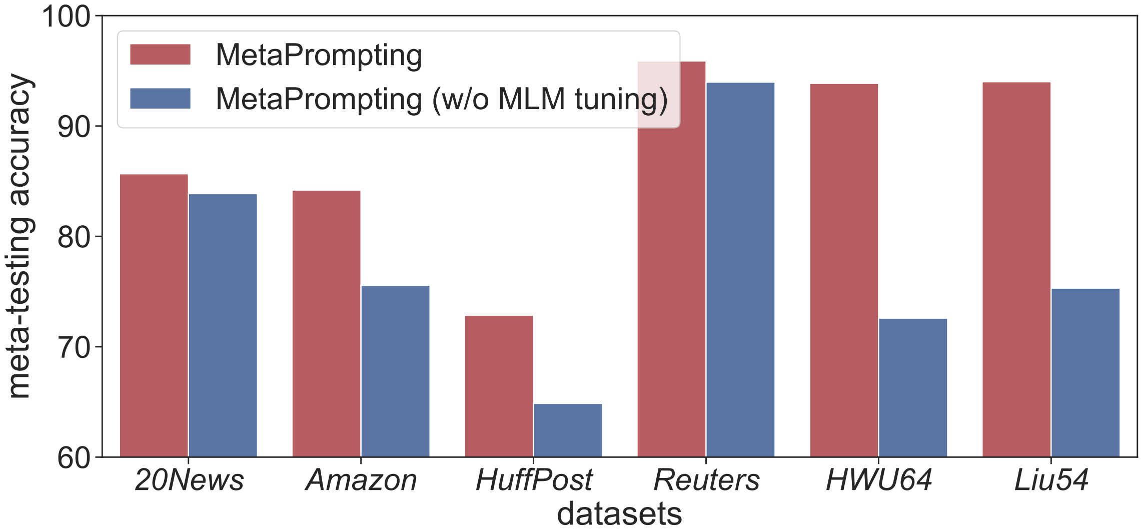


Figure 5.1: 5-way 5-shot classification meta-testing accuracy of MetaPrompting with or without MLM tuning on six data sets.

Prompt tuning is sensitive to the prompt initialization [119]. Recently, a number of approaches have been proposed to alleviate this problem [119, 122, 232]. In particular, MetaPrompting [81] is the state-of-the-art that uses meta-learning [223, 51] to learn a meta-initialization for all task-specific prompts. However, MetaPrompting 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-initialized prompt. (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 5.1 shows a large gap in meta-testing accuracies with and without MLM tuning (experimental details are in Section 5.3).

In this chapter, we use a pool of multiple prompts [122, 240, 241] to extract task knowledge from meta-training tasks, and then construct instance-dependent prompts as weighted combinations of all the prompts in the pool via attention [230]. The attention's query vector is the instance's feature embedding. The prompt pool is the shared meta-knowledge and learned by the MAML algorithm [51]. Specifically, given a task with a support set and a query set, the base learner takes the meta-parameter and the support set to build a task-specific prompt pool, then the meta-learner optimizes the meta-parameter on the query set. Meta-learning a prompt pool is more flexible than meta-learning only a single prompt initialization (as in MetaPrompting), and allows better adaptation of complex tasks. Moreover, as only the prompt pool is tuned, it is much more parameter-efficient than MetaPrompting (with 1000× fewer parameters).

We also propose a novel soft verbalizer called *representative verbalizer* (RepVerb), which constructs label embeddings by averaging feature embeddings of the corresponding training samples. Unlike manually-designed verbalizers, RepVerb does not incur human effort for label token selection. Moreover, as RepVerb does not require learning any additional parameters, empirical results in Section 5.3.2 demonstrate that RepVerb is more effective than the soft verbalizers in WARP [71], DART [275], ProtoVerb [33]. Besides, the feature embedding learned by RepVerb is more discriminative.

The whole procedure, which combines meta-learning the structured prompts and RepVerb, is called **MetaPrompter** in the sequel. Experiments are performed on six widely used classification data sets. Results demonstrate that RepVerb outperforms existing soft verbalizers, and is also beneficial to other prompt-based methods such as MetaPrompting. Moreover, MetaPrompter achieves better performance than the recent state-of-the-arts.

Our contributions are summarized as follows: (i) We propose a parameter-efficient algorithm MetaPrompter for effective structured prompting. (ii) We propose a simple and effective soft verbalizer (RepVerb). (iii) Experimental results demonstrate the effectiveness and parameter-efficiency of MetaPrompter.

5.2 The Proposed MetaPrompter

In this section, we propose a simple and effective soft verbalizer (representative verbalizer) without inducing additional parameters (Section 5.2.1). Moreover, while MetaPrompting uses a single meta-initialized prompt to build task-specific prompts, we propose in section 5.2.2 the extraction of task knowledge into a pool of multiple prompts, and constructs instance-dependent prompts by attention [230]. Figure 5.2 shows an overview of the proposed MetaPrompter.

5.2.1 Representative Verbalizer

Instead of explicitly learning an embedding $\mathbf{v}_y$ for each label y [71, 33, 275], we propose the *Representative Verbalizer* (RepVerb), which constructs $\mathbf{v}_y$ from feature embeddings of the corresponding training samples (Algorithm 5). It does not require learning additional parameters, and is thus more effective on limited data as in few-shot learning.

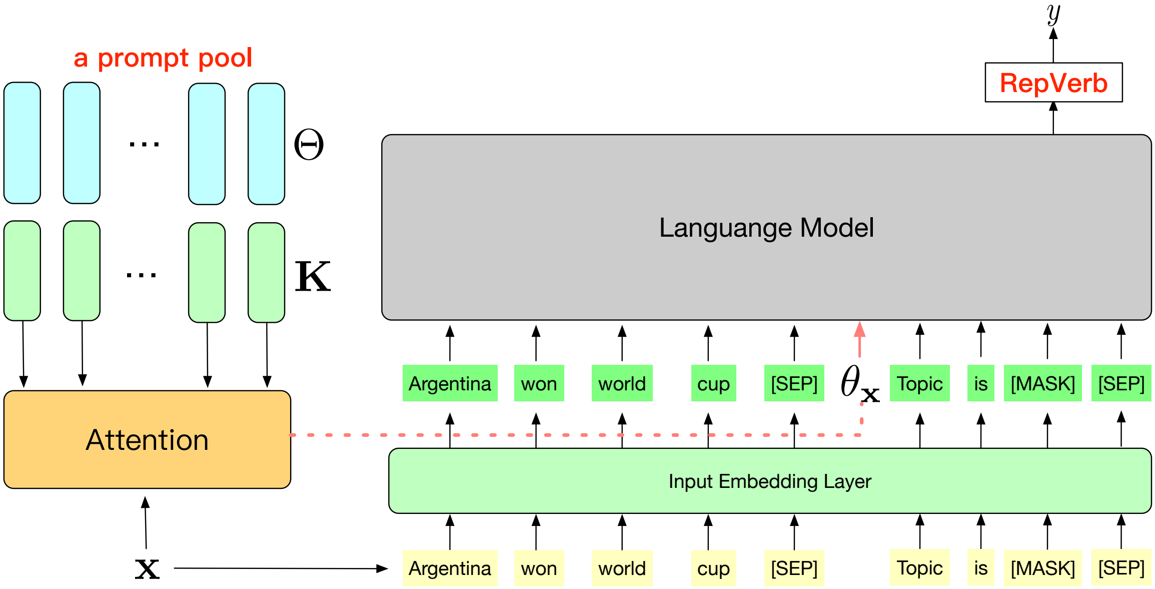


Figure 5.2: Overview of MetaPrompter..

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 pretrained MLM, and then obtain `[MASK]`'s embedding $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ as its feature embedding. Similar to ProtoNet [216], we propose to construct $\mathbf{v}_y$ for each y by averaging the corresponding samples' feature embeddings, as:

$$\mathbf{v}_y = \frac{1}{|\mathcal{S}_{\tau,y}|} \sum_{(\mathbf{x},y)\in\mathcal{S}_{\tau,y}} \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}}). \tag{5.1}$$

To predict the label of a given $\mathbf{x}$, we measure the cosine similarity[1] between $\mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})$ and each $\mathbf{v}_y$ ($y \in \mathcal{Y}_\tau$):

$$\tilde{\mathbb{P}}(y|\mathbf{x};\boldsymbol{\phi},\boldsymbol{\theta}) = \frac{\exp(\rho\cos(\mathbf{v}_y, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}{\sum_{y'\in\mathcal{Y}_\tau}\exp(\rho\cos(\mathbf{v}_{y'}, \mathbf{h}_{\texttt{[MASK]}}(\tilde{\mathbf{x}})))}, \tag{5.2}$$

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. [164].

5.2.2 Meta Structured-Prompting

In the following, we propose the use of MAML and attention mechanism [230] to meta-learn a prompt pool. While MetaPrompting uses task-specific prompts [81], we propose the construction of instance-specific prompts, which allows more flexibility.

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

Algorithm 5 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
```

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 $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1, \dots, K\}$, with key $\mathbf{k}_i \in \mathbb{R}^{d_o}$ and value $\boldsymbol{\theta}_i \in \mathbb{R}^{L_p \times d_i}$ [122, 240, 241]. 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 $\mathbf{K} = [\mathbf{k}_1^\top; \dots; \mathbf{k}_K^\top]$, and $\mathbf{q}_\mathbf{x} \in \mathbb{R}^{d_o}$ is the embedding of the `[MASK]` output by a pretrained and frozen MLM with the wrapped input (e.g., ($\mathbf{x}$. `Topic is [MASK]`)) [240, 241]. 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):

$$\boldsymbol{\theta}_\mathbf{x}(\mathbf{K}, \boldsymbol{\Theta}) = \sum_{i=1}^{K} a_i \boldsymbol{\theta}_i, \tag{5.3}$$

where $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1; \dots; \boldsymbol{\theta}_K]$. While [240, 241] only selects the top-N most similar prompts from the pool, in (5.3) all the prompts are used and updated simultaneously.

The proposed procedure, which will be called MetaPrompter, is shown in Algorithm 6. At iteration t, the base learner takes $(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})$ and a task τ to optimize for a task-specific prompt pool by gradient descent (steps 4-15). $(\mathbf{K}_{t-1}, \boldsymbol{\Theta}_{t-1})$ is used as the initialization (step 4). For each inner iteration j, $(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})$ constructs the instance-dependent prompts $\boldsymbol{\theta}_{\mathbf{x},j}(\mathbf{K}_{t,j-1}, \boldsymbol{\Theta}_{t,j-1})$ in (5.3) (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)

Algorithm 6 MetaPrompter.

```
Require: prompt length $L_p$; size of prompt pool $K$; $\lambda = 0.5$; step size $\alpha, \eta$; meta-parameters $(\mathbf{K}, \mathbf{\Theta})$; query function $q(\cdot)$;
 1: for $t = 1, \dots, T$ 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 $j = 1, \dots, J$ 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 (2.16);
10:         end for
11:         call COMPUTE_LABEL_EMB($\mathcal{S}_\tau$) of Algorithm 5 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 5 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 (5.4);
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 5 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 (2.16) and (5.4), 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 $(\mathbf{K}_T, \mathbf{\Theta}_T)$.
```

and soft verbalizer (steps 11 and 12):

$$\mathbb{P}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) = (1-\lambda)\hat{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}) + \lambda\tilde{\mathbb{P}}(y|\mathbf{x}; \boldsymbol{\theta}_{\mathbf{x},j}), \tag{5.4}$$

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 ($j = 1, \dots, J$) 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 number of inner update steps J. To reduce computation costs, we discard the second-order derivatives and use the first-order approximation (step 24) as in [51, 81], i.e,

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

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 (5.4).

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 and feature embeddings, respectively). This is much smaller[2] 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.

5.3 Experiments

5.3.1 Setup

Data sets. Following [18], we perform few-shot classification on six popularly used data sets: (i) *20News* [114], which contains informal discourses from news discussion forums of 20 topics; (ii) *Amazon* [74] consists of customer reviews from 24 products. The task is to classify reviews into product categories; (iii) *HuffPost* [152], 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

[2] 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 5.1: Statistics of the data sets.

	#classes (meta-train/valid/test)	#samples	#tokens per sample (mean ± 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

classification; (iv) *Reuters* [120] is a collection of Reuters newswire articles of 31 topics from 1996 to 1997; (v) *HWU64* [141] is an intent classification data set, containing user utterances of 64 intents; (vi) *Liu54* [141] 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 [18]. A summary of data sets is in Table 5.1.

Following [9, 72, 18, 81], we perform experiments in the 5-way 1-shot and 5-way 5-shot settings with 15 query samples per class. The pretrained BERT (*bert-base-uncased*) from HuggingFaces [248] is used as the pretrained MLM as [18, 81]. Experiments are run on a DGX station with 8 V100 32GB GPUs. The experiment is repeated three times with different random seeds.

5.3.2 Evaluation on RepVerb

First, we compare the performance of the proposed RepVerb with the state-of-the-art soft verbalizers of: (i) WARP [71][3], and (ii) ProtoVerb [33]. As the focus is on evaluating verbalizers, all methods use the same discrete prompt "`Topic is [MASK]`", and finetune all parameters for 5 steps with a learning rate of 0.00005 as in [33].

Results. Table 5.2 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 5.3 shows the t-SNE visualization of the embeddings ($\mathbf{h}_{\texttt{[MASK]}}(\mathbf{x})$'s) of 100 samples ($\mathbf{x}$'s)[4] and learned label embeddings

[3] Note that the verbalizer of WARP is the same as that of DART [275].

[4] 5-way × (5 support samples + 15 query samples) = 100.

Table 5.2: Meta-testing accuracy of 5-way few-shot classification.

		20News	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
5-shot	WARP [71]	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 [33]	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	**78.81** ± 0.08	**77.56** ± 0.16	**61.90** ± 0.08	**88.33** ± 0.40	**78.37** ± 0.49	**82.14** ± 0.23
1-shot	WARP [71]	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 [33]	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	**59.86** ± 0.38	**59.18** ± 0.31	**44.65** ± 0.20	**63.63** ± 0.41	**59.83** ± 0.71	**66.17** ± 0.40

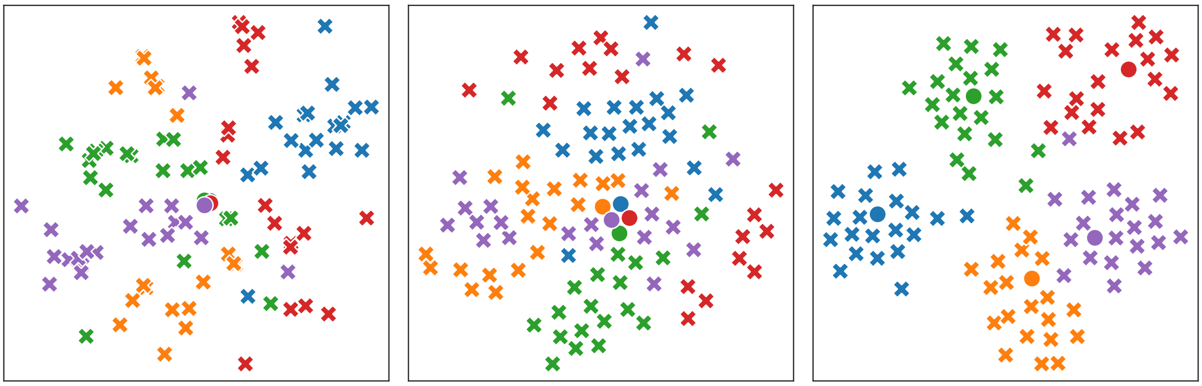

(a) WARP. (b) ProtoVerb. (c) RepVerb.

Figure 5.3: t-SNE visualization of `[MASK]`'s embeddings (crosses) and label embeddings (circles) for a 5-way 5-shot task randomly sampled from *Reuters*.

($\mathbf{v}_y$'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.

5.3.3 Evaluation on MetaPrompter

Setup. 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 [108] 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 [81, 86]. Following [119], prompts are initialized from the input embeddings of randomly sampled label tokens for both MetaPrompting and MetaPrompter.

Baselines. We compare with a variety of methods, including state-of-the-art prompt-based methods of (i) MetaPrompting [81], and its variants (ii) MetaPrompting+WARP /

Table 5.3: 5-way 5-shot classification meta-testing accuracy. Results marked with † are from [18]. "–" indicates that the corresponding result is not reported in [18].

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† [59]	0.07	55.00	66.00	56.30	56.20	-	-
DS† [9]	1.73	68.30	81.10	63.50	96.00	-	-
MLADA† [72]	0.73	77.80	86.00	64.90	96.70	-	-
ConstrastNet† [18]	109.52	71.74	85.17	65.32	95.33	92.57	93.72
MetaPrompting [81]	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 5.4: 5-way 1-shot classification meta-testing accuracy. Results marked with † are from [18]. "–" indicates that the corresponding result is not reported in [18].

	#param ($\times 10^6$)	*20News*	*Amazon*	*HuffPost*	*Reuters*	*HWU64*	*Liu54*
HATT† [59]	0.07	44.20	49.10	41.10	43.20	-	-
DS† [9]	1.73	52.10	62.60	43.00	81.80	-	-
MLADA† [72]	0.73	59.60	68.40	64.90	82.30	-	-
ConstrastNet† [18]	109.52	71.74	76.13	53.06	86.42	86.56	85.89
MetaPrompting [81]	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	0.06	$\mathbf{84.62} \pm 0.29$	$\mathbf{79.05} \pm 0.21$	67.12 ± 0.23	$\mathbf{96.34} \pm 0.20$	$\mathbf{92.11} \pm 0.30$	$\mathbf{93.72} \pm 0.18$

MetaPrompting+ProtoVerb / MetaPrompting+RepVerb, which combine meta-prompting with the soft verbalizer of WARP / ProtoVerb / RepVerb, respectively. Moreover, we also compare with the non-prompt-based methods of: (iii) HATT [59], which meta-learns a prototypical network [216] with a hybrid attention mechanism; (iv) DS [9], which learns attention scores based on word frequency; (v) MLADA [72], which uses an adversarial domain adaptation network to extract domain-invariant features during meta-training; and (vi) ConstrastNet [18], which performs feature extraction by contrastive learning.

Results. Table 5.3 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 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 5.4 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.

5.3.4 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 5.5 shows the nearest tokens to each of

Table 5.5: 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

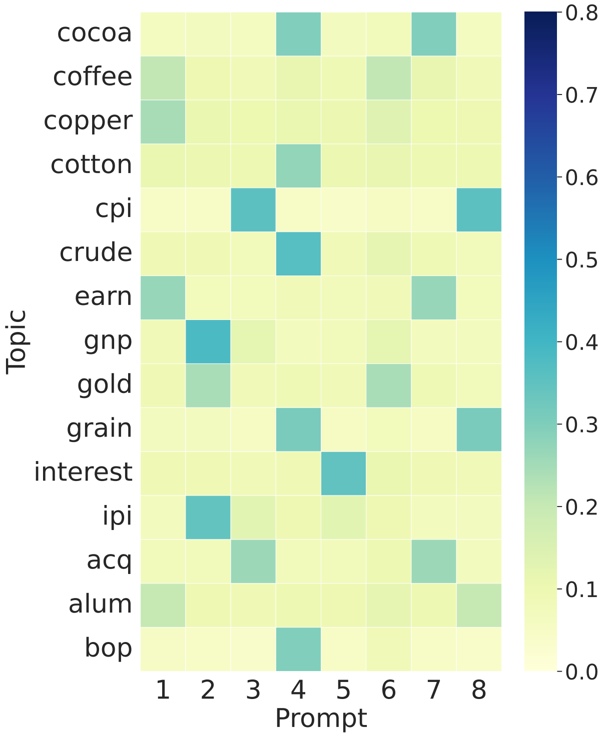


Figure 5.4: Distribution of attention weights on 5-way 5-shot classification of *Reuters* (15 topics).

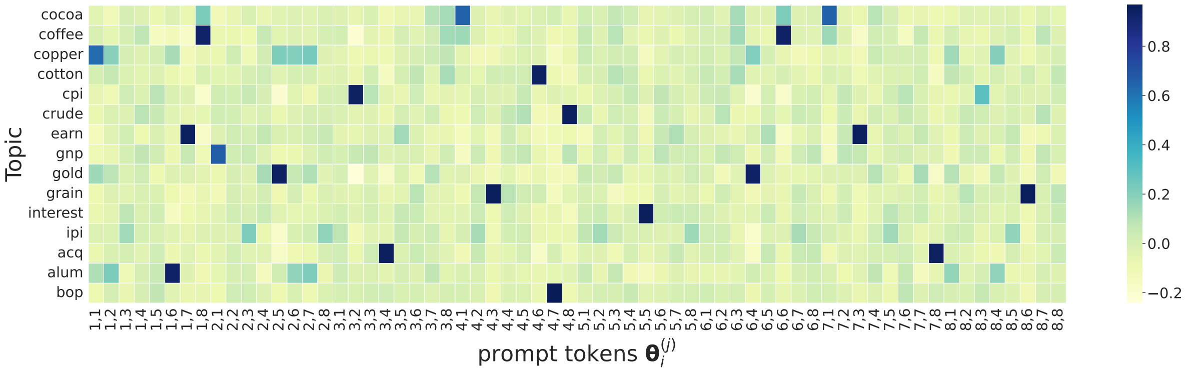


Figure 5.5: Cosine similarities between learned prompt tokens and topic embeddings on 5-way 5-shot classification of *Reuters*. In the x-axis, (i, j) stands for the jth row of $\boldsymbol{\theta}_i$ (i.e., $\boldsymbol{\theta}_i^{(j)}$)

the K $(= 8)$ learned prompts. Figure 5.4 shows the average attention weights between the K prompts and meta-training samples belonging to class (topic) y:

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

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 5.5), 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 $\{(\mathbf{k}_i, \boldsymbol{\theta}_i) : i = 1, \dots, K\}$, 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. [81]), and $\mathcal{E}(\cdot)$ is the input embedding. Figure 5.5 shows the cosine similarities between the learned prompt tokens $\{\boldsymbol{\theta}_i^{(j)} : i = 1, \dots, K, j = 1 \dots, L_p\}$ 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 5.4). 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 5.4).

5.4 Conclusion

In this chapter, we propose MetaPrompter, an effective and parameter-efficient algorithm for prompt tuning. It combines structured prompting and a novel verbalizer called RepVerb. A prompt pool structure is used to construct instance-dependent prompts by attention, while RepVerb builds label embedding by averaging feature embeddings of the corresponding training samples. The pool of prompts is meta-learned from the meta-training tasks. Experimental results demonstrate the effectiveness of the proposed MetaPrompter and RepVerb.

CHAPTER 6

Forward-Backward Reasoning in LLMs for Mathematical Verification

6.1 Introduction

Pretrained large language models (LLMs) [29, 163, 249] generalize well on unseen tasks by few-shot prompting (or in-context learning (ICL) [16, 150, 25]. This is performed by concatenating a few examples (e.g., question-answer pairs) as a prompt, and then appending the testing question. However, it is still challenging for LLMs to answer mathematical questions by simply prompting the question-answer pairs, as mathematics is *more complex* and often requires *many steps* to derive the answer.

One promising direction is using the *meta-knowledge of forward reasoning* extracted from pretraining tasks, i.e., start with the question, then generate several reasoning steps before giving the answer. Recently, Wei et al. [243] propose *chain-of-thought (CoT) prompting* for LLMs, which generates explicit intermediate steps that are used to reach the answer. Specifically, each in-context example is augmented with several thinking steps described in natural language. A few examples are concatenated as a CoT prompt. In inference, the testing question is appended to the prompt and then fed to an LLM. The LLM is expected to imitate the in-context examples, i.e., generating several reasoning steps before giving the answer. CoT prompting has achieved promising performance on mathematical reasoning tasks [243, 237, 281, 279], and many works have been proposed to improve its effectiveness [57, 281, 283, 262, 177] and efficiency [279, 109, 39, 144].

Self-Consistency [237] is a simple but effective approach to improve CoT prompting. Using temperature sampling [1, 50], it samples a diverse set of reasoning chains which may lead to multiple candidate answers. The one that receives the most votes is then chosen as the final answer. Self-Consistency is based on the forward reasoning meta-knowledge of LLMs, i.e., starts with the origin question and generates the reasoning chains with the answer. Figure 6.6 in Section 6.4.1 shows the testing accuracy (averaged over six data sets) of Self-Consistency with different numbers. As shown, simply sampling more reasoning paths may not lead to performance improvement, particularly when M_F is large. Moreover, among the failure questions of Self-Consistency, about 60% have at least one reasoning chain that reaches the correct answer (Table 6.4 in Section 6.4.1). Hence, the majority voting of Self-Consistency can be improved using a more reliable verifier.

In this chapter, we use the *meta-knowledge of backward reasoning* (or *backward chaining*) to verify candidate answers [176, 200, 107, 128, 266]. Backward reasoning works backward from a candidate answer to the antecedent for checking if any data supports this answer. To active backward reasoning meta-knowledge for verification, we mask an informative word in the question and ask the LLM to predict the masked word when a candidate answer $\hat{A}_c$ is provided. We focus on mathematical reasoning tasks, where numbers are the informative words being masked. For each candidate $\hat{A}_c$, we mask a number in the question by $\mathbf{x}$ and design a template *"If we know the answer to the above question is $\hat{A}_c$, what is the value of unknown variable* $\mathbf{x}$*?"* to form a backward question, which is then fed to the LLM to sample multiple backward reasoning chains to predict the masked number. Then, by defining the vote of $\hat{A}_c$ as the number of chains that predict the masked number exactly, we estimate the *backward probability* $\mathbb{P}_\mathrm{B}(\hat{A}_c)$ as the proportion of votes $\hat{A}_c$ gets in the backward direction. When using backward reasoning alone, the prediction is $\mathrm{argmax}_{\hat{A}_c}\mathbb{P}_\mathrm{B}(\hat{A}_c)$.

As the meta-knowledge of forward and backward reasoning are complementary, we propose a FOrward-BAckward Reasoning (FOBAR) method to combine them. By estimating the *forward probability* $\mathbb{P}_\mathrm{F}(\hat{A}_c)$ as the proportion of votes $\hat{A}_c$ gets in the forward direction, we propose to estimate the probability that $\hat{A}_c$ is correct (denoted $\mathbb{P}(\hat{A}_c)$) as the geometric mean of forward and backward probabilities, i.e., $\mathbb{P}(\hat{A}_c) \propto \sqrt{\mathbb{P}_\mathrm{F}(\hat{A}_c)\mathbb{P}_\mathrm{B}(\hat{A}_c)}$. Extensive experiments on six data sets and three OpenAI's LLMs (i.e., *text-davinci-003* [161], *GPT-3.5-Turbo* [162], and *GPT-4* [163]) show that FOBAR achieves state-of-the-art (SOTA) performance.

Our contributions are summarized as follows. (i) We use the meta-knowledge of backward reasoning for mathematical verification by masking a number in the original question and asking the LLM to predict the masked number when a candidate answer is provided. (ii) We propose FOBAR to combine forward and backward reasoning meta-knowledge for verification. (iii) Experimental results on six standard mathematical benchmarks and three LLMs show that FOBAR achieves SOTA performance. In particular, FOBAR outperforms Self-Consistency which uses forward reasoning alone, demonstrating that combining forward and backward reasoning together is better. Additionally, FOBAR outperforms Self-Verification, confirming that using the simple template and the proposed combination is more effective. (iv) Empirical results on two non-mathematical reasoning tasks show that FOBAR also performs well.

6.2 Forward-Backward Reasoning for Verification

In this section, we propose the FOBAR method for verification. An overview of FOBAR is shown in Figure 6.1. We first consider mathematical reasoning tasks. A set of candidate answers are generated in the forward direction, and we estimate each answer's probability based on the votes it receives (Section 6.2.1). Next, we mask a number in the question and propose a simple template to create backward questions for verifying candidate answers (Section 6.2.2). We further propose FOBAR (Section 6.2.3) to combine forward and backward reasoning. Extension to non-mathematical tasks is discussed in Section 6.2.4.

6.2.1 Forward Reasoning

Forward reasoning starts with a question and generates multiple intermediate steps toward the answer. Specifically, for a question Q, we prepend it with a base prompt $\mathbf{P}_{\text{F}}$ (e.g., CoT prompting [243] or ComplexCoT prompting [57]) and feed the tuple $(\mathbf{P}_{\text{F}}, Q)$ to the LLM for generating a reasoning chain and candidate answer. Using temperature sampling [1, 50], we sample M_{F} candidate reasoning chains $\{R_i\}_{i=1}^{M_{\text{F}}}$ and extract the corresponding candidate answers $\{A_i\}_{i=1}^{M_{\text{F}}}$ (see Figure 6.1, top). Let $\mathcal{A} = \{\hat{A}_c\}_{c=1}^{|\mathcal{A}|}$ be the set of answers deduplicated from $\{A_i\}_{i=1}^{M_{\text{F}}}$. Unlike greedy decoding [243], we may have several different candidate answers (i.e., $|\mathcal{A}| > 1$). We propose to estimate the probability that candidate $\hat{A}_c \in \mathcal{A}$ is correct as the proportion of votes it receives from the reasoning paths:

$$\mathbb{P}_{\text{F}}(\hat{A}_c) = \frac{1}{M_{\text{F}}} \sum_{i=1}^{M_{\text{F}}} \mathbb{I}(A_i = \hat{A}_c), \tag{6.1}$$

where $\mathbb{I}(\cdot)$ is the indicator function. Choosing $\hat{A}_c$ with the largest $\mathbb{P}_{\text{F}}(\hat{A}_c)$ corresponds to the state-of-the-art method of Self-Consistency [237]. However, as shown in Figure 6.6, **the performance of Self-Consistency saturates when M_{F} is sufficiently large.** Thus, simply sampling more reasoning paths brings negligible performance improvement.

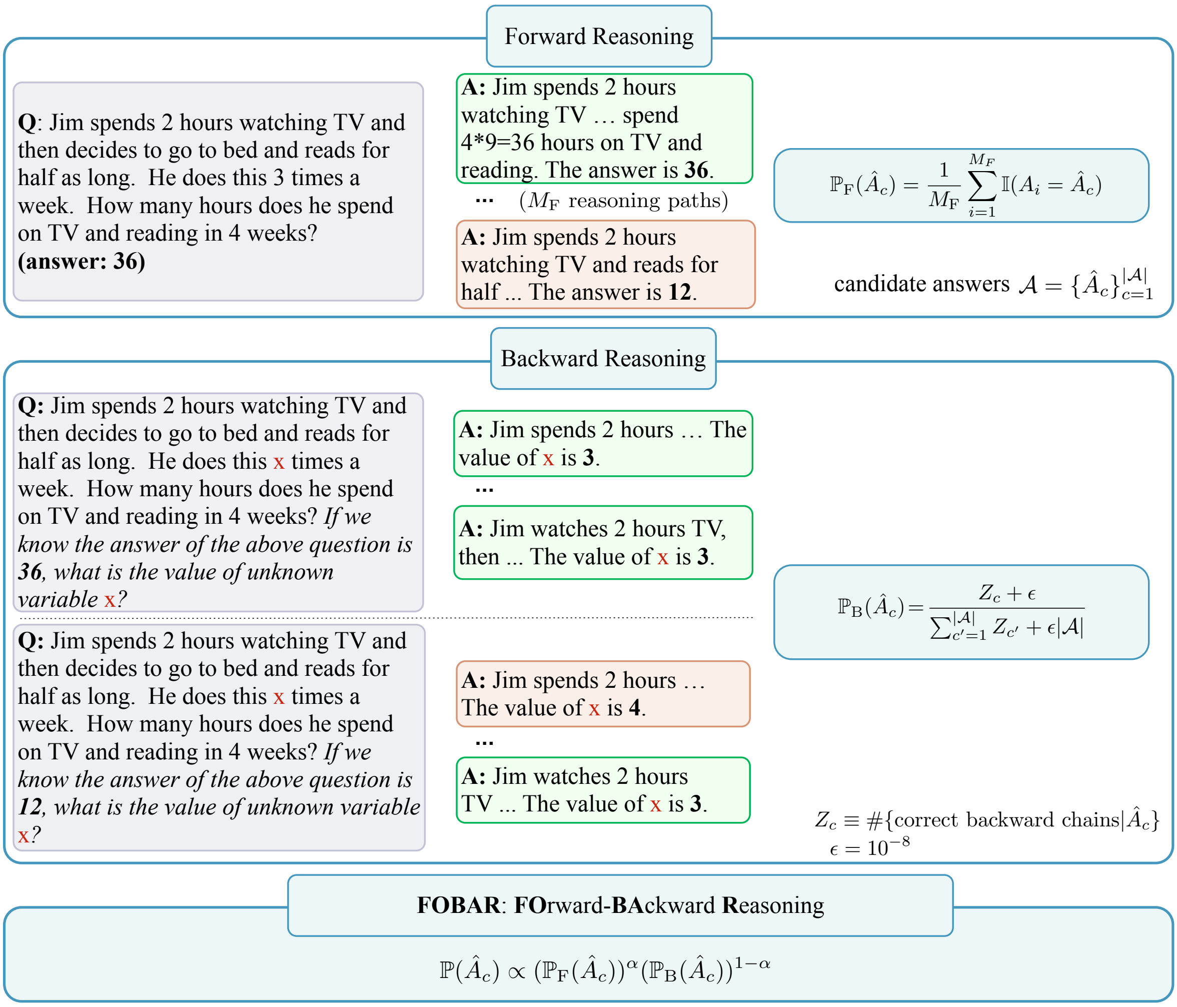

Forward Reasoning
Q: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this 3 times a week. How many hours does he spend on TV and reading in 4 weeks? (answer: 36)
A: Jim spends 2 hours watching TV … spend 4*9=36 hours on TV and reading. The answer is 36.
… (M_F reasoning paths)
A: Jim spends 2 hours watching TV and reads for half ... The answer is 12.
$\mathbb{P}_F(\hat{A}_c) = \frac{1}{M_F}\sum_{i=1}^{M_F}\mathbb{I}(A_i = \hat{A}_c)$
candidate answers $\mathcal{A} = \{\hat{A}_c\}_{c=1}^{|\mathcal{A}|}$
Backward Reasoning
Q: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this x times a week. How many hours does he spend on TV and reading in 4 weeks? If we know the answer of the above question is 36, what is the value of unknown variable x?
A: Jim spends 2 hours … The value of x is 3.
…
A: Jim watches 2 hours TV, then ... The value of x is 3.
Q: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this x times a week. How many hours does he spend on TV and reading in 4 weeks? If we know the answer of the above question is 12, what is the value of unknown variable x?
A: Jim spends 2 hours … The value of x is 4.
…
A: Jim watches 2 hours TV ... The value of x is 3.
$\mathbb{P}_B(\hat{A}_c) = \frac{Z_c + \epsilon}{\sum_{c'=1}^{|\mathcal{A}|} Z_{c'} + \epsilon|\mathcal{A}|}$
$Z_c \equiv \#\{\text{correct backward chains}|\hat{A}_c\}$
$\epsilon = 10^{-8}$
FOBAR: FOrward-BAckward Reasoning
$\mathbb{P}(\hat{A}_c) \propto (\mathbb{P}_F(\hat{A}_c))^{\alpha}(\mathbb{P}_B(\hat{A}_c))^{1-\alpha}$

Figure 6.1: Overview of forward/backward reasoning and the proposed FOBAR. The detailed procedure is shown in Algorithm 7.

6.2.2 Backward Reasoning

In backward reasoning, we mask a number contained in the question and ask the LLM to predict the masked number by using a provided candidate answer. Specifically, suppose that question Q involves N_Q numbers $\{\text{num}^{(n)}\}_{n=1}^{N_Q}$. We replace each of them one by one with **x**. The resultant masked question $\hat{Q}^{(n)}$ is then concatenated with the following template, which contains a candidate answer $\hat{A}_c \in \mathcal{A}$.

Template For Creating Backward Question

$\mathcal{T}(\hat{A}_c)$ = *If we know the answer to the above question is* $\{\hat{A}_c\}$, *what is the value of unknown variable* x*?*

Each $(\hat{Q}^{(n)}, \mathcal{T}(\hat{A}_c))$ pair is called a *backward question*. In total, we obtain N_Q backward questions. Some examples of backward questions are shown in Example 6.2.1. Note that

Self-Verification [246] needs the assistance of an LLM to rewrite a (question, answer) pair into a declarative statement.[1] In contrast, the proposed template is simpler and avoids possible mistakes (an example illustrating Self-Verification's rewriting mistakes is shown in Example 6.2.2).

Example 6.2.1: Backward questions.

Question: Jim spends x hours watching TV and then decides to go to bed and reads for half as long. He does this 3 times a week. How many hours does he spend on TV and reading in 4 weeks? *If we know the answer to the above question is* $\{\hat{A}_c\}$, *what is the value of unknown variable* x?
Question: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this x times a week. How many hours does he spend on TV and reading in 4 weeks? *If we know the answer to the above question is* $\{\hat{A}_c\}$, *what is the value of unknown variable* x?
Question: Jim spends 2 hours watching TV and then decides to go to bed and reads for half as long. He does this 3 times a week. How many hours does he spend on TV and reading in x weeks? *If we know the answer to the above question is* $\{\hat{A}_c\}$, *what is the value of unknown variable* x?

Example 6.2.2: Mistake in Self-Verification

Question: A class of 50 students has various hobbies. 10 like to bake, 5 like to play basketball, and the rest like to either play video games or play music. How many like to play video games if the number that like to play music is twice the number that prefer playing basketball? (answer: 25)
Question (Self-Verification): A class of x students has various hobbies. 10 like to bake, 5 like to play basketball, and the rest like to either play video games or play music. The number of people who like to play video games is equal to the number of people who prefer playing basketball multiplied by two. The number of people who like to play video games is 25. What is the answer of x?
Question (FOBAR): A class of x students has various hobbies. 10 like to bake, 5 like to play basketball, and the rest like to either play video games or play music. How many like to play video games if the number that like to play music is twice the number that prefer playing basketball? *If we know the answer to the above question is 25, what is the value of unknown variable* x?

To predict the masked number, we prepend the backward question with a prompt $\mathbf{P}_\text{B}$, which consists of several (backward) question-answer demos with reasoning chains. An example question-answer demo is shown in Example 6.2.3. We feed each of $(\mathbf{P}_\text{B}, \hat{Q}^{(n)}, \mathcal{T}(\hat{A}_c))$ (where $n = 1, \dots, N_Q$) to the LLM, which then imitates the in-context examples in $\mathbf{P}_\text{B}$ and generates a reasoning chain for the prediction of the masked number. We sample M_B such reasoning chains with predictions $\{\widehat{\text{num}}_{c,b}^{(n)}\}_{b=1}^{M_\text{B}}$ (see Figure 6.1,

[1] For example, "How many hours does he spend on TV and reading in 4 weeks?" with a candidate answer of 36 is rewritten to "He spends 36 hours on TV and reading in 4 weeks".

middle). For each candidate answer $\hat{A}_c$, we count the number of times that the masked number is exactly predicted:

$$Z_c = \sum_{n=1}^{N_{\mathrm{Q}}} \sum_{b=1}^{M_{\mathrm{B}}} \mathbb{I}(\widehat{\mathrm{num}}_{c,b}^{(n)} = \mathrm{num}^{(n)}). \tag{6.2}$$

The probability that candidate answer $\hat{A}_c$ is correct is estimated as

$$\mathbb{P}_{\mathrm{B}}(\hat{A}_c) = \frac{Z_c + \epsilon}{\sum_{c'=1}^{|\mathcal{A}|} Z_{c'} + \epsilon|\mathcal{A}|}, \tag{6.3}$$

where $\epsilon = 10^{-8}$ is a small positive constant to avoid division by zero. One can simply choose $\hat{A}_c$ with the largest $\mathbb{P}_{\mathrm{B}}(\hat{A}_c)$ as the prediction. A more effective method, as will be shown in Section 6.2.3, is to combine the probabilities obtained from forward and backward reasoning.

Example 6.2.3: Backward Reasoning.

Question: Randy has 60 mango trees on his farm. He also has x less than half as many coconut trees as mango trees. How many trees does Randy have in all on his farm? *If we know the answer to the above question is 85, what is the value of unknown variable* x*?*
Answer: Let's think step by step. We know that Randy has 60 mango trees on his farm. We also know that he has x less than half as many coconut trees as mango trees. Let's use C to represent the number of coconut trees. So we can write: C = (1/2)*60 - x = 30 - x. The total number of trees on Randy's farm is the sum of the number of mango trees and coconut trees: 60 + (30 - x) = 90 - x. We are given that the total number of trees on Randy's farm is 85, so we can write: 90 - x = 85. Solving for x, we get: x = 5. The value of x is 5.

6.2.3 FOBAR (FOrward and BAckward Reasoning)

As forward and backward reasoning are complementary (i.e., backward reasoning may succeed in the cases where forward reasoning fails and vice versa, as Examples 6.2.4 and 6.2.5), we propose to combine them for verification. Intuitively, a candidate answer is likely to be correct when it receives many votes in forward reasoning and also helps the LLM to predict the masked numbers in backward reasoning. We estimate the probability that $\hat{A}_c$ is correct as

$$\mathbb{P}(\hat{A}_c) \propto \left(\mathbb{P}_{\mathrm{F}}(\hat{A}_c)\right)^{\alpha} \left(\mathbb{P}_{\mathrm{B}}(\hat{A}_c)\right)^{1-\alpha}, \tag{6.4}$$

with weight $\alpha \in [0,1]$ (see Figure 6.1, bottom). When $\alpha = 1$, it reduces to Self-Consistency [237]; When α equals 0, it reduces to backward reasoning for verification. In the experiments, we combine the forward and backward probabilities by the geometric mean (i.e., $\alpha = 0.5$). Finally, we select the answer as $\arg\max_{\hat{A}_c \in \mathcal{A}} \mathbb{P}(\hat{A}_c)$. The whole procedure is shown in Algorithm 7.

Example 6.2.4: Forward reasoning fails but backward reasoning succeeds.

Question: The sum of three consecutive odd numbers is 69. What is the smallest of the three numbers?
Ground-truth answer: 21
Forward reasoning: $\mathbb{P}_{\mathrm{F}}(21) = 0.4, \mathbb{P}_{\mathrm{F}}(23) = 0.6$
Backward reasoning: $\mathbb{P}_{\mathrm{B}}(21) = 0.8, \mathbb{P}_{\mathrm{B}}(23) = 0.2$
FOBAR: $\mathbb{P}(21) = 0.62, \mathbb{P}(23) = 0.38$
A backward question: The sum of three consecutive odd numbers is **x**. What is the smallest of the three numbers? If we know the answer to the above question is 21, what is the value of unknown variable **x**?

Example 6.2.5: Forward reasoning succeeds but backward reasoning fails.

Question: While digging through her clothes for ice cream money, Joan found 15 dimes in her jacket, and 4 dimes in her shorts. How much money did Joan find?
Ground-Truth answer: 1.9
Forward reasoning: $\mathbb{P}_{\mathrm{F}}(1.9) = 0.7, \mathbb{P}_{\mathrm{F}}(190) = 0.3$
Backward reasoning: $\mathbb{P}_{\mathrm{B}}(1.9) = 0.43, \mathbb{P}_{\mathrm{B}}(190) = 0.57$
FOBAR: $\mathbb{P}(1.9) = 0.57, \mathbb{P}(190) = 0.43$
A backward question: While digging through her clothes for ice cream money, Joan found 15 dimes in her jacket, and **x** dimes in her shorts. How much money did Joan find? If we know the answer to the above question is 1.9, what is the value of unknown variable **x**?

6.2.4 Extension to Non-Mathematical Reasoning Tasks

In mathematical questions, numbers are the most informative words, which are chosen to be masked. For non-mathematical questions, we can analogously mask an informative word and ask the LLM to guess the masked word when a candidate answer is provided.

For example, consider the following question-answer pair from the *Last Letter Concatenation* task [243, 283]: "Take the last letters of each word in '*Whitney Erika Tj Benito*' and concatenate them" with ground-truth answer "*yajo*". We can mask one of the four words (e.g., "*Erika*"). Given a candidate answer $\hat{A}_c$, we create a backward question as "Take the last letters of each word in '*Whitney* ___ *Tj Benito*' and concatenate them. If we know the answer to the above question is $\hat{A}_c$, which is the word at the blank, *Erika* or *Dqhjz*", where "*Dqhjz*" is obtained by shifting each letter of "*Erika*". The LLM is more

Algorithm 7 FOBAR.

```
Require: number of reasoning chains $M_{\mathrm{F}}$ and $M_{\mathrm{B}}$, prompts $\mathbf{P}_{\mathrm{F}}$ and $\mathbf{P}_{\mathrm{B}}$; $\epsilon = 10^{-8}$;
    $\alpha = 0.5$;
Input: a question $Q$ with $N_Q$ numbers;
feed $(\mathbf{P}_{\mathrm{F}}, Q)$ to LLM, sample $M_{\mathrm{F}}$ reasoning chains with candidate answers $\{A_i\}_{i=1}^{M_{\mathrm{F}}}$;
deduplicate $\{A_i\}_{i=1}^{M_{\mathrm{F}}}$ to $\mathcal{A} = \{\hat{A}_c\}_{c=1}^{|\mathcal{A}|}$;
compute $\mathbb{P}_{\mathrm{F}}(\hat{A}_c)$ by Eq. (6.1) for $\hat{A}_c \in \mathcal{A}$;
for $\hat{A}_c \in \mathcal{A}$ do
    for $n = 1, \dots, N_Q$ do
        create $\hat{Q}^{(n)}$ by masking the $n$th number $\text{num}^{(n)}$ in $Q$;
        feed $(\mathbf{P}_{\mathrm{B}}, \hat{Q}^{(n)}, \mathcal{T}(\hat{A}_c))$ to LLM;
        sample $M_{\mathrm{B}}$ predictions $\{\widehat{\text{num}}_{c,b}^{(n)}\}_{b=1}^{M_{\mathrm{B}}}$;
    end for
    compute $Z_c$ by Eq. (6.2);
end for
compute $\mathbb{P}_{\mathrm{B}}(\hat{A}_c)$ by Eq. (6.3) for $\hat{A}_c \in \mathcal{A}$;
compute $\mathbb{P}(\hat{A}_c)$ by Eq. (6.4) for $\hat{A}_c \in \mathcal{A}$;
return $\arg\max_{\hat{A}_c \in \mathcal{A}} \mathbb{P}(\hat{A}_c)$.
```

likely to choose "*Erika*" if the second letter in $\hat{A}_c$ is "*a*".

For more general problems like *StrategyQA* [61], verbs are usually informative (The NLTK tool [142] can extract verbs). Like the *Last Letter Concatenation* task, we create backward questions as multiple-choice questions to restrict the search space of answers. We use LLMs to generate an antonym word of the masked verb. The masked word and its antonym are two options in the multiple-choice question. We illustrate the procedure by an example taken from *StrategyQA*.

Question: Yes or No: Would a pear sink in water?
Masked Question: Yes or No: Would a pear ___ in water?
Generate Optional Word: We use the LLM (e.g., GPT-3.5-Turbo) to output the antonym of "sink" by the below instruction:
Following the examples below, please give me an antonym verb for the last word.
import: export
lead: follow
include: exclude
stay: go
sink: (GPT-3.5-Turbo outputs "float")
Backward Question: Yes or No: Would a pear ___ in water? If we know the answer to the above question is **No**, what is the word at the blank, **sink** or **float**?

FOBAR can be extended to more complex non-mathematical questions (contain many sentences): (i) masking a sentence in the question; (ii) using LLMs to generate its opposite sentence; (iii) the masked sentence and its opposite sentence are two options in the multiple-choice backward question.

6.3 Experiments on Mathematical Tasks

6.3.1 Setup

Datasets. Experiments are conducted on six mathematical data sets which are commonly used in evaluating CoT reasoning ability [281, 237]: (i) *AddSub* [80], (ii) *MultiArith* [196], (iii) *SingleEQ* [110], (iv) *SVAMP* [170], (v) *GSM8K* [30], (vi) *AQuA* [132]. Some statistics and example question-answer pairs are shown in Table 6.1.

Table 6.1: Statistics of data sets used in the experiments.

		#samples	N_Q (mean ± std)	example
Math	*AddSub*	395	2.6 ± 0.7	Benny picked 2 apples and Dan picked 9 apples from the apple tree. How many apples were picked in total?
	MultiArith	600	3.1 ± 0.3	Katie picked 3 tulips and 9 roses to make flower bouquets. If she only used 10 of the flowers though, how many extra flowers did Katie pick?
	SingleEQ	508	2.2 ± 0.7	Joan went to 4 football games this year. She went to 9 football games last year. How many football games did Joan go to in all?
	SVAMP	1000	2.8 ± 0.7	Rachel has 4 apple trees. She picked 7 apples from each of her trees. Now the trees have a total 29 apples still on them. How many apples did Rachel pick in all?
	GSM8K	1319	3.8 ± 1.6	A robe takes 2 bolts of blue fiber and half that much white fiber. How many bolts in total does it take?
	AQuA	254	2.9 ± 1.3	If the population of a city increases by 5% annually, what will be the population of the city in 2 years time if its current population is 78000? Answer Choices: (A) 81900 (B) 85995 (C) 85800 (D) 90000 (E) None of these
Non-Math	*Last Letter*	500	4.0 ± 0.0	Take the last letters of each word in "*Whitney Erika Tj Benito*" and concatenate them.
	DateU	369	1.2 ± 0.7	The deadline is Jun 1, 2021, which is 2 days away from now. What is the date a month ago in MM/DD/YYYY?

Baselines. We compare the proposed FOBAR with (i) In-Context Learning (ICL) using question-answer pairs as demonstrations [16], and recent CoT prompting methods, including: (ii) CoT prompting [243]; (iii) ComplexCoT prompting [57] which selects demonstrations with complex reasoning steps; (iv) RE2 [253] which re-reads the question in the prompt; (v) PHP [281] which iteratively uses the previous answers as hints in designing prompts; (vi) RCoT [254] which reconstructs the question based on the candidate answer and checks the factual inconsistency for verification; (vii) Self-

-Consistency [237], which samples multiple reasoning chains and selects the answer by majority voting; (viii) Self-Verification [246], which chooses the top-2 candidate answers obtained from Self-Consistency and re-ranks them based on the verification scores computed in the backward procedure.

Following Zheng et al. [281], we experiment with three LLMs: (i) *text-davinci-003* [161], (ii) *GPT-3.5-Turbo* [162], and (iii) *GPT-4* [163]. *GPT-3.5-Turbo* and *GPT-4* are more powerful than *text-davinci-003*. The proposed FOBAR is general and can be integrated into any prompting method. Here, we choose the CoT prompting and ComplexCoT prompting as base prompts as in Zheng et al. [281].

Implementation Details. Following [237, 283, 281], the temperature for sampling is 0.7 for both forward and backward reasoning. The α in Eq. (6.4) is set to 0.5. For *text-davinci-003*, M_{F} is 40 as in [237, 281]; whereas the more powerful LLMs (*GPT-3.5-Turbo* and *GPT-4*) use a smaller M_{F} (i.e., 10). M_{B} is set to 8 for all three LLMs. We do not repeat the experiments using different seeds as conducting experiments without using different seeds is a standard protocol in the field of CoT-based research due to budget considerations (e.g., ComplexCoT [57], Self-Consistency [237], PHP [281]).

6.3.2 Main Results

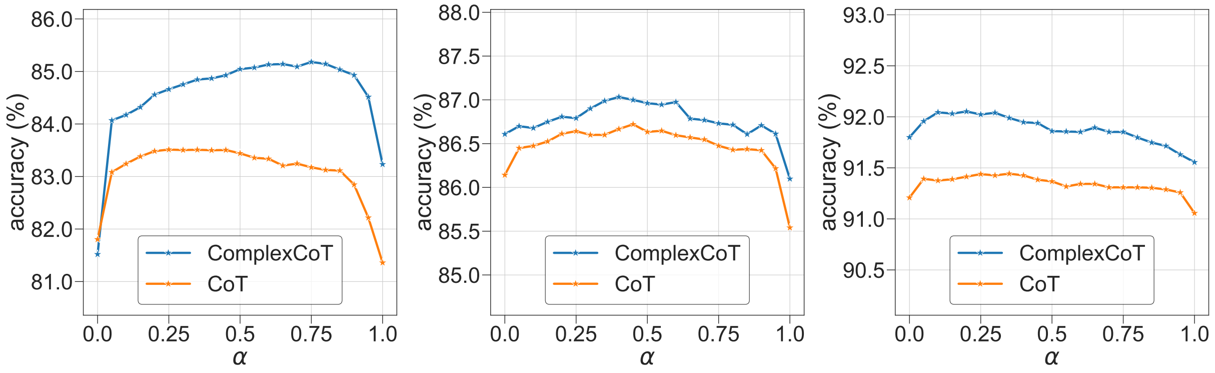
Table 6.2 shows the testing accuracies. As can be seen, for all three LLMs, FOBAR with ComplexCoT prompting achieves the highest average accuracy. When using CoT as the base prompt, FOBAR outperforms Self-Consistency most of the time, demonstrating that combining forward and backward reasoning is more accurate than using forward reasoning alone. Furthermore, FOBAR performs better than Self-Verification on almost all datasets, demonstrating that using the proposed simple template in backward reasoning and the proposed combination is more effective in verification. FOBAR (with either CoT or ComplexCoT) on *GPT-4* achieves the highest average accuracy, as *GPT-4* is currently the SOTA LLM. Moreover, for all three LLMs, FOBAR using ComplexCoT as the base prompt achieves higher accuracy than using CoT on average, which aligns with observations in [57, 281] that ComplexCoT is better than CoT.

Table 6.2: Testing accuracies (%) on six data sets using three LLMs. For each LLM, methods are grouped according to the base prompt they used. The best in each group is in **bold**. Results with † are from the original publications. "–" means that the result is not reported in the original publication.

			AddSub	*MultiArith*	*SingleEQ*	*SVAMP*	*GSM8K*	*AQuA*	Average
text-davinci-003		ICL [16]	90.4	37.6	84.3	69.1	16.9	29.1	54.5
	CoT	CoT [243]	91.4	93.6	92.7	79.5	55.8	46.5	76.6
		PHP† [281]	91.1	94.0	93.5	81.3	57.5	44.4	77.0
		RE2† [253]	91.7	93.3	93.3	81.0	61.6	44.5	77.6
		Self-Consistency [237]	91.7	95.9	94.5	83.1	67.9	**55.1**	81.4
		Self-Verification [246]	87.4	95.3	92.9	82.2	59.8	37.4	75.8
		FOBAR	**91.9**	**100.0**	**96.1**	**86.8**	**70.8**	**55.1**	**83.5**
	ComplexCoT	ComplexCoT [57]	88.9	95.3	93.7	78.0	67.7	48.8	78.7
		PHP† [281]	**91.6**	96.6	95.0	83.7	68.4	53.1	81.4
		Self-Consistency [237]	89.4	98.5	91.1	82.7	**79.1**	**58.7**	83.2
		Self-Verification [246]	89.9	95.5	94.1	80.1	72.0	38.2	78.3
		FOBAR	90.6	**100.0**	**95.3**	**87.0**	78.7	**58.7**	**85.0**
GPT-3.5-Turbo		ICL [16]	88.6	87.6	88.8	80.6	32.2	31.1	68.2
	CoT	CoT [243]	89.4	97.9	92.9	84.2	77.2	54.3	82.7
		RE2† [253]	89.9	96.5	**95.3**	80.0	80.6	58.3	83.4
		Self-Consistency [237]	**90.6**	98.6	93.1	86.4	81.9	**62.6**	85.5
		Self-Verification [246]	90.4	97.4	92.9	83.1	74.9	60.6	83.2
		FOBAR	89.4	**99.3**	94.5	**88.9**	**85.1**	**62.6**	**86.6**
	ComplexCoT	Complex CoT [57]	87.9	98.3	**94.5**	81.1	80.7	59.1	83.6
		RCoT† [254]	88.2	–	93.0	84.9	84.6	53.3	–
		PHP† [281]	85.3	98.0	92.9	83.1	85.1	60.6	84.2
		Self-Consistency [237]	88.1	98.8	**94.5**	85.0	86.4	63.0	86.0
		Self-Verification [246]	87.9	96.6	93.3	81.0	78.2	61.4	83.1
		FOBAR	**88.4**	**99.8**	94.3	**88.5**	**87.4**	**63.4**	**87.0**
GPT-4		ICL [16]	92.1	98.6	94.3	90.9	48.5	48.0	78.7
	CoT	CoT [243]	**92.7**	**99.0**	95.7	92.9	93.4	69.7	90.6
		Self-Consistency [237]	92.2	**99.0**	95.9	93.3	94.8	**71.3**	91.1
		Self-Verification [246]	**92.7**	**99.0**	95.7	93.1	93.7	70.1	90.7
		FOBAR	92.4	**99.0**	**96.1**	**94.1**	**95.4**	**71.3**	**91.4**
	ComplexCoT	Complex CoT [57]	**91.9**	98.3	94.5	92.4	95.1	72.4	90.8
		PHP† [281]	89.6	98.1	93.1	91.9	95.5	**79.9**	91.3
		Self-Consistency [237]	91.4	98.5	**94.7**	93.4	96.2	75.2	91.6
		Self-Verification [246]	91.6	98.5	**94.7**	93.0	95.7	75.6	91.5
		FOBAR	**91.9**	**98.6**	**94.7**	**94.4**	**96.4**	75.2	**91.9**

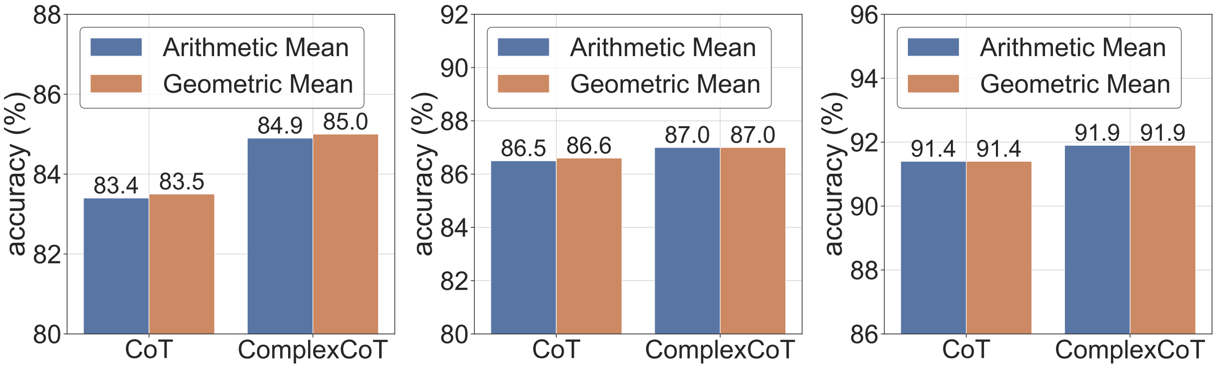
6.3.3 Combining Forward and Backward Probabilities

In this experiment, we study how the combination weight α in Eq. (6.4) affects performance. Figure 6.2 shows the testing accuracies (averaged over the six data sets) with $\alpha \in [0, 1]$ using the three LLMs. As can be seen, FOBAR is insensitive to α over a large range for all three LLMs. In the sequel, we use $\alpha = 0.5$, which corresponds to the geometric mean of the forward and backward probabilities.



(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 6.2: Testing accuracy (averaged over the six data sets) of FOBAR w.r.t. α.



(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 6.3: Testing accuracy of FOBAR (averaged over the six data sets) with geometric/arithmetic mean of forward and backward probabilities.

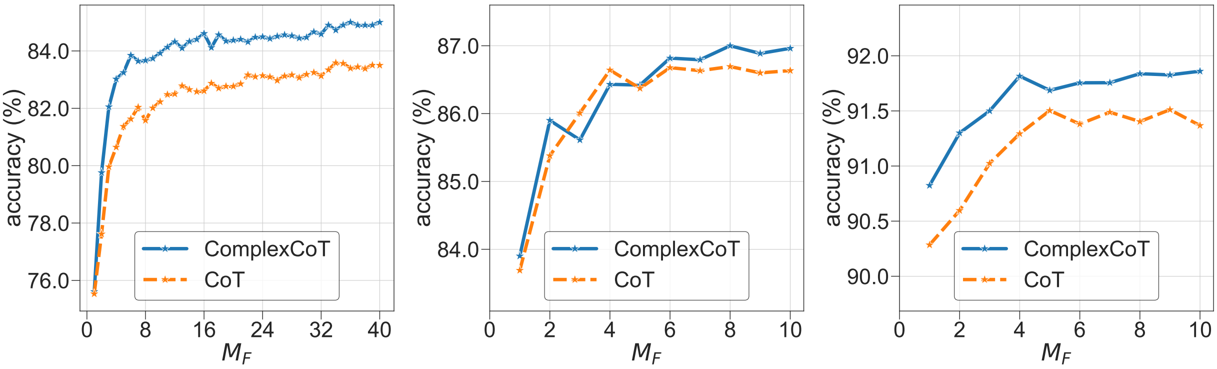
Alternatively, one can combine the forward and backward probabilities by the arithmetic mean, i.e., $\mathbb{P}(\hat{A}_c) = \frac{1}{2}\left(\mathbb{P}_{\mathrm{F}}(\hat{A}_c) + \mathbb{P}_{\mathrm{B}}(\hat{A}_c)\right)$. Figure 6.3 shows the testing accuracies for the three LLMs. As shown, The arithmetic mean has a performance comparable to that of the geometric mean. Hence, Figures 6.2 and 6.3 together suggest that FOBAR is robust to the combination of forward and backward probabilities.

6.3.4 Usefulness of Forward and Backward Reasoning

We perform an ablation study on forward (FO) and backward (BA) reasoning. We consider the four combinations: (i) using neither forward nor backward reasoning (which reduces to greedy decoding [243]); (ii) use only forward reasoning (i.e., Self--Consistency); (iii) use only backward reasoning in selecting answers (i.e., $\alpha = 0$ in Algorithm 7); (iv) use both forward and backward reasoning (i.e., the proposed FOBAR). Table 6.3 shows the testing accuracies (averaged over the six data sets) for the three LLMs. As can be seen, in all the settings, using forward or backward reasoning

Table 6.3: Average testing accuracies (%) with different combinations of forward (FO) and backward (BA) reasoning.

	Forward	Backward	*text-davinci-003*	*GPT-3.5-Turbo*	*GPT-4*
CoT	✗	✗	76.6	82.7	90.6
	✓	✗	81.4	85.5	91.1
	✗	✓	82.1	86.2	91.2
	✓	✓	**83.5**	**86.6**	**91.4**
ComplexCoT	✗	✗	78.7	83.6	90.8
	✓	✗	83.2	86.0	91.6
	✗	✓	81.3	86.3	91.8
	✓	✓	**85.0**	**87.0**	**91.9**



(a) *text-davinci-003*. (b) *GPT-3.5-turbo*. (c) *GPT-4*.

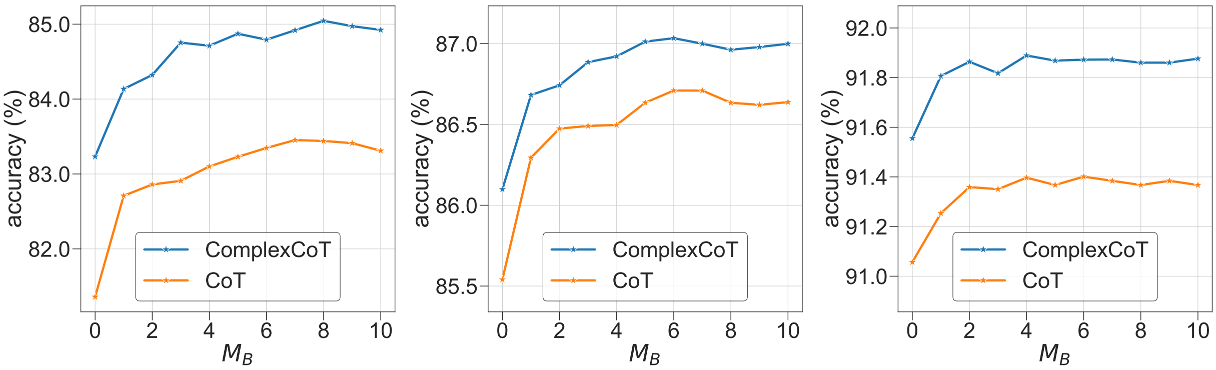
Figure 6.4: Testing accuracy of FOBAR (averaged over the six data sets) with M_{F}.

is consistently better than using neither of them. Moreover, combining forward and backward reasoning is always the best. Examples 6.2.4 and 6.2.5 show that FOBAR is able to rectify some cases of failure in forward and backward reasoning, respectively.

6.3.5 Number of Forward and Backward Reasoning Chains

6.3.5.1 Varying M_{F}

In this section, we study how the performance of FOBAR varies with the number of forward reasoning chains M_{F}. Figure 6.4 shows the testing accuracies (averaged over the six data sets) for the three LLMs. As can be seen, using a very small M_{F} (e.g., ≤ 5) is clearly undesirable, but the accuracy saturates quickly with increasing M_{F}. This suggests that one can use a small M_{F} to reduce the computational cost. Moreover, the accuracy curves of FOBAR are higher than those of Self-Consistency in Figure 6.6, again



(a) *text-davinci-003*. (b) *GPT-3.5-turbo*. (c) *GPT-4*.

Figure 6.5: Testing accuracy of FOBAR (averaged over the six data sets) with M_{B}.

demonstrating that integrating backward reasoning into verification is effective.

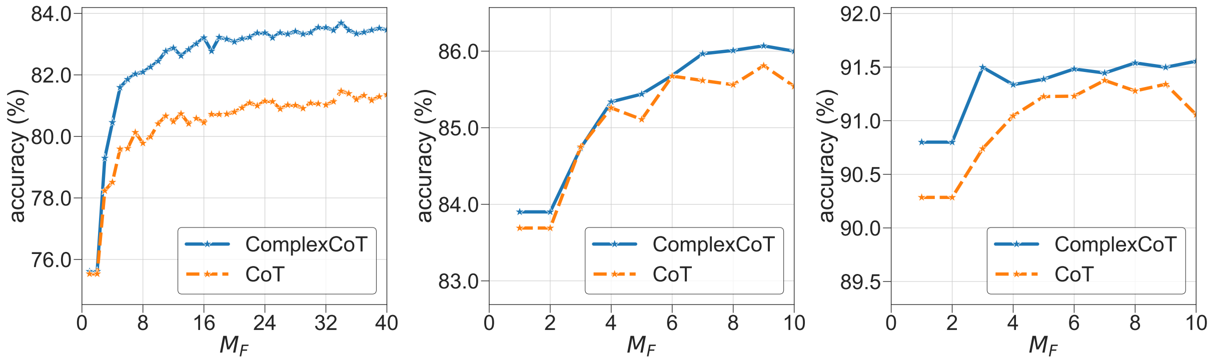
6.3.5.2 Varying M_{B}

Next, we study how the performance of FOBAR varies with the number of backward reasoning chains M_{B}. Figure 6.5 shows the testing accuracies (averaged over the six data sets) for the three LLMs. Note that $M_{\mathrm{B}} = 0$ corresponds to using only forward reasoning. As shown, using a very small M_{B} (e.g., ≤ 4) is clearly undesirable, but the accuracy saturates quickly when M_{B} increases. Hence, using a small M_{B} can achieve a good balance between performance and efficiency.

6.4 Analysis on Forward/Backward Reasoning

6.4.1 Saturated Performance of Self-Consistency

In this section, we study how the performance of Self-Consistency varies with the number of forward reasoning chains M_{F}. Figure 6.6 shows the testing accuracies (averaged over the six data sets) for the three LLMs. As can be seen, simply sampling more reasoning paths may not lead to performance improvement, particularly when M_{F} is large. Moreover, among the failure questions of Self-Consistency, about 60% have at least one reasoning chain reaches the correct answer (Table 6.4).

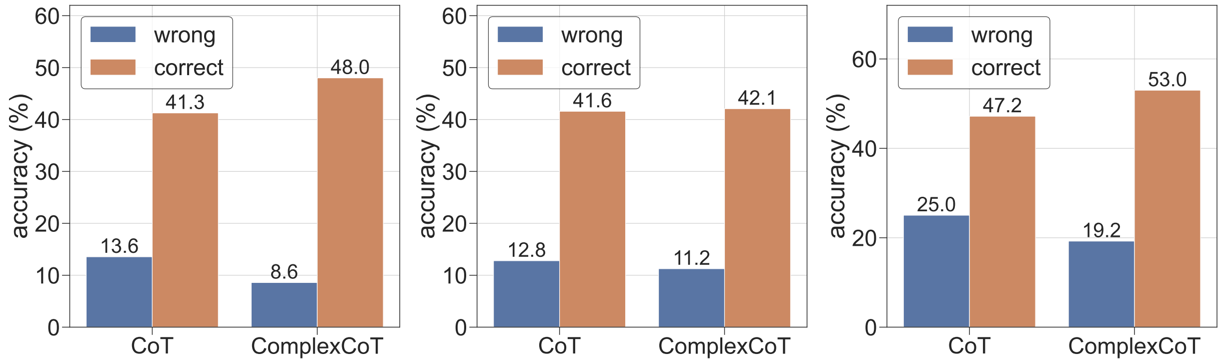


(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 6.6: Accuracy (averaged over six data sets) of Self-Consistency versus number of sampling paths (M_F).

Table 6.4: Statistics on the failure cases of Self-Consistency on the six data sets.

	AddSub	*MultiArith*	*SingleEQ*	*SVAMP*	*GSM8K*	*AQuA*	Total
#failures	47	7	28	150	179	94	505
#failures with no correct answer	28	0	14	57	60	52	211 (≈ 40%)
#failures with at least one correct answer	19	7	14	93	119	42	294 (≈ 60%)



(a) *text-davinci-003*. (b) *GPT-3.5-Turbo*. (c) *GPT-4*.

Figure 6.7: Accuracy (averaged over all backward questions across the six data sets) of predicting the masked number in backward questions with correct/wrong candidate answers.

6.4.2 Correct Candidate Helps Backward Reasoning

In this experiment, we verify the intuition that the correct candidate answer helps LLM to predict the masked numbers. Figure 6.7 compares the accuracies of predicting the masked numbers in backward questions with the correct/wrong candidates. As can be seen, using the correct candidate has 2× higher accuracy (averaged over the six data sets) than the wrong ones in predicting masked numbers, demonstrating that using backward reasoning to verify candidate answers is reasonable.

Table 6.5: Accuracies on the non-mathematical tasks of *Date Understanding* and *Last Letter Concatenation* using *GPT-3.5-Turbo*. Results with † are from the original publications. "–" means that the result is not reported in the original publication.

		Date Understanding	*Last Letter Concatenation*
	ICL [16]	52.0	8.0
CoT	CoT [243]	61.3	81.0
	RE2† [253]	47.2	–
	Self-Consistency [237]	65.6	81.4
	Self-Verification [246]	66.1	81.8
	FOBAR	**66.4**	**82.6**
ComplexCoT	ComplexCoT [57]	74.8	81.4
	RCoT† [254]	71.7	–
	Self-Consistency [237]	77.5	81.2
	Self-Verification [246]	76.2	81.6
	FOBAR	**78.0**	**82.4**

6.5 Experiments on Non-Mathematical Tasks

In this section, we perform experiments on three standard non-mathematical tasks: *Date Understanding* [243, 57], *Last Letter Concatenation* [243, 283], *StrategyQA* [61]. Examples are shown in Table 6.1. For *Date Understanding*, numbers are chosen as informative tokens; for *Last Letter Concatenation*, the four words in the questions are informative; for *StrategyQA*, verbs are chosen as informative tokens. For *Last Letter Concatenation* and *StrategyQA*, we create backward questions as multiple-choice questions to restrict the search space of answers, see Section 6.2.4 for detailed descriptions. We compare FOBAR with other CoT-based methods and ICL using *GPT-3.5-Turbo*. PHP does not report results on non-mathematical tasks.

Table 6.5 shows the testing accuracies. As shown, FOBAR outperforms all the baselines with either CoT or ComplexCoT as the base prompt. Moreover, all CoT-based methods are better than ICL significantly.

6.6 Conclusion

In this chapter, we study the problem of verifying candidate answers to mathematical problems using chain-of-thought prompting. To complement the use of only forward reasoning for verification, we use the meta-knowledge of backward reasoning : A simple template is introduced to create questions and a prompt is designed to ask the LLM to predict a masked word when a candidate answer is provided. Furthermore, we proposed FOBAR to combine the meta-knowledge of forward and backward reasoning for verification. Extensive experiments on six standard mathematical data sets and three LLMs show that the proposed FOBAR achieves state-of-the-art performance on mathematical reasoning tasks. FOBAR can also be used on non-mathematical tasks and achieves superior performance.

CHAPTER 7

MetaMathQA: Bootstrap Math Questions for LLMs

7.1 Introduction

Recent years have witnessed the rapid development of large language models (LLMs) which emerge as the favored approach for various applications and demonstrate multi-dimensional abilities, including instruction following [16, 222, 165], coding assistance [19, 158, 146, 123], and mathematical problem-solving [268, 88, 145, 31]. Among various tasks, solving mathematical problems is more challenging as they often require highly complex multi-step reasoning capabilities. Although some close-sourced models, e.g., *GPT-3.5-Turbo* [161], *GPT-4* [163] and *PaLM-2* [227], have demonstrated promising performance on some mathematical problem-solving benchmarks, it is still a mystery how these models are trained and what data these models use. Therefore, how to equip open-source LLMs (e.g., *LLaMA* [226, 227]) with good mathematical problem-solving meta-knowledge remains an open challenge.

To tackle this challenge, two popular lines of research to improve the mathematical problem-solving abilities of LLMs are *prompt-based methods* and *finetuning-based methods*. Prompt-based methods [243, 57, 237, 283, 57, 237] aim to activate the meta-knowledge of forward reasoning by choosing suitable prompting inputs without modifying the model parameters. Finetuning-based methods enhance the meta-knowledge of mathematical reasoning by finetuning on mathematical data. While prompt-based methods are model-dependent and sensitive to many factors, finetuning-based methods, despite being simple and model-agnostic, heavily rely on effective training data on downstream mathematical questions.

In this chapter, we aim to augment data for training LLMs to enhance the meta-knowledge of solving mathematical problems. Specifically, we propose to bootstrap the questions in both forward and backward reasoning directions. For the *forward* direction, we have the original and LLM-rephrased questions. For the *backward* direction, we have the self-verification question [246] and FOBAR question proposed in Chapter 6. To construct backward reasoning questions, we mask a number in a question using

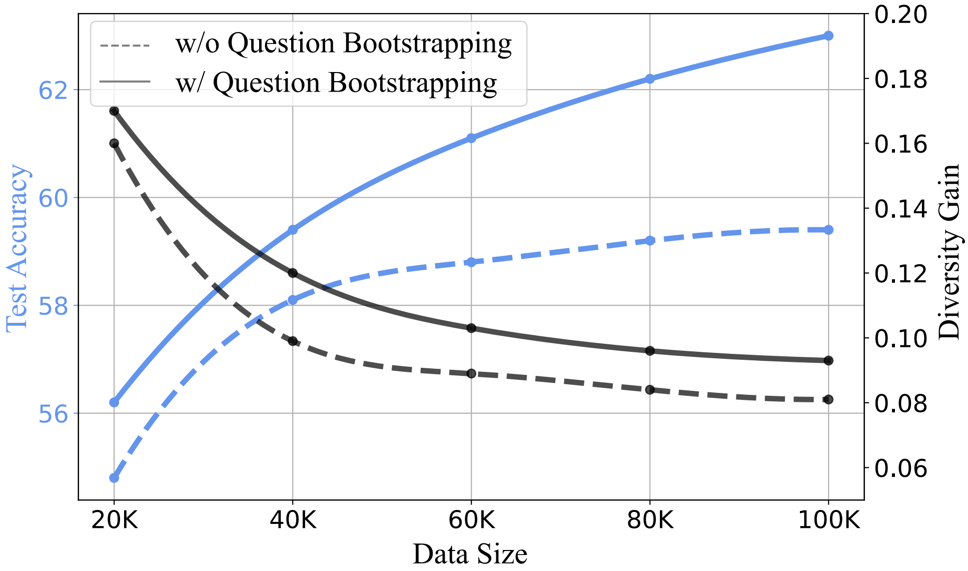


Figure 7.1: *GSM8K* accuracy of *LLaMA-2-7B* finetuned on different sizes of answer augmentation data. Larger diversity gain indicates the question is more diverse compared to the existing questions. Detailed experimental setup is given in Section 7.3.2.

an identifier "x" and ask the model to predict the masked number when the answer is provided. Different from [246, 103] that apply backward reasoning for *inference* verification, we use it as a form of question for language model *finetuning*. For answers, we adopt an answer augmentation method based on rejection sampling [268], where diverse reasoning paths are generated and only those with correct answers are used. After combining both forward and backward mathematical questions with augmented answers, we construct a new dataset for fine-tuning, called *MetaMathQA*. By fine-tuning *LLaMA-2* on *MetaMathQA*, we obtain a series of MetaMath models. Question bootstrapping is guided by the insight that a mathematical question represents merely a single view of the underlying meta-knowledge. Therefore, question bootstrapping can be viewed as a form of multi-view augmentation in order to enable the transfer of the meta-knowledge. Leveraging the *MetaMathQA* dataset, MetaMath demonstrates exceptional performance in mathematical reasoning, positioning it among the top performers on widely recognized evaluation benchmarks.

Another motivation behind question bootstrapping is to enlarge the question diversity [46] such that the question distribution can be rich enough to cover more unseen scenarios. We quantify the question diversity of the original questions and the proposed *MetaMathQA* dataset in Figure 7.1. The diversity gain [12] indicates how diverse the question is compared to the existing dataset, and a larger diversity gain means the new question is more different from the existing dataset. With question bootstrapping, the

MetaMathQA dataset is much more diverse than the original dataset. We also observe that the test accuracy without bootstrapped questions rapidly reaches a state of saturation. In contrast, the test accuracy, when using bootstrapped questions, continues to exhibit a steady increase. Our contributions are summarized as follows.

- We propose a novel question bootstrapping method to augment the training dataset, resulting in *MetaMathQA*. Question bootstrapping rewrites questions with both forward and backward reasoning paths and also leverages LLMs to rephrase the question text.
- Based on the *MetaMathQA* dataset, MetaMath is finetuned from state-of-the-art open-source LLMs (e.g., *LLaMA-2*), showing excellent elementary mathematical problem-solving capability.

7.2 The Proposed MetaMathQA

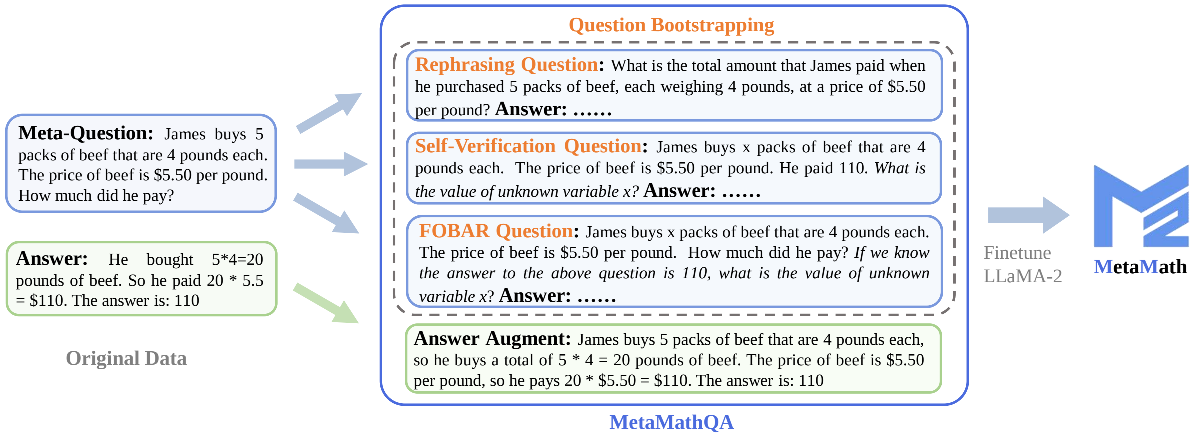

Question Bootstrapping
Rephrasing Question: What is the total amount that James paid when he purchased 5 packs of beef, each weighing 4 pounds, at a price of $5.50 per pound? Answer: ......
Meta-Question: James buys 5 packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. How much did he pay?
Self-Verification Question: James buys x packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. He paid 110. What is the value of unknown variable x? Answer: ......
Answer: He bought 5*4=20 pounds of beef. So he paid 20 * 5.5 = $110. The answer is: 110
FOBAR Question: James buys x packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. How much did he pay? If we know the answer to the above question is 110, what is the value of unknown variable x? Answer: ......
Finetune LLaMA-2
MetaMath
Original Data
Answer Augment: James buys 5 packs of beef that are 4 pounds each, so he buys a total of 5 * 4 = 20 pounds of beef. The price of beef is $5.50 per pound, so he pays 20 * $5.50 = $110. The answer is: 110
MetaMathQA

Figure 7.2: Overview of MetaMath.

The overview of our method is illustrated in Figure 7.2. Given a **meta-question** (a sample in the original mathematical training set), we can generate a series of variants. Specifically, we perform three types of question bootstrapping. Combined with answer augmentation, we present *MetaMathQA*, a diverse and high-quality mathematical dataset based on *GSM8K* and *MATH*. We then present MetaMath, a family of LLMs finetuned on *MetaMathQA* focusing on improving open-source models' mathematical reasoning ability.

7.2.1 Answer Augmentation

Generating more reasoning paths is a simple but effective way to augment the training set. For a question q_i, we use few-shot ComplexCoT prompting with temperature sampling to generate K_{AnsAug} more reasoning paths $\{(r_i^{(j)}, a_i^{(j)}) : j = 1, \ldots, K_{\text{AnsAug}}\}$: the question is appended to a few in-context reasoning examples, then fed to the LLM for generating its reasoning path $r_i^{(j)}$ and answer $a_i^{(j)}$. We filter out reasoning paths with correct answers as:

$$\mathcal{D}_{\text{AnsAug}} = \{(q_i, r_i^{(j)}, a_i^{(j)}) : a_i^{(j)} = a_i^\star; i = 1, \ldots, N_q; j = 1, \ldots, K_{\text{AnsAug}}\}. \tag{7.1}$$

Example 7.2.1: Answer Augmentation

Question: James buys 5 packs of beef that are 4 pounds each. The price of beef is \$5.50 per pound. How much did he pay?
Answer: (sample answers from GPT-3.5-Turbo)

7.2.2 Question Bootstrapping by LLM Rephrasing

Generating more answers for mathematical questions with LLMs is straightforward, but *creating questions is more difficult*. Math Questions are written by well-educated teachers. Hence, enlarging the question set through manual creation is time-consuming and labor-intensive. To address this issue, we propose *rephrasing prompting* to generate more questions through LLMs to **enhance the meta-knowledge of forward reasoning**.

Example 7.2.2: Rephrasing Question

Question: What is the total amount that James paid when he purchased 5 packs of beef, each weighing 4 pounds, at a price of \$5.50 per pound?
Answer: Each pack of beef weighs 4 pounds, so 5 packs weigh 4 * 5 = 20 pounds in total. The price per pound of beef is \$5.50, so the total cost for 20 pounds is 20 * \$5.50 = \$110. ... The answer is: 110.

Specifically, for a question q_i, we append it to the prompt, which is then fed to the LLM for generating the rephrased question. Example 7.2.2 shows a rephrased question generated by *GPT-3.5-Turbo* and the rephrasing prompt for *GSM8K* is shown in Example 7.2.3. We adopt temperature sampling to sample K_{rephrase} rephrased questions for each meta-question. It is time-consuming to manually check the consistency between the rephrased questions and the original questions. To overcome this difficulty, we propose a supervised method to evaluate the correctness between the rephrased questions and

Example 7.2.3: Prompt for Rephrasing *GSM8K* Questions

You are an AI assistant to help me rephrase questions. Follow the given examples.

Question: Angelo and Melanie want to plan how many hours over the next week they should study together for their test next week. They have 2 chapters of their textbook to study and 4 worksheets to memorize. They figure out that they should dedicate 3 hours to each chapter of their textbook and 1.5 hours for each worksheet. If they plan to study no more than 4 hours each day, how many days should they plan to study total over the next week if they take a 10-minute break every hour, include 3 10-minute snack breaks each day, and 30 minutes for lunch each day?
Rephrase the above question: Angelo and Melanie need to study 2 chapters in their textbook and 4 worksheets for their upcoming test. They have planned to dedicate 3 hours for each chapter and 1.5 hours for each worksheet. They can study for a maximum of 4 hours each day, taking into account 10-minute breaks every hour, 3 10-minute snack breaks per day, and 30 minutes for lunch. How many days do they need to study in total over the next week to complete their study plan?

Question: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
Rephrase the above question: If Leah had 32 chocolates and her sister had 42, and they both consumed 35 chocolates, what is the total number of chocolates that they have left?

Question: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
Rephrase the above question: What is the amount of money that Olivia has left after purchasing five bagels for $3 each, if she initially had $23?

Question: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
Rephrase the above question: If there were initially nine computers in the server room and five more computers were added each day from Monday to Thursday, what is the current total number of computers in the server room?

Question: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
Rephrase the above question: After losing 23 golf balls on Tuesday and an additional 2 on Wednesday, how many golf balls does Michael have left if he initially had 58 golf balls?

Question: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
Rephrase the above question: If Jason initially had 20 lollipops and now has 12 after giving some to Denny, how many lollipops did he give to Denny?

Question: Sam bought a dozen boxes, each with 30 highlighter pens inside, for $10 each box. He rearranged five of these boxes into packages of six highlighters each and sold them for $3 per package. He sold the rest of the highlighters separately at the rate of three pens for $2. How much profit did he make in total, in dollars?
Rephrase the above question: Sam purchased 12 boxes, each containing 30 highlighter pens, at $10 per box. He repackaged five of these boxes into sets of six highlighters and sold them for $3 per set. He sold the remaining highlighters individually at a rate of three pens for $2. What is the total profit he made in dollars?

Question: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
Rephrase the above question: If there were initially 15 trees in the grove and the grove workers are planning to plant more trees today, resulting in a total of 21 trees, how many trees did the workers plant today?

Question: {Q}
Rephrase the above question:

the meta-questions. For each rephrased question $\hat{q}_i^{(j)}$, we use few-shot ComplexCoT prompting to generate its reasoning path $\hat{r}_i^{(j)}$ and answer $\hat{a}_i^{(j)}$, which is then compared with the ground-truth answer $a_i^\star$. The accuracy for answering the rephrased questions by *GPT-3.5-Turbo* is 76.30%, which is comparable to that of answering the original training questions (80.74%). This suggests that the quality of rephrased questions is preserved high while the question diversity is improved. We collect the rephrased questions with correct answers (i.e., $\hat{a}_i^{(j)} = a_i^\star$) as the augmented data:

$$\mathcal{D}_{\text{rephrase}} = \{(\hat{q}_i, \hat{r}_i^{(j)}, \hat{a}_i^{(j)}) : \hat{a}_i^{(j)} = a_i^\star; i = 1, \dots, N_q; j = 1, \dots, K_{\text{rephrase}}\}. \tag{7.2}$$

7.2.3 Question Bootstrapping by Backward Reasoning

Backward reasoning plays an important role in answering many mathematical questions, i.e., starting with a given condition and thinking backward to determine an unknown variable in the question. One specific example between a question and a backward question is illustrated in Example 7.2.4. Compared with forward questions, existing methods (SFT, RFT, WizardMath) have significantly lower accuracy on backward questions, as shown in Figure 7.3, motivating us to bootstrap backward questions to **improve the meta-knowledge of backward reasoning**.

Example 7.2.4: Question and Backward Question

Question: James buys 5 packs of beef that are 4 pounds each. The price of beef is \$5.50 per pound. How much did he pay?
Answer: He bought 5*4=20 pounds of beef. He paid 20*5.5=\$110. The answer is: 110 ✓

Backward Question: James buys **x** packs of beef that are 4 pounds each. The price of beef is \$5.50 per pound. How much did he pay? If we know the answer to the above question is 110, what is the value of unknown variable **x**?
Answer: The total weight of the beef is 4***x** because 4*5.5 = 22. ... The answer is: 27 ✗

To improve the backward reasoning ability of finetuned models, we generate more questions which can be solved in a backward manner: a number in the question q_i is masked by "**x**", while the LLM is asked to predict the value of "**x**" when its answer $a_i^\star$ is provided. Different from forward reasoning, which generates explicit intermediate steps towards the final answer, backward reasoning *starts with the answer* and generates multiple reasoning steps to predict the masked number. Representative backward reasoning methods include Self-Verification [246] and FOBAR proposed in Chapter 6.

Example 7.2.5: Rewriting Prompt

You are an AI assistant to help me rewrite question into a declarative statement when its answer is provided. Follow the given examples and rewrite the question.

Question: How many cars are in the parking lot? The answer is 5.
Result: There are 5 cars in the parking lot.

Question: How many trees did the grove workers plant today? The answer is 6.
Result: The grove workers planted 6 trees today.

Question: If they ate 35, how many pieces do they have left in total? The answer is 39.
Result: They have 39 pieces left in total if they ate 35.

Question: How many lollipops did Jason give to Denny? The answer is 8.
Result: Jason gave 8 lollipops to Denny.

Question: How many toys does he have now? The answer is 9.
Result: He now has 9 toys.

Question: How many computers are now in the server room? The answer is 29.
Result: There are 29 computers now in the server room.

Question: How many golf balls did he have at the end of wednesday? The answer is 33.
Result: He had 33 golf balls at the end of Wednesday.

Question: How much money does she have left? The answer is 8.
Result: She has 8 money left.

Question: {Q} The answer is {A}.
Result:

In Self-Verification (SV) [246], the question with the answer is first rewritten into a declarative statement, e.g., "How much did he pay?" (with the answer 110) is rewritten into "He paid $10". Then, a question for asking the value of **x** is appended, e.g., "What is the value of unknown variable **x**?". We propose using in-context learning to rewrite the question with the answer into a declarative statement, as shown in Example 7.2.5. Additionally, Example 7.2.6 gives an augmented example after rewriting. We collect the new questions and their generated reasoning paths with correct answers as the augmented data:

$$\mathcal{D}_{\text{SV}} = \{(\tilde{q}_i^{(j)}, \tilde{r}_i^{(j)}, \tilde{a}_i^{(j)}) : \tilde{a}_i^{(j)} = a_i^{\star}; i = 1, \ldots, N_q; j = 1, \ldots, K_{\text{SV}}\}. \tag{7.3}$$

Example 7.2.6: Self-Verification [246] Question

Question: James buys **x** packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. He paid 110. What is the value of unknown variable **x**?
Answer: To solve this problem, we need to determine the value of **x**, which represents the number of packs of beef that James bought. Each pack of beef weighs 4 pounds and ... The value of **x** is 5.

Example 7.2.7: FOBAR Question

Question: James buys **x** packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. How much did he pay? If we know the answer to the above question is 110, what is the value of unknown variable **x**?
Answer: James buys **x** packs of beef that are 4 pounds each, so he buys a total of 4x pounds of beef. The price of beef is $5.50 per pound, so the total cost of the beef is 5.50 * 4**x** = 22**x**. ... The value of **x** is 5.

Self-Verification needs to rewrite the question with an answer into a declarative statement, which is challenging for complex questions. To address this issue, The FOBAR method directly appends the answer to the question, i.e., "*If we know the answer to the above question is* $\{a_i^\star\}$ *, what is the value of unknown variable* x?" Example 7.2.7 shows an example. We collect the backward questions along with their correct answers (generated by *GPT-3.5-Turbo*) as our augmented data:

$$\mathcal{D}_{\text{FOBAR}} = \{(\bar{q}_i^{(j)}, \bar{r}_i^{(j)}, \bar{a}_i^{(j)}) : \bar{a}_i^{(j)} = a_i^\star; i = 1, \ldots, N_q; j = 1, \ldots, K_{\text{FOBAR}}\}. \tag{7.4}$$

7.2.4 Finetuning the LLMs

We merge all the augmented data, including answer-augmented data and bootstrapped questions (Rephrasing, Self-Verification, FOBAR) as

$$\mathcal{D}_{\text{MetaMathQA}} = \mathcal{D}_{\text{AnsAug}} \cup \mathcal{D}_{\text{rephrase}} \cup \mathcal{D}_{\text{SV}} \cup \mathcal{D}_{\text{FOBAR}}.$$

We finetune an LLM model (parameterized by $\boldsymbol{\theta}$) on $\mathcal{D}_{\text{MetaMathQA}}$ to obtain the MetaMath model by maximizing the log-likelihood of the reasoning path conditioned on the question, i.e.,

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{(q,r,a)\in\mathcal{D}_{\text{MetaMathQA}}} \log \mathbb{P}(r \mid q; \boldsymbol{\theta}).$$

Although we only consider *LLaMA-2* and *Mistral* in this chapter, *MetaMathQA* can also be used to finetune other LLMs.

7.3 Experiments

7.3.1 Proposed *MetaMathQA* Dataset

Seed Datasets. We use two popular mathematical reasoning benchmarks (the seed datasets): (i) *GSM8K* [30] is a dataset consisting of high-quality grade school math problems, containing $7,473$ training samples and $1,319$ testing samples; and (ii) *MATH* [77] dataset consists of high school math competition problems that span seven subjects including *PreAlgebra*, *Algebra*, *Number Theory*, *Counting and Probability*, *Geometry*, *Intermediate Algebra*, and *PreCalculus*. It contains $7,500$ and $5,000$ samples for training and testing, respectively. Questions in *GSM8K* [30] take between 2 and 8 steps to reach the answer, while *MATH* is much more challenging and needs more steps.

Implementation Details. *GPT-3.5-Turbo* is used for rephrasing questions as well as generating answers in all four augmentations, where the temperature is set to 0.7 as in [237].

Results. Table 7.1 illustrates the detailed description of our *MetaMathQA* dataset. Specifically, it contains 155K, 130K, 55K, and 55K augmented samples from AnsAug, Rephrasing, Self-Verification, and FOBAR augmentations, respectively. Moreover, 240K samples are augmented from *GSM8K*, while 155K samples are augmented from *MATH*. In the next section, we will conduct experiments to verify that MetaMathQA is beneficial to improving open-source models' mathematical reasoning ability.

Table 7.1: Number of samples in the proposed *MetaMathQA*.

Datasets	AnsAug	Rephrasing	SV	FOBAR	Overall
MetaMathQA-GSM8K	80K	80K	40K	40K	240K
MetaMathQA-MATH	75K	50K	15K	15K	155K
MetaMathQA	155K	130K	55K	55K	395K

7.3.2 Usefulness of *MetaMathQA*

In this section, we validate whether the proposed *MetaMathQA* dataset is useful to finetune open-source models for improving their mathematical reasoning abilities. **All experiments in Section 7.3.2 were conducted by Longhui Yu, another co-author of MetaMath [267]. Longhui Yu and I jointly worked on designing the experiments and**

analyzing the experimental results. Section 7.3.2.2 shows the key findings. More results can be found in [267].

7.3.2.1 Setup

Models. We use the current state-of-the-art open-source model *LLaMA-2* [227], including three different parameter sizes: 7B, 13B, and 70B, as the base model for fine-tuning. *GPT-3.5-Turbo* is used for rephrasing questions as well as generating answers in all four augmentations, where the temperature is set to 0.7 as in [237]. The *LLaMA-2-7B* and *LLaMA-2-13B* are trained by fully fine-tuning. *LLaMA-2-70B* is finetuned by QLoRA [37] for computational efficiency.

Implementation Details. For the fully fine-tuning setting, we use the AdamW optimizer to train the model with 3 epochs and the batch size is 128. We use 8 NVIDIA A100 GPUs to train the 7B and 13B models, the learning rate is set as 2e-5 with a 3% learning rate warmup. For the 70B model QLoRA fine-tuning, the LoRA rank and alpha are 96 and 16, with a 0.05 dropout between the two matrices. The LoRA matrices are appended to both the attention layer and the MLP layer. We use the same AdamW optimizer but with a 1e-4 learning rate and without a learning rate warmup. The Training Prompt (Example 7.3.1) is from Alpaca [222], where the instruction is replaced by the *MetaMathQA* question.

Example 7.3.1: Training Prompt

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction: {instruction}

Response:

Example 7.3.2: Evaluation Prompt

Below is an instruction that describes a task. Write a response that appropriately completes the request.

Instruction: {instruction}

Response: Let's think step by step.

Evaluation Prompting. Different from the few-shot prompting evaluation for closed-source models, we find that zero-shot prompting is better for finetuned LLMs, which also saves more inference costs. Specifically, MetaMath uses the zero-shot Evaluation Prompt (Example 7.3.2) for *GSM8K* and *MATH*, where the instruction is replaced by the testing question. We set the temperature as 0 for fine-tuned models.

Answer Extraction. Different from the Wei et al. [243], where they use complex string rules to extract the final answer. In line with WizardMath [145], MetaMath only extracts the string behind "*The answer is:*" as the final answer. To teach the model this extraction method, we append "*The answer is: {ground-truth answer}*" to the end of answers in the *MetaMathQA* dataset.

Baselines. The proposed methods are compared with (i) closed-source models such as *GPT-3.5-Turbo* [162], *PaLM* [29]; (ii) open-source models such as *LLaMA-1* [226], *LLaMA-2* [227]; (iii) Supervised Fine-Tuning (SFT), which uses the training set of the original *GSM8K* or *MATH* datasets; (iv) Rejection sampling Fine-Tuning (RFT) [268] generates and collects correct reasoning paths as augmented data for fine-tuning; (v) WizardMath [145] which generates samples and trains two reward models using *GPT-3.5-Turbo* to select samples for fine-tuning.

Diversity Gain. We use the diversity gain [12] to measure to what extent a new dataset added to a basic dataset can improve the overall data diversity. For a base dataset $\mathcal{D}_{base} = \{x_i = (q_i, r_i, a_i)\}_{i=1}^{N}$ with N samples, and a new dataset $\mathcal{D}_{new} = \{x_i = (q_i, r_i, a_i)\}_{i=1}^{M}$ with M samples, the diversity gain is defined as: $\mathcal{D}_{new}$ relative to $\mathcal{D}_{base}$ as: $d_{gain} = \frac{1}{M}\sum_{x_i \in \mathcal{D}_{new}} \min_{x_j \in \mathcal{D}_{base}}(\|f(x_i) - f(x_j)\|_2^2)$, where f is the feature extractor and we use the OpenAI Embedding API *text-embedding-ada-002* for feature extraction. For Figure 7.1, we change the data size of base data and select a fixed set of 20K new data points that the model has not encountered to form $\mathcal{D}_{new}$.

7.3.2.2 Main Results

Main Results. Table 7.2 shows the testing accuracy on *GSM8K* and *MATH*. As can be seen, for all three groups (1-10B, 11-50B, 51-70B), MetaMath achieves the state-of-the-art performance, demonstrating that *MetaMathQA* is useful for improving open-source models' mathematical reasoning ability.

We conduct experiments to **study the effect of augmentations** in *MetaMathQA*. We first finetune the *LLaMA-2-7B* model on augmented *GSM8K* (*MetaMath-GSM8K*) data, and test the finetuned model on *GSM8K* and *MATH*. Table 7.3 shows the testing accuracy of

Table 7.2: Comparison of testing accuracy to existing LLMs on *GSM8K* and *MATH*.

	#params	*GSM8K*	*MATH*
	closed-source models		
GPT-4 [163]	-	92.0	42.5
GPT-3.5-Turbo [162]	-	80.8	34.1
PaLM [29]	8B	4.1	1.5
PaLM [29]	62B	33.0	4.4
PaLM [29]	540B	56.5	8.8
PaLM-2 [4]	540B	80.7	34.3
Flan-PaLM 2 [4]	540B	84.7	33.2
Minerva [121]	8B	16.2	14.1
Minerva [121]	62B	52.4	27.6
Minerva [121]	540B	58.8	33.6
	open-source models (1-10B)		
LLaMA-2 [227]	7B	14.6	2.5
MPT [153]	7B	6.8	3.0
Falcon [172]	7B	6.8	2.3
InternLM [89]	7B	31.2	-
GPT-J [233]	6B	34.9	-
ChatGLM 2 [272]	6B	32.4	-
Qwen [6]	7B	51.6	-
Baichuan-2 [256]	7B	24.5	5.6
SFT [227]	7B	41.6	-
RFT [268]	7B	50.3	-
MAmooTH-CoT [269]	7B	50.5	10.4
WizardMath [145]	7B	54.9	10.7
MetaMath	7B	**66.5**	**19.8**
	open-source models (11-50B)		
LLaMA-2 [227]	13B	28.7	3.9
LLaMA-2 [227]	34B	42.2	6.2
MPT [153]	30B	15.2	3.1
Falcon [172]	40B	19.6	2.5
Vicuna [28]	13B	27.6	-
SFT [227]	13B	50.0	-
RFT [268]	13B	54.8	-
MAmooTH-CoT [269]	13B	56.3	12.9
WizardMath [145]	13B	63.9	14.0
MetaMath	13B	**72.3**	**22.4**
	open-source models (51-70B)		
LLaMA-2 [227]	70B	56.8	13.5
RFT [268]	70B	64.8	-
Platypus [115]	70B	70.6	15.6
MAmooTH-CoT [269]	70B	72.4	21.1
WizardMath [145]	70B	81.6	22.7
MetaMath	70B	**82.3**	**26.6**

Table 7.3: Effect of different question augmentation with *LLaMA-2-7B* finetuned on *GSM8K* or *MATH*.

	finetuned on *GSM8K*						finetuned on *MATH*					
	AnsAug	Rep.	SV	FOBAR	*GSM8K*	*MATH*	AnsAug	Rep.	SV	FOBAR	*GSM8K*	*MATH*
SFT [227]	✗	✗	✗	✗	41.6	3.0	✗	✗	✗	✗	13.8	4.7
MetaMath	✓	✗	✗	✗	59.6	4.4	✓	✗	✗	✗	28.4	12.9
	✗	✓	✗	✗	59.7	4.4	✗	✓	✗	✗	30.4	12.4
	✓	✓	✗	✗	60.6	4.4	✓	✓	✗	✗	29.1	15.3
	✓	✓	✓	✓	**64.4**	**5.7**	✓	✓	✓	✓	**34.6**	**17.7**

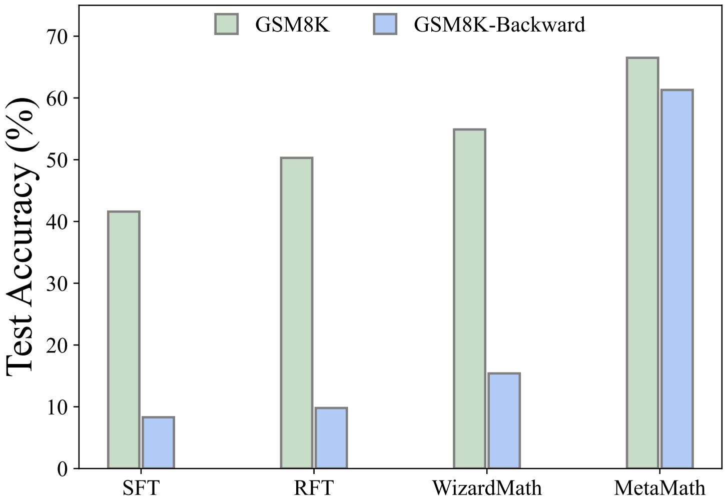


Figure 7.3: The accuracy gap between *GSM8K* and *GSM8K-Backward*.

different combinations of augmentations. As can be seen, using all augmentations is significantly better than using answer augmentation alone, showing the effectiveness of three proposed question augmentations. Furthermore, combining backward questions also improves the testing accuracy. These observations also can be seen from Table 7.3, where we fine-tune *LLaMA-2-7B* on the augmented *MATH* (*MetaMathQA-MATH*) data then evaluate the model on *GSM8K* and *MATH*.

To **evaluate the backward mathematical capability**, we propose a *GSM8K-Backward* test set, including 1270 backward questions by using SV and FOBAR to augment the original *GSM8K* test set (as shown in Example 7.2.6 and Example 7.2.7). Figure 7.3 shows the accuracy comparison of different 7B mathematical LLMs between the *GSM8K* and *GSM8K-Backward* datasets. As can be seen, existing LLMs struggle to solve mathematical problems in backward rationales and our MetaMath has a significant improvement on both datasets. Specifically, the ways where different LLMs solve the backward mathematical problem are illustrated through Example 7.3.3.

Example 7.3.3: Case study about Reverse Mathematical problem

Question: A robe takes 2 bolts of blue fiber and x% that much white fiber. It takes a total of 3 bolts. What is the value of unknown variable x? (Ground-truth answer is 50)
SFT Answer: We know that the robe takes 2 * x% = 3 bolts. So 2 * x% = 3. So x% = 3/2 = 60%. So x = 60. #### 60. The answer is: 60
RFT Answer: We know that the robe takes 2 bolts of blue fiber and x% that much white fiber, so it takes 2 * 0.01 = 0.02 bolts of white fiber. So in total, it takes 2 + 0.02 = «2+0.02=2.02»2.02 bolts of blue and white fiber. Since we know that it takes 3 bolts in total, we can write the equation 2.02 = 3. We can solve for x by multiplying both sides by 3, getting 6.04 = 3 * 0.02. We can then write the equation 6.04 = 3, and solve for x by dividing both sides by 6.04, getting x = 0.02. #### 2 The answer is: 2
WizardMath Answer: Step 1: Define the variables. Let x% of the blue fiber be represented by the variable y. Then, the robe takes 2y bolts of blue fiber. Step 2: Set up the equation. Since the robe takes a total of 3 bolts, we can write the equation as follows: 2y + xy = 3 Step 3: Simplify the equation. We can simplify the equation by combining the terms with y: 2y + xy = 3 3y + xy = 3 Step 4: Solve for x. To solve for x, we need to isolate xy on one side of the equation. We can do this by subtracting 3y from both sides: 3y + xy - 3y = 3 - 3y xy = 3 - 3y Step 5: Find the value of x. Now, we need to find the value of x. We know that xy = 3 - 3y, and we want to express y in terms of x. To do this, we can divide both sides of the equation by x: (3 - 3y) / x = y Step 6: Solve for y. Now, we can solve for y by multiplying both sides of the equation by x: 3 - 3y = xy 3 - 3y = x(3 - 3y) Step 7: Simplify the equation. We can simplify the equation by expanding and combining like terms: 3 - 3y = 3 - 3y Step 8: Find the value of x. Now, we can see that y = 0, which means that x% of the blue fiber is 0%. Since we are dealing with a percentage, the only possible value for x is 0. Therefore, the value of unknown variable x is 0. Final Answer: x = 0. The answer is: x
Our Answer: The robe takes 2 bolts of blue fiber. It takes x% of that much white fiber, which means it takes (x/100) * 2 bolts of white fiber. In total, it takes 3 bolts. So, we can write the equation as: 2 + (x/100) * 2 = 3 Let's simplify and solve for x: 2 + (2x/100) = 3 To isolate x, we subtract 2 from both sides of the equation: 2 + (2x/100) - 2 = 3 - 2 (2x/100) = 1 To solve for x, we multiply both sides of the equation by 100/2: (2x/100) * (100/2) = 1 * (100/2) x = 50 The value of x is 50. #### 50 The answer is: 50

7.4 Conclusion

In this chapter, we focus on improving open-source LLMs' meta-knowledge of solving mathematical problems. By bootstrapping mathematical questions on *GSM8K* and MATH, we present a high-quality and diverse dataset *MetaMathQA*, involving forward reasoning and backward reasoning samples. Our family of LLMs finetuned on *MetaMathQA*, called MetaMath, have achieved state-of-the-art on mathematical benchmarks among all open-source LLMs, demonstrating that MetaMathQA is useful for boosting LLM mathematical problem-solving capabilities.

CHAPTER 8

Conclusion & Future Works

8.1 Conclusion

Meta-learning is used to accelerate learning new tasks from meta-knowledge extracted from historical tasks. In this thesis, we studied the meta-learning problems when tasks are complex. We first extend learning a meta-regularization for simple linear regression tasks to *nonlinear* tasks by proximal kernelized extension (Chapter 3). To deal with complex tasks, whose model weights are diverse, we formulate task-specific knowledge into *a subspace mixture*, where each subspace represents one type of knowledge (Chapter 4). As language models are usually large, learning multiple meta-models causes a heavy burden on computation and memory. We further propose to learn *a pool of multiple meta-prompts* for prompt learning in language models (Chapter 5). For more challenging mathematical tasks, we activated the *backward reasoning meta-knowledge* (based on CoT prompting) to verify candidate answers and proposed to combine the forward and backward reasoning for verification by LLMs (Chapter 6). As open-source models struggle to solve challenging mathematical problems, we proposed *bootstrapping questions from both forward and backward directions* to enhance the diversity of mathematical reasoning meta-knowledge (Chapter 7). Experiments are conducted in each chapter to verify the usefulness of the proposed algorithms. Detailed summaries for each chapter are listed as follows.

1. In Chapter 3, we proposed an efficient algorithm to learn a meta-regularization for *nonlinear* models by kernelized proximal regularization. Nonlinearity allows more powerful models like deep networks to deal with complex tasks. We formulated the inner problem as a dual problem and introduced a learnable proximal regularizer to the base learner. We theoretically established the local and global convergence of the proposed algorithm. Experiments on benchmark regression and classification datasets demonstrate that learning a nonlinear meta-regularizer is effective. Moreover, for regression tasks, the proposed algorithm has a closed-form solution in the base learner and, thus, is very efficient.

2. In Chapter 4, we formulated task model parameters into multiple subspaces and proposed a novel meta-learning algorithm MUSML to learn the subspace bases. MUSML is very general, thus, can be used on linear and nonlinear models. Generalization of the proposed MUSML is analyzed theoretically. Experiments on synthetic demonstrate that MUSML can discover the underlying subspaces of task model parameters. Empirical results on real-world datasets show that MUSML achieves state-of-the-art performance on complex tasks.

3. In Chapter 5, we proposed using a pool of meta-prompts to extract knowledge from meta-training tasks. We construct instance-dependent prompts by combining the meta-prompts via attention. We propose a novel algorithm MetaPrompter to combine learning a prompt pool with a novel soft verbalizer. Language models are frozen and only the pool is learnable, thus, the proposed MetaPrompter is parameter-efficient. Meta-learning a prompt pool is more flexible than meta-learning only a single prompt initialization (as in MetaPrompting) and allows better adaptation of complex tasks.

4. In Chapter 6, we studied the problem of verifying candidate answers to mathematical problems using chain-of-thought prompting. To complement the use of only forward reasoning meta-knowledge for verification, we activated the meta-knowledge backward reasoning: A simple template is introduced to create questions and a prompt is designed to ask the LLM to predict a masked word when a candidate answer is provided. Furthermore, we proposed FOBAR to combine forward and backward reasoning meta-knowledge for verification. Extensive experiments on six standard mathematical data sets and three LLMs show that the proposed FOBAR achieves state-of-the-art performance on mathematical reasoning tasks. Additionally, FOBAR can also be used on non-mathematical tasks and achieves superior performance on three datasets.

5. In Chapter 7, we focused on data augmentation for training open-source LLMs to improve the mathematical meta-knowledge. We proposed to bootstrapping mathematical questions on *GSM8K* and *MATH* by rewriting questions with both forward and backward reasoning paths. We presented a high-quality and diverse dataset *MetaMathQA*, involving forward reasoning and backward reasoning samples. By finetuning open-source LLMs on *MetaMathQA*, we obtain a family of models MetaMath, which have achieved state-of-the-art on mathematical

benchmarks among all open-source LLMs. Remarkably, MetaMath-7B reaches 66.5% on *GSM8K* and 19.8% on *MATH*, surpassing previous open-source LLMs by a significant margin. This chapter further emphasizes the importance of the characteristics of the training data in boosting LLM problem-solving capabilities.

8.2 Future Works

In the future, we would like to work on the following topics.

- **Meta-learning low-rank matrices for subspace learning.** In Chapter 4, we studied the problem of learning multiple subspaces for building task-specific models. For task-specific models with parameter $\boldsymbol{\theta}_\tau \in \mathbb{R}^{d_{in} \times d_{out}}$, the size of meta-parameters is $K \times m \times d_{in} \times d_{out}$ (K is the number of subspaces, m is the subspace dimension). For large models (e.g., *LLaMA-2-7B* [227]), this approach suffers from a heavy burden on memory and computation. One promising solution is to meta-learn low-rank matrices for subspace adaptation. Specifically, the task-specific model is built as $\boldsymbol{\theta} + \sum_{k=1}^{K} \lambda_{k,\tau} \mathbf{B}_k \boldsymbol{\Sigma}_\tau \mathbf{A}_k$, where $\boldsymbol{\theta}$ and $\{\mathbf{B}_k \in \mathbb{R}^{d_{in} \times m}, \mathbf{A}_k \in \mathbb{R}^{m \times d_{out}} : k = 1, \dots, K\}$ are meta-parameters shared across all tasks, $\lambda_{k,\tau}$ is a weight to indicate whether the kth subspace is suitable for task τ, and $\boldsymbol{\Sigma}_\tau$ is task-specific mixing coefficients learned in the base learner. By learning low-rank matrices, the number of meta-parameters reduces to $d_{in} \times d_{out} + K \times m \times (d_{in} + d_{out})$.

- **Structured MetaPrompter for black-box LLMs.** In Chapter 5, we propose MetaPrompter to learn a pool of continuous meta-prompts for constructing instance-dependent prompts. As the instance-dependent prompt is appended to $\mathbf{x}$ after the input embedding layer, MetaPrompter is only suitable for open-source models. Most commercial models like ChatGPT are close-source, thus, only accept discrete tokens (e.g,. "James buys x packs of beef that are 4 pounds each. The price of beef is $5.50 per pound. How much did he pay?") as inputs. To extend our MetaPrompter to closed-source models, we can train a generator $\mathcal{G}$ to produce a pool of discrete prompts and a retriever $\mathcal{R}$ to choose the suitable prompt. Specifically, for an input $\mathbf{x}$, we wrap it by a template "the question $\{\mathbf{x}\}$ is related to below knowledge" and feed it to the generator multiple times with different seeds to obtain a pool of discrete prompts $\{\mathbf{p}_1, \dots, \mathbf{p}_K\}$ (e.g., "mathematics", "food"). For each prompt $\mathbf{p}_i$, we append it to the input, which is then fed to the closed-source LLM to obtain an output $\hat{y}_i = \mathsf{LLM}([\mathbf{x}, \mathbf{p}_i])$. By comparing $\hat{y}_i$ with the ground-truth, we obtain a score to measure the quality of the prompt. The key challenge is designing the training objective and collecting diverse datasets to learn universal generator and retriever.

- **Backward reasoning for non-mathematical tasks.** In Chapter 6, we leveraged the backward reasoning meta-knowledge to verify candidate answers to mathematical problems. Backward reasoning is a general method and we extend it to two non-mathematical tasks. For general non-mathematical reasoning tasks, the key challenge is to find the informative words to be masked. A possible direction is training a model or designing a prompt to identify the informative words/sentences automatically.

Appendix

Proof for Theoretical Results in Chapter 3

Proof of Proposition 3.2.1. (i) Notice that $\mathbf{w}_\tau^{(\text{prox})}$ is affine in $\boldsymbol{\theta}$, thus, $\mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{prox})} - y)^2$ is convex in $\boldsymbol{\theta}$. The CommonMean algorithm is using stochastic gradient descent to minimize the population risk, and the global convergence of $\boldsymbol{\theta}_t$ follows from the stochastic convex optimization [44].

(ii) Similarly, $\mathbf{w}_\tau^{(\text{gd})}$ is affine in $\boldsymbol{\psi}$, thus, the loss $\mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{gd})} - y)^2$ is convex in $\boldsymbol{\psi}$. Using stochastic gradient descent, $\boldsymbol{\psi}_t$ achieves global convergence [44]. By the below Proposition 3.2.2, $\bar{\mathbf{w}}$ is the unique optima, and we finish the proof. □

Proof of Proposition 3.2.2. For each task τ, let $\mathbf{v}_\tau = \mathbf{w}_\tau^\star - \bar{\mathbf{w}}$, then $\{\mathbf{v}_\tau\}$ are i.i.d. random variables with zero mean. Denote $\mathbf{C}_\tau = \left(\lambda \mathbf{I} + \mathbf{X}_\tau^\top \mathbf{X}_\tau\right)^{-1}$. As $\mathbf{w}_\tau^{(\text{prox})} = \mathbf{C}_\tau \left(\lambda \boldsymbol{\theta} + \mathbf{X}_\tau^\top \mathbf{y}_\tau\right)$ and $\mathbf{y}_\tau = \mathbf{X}_\tau \mathbf{w}_\tau^\star + \boldsymbol{\xi}_\tau$, it follows that

$$\begin{aligned}
&\mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\mathbf{x}^\top \mathbf{w}_\tau^{(\text{prox})} - y)^2 \\
=\ & \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\lambda \mathbf{x}^\top \mathbf{C}_\tau \boldsymbol{\theta} + \mathbf{x}^\top \mathbf{C}_\tau \mathbf{X}_\tau^\top (\mathbf{X}_\tau \mathbf{w}_\tau^\star + \boldsymbol{\xi}_\tau) - \mathbf{x}^\top \mathbf{w}_\tau^\star - \xi)^2 \\
=\ & \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\lambda \mathbf{x}^\top \mathbf{C}_\tau \boldsymbol{\theta} + \mathbf{x}^\top \mathbf{C}_\tau \mathbf{X}_\tau^\top (\mathbf{X}_\tau \bar{\mathbf{w}} + \mathbf{X}_\tau \mathbf{v}_\tau + \boldsymbol{\xi}_\tau) - \mathbf{x}^\top \bar{\mathbf{w}} - \mathbf{x}^\top \mathbf{v}_\tau - \xi)^2 \\
=\ & \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\lambda \mathbf{x}^\top \mathbf{C}_\tau \boldsymbol{\theta} + \mathbf{x}^\top \mathbf{C}_\tau \mathbf{X}_\tau^\top \mathbf{X}_\tau \bar{\mathbf{w}} - \mathbf{x}^\top \bar{\mathbf{w}})^2 + \text{constant} \qquad (8.1) \\
=\ & \mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbb{E}_{Q_\tau} \sum_{(\mathbf{x},y)\in Q_\tau} (\lambda \mathbf{x}^\top \mathbf{C}_\tau (\boldsymbol{\theta} - \bar{\mathbf{w}}))^2 + \text{constant} \\
=\ & \lambda^2 \sigma_{\mathbf{x}}^2 n_q \mathbb{E}_\tau \mathbb{E}_{S_\tau} (\boldsymbol{\theta} - \bar{\mathbf{w}})^\top \mathbf{C}_\tau^2 (\boldsymbol{\theta} - \bar{\mathbf{w}}) + \text{constant},
\end{aligned}$$

where we have used the setting that $\mathbf{x}, \xi, \mathbf{X}_\tau, \boldsymbol{\xi}_\tau$, and $\mathbf{v}_\tau$ are independent to obtain (8.1). Since $\mathbb{E}_\tau \mathbb{E}_{S_\tau} \mathbf{C}_\tau^2 \succeq \lambda^{-2} \mathbf{I}$, we conclude that $\boldsymbol{\theta} = \bar{\mathbf{w}}$ is the unique optima.

For MAML with one gradient step $\mathbf{w}_\tau^{(\mathrm{gd})} = \boldsymbol{\psi} - \gamma \mathbf{X}_\tau^\top(\mathbf{X}_\tau\boldsymbol{\psi} - \mathbf{y}_\tau)$, it follows that

$$
\begin{aligned}
&\mathbb{E}_\tau\mathbb{E}_{S_\tau}\mathbb{E}_{Q_\tau}\sum_{(\mathbf{x},y)\in Q_\tau}(\mathbf{x}^\top\mathbf{w}_\tau^{(\mathrm{gd})} - y)^2\\
&= \mathbb{E}_\tau\mathbb{E}_{S_\tau}\mathbb{E}_{Q_\tau}\sum_{(\mathbf{x},y)\in Q_\tau}(\mathbf{x}^\top(\mathbf{I} - \gamma\mathbf{X}_\tau^\top\mathbf{X}_\tau)\boldsymbol{\psi} + \gamma\mathbf{x}^\top\mathbf{X}_\tau^\top\mathbf{y}_\tau - y)^2\\
&= \mathbb{E}_\tau\mathbb{E}_{S_\tau}\mathbb{E}_{Q_\tau}\sum_{(\mathbf{x},y)\in Q_\tau}(\mathbf{x}^\top(\mathbf{I} - \gamma\mathbf{X}_\tau^\top\mathbf{X}_\tau)\boldsymbol{\psi} + \gamma\mathbf{x}^\top\mathbf{X}_\tau^\top(\mathbf{X}_\tau\bar{\mathbf{w}} + \mathbf{X}_\tau\mathbf{v}_\tau + \boldsymbol{\xi}_\tau) - \mathbf{x}^\top\bar{\mathbf{w}} - \mathbf{x}^\top\mathbf{v}_\tau - \xi)^2\\
&= \mathbb{E}_\tau\mathbb{E}_{S_\tau}\mathbb{E}_{Q_\tau}\sum_{(\mathbf{x},y)\in Q_\tau}\Big(\mathbf{x}^\top(\mathbf{I} - \gamma\mathbf{X}_\tau^\top\mathbf{X}_\tau)(\boldsymbol{\psi} - \bar{\mathbf{w}})\Big)^2 + \text{constant}\\
&= n_q\sigma_{\mathbf{x}}^2\mathbb{E}_\tau\mathbb{E}_{S_\tau}\|(\mathbf{I} - \gamma\mathbf{X}_\tau^\top\mathbf{X}_\tau)(\boldsymbol{\psi} - \bar{\mathbf{w}})\|^2 + \text{constant}.
\end{aligned}
$$

As $\gamma < 1/\sigma_{\mathbf{x}}^2$, we conclude that $\boldsymbol{\psi} = \bar{\mathbf{w}}$ is the unique optima. □

Proof of Proposition 3.2.4. The ridge regression has an efficient closed-form solution

$$
\mathbf{w}^{(\mathrm{prox})} = \Big(\lambda\mathbf{I} + \mathbf{X}^\top\mathbf{X}\Big)^{-1}\Big(\lambda\boldsymbol{\theta} + \mathbf{X}^\top\mathbf{y}\Big).
$$

Using the SVD decomposition of $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$ and $\mathbf{y} = \mathbf{X}\mathbf{w}^\star + \boldsymbol{\xi}$, we obtain

$$
\begin{aligned}
\mathbf{w}^{(\mathrm{prox})} &= \Big(\mathbf{I} + \lambda^{-1}\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top\Big)^{-1}\Big(\mathbf{V}\mathbf{a}_0 + \mathbf{V}^\perp\mathbf{b}_0 + \lambda^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^\top\mathbf{y}\Big)\\
&= \Big(\mathbf{I} + \lambda^{-1}\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^\top\Big)^{-1}\Big(\mathbf{V}\mathbf{a}_0 + \mathbf{V}^\perp\mathbf{b}_0 + \lambda^{-1}\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{a}^\star + \lambda^{-1}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}\boldsymbol{\xi}\Big) \qquad (8.2)\\
&= \mathbf{V}^\perp\mathbf{b}_0 + \mathbf{V}(\mathbf{I} + \lambda^{-1}\boldsymbol{\Sigma}^2)^{-1}\Big(\mathbf{a}_0 + \lambda^{-1}\boldsymbol{\Sigma}^2\mathbf{a}^\star\Big) + \mathbf{V}\Big(\lambda\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}\Big)^{-1}\mathbf{U}^\top\boldsymbol{\xi}, \qquad (8.3)
\end{aligned}
$$

where we have used $\mathbf{U}^\top\mathbf{y} = \mathbf{U}^\top(\mathbf{X}\mathbf{w}^\star + \boldsymbol{\xi}) = \mathbf{U}^\top\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top(\mathbf{V}\mathbf{a}^\star + \mathbf{V}^\perp\mathbf{b}^\star) + \mathbf{U}^\top\boldsymbol{\xi} = \boldsymbol{\Sigma}\mathbf{a}^\star + \mathbf{U}^\top\boldsymbol{\xi}$ in (8.2) and the Woodbury identity in (8.3). Then the estimation error is

$$
\mathbf{w}^{(\mathrm{prox})} - \mathbf{w}^\star = \mathbf{V}^\perp(\mathbf{b}_0 - \mathbf{b}^\star) + \mathbf{V}(\mathbf{I} + \lambda^{-1}\boldsymbol{\Sigma}^2)^{-1}(\mathbf{a}_0 - \mathbf{a}^\star) + \mathbf{V}\Big(\lambda\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}\Big)^{-1}\mathbf{U}^\top\boldsymbol{\xi}.
$$

Taking the square ℓ_2-norm and then expectation over $\boldsymbol{\xi}$ on both sides, we have

$$
\begin{aligned}
&\mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{w}^{(\mathrm{prox})} - \mathbf{w}^\star\|^2\\
&= \|\mathbf{V}^\perp(\mathbf{b}_0 - \mathbf{b}^\star)\|^2 + \|\mathbf{V}(\mathbf{I} + \lambda^{-1}\boldsymbol{\Sigma}^2)^{-1}(\mathbf{a}_0 - \mathbf{a}^\star)\|^2 + \mathbb{E}_{\boldsymbol{\xi}}\|\mathbf{V}\Big(\lambda\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}\Big)^{-1}\mathbf{U}^\top\boldsymbol{\xi}\|^2 \qquad (8.4)\\
&= \|\mathbf{b}_0 - \mathbf{b}^\star\|^2 + \|(\mathbf{I} + \lambda^{-1}\boldsymbol{\Sigma}^2)^{-1}(\mathbf{a}_0 - \mathbf{a}^\star)\|^2 + \mathbb{E}_{\boldsymbol{\xi}}\|\Big(\lambda\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}\Big)^{-1}\mathbf{U}^\top\boldsymbol{\xi}\|^2\\
&= \|\tilde{\mathbf{b}}\|^2 + \sum_{j=1}^{n_s}\left(\frac{\lambda\tilde{\mathrm{a}}_j}{\lambda + \nu_j^2}\right)^2 + \sum_{j=1}^{n_s}\left(\frac{\nu_j\sigma_{\xi}}{\lambda + \nu_j^2}\right)^2,
\end{aligned}
$$

where (8.4) follows from the fact that $\mathbf{V}^{\perp}$ is $\mathbf{V}$'s orthogonal complement and $\boldsymbol{\xi}$ is independent with $\mathbf{X}$ (also the $\boldsymbol{\Sigma}$, $\mathbf{U}$ and $\mathbf{V}$). □

Proof of Lemma 3.4.2. As $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi}) \equiv \sum_{\tau \in \mathcal{T}} \sum_{(\mathbf{x},y) \in Q_\tau} \ell(\hat{y}, y)$, it suffices to show that $\ell(\hat{y}, y)$ is Lipschitz-smooth in $(\boldsymbol{\theta}, \boldsymbol{\phi})$.

Using the chain rule, we have

$$\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \ell(\hat{y}, y) = \nabla_1 \ell(\hat{y}, y) \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \hat{y}, \tag{8.5}$$

$$\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \hat{y} = \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} f_{\boldsymbol{\theta}}(\mathbf{z}) + (\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}))^\top \boldsymbol{\alpha}_\tau + (\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau)^\top \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}). \tag{8.6}$$

The Lipschitz properties of direct derivatives $\nabla_1 \ell(\hat{y}, y), \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} f_{\boldsymbol{\theta}}(\mathbf{z}), \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z})$, and $\mathcal{K}(\mathbf{Z}_\tau \mathbf{z})$ follow from the Assumption 1. It remains to claim $\boldsymbol{\alpha}_\tau$ and $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau$ are Lipschitz. Let $\mathbf{p} = \left[f_{\boldsymbol{\theta}}(\mathbf{z}_1); \ldots; f_{\boldsymbol{\theta}}(\mathbf{z}_{n_s}); \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_1); \ldots; \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_{n_s}) \right] \in \mathbb{R}^{n_s + n_s^2}$ be the input of the dual problem.

(i) Claim: $\underline{\boldsymbol{\alpha}_\tau \text{ is Lipschitz w.r.t. } (\boldsymbol{\theta}, \boldsymbol{\phi}) \text{ and } \boldsymbol{\alpha}_\tau(\mathbf{p}) \text{ is Lipschitz-smooth w.r.t. } \mathbf{p}.}$ To show $\boldsymbol{\alpha}_\tau$ is Lipschitz w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi})$, it suffices to show that $\|\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau\|$ is bounded. By the chain rule, $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau = \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathbf{p}$. Denote the dual objective by $g(\mathbf{p}, \boldsymbol{\alpha})$. By the implicit function theorem [198], $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau = -\left(\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) \right)^{-1} \frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)$, where

$$\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) = \sum_{(\mathbf{x}_i, y_i) \in S_\tau} \nabla^2_1 \ell(f_\tau(\mathbf{z}_i), y_i) \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_i) \mathcal{K}(\mathbf{Z}_\tau, \mathbf{z}_i)^\top + \mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau)$$

$$\frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau) = \left[\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau) \mathbf{D} \;\middle|\; (\mathcal{K}(\mathbf{Z}_\tau, \mathbf{Z}_\tau) \mathbf{D}) \otimes \boldsymbol{\alpha}_\tau^\top + \mathbf{v}^\top \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\alpha}_\tau^\top \right]$$

$$\mathbf{D} = \text{diag}([\nabla^2_1 \ell(f_\tau(\mathbf{z}_1), y_1); \ldots; \nabla^2_1 \ell(f_\tau(\mathbf{z}_{n_s}), y_{n_s})])$$

$$\mathbf{v} = [\nabla_1 \ell(f_\tau(\mathbf{z}_1), y_1); \ldots; \nabla_1 \ell(f_\tau(\mathbf{z}_{n_s}), y_{n_s})],$$

and $\otimes$ is the Kronecker product. It follows from the Assumption 3.4.1 that both $\nabla^2_{\boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)$ and $\frac{\partial^2}{\partial \mathbf{p} \partial \boldsymbol{\alpha}} g(\mathbf{p}, \boldsymbol{\alpha}_\tau)$ are Lipschitz w.r.t. $\mathbf{p}$. Hence, we conclude that $\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p})$ is Lipschitz, $\boldsymbol{\alpha}_\tau(\mathbf{p})$ is Lipschitz-smooth w.r.t. $\mathbf{p}$, and $\|\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p})\|$ is bounded. Again, the boundedness of $\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \mathbf{p}$ follows from the Lipschitz-smoothness of $\mathbf{p}$ w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi})$. We conclude that $\boldsymbol{\alpha}_\tau$ is Lipschitz w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi})$.

(ii) Claim: $\underline{\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau \text{ is Lipschitz w.r.t. } (\boldsymbol{\theta}, \boldsymbol{\phi}).}$ Given $(\boldsymbol{\theta}, \boldsymbol{\phi})$ and $(\boldsymbol{\theta}', \boldsymbol{\phi}')$, we show that

$$\|\nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau(\boldsymbol{\theta}, \boldsymbol{\phi}) - \nabla_{(\boldsymbol{\theta},\boldsymbol{\phi})} \boldsymbol{\alpha}_\tau(\boldsymbol{\theta}', \boldsymbol{\phi}')\| \leq \eta \|(\boldsymbol{\theta}, \boldsymbol{\phi}) - (\boldsymbol{\theta}', \boldsymbol{\phi}')\|$$

for some $\eta > 0$. For notation simplicity, let $\mathbf{\Phi} = (\boldsymbol{\theta}, \boldsymbol{\phi})$ and $\mathbf{\Phi}' = (\boldsymbol{\theta}', \boldsymbol{\phi}')$, then we have

$$\begin{aligned}
&\|\nabla_{\mathbf{\Phi}} \boldsymbol{\alpha}_\tau(\mathbf{\Phi}) - \nabla_{\mathbf{\Phi}} \boldsymbol{\alpha}_\tau(\mathbf{\Phi}')\| \\
&= \|\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi})) \nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}) - \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi}')) \nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}')\| \\
&= \|\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi})) \nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}) - \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi}')) \nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}') \pm \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi})) \nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}')\| \\
&\leq \|\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi}))\| \|\nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}) - \nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}')\| + \|\nabla_{\mathbf{\Phi}} \mathbf{p}(\mathbf{\Phi}')\| \|\nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi})) - \nabla_{\mathbf{p}} \boldsymbol{\alpha}_\tau(\mathbf{p}(\mathbf{\Phi}'))\|.
\end{aligned}$$

As $\mathbf{p}(\mathbf{\Phi})$ and $\boldsymbol{\alpha}_\tau(\mathbf{p})$ are Lipschitz-smooth, there exists $\eta > 0$ such that

$$\begin{aligned}
\|\nabla_{\mathbf{\Phi}} \boldsymbol{\alpha}_\tau(\mathbf{\Phi}) - \nabla_{\mathbf{\Phi}} \boldsymbol{\alpha}_\tau(\mathbf{\Phi}')\| &\leq \eta \|\mathbf{\Phi} - \mathbf{\Phi}'\| + \eta \|\mathbf{p}(\mathbf{\Phi}) - \mathbf{p}(\mathbf{\Phi}')\| \\
&\leq \eta \|\mathbf{\Phi} - \mathbf{\Phi}'\| + \eta \|\mathbf{\Phi} - \mathbf{\Phi}'\| \\
&= 2\eta \|\mathbf{\Phi} - \mathbf{\Phi}'\|.
\end{aligned}$$

We conclude that $\nabla_{\mathbf{\Phi}} \boldsymbol{\alpha}_\tau$ is 2η-Lipschitz.

By (i) and (ii), ℓ is Lipschitz-smooth w.r.t. the meta-parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$. Therefore, $\mathcal{L}_{\text{meta}}(\boldsymbol{\theta}, \boldsymbol{\phi})$ is Lipschitz-smooth w.r.t. $(\boldsymbol{\theta}, \boldsymbol{\phi})$ with a Lipschitz constant $\eta_{\text{meta}} > 0$. □

Proof of Theorem 3.4.1. Let $\mathbf{\Phi} = (\boldsymbol{\theta}, \boldsymbol{\phi})$. Let $\boldsymbol{\zeta}_t = \nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t) - \frac{1}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$, where $\frac{1}{b} \sum_{\tau \in \mathcal{B}_t} \mathbf{g}_\tau$ is an unbiased estimation of $\nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)$, Using the Taylor expansion, we have

$$\begin{aligned}
&\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_{t+1}) \\
&\leq \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t) + \nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)^\top (\mathbf{\Phi}_{t+1} - \mathbf{\Phi}_t) + \frac{1}{2} \eta_{\text{meta}} \|\mathbf{\Phi}_{t+1} - \mathbf{\Phi}_t\|^2 \\
&\leq \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t) - \eta_t (1 - \frac{\eta_{\text{meta}} \eta_t}{2}) \|\nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)\|^2 + \eta_t \nabla_{\mathbf{\Phi}_t}^\top \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t) \boldsymbol{\zeta}_t + \frac{1}{2} \eta_{\text{meta}} \eta_t^2 \sigma_{\mathbf{g}}^2.
\end{aligned}$$

Taking conditional expectation over $\boldsymbol{\zeta}_{t-1}$ on both sides and then taking the expectation over the random training samples, we have

$$\mathbb{E} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_{t+1}) \leq \mathbb{E} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t) - \frac{\eta_t}{2} \mathbb{E} \|\nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)\|^2 + \frac{1}{2} \eta_{\text{meta}} \eta_t^2 \sigma_{\mathbf{g}}^2, \tag{8.7}$$

where we have used $1 - \frac{\eta_{\text{meta}} \eta_t}{2} \geq \frac{1}{2}$. Rearranging the above inequality and summing over t, we have

$$\sum_{t=1}^T \frac{\eta_t}{2} \mathbb{E} \|\nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)\|^2 \leq \mathbb{E} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_1) + \eta_{\text{meta}} \sigma_{\mathbf{g}}^2 \sum_{t=1}^T \eta_t^2. \tag{8.8}$$

Since $\eta_t = \min({}^1/\sqrt{T}, {}^1/2\eta_{\text{meta}})$, we have $\sum_{t=1}^T \eta_t^2 \leq 1$. Diving both sides by ${}^1/\sqrt{T}$, we conclude that $\min_{1 \leq t \leq T} \mathbb{E} \|\nabla_{\mathbf{\Phi}_t} \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)\|^2 = \mathcal{O}\left(\sigma_{\mathbf{g}}^2/\sqrt{T}\right)$. □

Proof of Proposition 4.2.1. This proposition is a property of linear regression tasks and has been mentioned in [111, 228]. We include the proof here for completeness.

By the definition $y = \mathbf{x}^\top \mathbf{w} + \xi$, we have

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{(\mathbf{x},y)\sim\tau,(\mathbf{x}',y')\sim\tau} yy'\mathbf{x}\mathbf{x}'^\top \\
&= \mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{x}'\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\xi\sim\mathcal{N}(0,\sigma_\xi^2),\xi'\sim\mathcal{N}(0,\sigma_\xi^2)}(\mathbf{x}^\top\mathbf{w}^\star_\tau+\xi)(\mathbf{x}'^\top\mathbf{w}^\star_\tau+\xi')\mathbf{x}\mathbf{x}'^\top \\
&= \mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{x}'\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\xi\sim\mathcal{N}(0,\sigma_\xi^2),\xi'\sim\mathcal{N}(0,\sigma_\xi^2)}(\mathbf{x}^\top\mathbf{w}^\star_\tau\mathbf{x}'^\top\mathbf{w}^\star_\tau\mathbf{x}\mathbf{x}'^\top \\
&\qquad + \mathbf{x}^\top\mathbf{w}^\star_\tau\xi'\mathbf{x}\mathbf{x}'^\top + \xi\mathbf{x}'^\top\mathbf{w}^\star_\tau\mathbf{x}\mathbf{x}'^\top + \xi\xi'\mathbf{x}\mathbf{x}'^\top) \\
&= \mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{x}'\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbf{x}^\top\mathbf{w}^\star_\tau\mathbf{x}'^\top\mathbf{w}^\star_\tau\mathbf{x}\mathbf{x}'^\top,
\end{aligned}
$$

where the last equality follows from the independence of $\xi, \xi', \mathbf{x}$, and $\mathbf{x}'$. Using the independence of $\mathbf{x}$, and $\mathbf{x}'$, we obtain

$$
\begin{aligned}
&\mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{0},\mathbf{I}),\mathbf{x}'\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbf{x}^\top\mathbf{w}^\star_\tau\mathbf{x}'^\top\mathbf{w}^\star_\tau\mathbf{x}\mathbf{x}'^\top \\
&= \mathbb{E}_{\tau\sim p(\tau)}(\mathbb{E}_{\mathbf{x}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbf{x}\mathbf{x}^\top)\mathbf{w}^\star_\tau\mathbf{w}^{\star\top}_\tau(\mathbb{E}_{\mathbf{x}'\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\mathbf{x}'\mathbf{x}'^\top) \\
&= \mathbb{E}_{\tau\sim p(\tau)}\mathbf{w}^\star_\tau\mathbf{w}^{\star\top}_\tau.
\end{aligned}
$$

$\square$

Proof of Theorem 3.4.2. Let $\mathbf{\Phi} = (\boldsymbol{\theta}, \boldsymbol{\phi})$. By the chain rule, we have

$$
\begin{aligned}
\nabla_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}) &= \frac{1}{|\mathcal{T}|}\sum_{\tau\in\mathcal{T}}\sum_{(\mathbf{x},y)\in Q_\tau}\nabla_1\ell(\hat{y},y)\nabla_{\mathbf{\Phi}}\hat{y} && (8.9) \\
&= \frac{1}{|\mathcal{T}|}\mathcal{G}(\mathbf{\Phi})^\top\nabla_{\mathbf{\Phi}}\mathcal{M}(\mathbf{\Phi}), && (8.10)
\end{aligned}
$$

where $\mathcal{G}(\mathbf{\Phi}) \equiv \begin{bmatrix}\cdots & \nabla_1\ell(\hat{y},y) & \cdots\end{bmatrix} \in \mathbb{R}^{n_q|\mathcal{T}|}$ stacks all gradients of the losses on query examples as a vector. Hence, we establish the Polyak-Lojasiewicz (PL) inequality [179] as follows

$$
\begin{aligned}
\|\nabla_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi})\|^2 &= \frac{1}{|\mathcal{T}|^2}\left\|\mathcal{G}(\mathbf{\Phi})^\top\nabla_{\mathbf{\Phi}}\mathcal{M}(\mathbf{\Phi})\right\|^2 \\
&= \frac{1}{|\mathcal{T}|^2}\mathcal{G}(\mathbf{\Phi})^\top\nabla_{\mathbf{\Phi}}\mathcal{M}(\mathbf{\Phi})\nabla_{\mathbf{\Phi}}^\top\mathcal{M}(\mathbf{\Phi})\mathcal{G}(\mathbf{\Phi}) \\
&\geq \frac{\mu}{|\mathcal{T}|^2}\|\mathcal{G}(\mathbf{\Phi})\|^2 && \text{(uniform conditioning)} \\
&= \frac{\mu}{|\mathcal{T}|^2}\sum_{\tau\in\mathcal{T}}\sum_{(\mathbf{x},y)\in Q_\tau}(\nabla_1\ell(\hat{y},y))^2 \\
&\geq \frac{\mu\rho}{2|\mathcal{T}|^2}\sum_{\tau\in\mathcal{T}}\sum_{(\mathbf{x},y)\in Q_\tau}(\ell(\hat{y},y)-\min_{y'}\ell(y',y)) && \text{(strongly convex)} \\
&\geq \frac{\mu\rho}{2|\mathcal{T}|}\left(\mathcal{L}_{\text{meta}}(\mathbf{\Phi})-\min_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi})\right).
\end{aligned}
$$

The PL inequality is commonly used in proving the global convergence of nonconvex optimization [105, 134]. Then, $\min_{1\leq t\leq T}\mathbb{E}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)-\min_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}) = \mathcal{O}\left(\sigma_{\text{g}}^2/\sqrt{T}\right)$ follows directly from Theorem 1.

For full gradient descent, the gradient noise $\boldsymbol{\zeta}_t = \nabla_{\mathbf{\Phi}_t}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t) - \frac{1}{b}\sum_{\tau\in\mathcal{B}_t}\mathbf{g}_\tau = \mathbf{0}$, thus, the noisy gradient will be the true gradient. By the Taylor expansion, it follows that

$$
\begin{aligned}
&\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_{t+1})-\min_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}) \\
&\leq \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)+\nabla_{\mathbf{\Phi}_t}^\top\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)(\mathbf{\Phi}_{t+1}-\mathbf{\Phi}_t)+\frac{\beta_{\text{meta}}}{2}\|\mathbf{\Phi}_{t+1}-\mathbf{\Phi}_t\|^2-\min_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}) \\
&= \mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)-\eta\|\nabla_{\mathbf{\Phi}_t}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)\|^2+\frac{\eta^2\beta_{\text{meta}}}{2}\|\nabla_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)\|^2-\min_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi}) \\
&\leq \left(1-\frac{\eta\mu\rho}{4|\mathcal{T}|}\right)(\mathcal{L}_{\text{meta}}(\mathbf{\Phi}_t)-\min_{\mathbf{\Phi}}\mathcal{L}_{\text{meta}}(\mathbf{\Phi})),
\end{aligned}
$$

and we obtain the exponential convergence. □

Proof for Theoretical Results in Chapter 4

Proof of Lemma 4.3.3. **Claim 1**: For $k\in\{1,\ldots,K\}$ and $i\in\{1,\ldots,n_s\}$, it holds that

$$
\|\mathbf{v}_{\tau,k}-\mathbf{v}_{\tau,k,i}\| \leq \frac{2\varrho(1+\alpha\varrho\rho^2 m)^J}{\rho\beta\sqrt{m}n_s}.
$$

By the update rule in the base learner, we have

$$\begin{aligned}\|\mathbf{v}_{\tau,k}^{(t'+1)} - \mathbf{v}_{\tau,k,i}^{(t'+1)}\| &= \|\mathbf{v}_{\tau,k}^{(t')} - \alpha\nabla_{\mathbf{v}_{\tau,k}^{(t')}}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')}) - \mathbf{v}_{\tau,k,i}^{(t')} + \alpha\nabla_{\mathbf{v}_{\tau,k,i}^{(t')}}\mathcal{L}(\mathcal{S}_\tau^{(i)};\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')})\| \\ &\leq \|\mathbf{v}_{\tau,k}^{(t')} - \mathbf{v}_{\tau,k,i}^{(t')}\| + \alpha\|\nabla_{\mathbf{v}_{\tau,k}^{(t')}}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')}) - \nabla_{\mathbf{v}_{\tau,k,i}^{(t')}}\mathcal{L}(\mathcal{S}_\tau^{(i)};\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')})\|.\end{aligned}$$

For the second term, by the chain rule, it follows that

$$\begin{aligned}&\left\|\nabla_{\mathbf{v}_{\tau,k}^{(t')}}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')}) - \nabla_{\mathbf{v}_{\tau,k,i}^{(t')}}\mathcal{L}(\mathcal{S}_\tau^{(i)};\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')})\right\| \\ =\ &\left\|\mathbf{S}_k^\top\left(\nabla_{\mathbf{w}}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')}) - \nabla_{\mathbf{w}}\mathcal{L}(\mathcal{S}_\tau^{(i)};\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')})\right)\right\| \\ \leq\ &\|\mathbf{S}_k\|_{\mathrm{F}} \cdot \|\frac{1}{n_s}\sum_{j\neq i}\Big(\nabla_{\mathbf{w}}\ell(f(\mathbf{z}_j;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')})) - \nabla_{\mathbf{w}}\ell(f(\mathbf{z}_j;\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')}))\Big) \\ &+ \frac{1}{n_s}\Big(\nabla_{\mathbf{w}}\ell(f(\mathbf{z}_i;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')})) - \nabla_{\mathbf{w}}\ell(f(\mathbf{z}'_i;\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')}))\Big)\ \| \end{aligned} \tag{8.11}$$

$$\begin{aligned}\leq \rho\sqrt{m}\Bigg(&\frac{1}{n_s}\sum_{j\neq i}\left\|\nabla_{\mathbf{w}}\ell(f(\mathbf{z}_j;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')})) - \nabla_{\mathbf{w}}\ell(f(\mathbf{z}_j;\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')}))\right\| \\ &+ \frac{1}{n_s}\left\|\nabla_{\mathbf{w}}\ell(f(\mathbf{z}_j;\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')}))\right\| + \frac{1}{n_s}\left\|\nabla_{\mathbf{w}}\ell(f(\mathbf{z}'_i;\mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')}))\right\|\Bigg)\end{aligned} \tag{8.12}$$

$$\leq \rho\sqrt{m}\left(\frac{1}{n_s}\sum_{j\neq i}\beta\|\mathbf{S}_k\mathbf{v}_{\tau,k}^{(t')} - \mathbf{S}_k\mathbf{v}_{\tau,k,i}^{(t')}\| + \frac{2\varrho}{n_s}\right) \tag{8.13}$$

$$\leq \rho\sqrt{m}\left(\frac{n_s-1}{n_s}\beta\|\mathbf{S}_k\|\|\mathbf{v}_{\tau,k}^{(t')} - \mathbf{v}_{\tau,k,i}^{(t')}\| + \frac{2\varrho}{n_s}\right) \tag{8.14}$$

$$\leq m\rho^2\beta\|\mathbf{v}_{\tau,k}^{(t')} - \mathbf{v}_{\tau,k,i}^{(t')}\| + \frac{2\varrho\rho\sqrt{m}}{n_s}, \tag{8.15}$$

where Eq.(8.11) uses the norm inequality $\|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$ and $\|\mathbf{S}_k\| \leq \|\mathbf{S}_k\|_{\mathrm{F}}$, Eq.(8.12) uses the compactness assumption (thus $\|\mathbf{S}_k\|_{\mathrm{F}} \leq \rho\sqrt{m}$) and the triangle inequality, Eq.(8.13) uses the Lipschitzness of $\nabla_{\mathbf{w}}\ell(f(\mathbf{x};\mathbf{w}),y)$, Eq.(8.14) uses the Lipschitzness of $\ell(f(\mathbf{x};\mathbf{w}),y)$, and Eq.(8.15) uses the boundedness of $\|\mathbf{S}_k\|$ again. Hence, we obtain a recursive inequality

$$\|\mathbf{v}_{\tau,k}^{(t'+1)} - \mathbf{v}_{\tau,k,i}^{(t'+1)}\| \leq (1+\alpha m\rho^2\beta)\|\mathbf{v}_{\tau,k}^{(t')} - \mathbf{v}_{\tau,k,i}^{(t')}\| + \frac{2\alpha\varrho\rho\sqrt{m}}{n_s}. \tag{8.16}$$

By induction, we obtain a bound for $\mathbf{v}_{\tau,k}^{(J)} - \mathbf{v}_{\tau,k,i}^{(J)}$:

$$\begin{aligned}
\|\mathbf{v}_{\tau,k}^{(J)} - \mathbf{v}_{\tau,k,i}^{(J)}\| &\leq (1+\alpha m\rho^2\beta)\|\mathbf{v}_{\tau,k}^{(0)} - \mathbf{v}_{\tau,k,i}^{(0)}\| + \frac{2\alpha\varrho\rho\sqrt{m}}{n_s}\sum_{t'=0}^{J-1}(1+\alpha\beta\rho^2 m)^{t'} \\
&\leq \frac{2\varrho(1+\alpha\varrho\rho^2 m)^J}{\rho\beta\sqrt{m}n_s},
\end{aligned} \tag{8.17}$$

where we have used the fact $\mathbf{v}_{\tau,k}^{(0)} = \mathbf{v}_{\tau,k,i}^{(0)}$.

Claim 2: The stability constant of the base learner is $\frac{2\varrho(1+\alpha\beta\rho^2 m)^J}{n_s}$.

Next, we analyze the stability constant of the base learner:

$$\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}_i'\sim\tau}\left|\ell(f(\mathbf{x}_i;\mathbf{S}_k\mathbf{v}_{\tau,k}),y_i) - \ell(f(\mathbf{x}_i;\mathbf{S}_k\mathbf{v}_{\tau,k,i}),y_i)\right|$$

$$\leq \beta\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}_i'\sim\tau}\|\mathbf{S}_k\mathbf{v}_{\tau,k,i} - \mathbf{S}_k\mathbf{v}_{\tau,k}\| \tag{8.18}$$

$$\leq \beta\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}_i'\sim\tau}\|\mathbf{S}_k\|_{\mathrm{F}}\|\mathbf{v}_{\tau,k,i} - \mathbf{v}_{\tau,k}\| \tag{8.19}$$

$$\leq \beta\rho\sqrt{m}\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}_i'\sim\tau}\|\mathbf{v}_{\tau,k,i} - \mathbf{v}_{\tau,k}\| \tag{8.20}$$

$$\leq \beta\rho\sqrt{m}\cdot\frac{2\varrho(1+\alpha\beta\rho^2 m)^J}{\rho\beta\sqrt{m}n_s} \tag{8.21}$$

$$= \frac{2\varrho(1+\alpha\beta\rho^2 m)^J}{n_s}, \tag{8.22}$$

where Eq.(8.18) uses the Lipschitz property of ℓ, Eq.(8.19) uses the norm inequality, Eq.(8.20) uses the boundedness of $\|\mathbf{S}_k\|_{\mathrm{F}}$, Eq.(8.21) uses the inequality (8.17). The above equality reveals that the stability constant in Theorem 11 of [14] (β_2 there) is $\frac{2\varrho(1+\alpha\beta\rho^2 m)^J}{n_s}$. □

Proof of Theorem 4.3.1. The proof is based on the connection between generalization and stability [14].

We adopt the notations used in the proof of Lemma 4.3.3. We apply Lemma 4.3.2 to our algorithm and obtain

$$\begin{aligned}
&\mathbb{E}_{\mathcal{S}_\tau}\left[\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_k\mathbf{v}_{\tau,k}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k})\right]^2 \\
&\leq \frac{\nu^2}{2n_s} + 3\nu\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}_i'\sim\tau}\left|\ell(f(\mathbf{x}_i;\mathbf{S}_k\mathbf{v}_{\tau,k}),y_i) - \ell(f(\mathbf{x}_i;\mathbf{S}_k\mathbf{v}_{\tau,k,i}),y_i)\right| \\
&\leq \frac{\nu^2}{2n_s} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{n_s},
\end{aligned} \tag{8.23}$$

where (8.23) uses the equality (8.22) in Lemma 4.3.3. By the Cauchy-Schwarz inequality, we have

$$\mathbb{E}_{\mathcal{S}_\tau} \left|\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_k\mathbf{v}_{\tau,k}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k})\right| \leq \sqrt{\frac{\nu^2}{2n_s} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{n_s}}. \tag{8.24}$$

To provide an upper bound of $\mathcal{R}(\mathcal{S}) - \hat{\mathcal{R}}(\mathcal{S})$, we need to address the randomness in k_τ:

$$\begin{aligned}
\mathcal{R}(\mathcal{S}) - \hat{\mathcal{R}}(\mathcal{S}) &= \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau})\right] \\
&= \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\sum_{k=1}^{K}\mathbb{I}_{[k_\tau=k]}\left[\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_k\mathbf{v}_{\tau,k}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k})\right] \\
&\leq \mathbb{E}_\tau\sum_{k=1}^{K}\mathbb{E}_{\mathcal{S}_\tau}\mathbb{I}_{[k_\tau=k]}\left|\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_k\mathbf{v}_{\tau,k}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k})\right| \\
&\leq \mathbb{E}_\tau\sum_{k=1}^{K}\mathbb{E}_{\mathcal{S}_\tau}\left|\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_k\mathbf{v}_{\tau,k}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_k\mathbf{v}_{\tau,k})\right| \\
&\leq K\sqrt{\frac{\nu^2}{2n_s} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{n_s}},
\end{aligned}$$

where the first inequality is because the empirical loss can be smaller than the population loss, and the last inequality follows from the Eq.(8.24). □

Proof of Theorem 4.3.2. By the definition of excess risk, we have

$$\begin{aligned}
0 &\leq \mathcal{R}(\mathcal{S}) - \mathcal{R}^\star \\
&= \mathbb{E}_\tau[\mathbb{E}_{\mathcal{S}_\tau}\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}),y) - \mathbb{E}_{\mathcal{S}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}) + \mathbb{E}_{\mathcal{S}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}) \\
&\qquad - \mathbb{E}_{\mathcal{S}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau}) + \mathbb{E}_{\mathcal{S}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau}) - \mathbb{E}_{\mathbf{z}\sim\tau}\ell(\mathbf{z};\mathbf{w}^\star_\tau)] \qquad (8.25) \\
&= \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}),y) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau})\right] \\
&\quad + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau})\right] \\
&\quad + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\frac{1}{N_{tr}}\sum_{\mathbf{z}\in\mathcal{S}_\tau}\ell(f(\mathbf{x};\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau}),y) - \mathbb{E}_{\mathbf{z}\sim\tau}\ell(f(\mathbf{x};\mathbf{w}^\star_\tau),y)\right] \\
&\leq K\sqrt{\frac{\nu^2}{2N_{tr}} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{N_{tr}}} + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau})\right] \\
&\quad + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau}) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{w}^\star_{\tau,\,\mathbf{S}_{k_\tau}})\right] + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\|\nabla_{\mathbf{w}}\mathcal{L}(\mathcal{S}_\tau;\boldsymbol{\xi}_\tau)\|\|\mathbf{w}^\star_{\tau,\mathbf{S}^\perp_{k_\tau}}\| \qquad (8.26) \\
&\qquad (8.27)
\end{aligned}$$

$$\begin{aligned}&\leq K\sqrt{\frac{\nu^2}{2N_{tr}} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{N_{tr}}} + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau})\right]\\ &\quad + \varrho\mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\operatorname{dist}(\mathbf{w}^\star_\tau,\mathbb{S}_{k_\tau}), \qquad (8.28)\end{aligned}$$

where identity (8.25) follows by introducing two additional terms $\mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau})$ and $\mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau})$, Eq.(8.26) uses the bound in Theorem 4.3.1 and the mean value theorem (we decompose $\mathbf{w}^\star_\tau = \mathbf{w}^\star_{\tau,\mathbf{S}_{k_\tau}} + \mathbf{w}^\star_{\tau,\mathbf{S}^\perp_{k_\tau}}$ and $\boldsymbol{\xi}_\tau \in [\mathbf{w}^\star_{\tau,\mathbf{S}_{k_\tau}}, \mathbf{w}^\star_\tau]$), and Eq.(8.28) follows from the Lipschitzness assumption and $\mathbb{E}_{\mathcal{S}_\tau}\Big[\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau}) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{w}^\star_{\tau,\ \mathbf{S}_{k_\tau}})\Big] \leq 0$ as $\mathbf{v}^\star_{\tau,k_\tau}$ is an exact solution of the problem $\min_{\mathbf{v}_\tau}\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_\tau)$. We conclude that

$$\begin{aligned}&\mathcal{R}(\mathcal{S}) - \mathcal{R}^\star\\ &\leq K\sqrt{\frac{\nu^2}{2N_{tr}} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{N_{tr}}} + \mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\left[\mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}_{\tau,k_\tau}) - \mathcal{L}(\mathcal{S}_\tau;\mathbf{S}_{k_\tau}\mathbf{v}^\star_{\tau,k_\tau})\right]\\ &\quad + \varrho\mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\operatorname{dist}(\mathbf{w}^\star_\tau,\mathbb{S}_{k_\tau})\\ &\leq K\sqrt{\frac{\nu^2}{2N_{tr}} + \frac{6\nu\varrho(1+\alpha\beta\rho^2 m)^J}{N_{tr}}} + \rho\sqrt{m}\,\mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\|\mathbf{v}_{\tau,k_\tau} - \mathbf{v}^\star_{\tau,k_\tau}\| + \varrho\mathbb{E}_\tau\mathbb{E}_{\mathcal{S}_\tau}\operatorname{dist}(\mathbf{w}^\star_\tau,\mathbb{S}_{k_\tau}),\end{aligned}$$

where the last inequality is from the Lipschitzness of ℓ and $\|\mathbf{S}_{k_\tau}\| \leq \|\mathbf{S}_{k_\tau}\|_F \leq \rho\sqrt{m}$. □

References

[1] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive science*, 1985.

[2] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Neural Information Processing Systems*, 2019.

[3] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In *International Conference on Machine Learning*, 2018.

[4] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting,

Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. PaLM 2 technical report. Preprint arXiv:2305.10403, 2023.

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

[6] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. Preprint arXiv:2309.16609, 2023.

[7] Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *International Conference on Machine Learning*, 2019.

[8] Fan Bao, Guoqiang Wu, Chongxuan Li, Jun Zhu, and Bo Zhang. Stability and generalization of bilevel programming in hyperparameter optimization. In *Neural Information Processing Systems*, 2021.

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

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

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

[12] Jeff Bilmes. Submodularity in machine learning and artificial intelligence. Preprint arXiv:2202.00132, 2022.

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

[14] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2002.

[15] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[16] 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.

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

[18] 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*, 2022.

[19] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario

Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. Preprint arXiv:2107.03374, 2021.

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

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

[22] 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*, 2018.

[23] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of Thoughts Prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*, 2023.

[24] Xinyun Chen, Maxwell Lin, Nathanael Schaerli, and Denny Zhou. Teaching large language models to self-debug. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[25] 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*, 2022.

[26] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[27] 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*, 2018.

[28] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing GPT-4 with 90% ChatGPT quality. Technical report, 2023.

[29] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling language modeling with pathways. Preprint arXiv:2204.02311, 2022.

[30] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Hesse Christopher, and Schulman John. Training verifiers to solve math word problems. Preprint arXiv:2110.14168, 2021.

[31] Katherine M. Collins, Albert Q. Jiang, Simon Frieder, Lionel Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B. Tenenbaum, William Hart, Timothy Gowers, Wenda Li, Adrian Weller, and Mateja Jamnik. Evaluating language models for mathematics through interactions. Preprint arXiv:2306.01694, 2023.

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

[33] 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.

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

[35] 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.

[36] 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.

[37] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized llms. In *Neural Information Processing Systems*, 2023.

[38] 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.

[39] Shizhe Diao, Pengcheng Wang, Yong Lin, and Tong Zhang. Active prompting with chain-of-thought for large language models. Preprint arXiv:2302.12246, 2023.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] John C Duchi. Introductory lectures on stochastic optimization. *The Mathematics of Data*, 2018.

[45] 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.

[46] Ronen Eldan and Yuanzhi Li. TinyStories: How small can language models be and still speak coherent english? Preprint arXiv:2305.07759, 2023.

[47] 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.

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

[49] 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.

[50] Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. In *Workshop on Stylistic Variation*, 2017.

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

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

[53] 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.

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

[55] 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.

[56] Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, 2023.

[57] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. In *International Conference on Learning Representations*, 2023.

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

[59] 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.

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

[61] Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 2021.

[62] Saeed Ghadimi and Guanghui Lan. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013.

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

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

[65] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 2021.

[66] Riccardo Grazzi, Luca Franceschi, Massimiliano Pontil, and Saverio Salzo. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning*, 2020.

[67] Liang-Yan Gui, Yu-Xiong Wang, Deva Ramanan, and José MF Moura. Few-shot human motion prediction via meta-learning. In *European Conference on Computer Vision*, 2018.

[68] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, 2018.

[69] 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.

[70] 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.

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

[72] 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.

[73] 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.

[74] 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.

[75] Tong He, Chunhua Shen, Zhi Tian, Dong Gong, Changming Sun, and Youliang Yan. Knowledge adaptation for efficient semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[76] Yun He, Steven Zheng, Yi Tay, Jai Gupta, Yu Du, Vamsi Aribandi, Zhe Zhao, Yaguang Li, Zhao Chen, Donald Metzler, Heng-Tze Cheng, and Ed H. Chi. Hy-

perPrompt: Prompt-based task-conditioning of transformers. In *International Conference on Machine Learning*, 2022.

[77] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Neural Information Processing Systems: Datasets and Benchmarks*, 2021.

[78] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. Preprint arXiv:1503.02531, 2015.

[79] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[80] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

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

[82] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, 2019.

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

[84] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[85] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[86] 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.

[87] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. Large language models can Self-Improve. Preprint arXiv:2210.11610, 2022.

[88] Shima Imani, Liang Du, and Harsh Shrivastava. MathPrompter: Mathematical reasoning using large language models. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[89] InternLM. InternLM: A multilingual language model with progressively enhanced capabilities. Technical report, 2023.

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

[91] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2016.

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

[93] 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.

[94] 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*, 2020.

[95] Kaiyi Ji, Junjie Yang, and Yingbin Liang. Theoretical convergence of multi-step model-agnostic meta-learning. *Journal of Machine Learning Research*, 2022.

[96] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B. Technical Report arXiv:2310.06825, 2023.

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

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

[99] Weisen Jiang, James Kwok, and Yu Zhang. Subspace learning for effective meta-learning. In *International Conference on Machine Learning*, 2022.

[100] Weisen Jiang, Hansi Yang, Yu Zhang, and James Kwok. An adaptive policy to employ sharpness-aware minimization. In *International Conference on Learning Representations*, 2023.

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

[102] Weisen Jiang, Baijiong Lin, Han Shi, Yu Zhang, Zhenguo Li, and James Kwok. BYOM: Building your own multi-task model for free. Preprint arXiv:2310.01886, 2024.

[103] Weisen Jiang, Han Shi, Longhui Yu, Zhengying Liu, Yu Zhang, Zhenguo Li, and James Kwok. Forward-backward reasoning in large language models for mathematical verification. In *Findings of Annual Meeting of the Association for Computational Linguistics*, 2024.

[104] 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.

[105] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2016.

[106] Seyed Mehran Kazemi, Najoung Kim, Deepti Bhatia, Xin Xu, and Deepak Ramachandran. LAMBADA: Backward chaining for automated reasoning in natural language. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[107] Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Text modular networks: Learning to decompose tasks in the language of existing models. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

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

[109] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Neural Information Processing Systems*, 2022.

[110] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 2015.

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

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

[113] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 2015.

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

[115] Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. Platypus: Quick, cheap, and powerful refinement of LLMs. Preprint arXiv:2308.07317, 2023.

[116] 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*, 2019.

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

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

[119] 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.

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

[121] Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Neural Information Processing Systems*, 2022.

[122] 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*, 2022.

[123] Raymond Li, Loubna Ben allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia LI, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Joel Lamy-Poirier, Joao Monteiro, Nicolas Gontier, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Ben Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason T Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Urvashi Bhattacharyya, Wenhao Yu, Sasha Luccioni, Paulo Villegas, Fedor Zhdanov, Tony Lee, Nadav Timor, Jennifer Ding, Claire S Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro Von Werra, and Harm de Vries. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023.

[124] Shiyang Li, Jianshu Chen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. Explanations from large language models make small reasoners better. In *Workshop on Sustainable AI*, 2024.

[125] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

[126] 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*, 2022.

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

[128] Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. Explainable multi-hop verbal reasoning through internal monologue. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

[129] Baijiong Lin, Weisen Jiang, Feiyang Ye, Yu Zhang, Pengguang Chen, Ying-Cong Chen, Shu Liu, and James Kwok. Dual-balancing for multi-task learning. Preprint arXiv:2308.12029, 2023.

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

[131] Zhaojiang Lin, Andrea Madotto, and Pascale Fung. Exploring versatile generative language model via parameter-efficient transfer learning. In *Empirical Methods in Natural Language Processing*, 2020.

[132] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Annual Meeting of the Association for Computational Linguistics*, 2017.

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

[134] 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*, 2022.

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

[136] 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*, 2022.

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

[138] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 2023.

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

[140] 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*, 2022.

[141] 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.

[142] Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *ACL Workshop*, 2002.

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

[144] Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations*, 2022.

[145] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. WizardMath: Empowering mathematical reasoning for large language models via reinforced Evol-Instruct. Preprint arXiv:2308.09583, 2023.

[146] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering code large language models with evol-instruct. In *International Conference on Learning Representations*, 2024.

[147] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative refinement with self-feedback. In *Neural Information Processing Systems*, 2023.

[148] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

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

[150] 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.

[151] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *AAAI Conference on Artificial Intelligence*, 2020.

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

[153] MosaicML. Introducing MPT-7B: A new standard for open-source, commercially usable llms. Techical report, 2023.

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

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

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

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

[158] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. CodeGen: An open large language model for code with multi-turn program synthesis. In *International Conference on Learning Representations*, 2022.

[159] 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.

[160] 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.

[161] OpenAI. GPT-3.5. Technical Report, 2022.

[162] OpenAI. Introducing ChatGPT. Technical Report, 2022.

[163] OpenAI. GPT-4. Technical Report, 2023.

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

[165] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Neural Information Processing Systems*, 2022.

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

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

[168] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[169] 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.

[170] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

[171] Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and Boi Faltings. REFINER: Reasoning feedback on intermediate representations. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2024.

[172] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. Preprint arXiv:2306.01116, 2023.

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

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

[175] 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.

[176] Philip Pettit and Robert Sugden. The backward induction paradox. *The Journal of Philosophy*, 1989.

[177] Silviu Pitis, Michael R Zhang, Andrew Wang, and Jimmy Ba. Boosted prompt ensembles for large language models. Preprint arXiv:2304.05970, 2023.

[178] Gabriel Poesia and Noah D Goodman. Peano: learning formal mathematical reasoning. *Philosophical Transactions of the Royal Society A*, 2023.

[179] Boris Teodorovich Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1963.

[180] Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. Grips: Gradient-free, edit-based instruction search for prompting large language models. Preprint arXiv:2203.07281, 2022.

[181] 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.

[182] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical Report, 2019.

[183] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. Openai blog, 2019.

[184] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[185] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 2020.

[186] 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.

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

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

[189] 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.

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

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

[192] 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.

[193] Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Ali Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. Few-shot autoregressive density estimation: Towards learning to learn distributions. In *International Conference on Learning Representations*, 2018.

[194] Tim Rocktäschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. In *Workshop on Automated Knowledge Base Construction*, 2016.

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

[196] Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Conference on Empirical Methods in Natural Language Processing*, 2015.

[197] 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.

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

[199] 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*, 2015.

[200] Staurt J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[201] 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.

[202] 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.

[203] 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.

[204] 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.

[205] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *International Conference on Computational Learning Theory*, 2001.

[206] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. Preprint arXiv:1707.06347, 2017.

[207] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. Preprint arXiv:1709.03410, 2017.

[208] Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. In *International Conference on Machine Learning*, 2023.

[209] Peng Shen, Xugang Lu, Sheng Li, and Hisashi Kawai. Feature representation of short utterances based on knowledge distillation for spoken language identification. In *Conference of the International Speech Communication Association*, 2018.

[210] 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.

[211] 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.

[212] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Neural Information Processing Systems*, 2023.

[213] Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics*, 2023.

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

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

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

[217] 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.

[218] Tianxiang Sun, Zhengfu He, Hong Qian, Yunhua Zhou, Xuanjing Huang, and Xipeng Qiu. BBTv2: Towards a gradient-free future with large language models. In *Empirical Methods in Natural Language Processing*, 2022.

[219] 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.

[220] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[221] 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.

[222] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. Techical report, 2023.

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

[224] 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.

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

[226] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume

Lample. LLaMA: Open and efficient foundation language models. Preprint arXiv:2302.13971, 2023.

[227] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. LLaMA 2: Open foundation and fine-tuned chat models. Preprint arXiv:2307.09288, 2023.

[228] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2020.

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

[230] 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.

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

[232] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

[233] Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 billion parameter autoregressive language model. Technical report, 2021.

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

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

[236] Mingzhe Wang and Jia Deng. Learning to prove theorems by learning to generate theorems. In *Neural Information Processing Systems*, 2020.

[237] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.

[238] 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*, 2020.

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

[240] 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*, 2022.

[241] 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*, 2022.

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

[243] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *Neural Information Processing Systems*, 2022.

[244] Yanbin Wei, Shuai Fu, Weisen Jiang, James Kwok, and Yu Zhang. Rendering graphs for graph reasoning in multimodal large language models. Preprint arXiv:2402.02130, 2024.

[245] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to Self-Correct. In *International Conference on Learning Representations*, 2023.

[246] Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. Large language models are reasoners with self-verification. In *Conference on Empirical Methods in Natural Language Processing*, 2023.

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

[248] 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. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

[249] Zhenyu Wu, YaoXiang Wang, Jiacheng Ye, Jiangtao Feng, Jingjing Xu, Yu Qiao, and Zhiyong Wu. OpenICL: An open-source framework for in-context learning. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[250] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond Pascal: A benchmark for 3D object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision*, 2014.

[251] 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.

[252] Benfeng Xu, Quan Wang, Zhendong Mao, Yajuan Lyu, Qiaoqiao She, and Yongdong Zhang. KNN Prompting: Beyond-context learning with calibration-free

nearest neighbor inference. In *The Eleventh International Conference on Learning Representations*, 2023.

[253] Xiaohan Xu, Chongyang Tao, Tao Shen, Can Xu, Hongbo Xu, Guodong Long, and Jian-guang Lou. Re-Reading improves reasoning in language models. Preprint arXiv:2309.06275, 2023.

[254] Tianci Xue, Ziqi Wang, Zhenhailong Wang, Chi Han, Pengfei Yu, and Heng Ji. RCoT: Detecting and rectifying factual inconsistency in reasoning by reversing chain-of-thought. Preprint arXiv:2305.11499, 2023.

[255] 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.

[256] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, JunTao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. Baichuan 2: Open large-scale language models. Preprint arXiv:2309.10305, 2023.

[257] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer Learning*. Cambridge University Press, 2020.

[258] 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.

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

[260] 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.

[261] 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.

[262] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of Thoughts: Deliberate problem solving with large language models. In *Neural Information Processing Systems*, 2023.

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

[264] 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.

[265] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. In *International Conference on Learning Representations*, 2020.

[266] Fei Yu, Hongbo Zhang, and Benyou Wang. Nature language reasoning, a survey. Preprint arXiv:2303.14725, 2023.

[267] Longhui Yu*, Weisen Jiang*, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, 2024.

[268] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. Preprint arXiv:2308.01825, 2023.

[269] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *International Conference on Learning Representations*, 2023.

[270] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *IEEE/CVF International Conference on Computer Vision*, 2019.

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

[272] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. GLM-130B: An open bilingual pre-trained model. Preprint arXiv:2210.02414, 2022.

[273] 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.

[274] Kechi Zhang, Zhuo Li, Jia Li, Ge Li, and Zhi Jin. Self-Edit: Fault-aware code editor for code generation. In *Annual Meeting of the Association for Computational Linguistics*, 2023.

[275] 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*, 2022.

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

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

[278] Yulong Zhang, Shuhao Chen, Weisen Jiang, Yu Zhang, Jiangang Lu, and James Kwok. Domain-guided conditional diffusion model for unsupervised domain adaptation. Preprint arXiv:2309.14360, 2023.

[279] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *International Conference on Learning Representations*, 2023.

[280] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. Multimodal chain-of-thought reasoning in language models. In *International Conference on Machine Learning*, 2023.

[281] Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting improves reasoning in large language models. Preprint arXiv: 2304.09797, 2023.

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

[283] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

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

[285] 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.

[286] Xiao Zhou, Weizhong Zhang, Hang Xu, and Tong Zhang. Effective sparsification of neural networks with global sparsity constraint. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

[287] 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.

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