NEW INSIGHTS INTO META-LEARNING: FROM THEORY TO ALGORITHMS

JIAXIN CHEN

PhD

The Hong Kong Polytechnic University

2021

The Hong Kong Polytechnic University
Department of Computing

New Insights into Meta-Learning: From Theory to Algorithms

Jiaxin Chen

A thesis submitted in partial fulfilment of the requirements

for the degree of Doctor of Philosophy

January 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Jiaxin Chen (Name of student)

Abstract

Deep learning has surpassed human-level performance in various domains where the training data is abundant. Human intelligence, however, has the ability to adapt to a new environment with little experiences or recognize new categories after seeing just a few training samples. To endow machines such humanlike *few-shot learning* skills, meta-learning provides promising solutions and has generated a surge of interest recently. A meta-learning algorithm (*meta-algorithm*) trains over a large number of i.i.d. tasks sampled from a task distribution and learns an algorithm (*inner-task algorithm*) that can quickly adapt to a future task with few training data. From a generalization view, traditional supervised learning studies the generalization of a learned hypothesis (predictor) to novel data samples in a given task, whereas meta-learning explores the generalization of a learned algorithm to novel tasks. In this thesis, we provide new insights into the generalization of modern meta-learning based on theoretical analysis and propose new algorithms to improve its generalization ability.

We provide theoretical investigations into *Support/Query (S/Q) Episodic Training Strategy* which is widely believed to improve the generalization and applied in modern meta-learning algorithms. We analyze the generalization error bound of generic meta-learning algorithms trained with such strategy via a stability analysis. We show that the S/Q episodic training strategy naturally leads to a counterintuitive generalization bound of $O(1/\sqrt{n})$, which only depends on the number of task n but independent of the inner-task sample size m. Under the common assumption $m << n$ for few-shot learning, the bound of $O(1/\sqrt{n})$ implies strong generalization guarantee for modern meta-learning algorithms in the few-shot regime.

We further point out that there still exist limitations of the existing modern meta-training strategy, i.e., the optimization procedure of model parameters following the strategy of differentiating through the inner-task optimization path. To satisfy this requirement, the inner-task

algorithms should be solved analytically and this significantly limits the capacity and performance of the learned inner-task algorithms. Hence, we propose an adaptation-agnostic meta-training strategy that removes such dependency and can be used to train inner-task algorithms with or without analytical expressions. Such general meta-training strategy naturally leads to an ensemble framework that can efficiently combine various types of algorithms to achieve better generalization.

Motivated by a closer look at metric-based meta-algorithms which shows high generalization ability over the few-shot classification problems, we propose a generic variational metric scaling framework which is compatible with metric-based meta-algorithms and achieves consistent improvements over the standard few-shot classification benchmarks. Finally, as majority of existing meta-algorithms focus on within-domain generalization, we further consider cross-domain generalization over a realistic yet more challenging few-shot classification problem, where a large discrepancy exists between the task distributions of training domains and a test domain. A gradient-based hierarchical meta-learning framework (M^2L) is proposed to solve such problem. Finally, we discuss open questions and future directions in meta-learning.

List of Publications

1. **Jiaxin Chen**, Xiao-Ming Wu, Yanke Li, Qimai Li, Li-Ming Zhan, Fu-lai Chung. A Closer Look at the Training Strategy for Modern Meta-Learning. **NeurIPS 2020**.

2. **Jiaxin Chen**, Li-Ming Zhan, Xiao-Ming Wu, Fu-lai Chung. Variational Metric Scaling for Metric-Based Meta-Learning. **AAAI 2020**.

3. **Jiaxin Chen***, Li-Ming Zhan*, Xiao-Ming Wu, Fu-lai Chung. Adaption-Agnostic Meta-Training. **ICML 2021 @AutoML Workshop.**

4. Li-Ming Zhan, Bo Liu, Lu Fan, **Jiaxin Chen**, Xiao-Ming Wu. Medical Visual Question Answering via Conditional Reasoning. **ACM MM 2020**.

5. **Jiaxin Chen***, Junjie Ye*, Xiao-Ming Wu, Guangyuan Shi, Li-Ming Zhan, Yifan Xue, Fu-lai Chung. Meta^2 Learning. *In Submission.*

Acknowledgements

First and foremost, I would like to appreciate Prof. Fu-lai Chung, my supervisor for his continuous support and guidance through my Ph.D.. He is very nice and open, providing me a lot of freedom to work on various research topics and opportunities to cooperate with excellent researchers. I am grateful to my collaborator Prof. Xiao-ming Wu for her continuous guidance and encouragement. She is rigorous and insightful who leads me to conduct impactful research.

I'm very happy and lucky to collaborate with a lot of excellent and talented researchers. For the works in this thesis, I worked with Junjie Ye, Liming Zhan, Qimai Li, Guangyuan Shi, Yanke Li and Sitong Mao. Without your insightful discussions, meaningful comments and collaboration in the experiments, this thesis will not be finished.

My campus life in PolyU is full of excitement and happiness. My time in PolyU will not be the same without my friends, Yuping Ke, Xiao Shen, Wei Lu, Yumeng Guo, Quanyu Dai, Han Liu, Zuoxia Yu, Kang Li, Jie You, Jingwei Zhao, Pengfei Chen and Yifan Xue.

Finally, this thesis is dedicated to my family and my friends. Thank you for all of your sincere love and support.

Table of contents

List of Figures

List of Tables

1

Introduction

1.1 Background

Deep learning has surpassed human-level performance in various domains where the training data is abundant. But human intelligence has remarkable ability to adapt to a new environment with little experience or recognize new categories after seeing few samples. To endow machines with such kind of capability, a promising paradigm is meta-learning [5], which learns general patterns from a large number of tasks for fast adaptation to unseen tasks. A *meta-algorithm* (so-called *meta-learner*) trains over a large amount of training tasks i.i.d. sampled from a task distribution and learns an *inner-task algorithm* (so-called *base-learner*) that can quickly adapt to a novel task with few training samples. To avoid confusion, the training set and the test set in each task are called *support* set and *query* set, respectively. The commonly used modern meta-training procedure is an interleaved process which includes *inner-task adaptation* and *meta-update*. For any training task, during the inner-task adaptation, the inner-task algorithm is adapted to its support set and outputs a task-specific predictor. And the error of the predictor over the query set is computed to measure the performance of the inner-task algorithm. During meta-update, this error over the query set is minimized to update the model parameters. Majority of meta-algorithms follow such training strategy [30, 34, 100] and they mainly differ in the choice of the inner-task algorithm. For instance, the gradient-based meta-algorithms apply a learned initialization with stochastic gradient decent as the inner-task algorithm; the metric-based meta-algorithms apply a comparison algorithm with a similarity

metric; and the model-based meta-algorithms make use of black-box (e.g. a neural network) to produce the parameters of a pre-defined task-specific predictor.

1.2 Related Work

Gradient-based meta-learning. Gradient-based meta-learning learns a model with prior knowledge which can adapt to a new task with a few gradient update steps [30, 32, 44, 41, 43, 92, 93, 3]. The pioneering work MAML [30] achieves this goal by learning a common initialization of deep neural networks. For each training task, the learned initialization is adapted to the support set with a few gradient steps and the initialization is updated by optimizing the adapted model's performance over the query set. MAML is general and model-agnostic which is compatible with various tasks such as regression, classification and reinforcement learning.

MAML [30] suffers from the expensive computational cost caused by the second-order gradient derivation. Many efforts have been devoted to address this issue with first-order approximation [81, 80], implicit gradients [89] or partially updating parameters in the inner loop [88]. Another major problem of gradient-based meta-learning is overfitting because the high-dimensional parameters of the entire network are updated using few support samples in each task. References [75] and [95, 64, 119] targeted to tackle this problem by latent representation learning, context adaptation and so on.

Metric-based meta-learning. Few-shot classification [66] aims to assign unseen samples (*query*) to the belonging categories with very few labeled samples (*support*) in each category. A promising paradigm for few-shot learning is metric-based meta-learning algorithms [37, 58, 100, 108]. Typically, they learn a general mapping, which projects queries and supports into an embedding space. These models are typically trained in an episodic manner [108] by minimizing the distances between a query and same-labeled supports in the embedding space. Given a new task in testing phase, a simple nearest neighbour classifier can be used to assign a query to its nearest class in the embedding space. Reference [58] proposed the first metric-based meta-algorithm for few-shot learning, in which a siamese network [17] is trained with the triplet loss to maximize the similarity between within-class samples and minimize that of inter-class samples in the embedding space. Matching networks [108] proposed the

episodic training strategy and used the cross-entropy loss where the logits are the distances between a query and supports. Prototypical networks [100] improved Matching networks by computing the distances between a query and the prototype (mean of supports) of each class. Many metric-based meta-algorithms [84, 33, 102, 67, 102] extended prototypical networks in different ways.

Model-based meta-learning. Model-based meta-learning is also called black box-based meta-learning which directly produces parameters for the inner-task algorithms [39, 75, 7, 78]. A typical model-based meta-algorithm called VERSA [42], interprets meta-learning from a probabilistic perspective and learns an amortization network to generate the parameters of the distributions of the task-specific parameters of the inner-task algorithms. Memory-augmented neural networks [96] directly memorize the old data and predict the labels for new data in a black box. LSTM-based meta-learning methods [3] implement the inner-task training by recurrently updating in the LSTM cell. Meta-networks [78] learn the task-specific parameters and meta-parameters by optimizing the fast weights and the slow weights.

1.3 Thesis Overview

The contributions of this thesis are summarized as follows:

- In Chapter 2, we theoretically study the generalization ability of the modern meta-algorithms with Episodic Support/Query training strategy via a stability analysis. We show that meta-algorithms with such strategy has a "counter-intuitive" generalization bound of $O(1/\sqrt{n})$ (n is the number of training task) which shows that the generalization gap converges to 0 given sufficient training tasks, despite of the limited training samples in each task. Such theoretical result is totally different from the traditional generalization bounds of $O(n, m)$ (m is the training sample size per task) which cannot converge under the common assumption of few-shot learning $m << n$ (e.g. $m = 5, 10$), providing a strong guarantee for modern meta-learning in few-shot learning scenario.

- In Chapter 3, we take a closer look at the limitations of the modern meta-training strategy and provide a unified perspective on existing meta-algorithms following this

strategy. The existing meta-algorithms' update rule relies on the analytical expressions of the task-specific parameters w.r.t. the model parameters which requires the inner-task algorithms having closed-form solutions. Such requirement limits the choice of the inner-task algorithm and correspondingly limits the model expressiveness of the models. To remove such limitations, we propose a general adaptation-agnostic meta-training strategy and such strategy naturally leads to a flexible and powerful ensemble framework as the inner-task algorithm.

- In Chapter 4, we focus on metric-based meta-learning which shows effectiveness, efficiency and high generalization ability in few-shot classification problems. As the metric scaling plays a crucial role in the performance of metric-based meta-learning algorithms but there still lacks a principled method for learning the metric scaling parameter automatically. In this chapter, we recast metric-based meta-learning from a Bayesian perspective and develop a variational metric scaling framework for learning a proper metric scaling parameter. To fit the learned embedding well and consider task-specific information, we further propose dimensional stochastic/amortized variational metric scaling methods which show consistent improvements.

- Majority of existing meta-learning algorithms focus on within-domain generalization. In Chapter 5, we consider a more challenging and realistic scenario, i.e., cross-domain fine-grained few-shot learning, under which a discrepancy exists between the task distributions of meta-training and meta-test. We provide new insights into such cross-domain generalization problem and proposed a domain-oriented meta-learning (DOM) algorithm to solve cross-domain fine-grained few-shot classification tasks.

2

Generalization of Modern Meta-Learning via a Stability Analysis

2.1 Overview

Early meta-algorithms directly minimize the averaged training error of a set of training tasks. To improve the generalization of meta-algorithms, the pioneering work [108] proposes a novel training strategy – support/query episodic training strategy. In particular, episodic training treats each task as a training instance and updates the inner-task algorithm by episode (task by task). Support/query (S/Q) training mimics the test process in each task, i.e., a training set (*support*) for *inner-task training* and a test set (*query*) for measuring the inner-task algorithm's performance. *Meta-training* proceeds by minimizing the error computed over the query set. This training strategy has been widely used to train modern meta-algorithms such as MAML [30] and ProtoNet [100].

Although it is widely believed that the S/Q training strategy can improve the generalization of meta-algorithms due to the match of training condition and test condition, there is barely any theoretical analysis of how it impacts generalization. Our key observation is that the generalization bound of meta-algorithms is closely related to the training strategy. The S/Q training strategy leads to a bound different from the existing meta-learning bounds which do not involve any specific meta-training strategy. In this chapter, we study the generalization error bounds of generic meta-algorithms trained with the S/Q scheme by employing tools from

stability analysis [73, 10, 48].

Based on stability analysis, we derive a generalization bound of $O(1/\sqrt{n})$ for meta-algorithms trained with S/Q strategy, which is independent of the sample size m of each task. The result seems counterintuitive at the first glance. However, it is natural if we carefully check the difference between S/Q training and traditional meta-training strategies. We explain the **key intuition of the bound** $O(1/\sqrt{n})$ as follows. For the traditional meta-training strategy, the bound of the generalization gap between the traditional empirical multi-task error and the transfer error consists of two terms, namely, an inner-task gap $\epsilon(m)$ caused by observing limited inner-task training samples and an outer-task gap $\epsilon(n)$ caused by observing limited training tasks. For S/Q training, for any training task, the inner-task algorithm minimizes the inner-task *training* error of the support set and outputs a hypothesis. The S/Q training error, i.e., the error of this inner-task hypothesis computed over the query set (unseen during inner-task training), is exactly the inner-task *test* error of this inner-task hypothesis, and thereby is an unbiased estimate to the inner-task generalization error of the inner-task hypothesis. Intuitively, the inner-task gap depending on the inner-task sample size m vanishes because S/Q training directly minimizes the inner-task test error. Correspondingly, the bound of the generalization gap between the S/Q training error and the transfer error equals to the outer-task gap that only depends on the number of task n.

To further explore the influence of training strategies on the generalization of meta-algorithms, we want to compare S/Q training with existing meta-training strategies. However, the traditional meta-training strategy cannot be used to train modern meta-algorithms such as MAML [30], Bilevel Programming [34] and ProtoNet [100] which require support samples for inner-task training and query samples for meta-training (see more discussions in Sec. 2.3). To compare with S/Q training, we introduce a new strategy for training modern meta-algorithms – leave-one-out (LOO) training, which minimizes the *leave-one-out* errors of *training* tasks. The key reason of studying LOO training is that it is similar to the traditional training strategy which computes the empirical error over the support set instead of the query set while can still be used to train modern meta-algorithms. Interestingly, although LOO training error is an "almost" unbiased estimate to the generalization error of the inner-task hypotheses of training

tasks [27], the generalization bound of meta-algorithms with LOO training still depends on both the inner-task sample size m and the task number n. See Table 2.1 for a summary of these three training strategies.

Table 2.1: Comparisons of three meta-training strategies. Data set: the data set used for computing the empirical error for meta-training. Estimate: in each training task, the empirical error for meta-training is an unbiased/biased estimate to the generalization error of the inner-task hypothesis. Compatibility: the training strategy's compatibility with modern meta-algorithms. Bound: the generalization bound.

Strategy	Data set	Estimate	Compatibility	Bound
Traditional	Support set	biased	$\times$	$\epsilon(n,m)$
Leave-one-out	Support set	"almost" unbiased	✓	$\epsilon(n,m)$
Support/Query	Query set	unbiased	✓	$\epsilon(n)$

From a generalization perspective, our results clearly explain the success of the S/Q training strategy. The sample-size free bound provides a firm theoretical support for the generalization of modern meta-learning algorithms in the few-shot learning setting. Furthermore, our theoretical results are empirically verified by experiments on standard few-shot classification and regression tasks implemented with popular meta-algorithms [30, 100, 34].

2.2 Preliminaries

Table 2.2: Notations.

	Single-task learning	Meta-learning	
Domain	$\mathcal{Z}$	$\mathcal{Z}^m$	
Unknown distribution	$\mathcal{D}$	τ	
Training instance	$S^{tr}=\{z_j=(x_j,y_j)\}_{j=1}^m$, $S^{tr}\sim\mathcal{D}^m$	LOO	$\mathbf{S}=\{S_i=S_i^{tr}\}_{i=1}^n, S_i^{tr}\sim\mathcal{D}_i^m, \mathcal{D}_i\sim\tau$
		S/Q	$\mathbf{S}=\{S_i=S_i^{tr}\cup S_i^{ts}\}_{i=1}^n, S_i^{tr}\sim\mathcal{D}_i^m, S_i^{ts}\sim\mathcal{D}_i^q, \mathcal{D}_i\sim\tau$
Test instance	$z\sim\mathcal{D}$	$S^{tr}\sim\mathcal{D}^m, z\sim\mathcal{D}, \mathcal{D}\sim\tau$	
Leave-one-out set	$S^{tr\setminus j}=(\cdots,z_{j-1},z_{j+1},\cdots)$	$\mathbf{S}^{\setminus i}=(\cdots,S_{i-1},S_{i+1},\cdots)$	
Resubstitution set	$S^{tr(j)}=(\cdots,z_j',\cdots)$	$\mathbf{S}^{(i)}=(\cdots,S_i',\cdots)$	
Target	$A(S^{tr}):\mathcal{X}\to\mathcal{Y}$	$\mathbf{A}(\mathbf{S}):\mathcal{Z}^m\to\mathcal{H}$	
Training error	$\hat{L}(A(S^{tr}),S^{tr})$ $=\frac{1}{m}\sum_{j=1}^m l(A(S^{tr}),z_j)$	LOO	$\hat{\mathcal{R}}_{loo}(\mathbf{A}(\mathbf{S}),\mathbf{S})=\frac{1}{n}\sum_{i=1}^n \hat{L}_{loo}(\mathbf{A}(\mathbf{S})(S_i^{tr}),S_i^{tr})$
		S/Q	$\hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}),\mathbf{S})=\frac{1}{n}\sum_{i=1}^n \hat{L}(\mathbf{A}(\mathbf{S})(S_i^{tr}),S_i^{ts})$
Quantity of interest	$L(A(S^{tr}),\mathcal{D})$ $=\mathbb{E}_{z\sim\mathcal{D}}l(A(S^{tr}),z)$	$\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)=\mathbb{E}_{\mathcal{D}\sim\tau}\mathbb{E}_{S^{tr}\sim\mathcal{D}^m}\mathbb{E}_{z\sim\mathcal{D}}l(\mathbf{A}(\mathbf{S})(S^{tr}),z)$	

Single-Task Learning. Let $\mathcal{Z}=\mathcal{X}\times\mathcal{Y}$ be a domain, where $\mathcal{X}$ denotes an input space and $\mathcal{Y}$ denotes an output space. Furthermore, $\mathcal{H}=\{h_w:w\in\mathcal{W}\}$ is the hypothesis set where

the hypothesis $h_w \in \mathcal{H}$ is parametrized by parameters w in the parameter space $\mathcal{W}$. Given a non-negative loss function $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$, the loss of a hypothesis h_w over a sample z is denoted by $l(h_w, z)$ or $l(w, z)$. In single-task learning, an algorithm A receives a training set $S^{tr} = \{z_j = (x_j, y_j)\}_{j=1}^m$ drawn i.i.d. from an unknown distribution $\mathcal{D}$ on $\mathcal{Z}$. Then, the algorithm selects a hypothesis denoted by $A(S^{tr})$ from $\mathcal{H}$ by minimizing the empirical error $\hat{L}(A(S^{tr}), S^{tr}) \stackrel{def}{=} \frac{1}{m}\sum_{j=1}^m l(A(S^{tr}), z_j)$. The performance of the learned $A(S^{tr})$ is measured by the generalization error $L(A(S^{tr}), \mathcal{D}) \stackrel{def}{=} \mathbb{E}_{z\sim\mathcal{D}} l(A(S^{tr}), z)$, which is the quantity of interest in statistical learning. To study the convergence of the empirical error to the generalization error, reference [10] upper bounded the gap between the generalization error and the empirical error by considering the stability of the algorithm A.

Stability theory analyses the sensitivity of an algorithm A in response to some small modifications of the training set, for example, the leave-one-out training set $S^{tr\setminus j} = (z_1, \cdots, z_{j-1}, z_{j+1}, \cdots, z_m)$ and the resubstitution training set $S^{tr(j)} = (z_1, \cdots, z_{j-1}, z'_j, z_{j+1}, \cdots, z_m)$, where $z'_j \sim \mathcal{D}$. Reference [10] proposed various notions of stability to derive the generalization bounds of learning algorithms. In this chapter, we mainly consider the uniform stability on the leave-one-out training set.

Definition 1 (Uniform stability [10]). *An algorithm A has uniform stability $\tilde{\beta}$ w.r.t. the loss function l, if the following holds $\forall j \in \{1, \ldots, m\}$:*

$$\forall S^{tr} \sim \mathcal{D}^m, \forall z \sim \mathcal{D}, |l(A(S^{tr}), z) - l(A(S^{tr\setminus j}), z)| \leq \tilde{\beta}.$$

As shown in [10], the uniform stability can be used to derive a generalization bound $O(m, \tilde{\beta})$. The bound converges to 0 as $m \to \infty$, if the algorithm is stable ($\tilde{\beta} \to 0$ as $m \to \infty$) and $\tilde{\beta} < O(1/\sqrt{m})$.

Meta-Learning. Unlike single-task learning where the training instances are data samples and the output is a hypothesis, the training instances of meta-learning are training tasks and the output is an algorithm. Assume that training tasks $\{\mathcal{D}_i\}_{i=1}^n$ are drawn i.i.d. from an unknown task distribution τ. A meta-learning algorithm (*meta-algorithm*) $\mathbf{A}$ observes a *meta-sample* $\mathbf{S} = \{S_i = S_i^{tr}\}_{i=1}^n$, where $S_i^{tr} \stackrel{i.i.d.}{\sim} \mathcal{D}_i^m$ of size m is the training set of i^{th} training task $\mathcal{D}_i$ and outputs an algorithm (*inner-task algorithm*) $\mathbf{A}(\mathbf{S}) : \mathcal{Z}^m \to \mathcal{H}$. To measure the

performance of the selected inner-task algorithm, the quantity of interest in meta-learning is the expectation of the generalization error with respect to the task distribution τ, which is termed as *transfer error* defined by [5] as follows,

$$\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) \stackrel{def}{=} \mathbb{E}_{\mathcal{D}\sim\tau}\mathbb{E}_{S^{tr}\sim\mathcal{D}^m}\mathbb{E}_{z\sim\mathcal{D}} l(\mathbf{A}(\mathbf{S})(S^{tr}), z).$$

Given a new task $\mathcal{D} \sim \tau$ with the training set $S^{tr} \sim \mathcal{D}^m$, the inner-task algorithm $\mathbf{A}(\mathbf{S})$ learns a hypothesis $\mathbf{A}(\mathbf{S})(S^{tr}) : \mathcal{X} \to \mathcal{Y}$. Given a new sample $x \sim \mathcal{D}$, the target assigned to x is $\mathbf{A}(\mathbf{S})(S^{tr})(x)$.

Meta-training proceeds by minimizing the average of the empirical error of the training tasks called *empirical multi-task error* which is defined by [73] as

$$\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S}) \stackrel{def}{=} \frac{1}{n}\sum_{i=1}^{n} \hat{L}(\mathbf{A}(\mathbf{S})(S_i^{tr}), S_i^{tr}). \tag{2.1}$$

In [73], the definition of uniform stability (Definition 1) was extended to meta-algorithms on the leave-one-out meta-sample $\mathbf{S}^{\setminus i} = (S_1, \cdots, S_{i-1}, S_{i+1}, \cdots, S_n)$ as follows.

Definition 2 (Uniform stability of meta-algorithms [73])**.** *A meta-algorithm* $\mathbf{A}$ *has uniform stability* β *w.r.t. the loss function* l *if the following holds for any meta-sample* $\mathbf{S}$ *and* $\forall i \in \{1, \ldots, n\}$*:*

$$\forall \mathcal{D} \sim \tau, \forall S^{tr} \sim \mathcal{D}^m, |\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{tr}) - \hat{L}(\mathbf{A}(\mathbf{S}^{\setminus i})(S^{tr}), S^{tr})| \leq \beta.$$

As shown in [73], the generalization bound of a meta-algorithm $\mathbf{A}$ can be obtained with the uniform stability β of the meta-algorithm $\mathbf{A}$ (Definition 2) and the uniform stability $\tilde{\beta}$ of the inner-task algorithm $\mathbf{A}(\mathbf{S})$ (Definition 1).

Theorem 1 (Generalization bound of meta-algorithms [73])**.** *For any task distribution* τ *and meta-sample* $\mathbf{S}$ *with* n *tasks, if a meta-algorithm* $\mathbf{A}$ *has uniform stability* β *and the inner-task algorithm* $\mathbf{A}(\mathbf{S})$ *has uniform stability* $\tilde{\beta}$ *w.r.t. a loss function* l *bounded by* M*, then the following statement holds with probability of at least* $1 - \delta$ *for any* $\delta \in (0, 1)$*:*

$$\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) \leq \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S}) + \epsilon(n, \beta, \tilde{\beta}), \tag{2.2}$$

where $\epsilon(n, \beta, \tilde{\beta}) = 2\beta + (4n\beta + M)\sqrt{\frac{\ln(1/\delta)}{2n}} + 2\tilde{\beta}$.

This upper bound implies that the empirical multi-task error converges to the transfer error as $n \to \infty$ and $m \to \infty$, only if $\beta < O(\frac{1}{\sqrt{n}})$ and the meta-algorithm is uniformly stable.

2.3 Generalization Bound of Meta-Algorithms with S/Q Training

The aforementioned traditional multi-task empirical error (2.1) studied in [73] can not be applicable to modern meta-algorithms such as metric-based meta-algorithms [100, 14, 112] and gradient-based meta-algorithms [30, 34]. For a metric-based meta-algorithm $\mathbf{A}$, it learns a parameterized mapping metric, and the inner-task algorithm $\mathbf{A}(\mathbf{S})$ can be regarded as a nearest neighbor algorithm with the learned metric. The traditional empirical error used in [73] is not applicable to the nearest neighbor algorithm because it will trivially equal to 0. Moreover, for a gradient-based meta-algorithm $\mathbf{A}$, it aims to achieve fast adaptation by learning an initialization of a neural network, and the inner-task algorithm can be considered as a gradient descent algorithm with a learned initialization w_t. When the inner-task training converges (w_t converges to $\bar{w}_t$), the gradient of the training error over the support set w.r.t. $\bar{w}_t$ equals to 0. If using the traditional empirical error which still uses the support set to compute the empirical error for meta-training (update w_t), meta-training cannot proceed due to gradient vanishing ($\nabla_{w_t}\hat{L}(\cdot, S_i^{tr}) = \nabla_{\bar{w}_t}\hat{L}(\cdot, S_i^{tr}) \times \nabla_{w_t}\bar{w}_t{=}0$).

S/Q training strategy. Instead of the inapplicable traditional training strategy, modern meta-algorithms follow the support/query training strategy proposed by [108] which uses the support/query training error defined as follows,

$$\hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}), \mathbf{S}) \overset{def}{=} \frac{1}{n}\sum_{i=1}^{n}\hat{L}(\mathbf{A}(\mathbf{S})(S_i^{tr}), S_i^{ts}) \overset{def}{=} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{q}\sum_{z_{ij}\in S_i^{ts}} l(\mathbf{A}(\mathbf{S})(S_i^{tr}), z_{ij}),$$

where the meta-sample is $\mathbf{S} = \{S_i = S_i^{tr} \cup S_i^{ts}\}_{i=1}^n$, where $S_i^{tr} \overset{i.i.d.}{\sim} \mathcal{D}_i^m$ of size m is the training set of i^{th} training task $\mathcal{D}_i$ and $S_i^{ts} \overset{i.i.d.}{\sim} \mathcal{D}_i^q$ of size q is the test set of $\mathcal{D}_i$. To avoid confusion, the training set and the test set are called *support set* and *query set* respectively.

2.3.1 Generalization Bound via Stability

We give the following definition of uniform stability of meta-algorithms w.r.t. S/Q training error.

Definition 3 (Uniform stability of meta-algorithms with S/Q training)**.** *A meta-algorithm* $\mathbf{A}$

has uniform stability β w.r.t. the loss function l if the following holds for any meta-sample $\mathbf{S}$ *and* $\forall i \in \{1, \ldots, n\}$*:*

$$\forall \mathcal{D} \sim \tau, \forall S^{tr} \sim \mathcal{D}^m, \forall S^{ts} \sim \mathcal{D}^q, |\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{ts}) - \hat{L}(\mathbf{A}(\mathbf{S}^{\setminus i})(S^{tr}), S^{ts})| \leq \beta.$$

Based on the defined uniform stability, we can analyze the generalization bound of meta-algorithms with S/Q training error.

In Theorem 1, the generalization gap of the traditional empirical multi-task error (2.1) can be written as

$$\begin{aligned}\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) - \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S}) =& \underbrace{\mathbb{E}_{\mathcal{D}\sim\tau, S^{tr}\sim\mathcal{D}^m}[\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{tr}) - \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S})]}_{\text{Outer-task gap}} \\ &+ \mathbb{E}_{\mathcal{D}\sim\tau, S^{tr}\sim\mathcal{D}^m} \underbrace{[\mathbb{E}_{z\sim\mathcal{D}} l(\mathbf{A}(\mathbf{S})(S^{tr}), z) - \hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{tr})]}_{\text{Inner-task gap}}.\end{aligned}$$

The generalization gap consists of two parts, namely, the outer-task gap and the inner-task gap. As the averaged empirical error of n tasks is a biased estimator of the expected empirical error w.r.t. the expectation of the task distribution, the bias leads to a gap at the task distribution τ level (outer-task gap). Similarly, for any task, the averaged training error of m samples is a biased estimator of the generalization error and this bias leads to a gap at the data distribution $\mathcal{D}$ level (inner-task gap). The outer-task gap and the inner-task gap can be bounded by the stability of the meta-algorithm (dependent on n) and the stability of the inner-task algorithm (dependent on m), respectively.

In contrast to the traditional empirical error, S/Q training error is computed over unseen samples (query set). Therefore, the training error $\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{ts}) = \frac{1}{q}\sum_{z_j \in S^{ts}} l(\mathbf{A}(\mathbf{S})(S^{tr}), z_j)$ is an unbiased estimate to the generalization error $\mathbb{E}_{z\sim\mathcal{D}} l(\mathbf{A}(\mathbf{S})(S^{tr}), z)$. As such, the inner-task gap vanishes under the expectation w.r.t. the query set and the generalization gap can be written as follows,

$$\begin{aligned}\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) - \hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}), \mathbf{S}) =& \mathbb{E}_{\mathcal{D}\sim\tau, S^{tr}\sim\mathcal{D}^m, z\sim\mathcal{D}} l(\mathbf{A}(\mathbf{S})(S^{tr}), z) - \hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}), \mathbf{S}) \\ =& \underbrace{\mathbb{E}_{\mathcal{D}\sim\tau, S^{tr}\sim\mathcal{D}^m, S^{ts}\sim\mathcal{D}^q}[\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{ts}) - \hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}), \mathbf{S})]}_{\text{Outer-task gap}}.\end{aligned}$$

The generalization gap can be upper bounded by measuring the stability of the meta-algorithm

β only. Motivated the relationship between stability and generalization of single-task learning [10], we derive the generalization bound as follows.

Theorem 2 (Generalization bound of meta-algorithms with S/Q training)**.** *For any task distribution τ and meta-sample $\mathbf{S}$ with n tasks, if a meta-algorithm $\mathbf{A}$ has uniform stability β w.r.t. a loss function l bounded by M, then the following statement holds with probability of at least $1-\delta$ for any $\delta \in (0,1)$:*

$$\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) \leq \hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}), \mathbf{S}) + \epsilon(n, \beta), \tag{2.3}$$

where $\epsilon = 2\beta + (4n\beta + M)\sqrt{\frac{\ln(1/\delta)}{2n}}$.

By Theorem 2, the generalization bound depends on the number of training tasks n and the uniform stability parameter β. If $\beta < O(\frac{1}{\sqrt{n}})$, we have $\epsilon(n, \beta) \to 0$ as $n \to \infty$. Hence, given a sufficiently small β, the transfer error converges to S/Q training error as the number of training tasks grows. Notice that the bound does not depend on the stability of the inner-task algorithm $\tilde{\beta}$. See Appendix A.1 for a proof.

2.3.2 Stability of Meta-Algorithms with Episodic Training

We have shown that the generalization bounds of meta-algorithms with S/Q depend on the uniform stability β of meta-algorithms. In this subsection, we focus on deriving β. Since modern meta-algorithms follow the episodic training strategy which observes a random ordered set of training tasks sequentially, we define randomized uniform stability and derive the stability parameter for generic meta-algorithms with episodic training.

Different from traditional meta-algorithms training over a large dataset, [108] proposed to train over mini-batches named episodes where each episode is a training task. The training idea can be considered as a meta-level stochastic gradient method (SGM). In single-task learning, SGM is an optimization algorithm which samples each data point randomly from the training set to compute the gradient of the objective function $l(w; z_j)$ and updates the parameters sample by sample. The episodic training procedure can be viewed in the same way. We randomly select a set of samples from a large dataset as a training task to compute the gradient of the loss function $\hat{L}(w; S_i)$ and update the parameters task by task.

Inspired by randomized uniform stability of SGM in single-task learning [48], we prove the stability of meta-algorithms. In the following, we define the randomized uniform stability of meta-algorithms on the leave-one-out meta-sample (Definition 4).

Definition 4 (Randomized uniform stability of meta-algorithms with S/Q training)**.** *A randomized meta-algorithm* $\mathbf{A}$ *has randomized uniform stability* β *w.r.t. the loss function* $\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{ts})$*, if the following holds for any task distribution* τ *and any meta-sample* $\mathbf{S}$*,* $\forall i \in \{1, \ldots, n\}, \forall \mathcal{D} \sim \tau, \forall S^{tr} \sim \mathcal{D}^m, \forall S^{ts} \sim \mathcal{D}^q$*:*

$$\mathbb{E}_{\mathbf{A}}[|\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{ts}) - \hat{L}(\mathbf{A}(\mathbf{S}^{\setminus i})(S^{tr}), S^{ts})|] \leq \beta.$$

Note that the definition of randomized uniform stability of meta-algorithms w.r.t. LOO error is similar to Definition 4, but the loss function is $\hat{L}_{loo}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{tr})$.

In single-task learning, given a training set $S = \{z_j\}_{j=1}^m$, a SGM updates the model parameters w_t at step $t \in \{0, \ldots, T-1\}$ by the rule: $G(w_{t+1}) = G(w_t) - \zeta_t \nabla_{w_t} f(w_t; z_{j_t})$, where ζ_t is the step size of the step t and z_{j_t} is selected from S with j_t generated from $\{1, \ldots, m\}$ uniformly at random. We now extend this update rule to meta-training. Given a meta-sample $\mathbf{S} = \{S_i\}_{i=1}^n$, the update rule of a meta-algorithm is $G(w_{t+1}) = G(w_t) - \zeta_t \nabla_{w_t} f(w_t; S_{i_t})$, where S_{i_t} is selected from $\mathbf{S}$ with i_t uniformly distributed over $\{1, \ldots, n\}$.

The update rule of meta-algorithms share the same properties with SGM under the assumption that the loss function $l(w; z)$ is Lipschitz continuous and smooth w.r.t. z. Given these properties, it can be shown that $\beta \leq O(1/n)$. A detailed proof is provided in the Appendix A.2.

Theorem 3. *Assume that the loss function* $\hat{L}(\mathbf{A}(\mathbf{S})(S), S)$ *is smooth, Lipschitz continuous w.r.t.* S *and bounded by* $M > 0$*. Suppose that a meta-algorithm* $\mathbf{A}$ *is implemented by episode,* $\mathbf{A}$ *has randomized uniform stability* $\beta \leq O(1/n)$*.*

Note that Theorem 3 holds for meta-algorithms with episodic training, regardless of the exact form of the loss function $\hat{L}(\mathbf{A}(\mathbf{S})(\cdot), \cdot)$, as long as the loss function satisfies Lipschitz continuity and smoothness w.r.t. the input.

2.3.3 Main Result

Based on the generalization bounds in Theorem 2 and the stability parameter in Theorem 3, we can obtain the following Theorem 4. Note that the result under the expectation w.r.t. the randomized meta-algorithm $\mathbf{A}$ is a straightforward extension of Theorem 2.

Theorem 4 (Generalization bound of meta-algorithms with S/Q episodic training strategy)**.** *Suppose that a meta-algorithm* $\mathbf{A}$ *is implemented by episodic training strategy with a loss function bounded by* M *and satisfying the conditions in Theorem 3. For any task distribution* τ *and meta-sample* $\mathbf{S}$ *consisting of* n *tasks, the following statement holds w.r.t. S/Q error with probability of at least* $1-\delta$*,* $\forall \delta \in (0,1)$*:*

$$\mathbb{E}_{\mathbf{A}}[\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)] \leq \mathbb{E}_{\mathbf{A}}[\hat{\mathcal{R}}_{s/q}(\mathbf{A}(\mathbf{S}),\mathbf{S})] + O(\sqrt{\frac{\ln(1/\delta)}{n}} + \frac{1}{n}). \tag{2.4}$$

Theorem 4 is applicable to modern meta-algorithms with S/Q episodic training strategy, if only the conditions are satisfied in Theorem 3. The generalization bound is of order $O(1/\sqrt{n})$ and independent of the sample size m. This indicates that given enough tasks, the generalization gap converges to zero, in spite of limited data samples in each task. Under the common assumption $m << n$ in few-shot learning, this result provides a strong generalization guarantee for meta-learning.

2.4 Leave-One-Out Training Strategy

LOO training strategy. In last section, we have shown that the S/Q training strategy leads to an inner-task sample-size free bound, which is very different from the existing bounds with traditional empirical multi-task error. Since the traditional empirical multi-task error cannot be used to train modern meta-algorithms, as a surrogate to the traditional scheme, we propose a leave-one-out (LOO) meta-training strategy that is compatible with gradient-based and metric-based meta-learning algorithms. Specifically, the LOO training error is defined as

$$\hat{\mathcal{R}}_{loo}(\mathbf{A}(\mathbf{S}),\mathbf{S}) \stackrel{def}{=} \frac{1}{n}\sum_{i=1}^{n} \hat{L}_{loo}(\mathbf{A}(\mathbf{S})(S_i^{tr}), S_i^{tr}) \stackrel{def}{=} \frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{z_{i,j}\in S_i^{tr}} l(\mathbf{A}(\mathbf{S})(S_i^{tr\setminus j}), z_{i,j}),$$

where the meta-sample is $\mathbf{S} = \{S_i^{tr}\}_{i=1}^n$. For any task $\mathcal{D}_i$, the LOO training strategy can be described as follows. For any data point in the support set $z_{i,j} \in S_i^{tr}$, we can form a new task with the rest of data $S_i^{tr\backslash j}$ being the leave-one-out support set and $z_{i,j}$ being the query. For inner-task training, the inner-task algorithm $\mathbf{A}(\mathbf{S})$ runs over each leave-one-out support set $S_i^{tr\backslash j}$ and outputs a hypothesis $\mathbf{A}(\mathbf{S})(S_i^{tr\backslash j})$ whose performance is measured by the corresponding query $z_{i,j}$. For meta-training, model parameters are updated by optimizing the average of the queries' errors $\frac{1}{m}\sum_{z_{i,j}\in S_i^{tr}} l(\mathbf{A}(\mathbf{S})(S_i^{tr\backslash j}), z_{i,j})$.

2.4.1 Generalization Bound of Meta-Algorithms with LOO Training

The following gives the uniform stability of meta-algorithms with LOO training.

Definition 5 (Uniform stability of meta-algorithms with LOO training)**.** *A meta-algorithm* $\mathbf{A}$ *has uniform stability* β *w.r.t. the loss function* l *if the following holds for any meta-sample* $\mathbf{S}$ *and* $\forall i \in \{1, \ldots, n\}, \forall \mathcal{D} \sim \tau, \forall S^{tr} \sim \mathcal{D}^m$*:*

$$|\hat{L}_{loo}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{tr}) - \hat{L}_{loo}(\mathbf{A}(\mathbf{S}^{\backslash i})(S^{tr}), S^{tr})| \leq \beta.$$

From a generalization perspective, the LOO training strategy is very different from the S/Q training strategy. For S/Q training, the query is unseen in the inner-task training since $S^{tr} \cap S^{ts} = \emptyset$, while for LOO training, the query has been seen, i.e., it is from S^{tr}. Similarly, the generalization gap of a meta-algorithm with LOO training can also be divided into the outer-task gap and the inner-task gap.

The proof of Theorem 2 can be straightforwardly extended to upper bound the outer-task gap. For the inner-task gap, given the stability of an inner-task algorithm $\tilde{\beta}$, the following holds for any meta-sample $\mathbf{S}$ and any test task $\mathcal{D} \sim \tau$,

$$\begin{aligned}\mathbb{E}_{S^{tr}\sim\mathcal{D}^m}\mathbb{E}_{z\sim\mathcal{D}} l(\mathbf{A}(\mathbf{S})(S^{tr}), z) \leq & \mathbb{E}_{S^{tr\backslash j}\sim\mathcal{D}^{m-1}}\mathbb{E}_{z\sim\mathcal{D}} l(\mathbf{A}(\mathbf{S})(S^{tr\backslash j}), z) + \tilde{\beta} \\ = & \mathbb{E}_{S^{tr}\sim\mathcal{D}}\hat{L}_{loo}(\mathbf{A}(\mathbf{S})(S^{tr}), S^{tr}) + \tilde{\beta}. \end{aligned} \tag{2.5}$$

Combining the upper bounds of the outer-task gap and the inner-task gap, we can obtain the following generalization bound which holds when the conditions of Theorem 2 are satisfied, i.e.,

$$\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) \leq \hat{\mathcal{R}}_{loo}(\mathbf{A}(\mathbf{S}), \mathbf{S}) + \epsilon(n, \beta, \tilde{\beta}), \tag{2.6}$$

where $\epsilon(n, \beta, \tilde{\beta}) = 2\beta + (4n\beta + M)\sqrt{\frac{\ln(1/\delta)}{2n}} + \tilde{\beta}$. This result indicates that the generalization bound of meta-algorithms with LOO training depends on both the uniform stability of the meta-algorithm and the inner-task algorithm. The bound converges to 0 as $n \to \infty$ and $\tilde{\beta} \to 0$, if $\beta < O(\frac{1}{\sqrt{n}})$.

Based on Theorem 3, we can derive the uniform stability parameter of meta-algorithms $\beta \leq O(1/n)$. Since $\tilde{\beta}$ is algorithmic-dependent, the derivation of $\tilde{\beta}$ depends on the specific inner-task algorithm. If $\tilde{\beta}$ exists, it depends on m and thus leads to a generalization bound of $\epsilon(n, m)$. As an example, we derive a generalization bound of order $O(1/\sqrt{n} + 1/m)$ for prototypical networks [100] with LOO training. The details are provided in Appendix A.3.

2.5 Experiments

To verify our analysis, we conduct experiments on few-shot regression and classification.[1]

Few-shot regression. We follow the experimental setting of MAML [30]. The problem aims to approximate a family of sine functions $f(x) = \alpha \sin(\beta x)$. The task distribution τ is the joint distribution $p(\alpha, \beta)$ of the amplitude parameter α and the phase parameter β. We set $p(\alpha) = U[0.1, 5]$ and $p(\beta) = U[0, \pi]$. All the training and test tasks are randomly generated from the task distribution $\tau = p(\alpha, \beta)$. We implement the meta-algorithms MAML [30] and Bilevel Programming [34] by using a MLP with two hidden layers of size 40 with ReLU activation function. Both the input layer and the output layer have dimensionality 1. The generalization gap of meta-algorithms is estimated by the gap between the training error and the test error. The test error is averaged over 600 test tasks with varying shots and 15 queries. The meta-training procedure exactly follows the episodic training strategy, i.e., the meta-algorithm observes a set of training tasks sequentially and applies stochastic gradient descent with one task per batch.

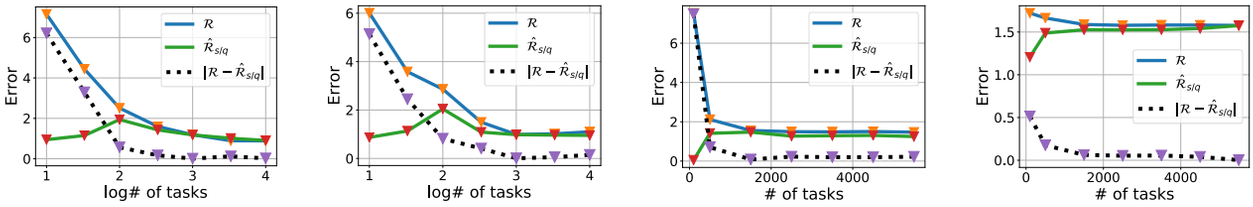
Few-shot classification. We follow the standard experimental setting proposed in [108] using the real-life dataset *mini*Imagenet. This dataset has 100 classes and is split into a training set

[1] The source code can be downloaded from https://github.com/jiaxinchen666/meta-theory.

of 64 classes, a test set of 20 classes and a validation set of 16 classes. Each task is formed by randomly selecting a few classes with m shots and q queries per class. We implement MAML [30] and ProtoNet [100] using the Conv-4 backbone and follow the implementation details in [16]. Few-shot classification on *mini*Imagenet is a benchmark task in modern meta-learning.

2.5.1 Convergence of the Generalization Gap as $n \to \infty$

Figure 2.1 reports the training error, the test error and the generalization gap of various meta-algorithms with S/Q training on few-shot classification and regression tasks. We set $m = 5, q = 1$ for regression and $m = 1, q = 1$ for classification. As expected, the test error decreases as $n \to \infty$, which demonstrates the benefit of meta-learning, i.e., training on more tasks makes the inner-task algorithm adapt well to a future task. More importantly, regardless of the inner-task sample size, the generalization gap always converges to 0 as $n \to \infty$. This phenomenon justifies our theoretical result, i.e., given enough training tasks, the generalization gap vanishes even though the inner-task training samples are very limited.



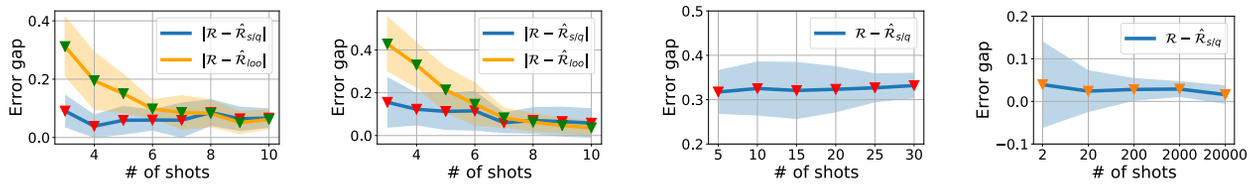
(a) Bilevel, regression (b) MAML, regression (c) ProtoNet, classification (d) MAML, classification

FIGURE 2.1: Generalization gaps of meta-algorithms with S/Q training on regression and classification.

2.5.2 Influence of Inner-Task Sample Size m on Generalization Gap

Note that our bound $O(1/\sqrt{n})$ in Theorem 4 is an upper bound of the generalization gap, which indicates that despite of a small sample size m, the generalization gap of a meta-algorithm with S/Q training can still converge to 0 as the task number n grows. However, the sample-size free upper bound cannot theoretically guarantee that the generalization gap is also sample-size free (independent of m). As such, we empirically study how the generalization gap changes as m increases.

Figure 2.2 (a) and (b) show the generalization gaps of Bilevel Programming [30] and MAML [34] respectively with both S/Q training and LOO training, by fixing the task number as $n = 1000$. It can be seen that for both meta-algorithms, the LOO error gap drops rapidly as the number of shots increases, which cannot be observed in the S/Q error gap. Meanwhile, S/Q training achieves a much smaller generalization error gap than LOO training in low-shot cases, indicating the advantage of S/Q training. Furthermore, Figure 2.2 (c) and (d) show the generalization gaps of ProtoNet [100] and Bilevel Programming [30] with S/Q training for classification and regression respectively, with m varying in a large range. In both scenarios, a flat trend can be observed, indicating that m may have little influence on generalization.



(a) Bilevel, regression (b) MAML, regression (c) ProtoNet, classification (d) Bilevel, regression

FIGURE 2.2: Generalization gaps of meta-algorithms trained with $n = 1000$ tasks. The horizontal axis represents the number of shots (sample size m). All results are averaged over 10 independent runs. (a) & (b): comparisons of S/Q training and LOO training. (c) & (d): S/Q training.

2.6 Related Work

To study the convergence of empirical error to generalization error, statistical learning theory provides two main ways: Vapnik-Chervonenkis (VC) theory and stability theory. VC theory studies model-free generalization bounds based on measures of the hypothesis set [106, 1, 8, 107]. The model-free bounds are extended to meta-learning to analyze the generalization of a learned hypothesis set. Based on PAC models and their variants, a generalization bound [5] of a learned hypothesis set is proposed and PAC-Bayes bounds [86, 87, 2] of a learned prior distribution of a hypothesis set are studied. However, they are not applicable to metric-based meta-algorithms, a branch of modern meta-learning, since the hypothesis set of a metric-based inner-task algorithm depends on training data and is uncertain [98].

Stability theory studies generalization bounds by considering stability of an algorithm instead of a measure of the hypothesis set. In single-task learning, it has been shown that if an algorithm (deterministic or randomized) is stable, the learned hypothesis can be generalized well [10, 26, 77, 99, 29, 74]. It is proved that algorithms with convex loss functions are stable [98]. And randomized algorithms with non-convex loss functions can also be stable [48, 60]. In meta-learning, the generalization bound of a meta-algorithm can be derived by the stability of the meta-algorithm and the inner-task algorithm [73]. However, the traditional training strategy processed all the data in one batch and did not consider the support/query strategy, which is not applicable to modern meta-algorithms.

Apart from the aforementioned works, many theoretical investigation were proposed in recent years. Some of them [53, 28, 22, 34] studied model-agnostic meta-algorithm [30] and explored the convergence guarantees for gradient-based meta-learning. Many others [19, 21, 12, 20, 54] focused on specific meta-algorithms and proposed the corresponding generalization guarantees. However, these results cannot be used to analyze practical meta-algorithms. As the loss function considered is convex or the mapping studied is linear, they are not applicable to deep neural network. Also, the training strategy is not episodic, which will fail to train practical popular meta-algorithms [30, 100, 34]. The most related work [117] also considered the support/query episodic training strategy but their theoretical results are still dependent on the inner-task sample size. In this chapter, we target for a sample-size-free bound.

3

Adaptation-Agnostic Meta-Learning

3.1 Overview

A meta-learning algorithm (meta-algorithm) is trained over a large number of tasks such that it can learn an algorithm (*base-learner*) which can adapt to a new task with few training samples. Existing meta-algorithms are characterized by the type of base-learners. For metric-based meta-algorithms [108, 100], the base-learners are nearest neighbor algorithms such as k-NN [108] and mean-centroid classification algorithm [100] or simple algorithms with closed-form solvers such as ridge regression [6] or SVM [63]. Such base-learners are stable, efficient and less prone to overfitting but the model expressiveness is low. For gradient-based meta-algorithms [30], the base-learner is a gradient decent algorithm with a learned initialization. Such base-leaner leads to a model-agnostic meta-algorithm with high model capacity but is prone to overfitting and suffers from the expensive computation of second-order gradient. The choice of base-learner has significant influence on the performance of meta-algorithms, but there are not many choices and the existing ones have their respective drawbacks such as low model expressiveness, overfitting issue and high computational cost.

In this chapter, we show that the restrictions on the choices of base-learners are due to the meta-training strategy which requires the base-learner to be solved analytically. The commonly used meta-training procedure is an interleaved process which includes inner-task adaptation and meta-update. During inner-task adaptation, the base-learner runs through the support set

and outputs a predictor parameterized by *task-specific parameters*. During meta-update, the loss of the task-specific predictor over the query set is minimized to update the *meta-parameters* that are shared by all tasks. Most meta-algorithms follow a *pathwise meta-training* strategy which integrates inner-task adaptation into meta-update, and thereby, the gradient of meta-parameters is the product of the gradient of the task-specific parameters with respect to the meta-parameters and the gradient of the task-specific predictor's loss function over the query set w.r.t. the task-specific parameters. In order to obtain an explicit and differentiable loss function to make the back-propagation of meta-update feasible, the task-specific parameters should have an analytical expression with respect to the meta-parameters. To satisfy this requirement, the choices of base-learners are significantly limited. Further, such meta-training strategy lacks the flexibility to combine different types of base-learners to take their advantages.

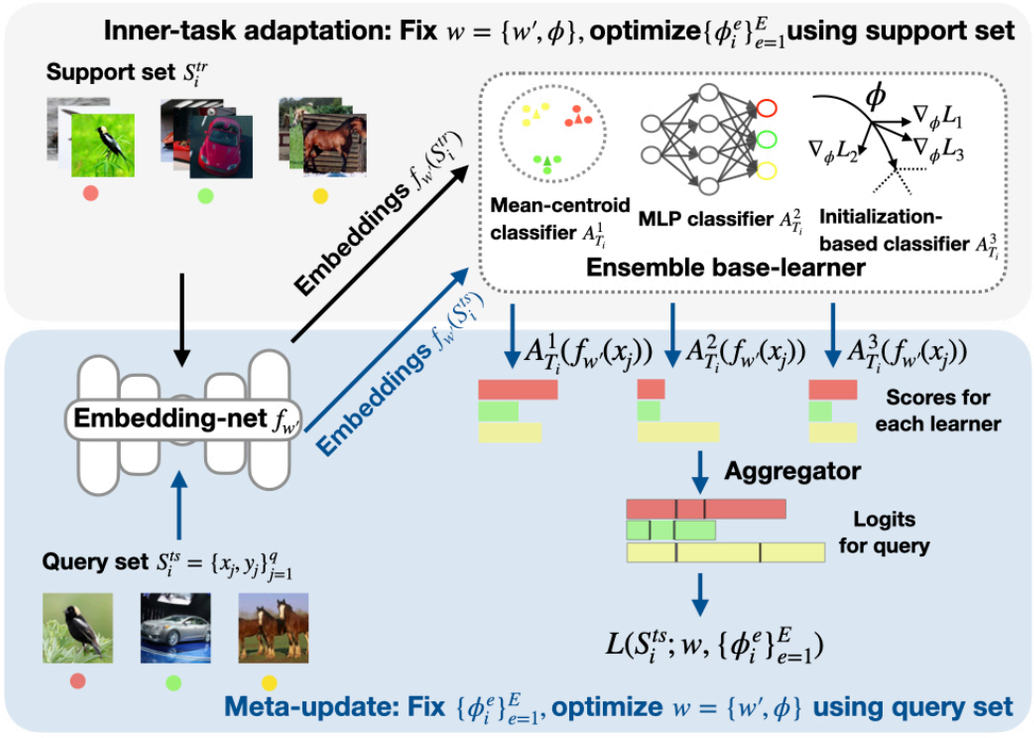


FIGURE 3.1: Diagram of the adaptation-agnostic ensemble framework (A2E).

To enrich the choices of base-learners and design stronger base-learners, we propose an adaptation-agnostic meta-training strategy to relax the analytical dependency between the task-specific parameters and the meta-parameters. As illustrated in Fig. 3.1, for inner-task

adaptation, we fix the meta-parameters and use the support set to optimize the task-specific parameters. For meta-update, we fix the task-specific parameters and optimize the meta-parameters using the query set. The meta-parameters are updated by minimizing the predictor's loss over the embedded query set. The proposed meta-training strategy is called adaptation-agnostic, since there is no assumption on the mathematical relationship between the task-specific parameters and the meta-parameters. When the base-learner has an analytical solution w.r.t. the meta-parameters, the proposed strategy reduces to normal meta-training. When the base-learner has an analytical solution w.r.t. a subset of meta-parameters, it becomes a partially decoupled meta-training procedure and encompasses newly proposed meta-algorithms such as ANIL [88]. When the base-learner has no analytical solutions, the meta-training procedure is fully decoupled, which opens up the opportunity of designing new base-learners.

Under this meta-training strategy, we further propose an adaptation-agnostic ensemble framework (A2E) for few-shot classification. Since the training data is very limited in the few-shot learning paradigm, an ensemble of base-learners can help to reduce the variance of algorithms and improve generalization [36, 23, 11]. Fortunately, the generality and flexibility of the proposed meta-training strategy makes it easy to combine different types of base-learners to exploit their advantages and alleviate their drawbacks. Since the proposed adaptation-agnostic meta-training strategy supports the normal, partially decoupled and fully decoupled meta-training procedures, it enables us to combine existing and new base-learners. For instance, as shown in Fig. 3.1, we combine the mean-centroid classification algorithm [100] (normal meta-training), the initialization-based inner-task algorithm [88] (partially decoupled meta-training), and a new non-linear multilayer perceptron classifier proposed by us (decoupled meta-training) as the ensemble base-learner. The ensemble base-learner has higher model expressiveness and is less prone to overfitting, which is verified by our experimental results.

3.2 Meta-Learning for Few-Shot Classification

3.2.1 Problem Formulation

The goal of a meta-algorithm $\mathbf{A}$ is to learn a *base-learner* $\mathcal{A}$ which can fast adapt to new tasks drawn from a task distribution τ. It involves two basic procedures: meta-training and meta-test.

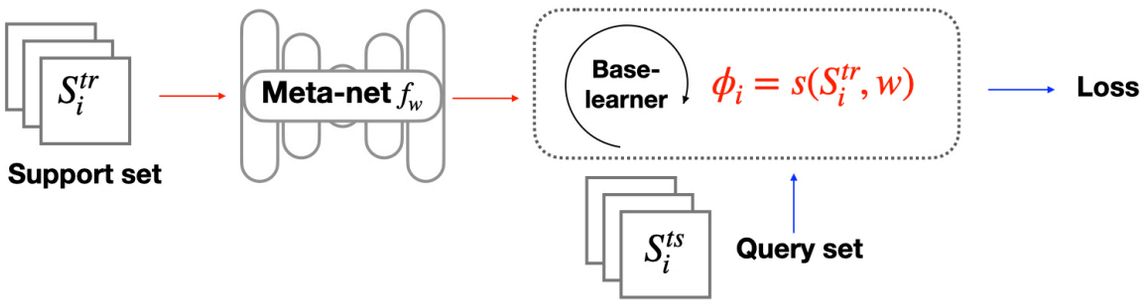


FIGURE 3.2: A common meta-training procedure of existing meta-algorithms. Note that $s(\cdot, \cdot)$ is an analytical expression.

During meta-training, given a set of training tasks $\{\mathcal{D}_i \sim \tau\}_{i=1}^n$, the meta-learner observes the meta-samples $\mathbf{S} = \{S_i = (S_i^{tr}, S_i^{ts})\}_{i=1}^n$, where S_i^{tr} is the training (*support*) set of task i and S_i^{ts} is the test (*query*) set of i. Let us denote a data sample by $z = (x, y) \in \mathcal{Z} = (\mathcal{X}, \mathcal{Y})$ a data sample, where $\mathcal{X}$ is the feature space and $\mathcal{Y}$ is the label space. For K-way classification, $y \in \{1, 2, \ldots, K\}$, a support set and a query set of a training task are of size m and q respectively, i.e., $S_i^{tr} = \{z_j = (x_j, y_j)\}_{j=1}^m$ and $S_i^{ts} = \{z_j = (x_j, y_j)\}_{j=m+1}^{m+q}$. After trained on these tasks, the meta-learner outputs a base-learner $\mathcal{A} = \mathbf{A}(\mathbf{S})$. During meta-test, given a new task $\mathcal{D} \sim \tau$ and the associated support set S^{tr}, the base-learner efficiently adapts to the support set and outputs a task-specific predictor $\mathcal{A}(S^{tr})$.

The base-learner is parametrized by *meta-parameters* w which is learned during meta-training. From a representation learning perspective, the base-learner $\mathcal{A}$ can be seen as a pre-selected algorithm with a feature extractor f_w parameterized by w, which is usually instantiated by a differentiable deep neural network.

Meta-training proceeds by improving the performance of the base-learner across the training tasks. Specifically, a meta-training episode consists of two steps: *inner-task adaptation* and *meta-update*. In inner-task adaptation, the base-learner adapts to the support set and outputs a task-specific predictor $g_{\phi_i} = \mathcal{A}(S_i^{tr}; w)$. We call ϕ_i *task-specific parameters* which is only computed and used for the current task i. For meta-update, the meta-algorithm updates the base-learner (the meta-parameters w) by optimizing its performance across the training tasks, which is measured by the loss of the task-specific predictor g_{ϕ_i} over the query set of each task.

3.2.2 A Unified View of Existing Meta-Learning Methods

We characterize that the existing meta-algorithms leverage a *pathwise meta-training procedure* as the gradient of the meta-parameters w is computed through the inner-task adaptation. As the meta-algorithm updates the meta-parameters w by minimizing the loss of the task-specific predictor g_{ϕ_i} over the query set of each task. The update rule of w is

$$w = w - \nabla_w \mathcal{L}(\{S_i^{ts}\}_{i=1}^n; w, \{\phi_i\}_{i=1}^n) = w - \nabla_w [\sum_{i=1}^{n} \sum_{z_j \in S_i^{ts}} l\left(g_{\phi_i}(x_j), y_j\right)]$$

$$= w - \nabla_{\phi_i} \sum_{i=1}^{n} \sum_{z_j \in S_i^{ts}} l\left(g_{\phi_i}(x_j), y_j\right) \times \nabla_w g_{\phi_i}, \tag{3.1}$$

where $g_{\phi_i} = \mathcal{A}(S_i^{tr}; w)$. It can be discovered that following the update rule (3.1), the gradients of the meta-parameters are back-propagated through the task-specific parameters ϕ_i. To make the back-propagation feasible, most existing meta-algorithms follow a common meta-training procedure as shown in Fig. 3.2. First, the task-specific parameters ϕ_i are directly computed w.r.t. w, i.e.,

$$g_{\phi_i} = \mathcal{A}(S_i^{tr}; w), \text{where } \phi_i = s(S_i^{tr}, w), \tag{3.2}$$

and $s(\cdot, \cdot)$ denotes an *analytical expression*. Then, ϕ_i is plugged back to the meta objective function (3.1) and w is optimized by the gradient propagated from ϕ_i.

For example, the task-specific parameters of a typical **gradient-based meta-algotithm**, MAML [30] ϕ_i is

$$\phi_i = w - l_{\phi_i} \nabla_w \mathcal{L}(S_i^{tr}; w) = w - \nabla_w \big(\sum_{(x_j, y_j) \in S_i^{tr}} l\big(f_w(x_j), y_j\big) \big). \tag{3.3}$$

Then, the gradients of w include a second-order gradient of w because

$$\nabla_w \phi_i = I - \nabla_w^2 \big(\sum_{(x_j, y_j) \in S_i^{tr}} l\big(f_w(x_j), y_j\big) \big). \tag{3.4}$$

Following pathwise meta-training procedure, if ϕ_i cannot be analytically computed by w, the gradient of w cannot be computed and the back-propagation cannot proceed. It turns out that the choice of base-learner with different analytical expressions characterizes the key

difference among existing meta-algorithms. Apart from gradient-based meta-algorithms such as MAML, the other popular meta-algorithms can be unified in this perspective.

Metric-based meta-algorithms. The base-learner of a metric-based meta-algorithm is a nearest neighbor algorithm with a distance function in the metric space, e.g., $d(x, x') = \|f_w(x) - f_w(x')\|_2^2$. For matching networks [108], the nearest neighbor algorithm is non-parametric, so there is no explicit training in inner-task adaptation. For prototypical networks [100], the task-specific parameters are the mean vectors of same-class support samples, which can be computed as

$$\phi_i = \{\frac{1}{N} \sum_{(x_j, y_j) \in S_i^{tr}, y_j = k} f_w(x_j)\}_{k=1}^K. \tag{3.5}$$

Model-based meta-algorithms. Some of the model-based meta-algorithms avoid inner-task training by learning a meta amortization network G parameterized by ψ to generate task-specific parameters ϕ_i using the support set as inputs [42, 43], i.e.,

$$\phi_i = G_\psi\big(f_w(S_i^{tr})\big). \tag{3.6}$$

Both w and ψ are global parameters to be optimized in meta-training.

Meta-algorithms with closed-form solvers. Several meta-algorithms adopt a simple algorithm with convex objective function as base-learner such that the task-specific parameters ϕ_i have a closed-form solution [6, 63]. For example, [6] uses ridge regression as base-learner, and the closed-form solution is

$$\phi_i = ({X_w}^T X_w + \lambda I)^{-1} {X_w}^T Y. \tag{3.7}$$

For brevity, $X_w = \{f_w(x_j)\}_{j=1}^m$ and $Y = \{y_j\}_{j=1}^m$, where $(x_j, y_j) \in S_i^{tr}$ [6].

Limitation of Pathwise Meta-Training Procedure. From this perspective, the key challenge in designing a meta-algorithm is to find a base-learner which has an explicit analytical expression of task-specific parameters. However, this requirement significantly limits the choice of base-learners and thereby limits the power of the corresponding meta-algorithms. For example, [6] and [63] are restricted to use simple algorithms with convex objective function such as ridge regression or support vector machines. To satisfy this requirement, gradient-based meta-algorithms such as MAML also suffer from computationally expensive second-order gradients.

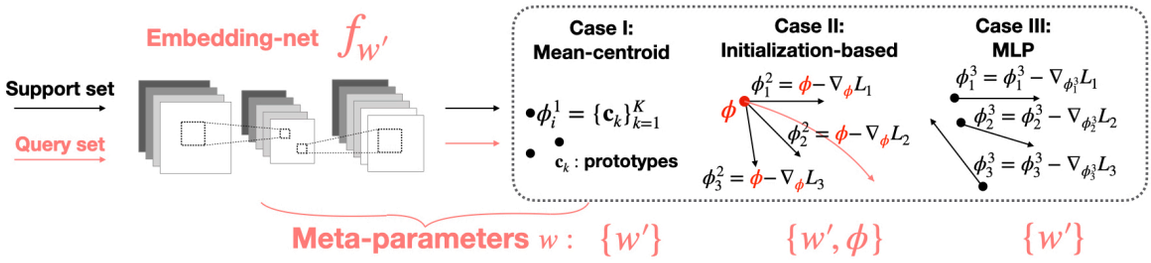


FIGURE 3.3: Inner-task adaptation of A2E (mean-centroid classification algorithm of [100], initialization-based base-learner in [88] and MLP proposed by us (Eq. (3.10)).

3.3 An Adaptation-Agnostic Meta-Training Procedure

To enrich the choices of base-learners and apply a powerful algorithm as the base-learner, in this section, we propose an adaptation-agnostic meta-training strategy that removes the analytical dependency between the task-specific parameters and the meta-parameters.

3.3.1 Adaptation-Agnostic Meta-Training

The key constraint of the normal meta-training strategy is that inner-task adaptation should be integrated into meta-update such that a meta objective function (Eq. (3.1)) for optimizing the meta-parameters w can be derived. Hence, the task-specific parameters should be formulated as an explicit analytical expression w.r.t. w so as to derive the meta objective function. To relax this constraint, we propose an adaptation-agnostic meta-training strategy which makes no assumption on such dependency.

In particular, we do not enforce inner-task adaptation to be integrated into meta-update but propose to conduct these two steps separately and iteratively. As shown in Fig. 3.1, in inner-task adaptation, the meta-parameters w is fixed, and the support set is fed to the shared embedding network and used to train the task-specific predictor $\mathcal{A}_{\phi_i}$. In meta-update, the task-specific parameters are fixed and the query set is used to optimize the meta-parameters w. The iteration scheme is formulated as follows:

$$\text{Inner-task adaptation: Fix } w, \phi_i = \arg\min_{x_{\phi_i}} \mathcal{L}(S_i^{tr}; w, x_{\phi_i}), \tag{3.8}$$

$$\text{Meta-update: Fix } \phi_i, w = w - l_w \nabla_w \mathcal{L}(S_i^{ts}; w, \phi_i), \tag{3.9}$$

where w refers to the meta-parameters, i.e., the global parameters shared by all the tasks, and

ϕ_i refers to the task-specific parameters, i.e., the local parameters which are different among the tasks. We call this training strategy *adaptation-agnostic*, since in Eq. (3.8) allows the use any inner-task algorithm with any optimization algorithm as long as the meta loss function $\mathcal{L}(S_i^{ts}; w, \phi_i)$ is differentiable w.r.t. w given ϕ_i, regardless of whether ϕ_i has an analytical expression w.r.t. w. The proposed adaptation-agnostic training strategy encompasses the following three cases.

Case I Normal meta-training. If the base-learner has an analytical solution, finding the minimum solution of ϕ_i in Eq. (3.8) is equivalent to solve ϕ_i analytically and the expression $\phi_i = s(S_i^{tr}, w)$ can be obtained as in Sec. 3.2.2. Then, $\phi_i = s(S_i^{tr}, w)$ will be plugged into Eq. (3.9) to optimize w. In this case, it is equivalent to the normal meta-training procedure introduced in Sec. 3.2.2.

Case II Partially decoupled meta-training. The proposed strategy can also be used to train a base-learner with an analytical solution w.r.t. a portion of meta-parameters, i.e., $\phi_i = s(f_{w'}(S_i^{tr}), \phi)$ where ϕ and w' are subsets of meta-parameters, i.e., $w = \{w', \phi\}$. Here, inner-task adaptation is to derive the analytical expression w.r.t. ϕ, and meta-update optimizes $w = \{w', \phi\}$, hence the name partially decoupled meta-training. For example, ANIL, a simplified version of MAML [30] recently proposed in [88], belongs to this case, where ϕ stands for the initialization of the final layers and w' stands for the parameters of the representation layers. In the inner loop, ANIL only updates the final layers ($\phi_i = \phi - l_\phi \nabla_\phi \mathcal{L}(f_{w'}(S_i^{tr}); \phi)$) with the representation layers (embedding-net $f_{w'}$ in Fig. 3.3) frozen. In the outer loop, it updates both ϕ and w'.

Case III Fully decoupled meta-training. In this case, the base-learner does not have an analytical expression w.r.t. the meta-parameters w, so a fully decoupled meta-training is adopted. In inner-task adaptation, the embedded support set $f_{w'}(S_i^{tr})$ ($w = w'$) is fed to Eq. (3.8), and the task-specific parameters ϕ_i are learned using some optimization algorithm such as stochastic gradient descent (SGD). Although we do not know the exact mathematical relationship between ϕ and w, we can still optimize the meta-parameters w by plugging ϕ_i into Eq. (3.9), i.e., optimizing w with fixed ϕ_i.

A new base-learner. Without the requirement of an analytical solution, the choice of

base-learner is of great flexibility. Naturally, we come up with a neural network with a non-convex loss function, i.e., cross-entropy loss. Since there is no restriction on the optimization algorithm or the network architecture, we simply use a multilayer perceptron (MLP) trained by SGD as an inner-task algorithm. The inner-task adaptation can be formulated as:

$$\phi_i = \phi_i - l_{\phi_i} \nabla_{\phi_i} \mathcal{L}(f_{w'}(S_i^{tr}); \phi_i), \tag{3.10}$$

where $w = w'$ and ϕ_i is randomly initialized for each task. Compared with the partially decoupled meta-training algorithm such as ANIL [88] in Case II, a significant difference is that for each task, here the task-specific parameters ϕ_i are randomly initialized instead of using a learned ϕ. This may make the base-learner more flexible and less prone to overfitting, as verified in our experiments.

3.3.2 Applying A Powerful Algorithm as the Base-Learner

The flexibility of the proposed adaptation-agnostic meta-training strategy enables us to combine the advantages of different types of base-learner to obtain a stronger base-learner that is more robust and can generalize better to new tasks. In this section, we propose an adaptation-agnostic ensemble framework (A2E) which can easily and efficiently combine a bag of diverse inner-task algorithms.

As illustrated in Fig. 3.1, during inner-task adaptation, we train a bag of diverse algorithms $\{\mathcal{A}^e\}_{e=1}^E$ separately with the embedded support set and obtain E predictors, i.e., $\{\mathcal{A}^e(S_i^{tr}; w)\}_{e=1}^E$. Next, meta-update is performed by aggregating the predictions of all the predictors on the query set to obtain final predictions and then using the final predictions to update the shared meta-parameters w. Formally, the meta-training procedure of A2E is formulated as follows,

$$\begin{aligned} &\text{Inner-task adaptation: Fix } w, \text{for } e \in \{1, 2, \ldots, E\}, \phi_i^e = \arg\min_{x_{\phi_i^e}} \mathcal{L}(S_i^{tr}; w, x_{\phi_i^e}), \\ &\text{Meta-update: Fix } \{\phi_i^e\}_{e=1}^E, w = w - l_w \nabla_w \mathcal{L}(S_i^{ts}; w, \{\phi_i^e\}_{e=1}^E). \end{aligned} \tag{3.11}$$

An instantiation of A2E. A2E (Eq. 3.11) is a very general framework, and it can basically integrate any inner-task algorithm as base-learner for diverse purposes. Since this chapter focuses on few-shot classification, we instantiate A2E with an ensemble of the mean-centroid classification algorithm of ProtoNets [100], the initialization-based inner-task algorithm as in

MAML [30], and a two-layer MLP as inner-task classifier proposed by us (Eq. (3.10)) as the ensemble base-learner for meta-learning. Note that for the inner-task algorithm of MAML, we use the partially decoupled version as in ANIL [88]. As such, the three diverse inner-task algorithms fall into Case I, Case II and Case III of our proposed adaptation-agnostic meta-training respectively (Fig. 3.3), and can be naturally combined in A2E framework. In addition, we choose to combine the mean-centroid classification algorithm and the initialization-based algorithm due to their complementary capabilities. The former has low model capacity but stable, while the latter has high model expressiveness but can easily overfit. The effectiveness of the proposed ensemble base-learner is empirically verified by our experiments in Sec. 3.4.

For inner-task adaptation, as illustrated in Fig. 3.3, the three algorithms are trained over the embedded support set independently, i.e,:

$$\begin{aligned}
\mathcal{A}^1\text{: } \phi_i^1 =& \{\mathbf{c}_k\}_{k=1}^K = \{\frac{1}{N}\sum_{z_j \in S_i^{tr}, y_j=k} f_{w'}(x_j)\}_{k=1}^K, \\
\mathcal{A}^2\text{: } \phi_i^2 =& \phi - l_\phi \nabla_\phi [\frac{1}{m}\sum_{z_j \in S_i^{tr}} -\log\left(\frac{e^{g_\phi(f_{w'}(x_j))[y_j]}}{\sum_{k'} e^{g_\phi(f_{w'}(x_j))[k']}}\right)], \\
\mathcal{A}^3\text{: } \phi_i^3 =& \phi_i^3 - l_{\phi_i^3} \nabla_{\phi_i^3} [\frac{1}{m}\sum_{z_j \in S_i^{tr}} -\log(\frac{e^{g_{\phi_i^3}(f_{w'}(x_j))[y_j]}}{\sum_{k'} e^{g_{\phi_i^3}(f_{w'}(x_j))[k']}})],
\end{aligned} \tag{3.12}$$

where $\mathcal{A}^1$, $\mathcal{A}^2$ and $\mathcal{A}^3$ denote the mean-centroid classification algorithm (Case I), the initialization-based algorithm (Case II) and the two-layer MLP (Case III) respectively. Note that for $\mathcal{A}^2$, ϕ is shared by each task and updated during meta-update.

In meta-update, as shown in Fig. 3.1 and Fig. 3.3, for any query $\{z_j = (x_j, y_j) \in S_i^{ts}\}$, the predictions of $\mathcal{A}^1$, $\mathcal{A}^2$ and $\mathcal{A}^3$ are aggregated to produce the final prediction. In our instantiation, we sum up all the predictions and use the output as the query's logits for computing the cross-entropy loss. Specifically, given the task-specific parameters ϕ_i^1, ϕ_i^2 and ϕ_i^3, the meta-update process is as follows,

$$w = w - l_w \nabla_w [\frac{1}{q}\sum_{z_j \in S_i^{ts}} -\log\left(\frac{e^{g_{\phi_i^3}(f_{w'}(x_j))[y_j] + g_{\phi_i^2}(f_{w'}(x_j))[y_j] - d(f_{w'}(x_j), \mathbf{c}_{y_j})}}{\sum_{k'} e^{g_{\phi_i^3}(f_{w'}(x_j))[y_j] + g_{\phi_i^2}(f_{w'}(x_j))[k'] - d(f_{w'}(x_j), \mathbf{c}_{k'})}}\right)],$$

where $d(\cdot, \cdot)$ is the distance between the query's embedding and the prototype and $w = \{w', \phi\}$.

3.4 Experimental Results

Experiments were designed to evaluate the performance of A2E introduced in Sec. 3.3.2 on standard and cross-domain few-shot classification tasks. In our ensemble framework, we combine the mean-centroid classification algorithm of ProtoNets [100], the two-layer MLP classifier proposed in Sec. 3.3.1 and the initialization-based (init-based) inner-task algorithm in ANIL (simplified MAML) [88] [1] into an ensemble base-learner.

3.4.1 Experimental Setup

Datasets. The ***mini*ImageNet** [108] consists of 100 classes with 600 images per class. The dataset is split into a training set with 64 classes, a testing set with 20 classes and a validation set with 16 classes [93]. Following the convention, the images are cropped into 3×84×84 and 3×224×224 when using CNN-based [108] and ResNet-based model architectures [16] respectively.

The **CUB** dataset [109] contains 200 classes and 11,788 images in total. The CUB dataset is split into 100 classes for training, 50 classes for validation and 50 classes for testing [16]. The input size of images in CUB is 3×224×224. **Implementation details.** In order to achieve a **fair** comparison, we employ the consistent experimental environment proposed in [16] and strictly follow its training details in it. Specifically, we compare the performance using the widely-used Conv-4 as in [100] and the ResNet-18 backbone adopted in their environment. We have not applied any high-way or high-shot training strategy. For the optimizer, we use Adam [55] as the meta-optimizer with a fixed learning rate 0.001. For the cross-domain tasks, we train models on the entire *mini*ImageNet dataset. The meta-validation and meta-test of the models use the validation set and test set of the CUB dataset respectively.

Comparison with the state-of-the-art

For **fair** comparison, here we compare with the state-of-the-art methods that have a similar implementation (e.g., using the same backbone network) as ours. We use a standard ResNet-18 backbone [49]. Differently, MetaOptNet [63] and TADAM [84] use a ResNet-12 backbone;

[1] [88] shows that the simplified MAML, i.e., ANIL, achieves the same performance as MAML [30]. Hence, it suffices to only compare with MAML.

Table 3.1: Results of 5-way classification tasks on *mini*ImageNet using Conv-4 (the above set) and ResNet-18 (the below set) respectively. Compared results are from references except $*$ re-implemented by [16].

***mini*ImageNet test accuracy**		
Model	5-way 1-shot	5-way 5-shot
Matching Net [108]	43.56 ± 0.84	55.31 ± 0.73
Relation Net [102] $*$	49.31 ± 0.85	66.60 ± 0.69
Meta LSTM [93]	43.44 ± 0.77	60.60 ± 0.71
SNAIL [75]	45.10	55.20
LLAMA [44]	49.40 ± 1.83	–
REPTILE [81]	49.97 ± 0.32	65.99 ± 0.58
PLATIPUS [32]	50.13 ± 1.86	–
GNN [37]	50.30	66.40
R2-D2 (high) [6]	49.50 ± 0.20	65.40 ± 0.20
MAML [30] $*$	46.70 ± 1.84	63.11 ± 0.92
Protonet [100] $*$	44.42 ± 0.84	64.24 ± 0.72
ANIL [88]	46.70 ± 0.40	61.50 ± 0.50
A2E (Mean-centroid + MLP+ Init-based)	$\mathbf{50.31 \pm 0.87}$	$\mathbf{68.55 \pm 0.67}$
Matching Net [108] $*$	52.91 ± 0.88	68.88 ± 0.69
Relation Net [102] $*$	52.48 ± 0.86	69.83 ± 0.68
MAML [30] $*$	49.61 ± 0.92	65.72 ± 0.77
Protonet [100] $*$	54.16 ± 0.82	73.68 ± 0.65
A2E (Mean-centroid + MLP+ Init-based)	$\mathbf{57.04 \pm 0.84}$	$\mathbf{75.65 \pm 0.71}$

LEO [95] uses a WRN-28-10 backbone. Besides, we do not use techniques such as DropBlock regularization, label smoothing and weight decay as adopted in MetaOptNet [63] to increase performance. Hence, we do not compare with these methods.

3.4.2 Main Results

Performance on *mini*Imagenet. For the standard few-shot scenario, we conduct experiments of 5-way 1-shot and 5-way 5-shot classification on *mini*ImageNet with the Conv-4 and

the ResNet-18 backbones. The results are shown in Table 3.1. For both 1-shot and 5-shot tasks, our model achieves comparable or superior performance compared with state-of-the-art meta-algorithms. Remarkably on the ResNet-18 backbone in Table 3.1, A2E outperforms the best meta-algorithms by achieving approximate 3% and 2% absolute increases in the 1-shot and 5-shot tasks respectively, demonstrating the effectiveness of A2E.

Cross-domain classification. To further examine the generalization ability of our method, we conduct experiments on the challenging cross-domain classification task proposed in [16]. The results are shown in Table 3.2. Here, D-MLP denotes the decoupled meta-training with a MLP base-learner proposed by us as in Sec. 3.3.1. For 5-way 5-shot classification, D-MLP achieves 6% absolute increases compared with MAML [30], which indicates that D-MLP is less prone to overfitting than MAML. Our ensemble framework A2E achieves 7%, 2.5%, 13% absolute increase over D-MLP, MAML [30] and PN [100] respectively. The results show that our adaptation-agnostic ensemble framework facilitates the meta-net to learn more general structures that can adapt better to new tasks with a domain shift.

Table 3.2: Results for a 5-way cross-domain classification task.

*mini*ImageNet→CUB		
	5-way 1-shot	5-way 5-shot
Matching networks [108]	41.10 ± 0.74	53.07 ± 0.74
Prototypical networks [100]	42.71 ± 0.78	62.02 ± 0.70
Relation net [102]	40.74 ± 0.76	57.71 ± 0.73
MAML [30]	32.77 ± 0.64	51.34 ± 0.72
D-MLP	35.88 ± 0.66	57.78 ± 0.76
A2E (Mean-centroid + MLP + Init-based)	$\mathbf{43.55 \pm 0.80}$	$\mathbf{64.63 \pm 0.82}$

3.4.3 An Ablation Study of A2E

To further study our proposed A2E, we provide an ablation study using the Conv-4 backbone. In Table 3.3, we can observe that A2E (mean-centroid + MLP + init-based) achieves best results when compared with each individual component or an ensemble of any two components. This further demonstrates that the ensemble method is effective.

Besides, we observe that A2E has the advantage of combining the strength of individual components while mitigating their drawbacks. On one hand, focusing on the results of 5-way 1-shot classification. It can be seen that all the variants of A2E achieve better results compared with the individual component. It is well known that models are extremely easy to overfit in the 1-shot scenario. Clearly, our framework is capable of reducing classification variance in such cases. On the other hand, inspecting the outcomes of the 5-way 5-shot classification, the results of the ensemble including the mean-centroid component are 66.61% and 67.55% which outperform the result of (MLP+init-based), i.e., 63.84% without the mean-centroid component. It demonstrates the power of the mean-centroid component in preventing overfit as the shot number increases and the ensemble method can obtain such advantage after incorporating the mean-centroid classification algorithm.

Table 3.3: An ablation study about components in A2E using the Conv-4 backbone. Results are obtained on *mini*Imagenet.

Mean-centroid	MLP	Init-based	5-way 1-shot	5-way 5-shot
✓			44.42 ± 0.84	64.24 ± 0.72
	✓		45.00 ± 0.39	64.38 ± 0.33
		✓	46.70 ± 1.84	63.11 ± 0.92
✓	✓		46.99 ± 0.43	66.61 ± 0.38
	✓	✓	49.74 ± 0.88	63.84 ± 0.73
✓		✓	50.10 ± 0.81	67.55 ± 0.37
✓	✓	✓	$\mathbf{50.31 \pm 0.87}$	$\mathbf{68.55 \pm 0.67}$

3.4.4 Efficiency

We provide a quantitative comparison by measuring the meta-training and meta-testing time for 100 episodes shown in Fig. 3.4 and our results are obtained on 5-way 1-shot models with the ResNet-18 backbone. Fig. 3.4 shows that our A2E merely increases the running time by a small margin even when combining three components in the ensemble and validates our statement that A2E is efficient.

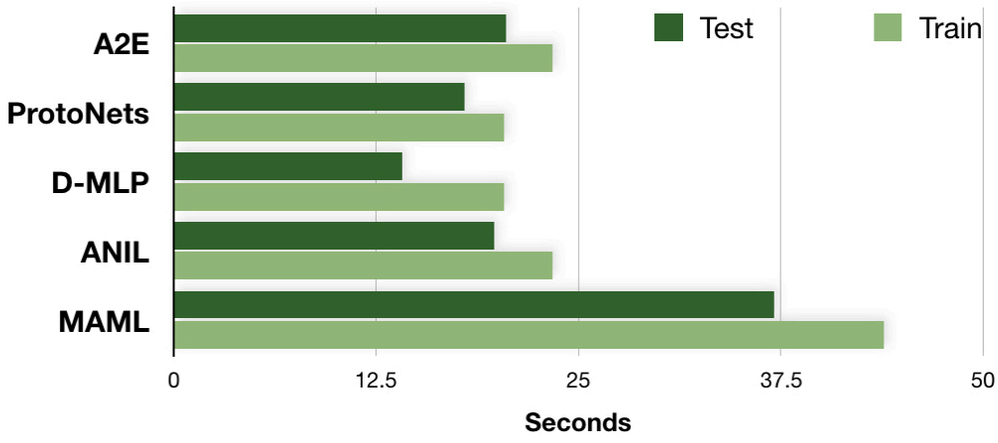


FIGURE 3.4: Comparison of running times. For brevity, A2E refers to A2E (mean-centroid + D-MLP + init-based) and MAML is the abbreviation for MAML first-order approximate version

3.5 Related Work

Decoupled training strategies have been explored in meta-learning or multi-task learning literature [34, 35, 115, 119]. Reference [34] proposed a bilevel programming for hyperparameter optimization and meta-learning, which is essentially similar to our proposed training strategy. Similar training strategies are adopted in [35] and [115] for reinforcement learning and multi-task learning respectively, which divides a network into shared layers and task-specific layers and updates them iteratively. CAVIA [119] trains a set of task-specific parameters and the model parameters iteratively. However, CAVIA is still a variant of MAML and it introduces a task-specific context parameter which serves as an additional input and is augmented into the model (e.g., hidden layers of the meta-net). However, the motivation and use of the training strategy in these works are totally different from ours. Our decoupled training procedure is motivated from an adaptation-agnostic perspective for meta-learning, which enables us to combine different types of base-leaner into an ensemble.

Existing ensemble methods for meta-learning include [69] and [25]. Reference [69] combines multiple versions of MAML with different hyper-parameters and [25] trains an ensemble of networks in a supervised-learning manner and then applies knowledge distillation to compress the ensemble model into a single network. Different from these works, our method is the first framework capable of combining various types of base-learners into an ensemble to

obtain a stronger base-learner.

Notice that [103] proposed a method Proto-MAML which combines prototypical networks and MAML by initializing the final layer of MAML with the prototype parameters learned in each inner-loop. Different from that, our framework is an ensemble learning framework, in which each base-learner is treated independently and updated collectively.

4

Variational Metric Scaling for Metric-Based Meta-Learning

4.1 Overview

Many metric-based meta-algorithms employ a softmax classifier with cross-entropy loss, which is computed with the logits being the distances between a query and supports in the embedding (metric) space. However, it has been shown that the scale of the logits – the metric scaling parameter, is critical to the performance of the learned model. Reference [100] found that Euclidean distance significantly outperforms cosine similarity in few-shot classification, while [84] and [112] pointed out that there is no clear difference between them if the logits are scaled properly. They supposed that there exists an optimal metric scaling parameter which is data and architecture related, but they only used cross validation to manually set the parameter, which requires pre-training and cannot find an ideal solution.

In this chapter, we aim to design an end-to-end method that can automatically learn an accurate metric scaling parameter. Given a set of training tasks, to learn a data-dependent metric scaling parameter that can generalize well to a new task, Bayesian posterior inference over learnable parameters is a theoretically attractive framework [43, 92]. We propose to recast metric-based meta-algorithms from a Bayesian perspective and take the metric scaling parameter as a global parameter. As exact posterior inference is intractable, we introduce a variational approach to efficiently approximate the posterior distribution with stochastic

variational inference.

While a proper metric scaling parameter can improve classification accuracy via adjusting the cross-entropy loss, it simply rescales the embedding space but does not change the relative locations of the embedded samples. To transform the embedding space to better fit the data distribution, we propose a dimensional variational scaling method to learn a scaling parameter for each dimension, i.e., a metric scaling vector. Further, in order to learn task-dependent embeddings [84], we propose an amortized variational approach to generate task-dependent metric scaling vectors, accompanied by an auxiliary training strategy to avoid time-consuming pre-training or co-training.

Our metric scaling methods can be used as pluggable modules for metric-based meta-algorithms. For example, it can be incorporated into prototypical networks (PN) [100] and all PN-based algorithms to improve their performance. To verify this, we conduct extensive experiments on the *mini*ImageNet benchmark for few-shot classification progressively. First, we show that the proposed stochastic variational approach consistently improves on PN, and the improvement is large for PN with cosine similarity. Second, we show that the dimensional variational scaling method further improves upon the one with single scaling parameter, and the task-dependent metric scaling method with amortized variational inference achieves the best performance. We also incorporate the dimensional metric scaling method into TADAM [84] in conjunction with other tricks to be proposed by the authors and observe notable improvement. Remarkably, after incorporating our method, TADAM achieves highly competitive performance compared with state-of-the-art methods.

4.2 Preliminaries

4.2.1 Notations and Problem Statement

Let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be a domain where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. Assume we observe a meta-sample $\mathbf{S} = \{S_i = S_i^{tr} \cup S_i^{ts}\}_{i=1}^{n}$ including n training tasks, where the i-th task consists of a support set of size m, $S_i^{tr} = \{z_{ij} = (x_{ij}, y_{ij})\}_{j=1}^{m}$, and a query set of size q, $S_i^{ts} = \{z_{ij} = (x_{ij}, y_{ij})\}_{j=m+1}^{m+q}$. Each training data point z_{ij} belongs to the domain $\mathcal{Z}$. Let us denote the model parameters by w and the metric scaling parameter by α . Given a new task

and a support set S^{tr} sampled from the task, the goal is to predict the label y of a query x.

4.2.2 Prototypical Networks

Prototypical networks (PN) [100] is a popular and highly effective metric-based meta-algorithm. PN learns a mapping ϕ_w which projects queries and the supports to an M-dimensional embedding space. For each class $k \in \{1, 2, \ldots, K\}$, the mean vector of the supports of class k in the embedding space is computed as the class *prototype* $\mathbf{c}_k$. The embedded query is compared with the prototypes and assigned to the class of the nearest prototype. Given a similarity metric $d : \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}^+$, the probability of a query z_{ij} belonging to class k is,

$$p_w(y_{ij} = k | x_{ij}, S_i^{tr}) = \frac{e^{-d(\phi_w(x_{ij}), \mathbf{c}_k)}}{\sum_{k'=1}^{K} e^{-d(\phi_w(x_{ij}), \mathbf{c}_{k'})}}. \tag{4.1}$$

Training proceeds by minimizing the cross-entropy loss, i.e., the negative log-probability $-\log p_w(y_{ij} = k | x_{ij}, S_i^{tr})$ of its true class k. After introducing the metric scaling parameter α, the classification loss of the i^{th} task becomes

$$\mathcal{L}(w; S_i) = -\sum_{j=m+1}^{m+q} \log \frac{e^{-\alpha * d(\phi_w(x_{ij}), \mathbf{c}_{y_{ij}})}}{\sum_{k'=1}^{K} e^{-\alpha * d(\phi_w(x_{ij}), \mathbf{c}_{k'})}}. \tag{4.2}$$

The metric scaling parameter α has been found to affect the performance of PN significantly.

4.3 Variational Metric Scaling

4.3.1 Stochastic Variational Scaling

In the following, we recast metric-based meta-learning from a Bayesian perspective. The predictive distribution can be parameterized as

$$p_w(y | x, S^{tr}, \mathbf{S}) = \int p_w(y | x, S^{tr}, \alpha) p_w(\alpha | \mathbf{S}) d\alpha. \tag{4.3}$$

The conditional distribution $p_w(y | x, S^{tr}, \alpha)$ is the discriminative classifier parameterized by w. Since the posterior distribution $p_w(\alpha | \mathbf{S})$ is intractable, we propose a variational distribution $q_\psi(\alpha)$ parameterized by parameters ψ to approximate $p_w(\alpha | \mathbf{S})$. By minimizing the KL divergence between the approximator $q_\psi(\alpha)$ and the real posterior distribution $p_w(\alpha | \mathbf{S})$, we obtain

the objective function

$$\begin{aligned}\mathcal{L}(\psi, w; \mathbf{S}) &= \int q_\psi(\alpha) \log \frac{q_\psi(\alpha)}{p_w(\alpha|\mathbf{S})} d\alpha \\ &= -\int q_\psi(\alpha) \log \frac{p_w(\mathbf{S}|\alpha)p(\alpha)}{q_\psi(\alpha)} d\alpha + \log p(\mathbf{S}) \\ &= -\int q_\psi(\alpha) \log p_w(\mathbf{S}|\alpha) d\alpha + KL(q_\psi(\alpha)|p(\alpha)) + \text{const} \\ &= -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \int q_\psi(\alpha) \log p_w(y_{ij}|x_{ij}, S_i^{tr}, \alpha) d\alpha + KL(q_\psi(\alpha)|p(\alpha)) + \text{const}. \end{aligned} \tag{4.4}$$

We want to optimize $\mathcal{L}(\psi, w; \mathbf{S})$ w.r.t. both the model parameters w and the variational parameters ψ. The gradient and the optimization procedure of the model parameters w are similar to the original metric-based meta-algorithms [108, 100] as shown in Algorithm 1.

To derive the gradients of the variational parameters, we leverage the re-parameterization trick proposed by [56] to derive a practical estimator of the variational lower bound and its derivatives w.r.t. the variational parameters. In this chapter, we use this trick to estimate the derivatives of $\mathcal{L}(\psi, w; \mathbf{S})$ w.r.t. ψ. For a distribution $q_\psi(\alpha)$, we can re-parameterize $\alpha \sim q_\psi(\alpha)$ using a differentiable transformation $\alpha = g_\psi(\epsilon)$, if exists, of an auxiliary random variable ϵ. For example, given a Gaussian distribution $q_{\mu,\sigma}(\alpha) = \mathcal{N}(\mu, \sigma^2)$, the re-parameterization is $g_{\mu,\sigma}(\epsilon) = \epsilon\sigma + \mu$, where $\epsilon \sim \mathcal{N}(0, 1)$. Hence, the first term in (4.4) is formulated as $-\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \mathbb{E}_{\epsilon \sim p(\epsilon)} \log p_w(y_{ij}|x_{ij}, S_i^{tr}, g_\psi(\epsilon))$.

We apply a Monte Carlo integration with a single sample $\alpha_i = g_\psi(\epsilon_i)$ for each task to get an unbiased estimator. Note that α_i is sampled for the task S_i rather than for each instance, i.e., $\{z_{ij}\}_{j=1}^{m+q}$ share the same α_i. The second term in (4.4) can be computed with a given prior distribution $p(\alpha)$. Then, the final objective function is

$$\mathcal{L}(\psi, w; \mathbf{S}) = -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \log p_w(y_{ij}|x_{ij}, S_i^{tr}, g_\psi(\epsilon_i)) + KL(q_\psi(\alpha)|p(\alpha)) \tag{4.5}$$

Estimation of gradients. The objective function (4.5) is a general form. Here, we consider $q_\psi(\alpha)$ as a Gaussian distribution $q_{\mu,\sigma}(\alpha) = \mathcal{N}(\mu, \sigma^2)$. The prior distribution is also a Gaussian distribution $p(\alpha) = \mathcal{N}(\mu_0, \sigma_0^2)$. By the fact that the KL divergence of two Gaussian

distributions has a closed-form solution, we obtain the following objective function

$$\mathcal{L}(\mu, \sigma, w; \mathbf{S}) = -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \log p_w(y_{ij}|x_{ij}, S_i^{tr}, g_{\mu,\sigma}(\epsilon_i)) + \log \frac{\sigma_0}{\sigma} + \frac{\sigma^2 + (\mu - \mu_0)^2}{2\sigma_0^2}, \tag{4.6}$$

where $g_{\mu,\sigma}(\epsilon_i) = \sigma\epsilon_i + \mu$. The derivatives of $\mathcal{L}(\mu, \sigma, w; \mathbf{S})$ w.r.t. μ and σ respectively are

$$\frac{\partial \mathcal{L}(\mu, \sigma, w; \mathbf{S})}{\partial \mu} = -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \frac{\partial \log p_w(y_{ij}|x_{ij}, S_i^{tr}, g_{\mu,\sigma}(\epsilon_i))}{\partial g_{\mu,\sigma}(\epsilon_i)} + \frac{\mu - \mu_0}{\sigma_0^2} \ , \tag{4.7}$$

$$\frac{\partial \mathcal{L}(\mu, \sigma, w; \mathbf{S})}{\partial \sigma} = -\sum_{i=1}^{n} \sum_{j=m+1}^{m+q} \frac{\partial \log p_w(y_{ij}|x_{ij}, S_i^{tr}, g_{\mu,\sigma}(\epsilon_i))}{\partial g_{\mu,\sigma}(\epsilon_i)} * \epsilon_i - \frac{1}{\sigma} + \frac{\sigma}{\sigma_0^2} \ . \tag{4.8}$$

In particular, we apply the proposed variational metric scaling method to Prototypical Networks with feature extractor ϕ_w. The details of the gradients and the iterative update procedure are shown in Algorithm 1. It can be seen that the gradients of the variational parameters are computed using the intermediate quantities in the computational graph of the model parameters w during back-propagation, hence the computational cost is very low. For meta-testing, we use μ (mean) as the metric scaling parameter for inference.

Algorithm 1 Stochastic Variational Scaling for Prototypical Networks

Input: Meta-sample $\{S_i\}_{i=1}^n$, learning rates l_w, l_ψ and μ_0, σ_0.
Random initialize μ, σ and w.
for i in $\{1, 2, \ldots, n\}$ **do**
  $\epsilon_i \sim \mathcal{N}(0, 1), \alpha_i = \sigma\epsilon_i + \mu$ // Sample α_i for i^{th} task.
  for k in $\{1, 2, \ldots, K\}$ **do**
    $\mathbf{c}_k = \frac{1}{N} \sum_{z_{ij} \in S_i^{tr}, y_{ij}=k} \phi_w(x_{ij})$ // Compute prototypes.
    for j in $\{m+1, 2, \ldots, m+q\}$ **do**
      $d(x_{ij}, \mathbf{c}_k) = \|\phi_w(x_{ij}) - \mathbf{c}_k\|_2^2$
      $p(y_{ij} = k) = \frac{e^{-\alpha_i * d(x_{ij}, \mathbf{c}_k)}}{\sum_{k'=1}^{K} e^{-\alpha_i * d(x_{ij}, \mathbf{c}_{k'})}}$
    end for
  end for
  $w = w - l_w * \nabla_w \mathcal{L}(\mu, \sigma, w; S_i)$ // Update the model parameters w.
  $\mu = \mu - l_\psi * (\sum_{j=m+1}^{m+q} (-d(x_{ij}, \mathbf{c}_{y_{ij}}) + \sum_{k'=1}^{K} p(y_{ij} = k') * d(x_{ij}, \mathbf{c}_{k'})) + \frac{\mu - \mu_0}{\sigma_0^2})$
  $\sigma = \sigma - l_\psi * (\sum_{j=m+1}^{m+q} \epsilon_i * (-d(x_{ij}, \mathbf{c}_{y_{ij}}) + \sum_{k'=1}^{K} p(y_{ij} = k') * d(x_{ij}, \mathbf{c}_{k'})) - \frac{1}{\sigma} + \frac{\sigma}{\sigma_0^2})$
   // Update the variational parameters $\psi = \{\mu, \sigma\}$.
end for

As mentioned, the proposed variational scaling framework is general. Note that training the scaling parameter α together with the model parameters [91] is a special case of our framework,

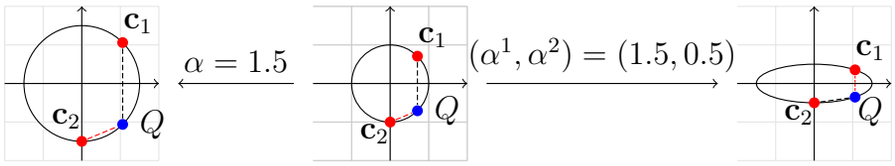


FIGURE 4.1: The middle figure shows a metric space in which the query (blue) and the support samples (red) are normalized to a unit ball. The left and right figures show the spaces scaled by a single parameter $\alpha = 1.5$ and a two-dimensional vector $(\alpha^1, \alpha^2) = (1.5, 0.5)$, respectively. The query Q is still assigned to class 2 in the left figure but to class 1 in the right one.

when $q_\psi(\alpha)$ is defined as $\mathcal{N}(\mu, 0)$, the variance of the prior distribution is $\sigma_0 \to \infty$, and the learning rate is fixed as $l_w = l_\psi$.

4.3.2 Dimensional Stochastic Variational Scaling

Metric scaling can be seen as a transformation of the metric (embedding) space. Multiplying the distances with the scaling parameter accounts to re-scaling the embedding space. By this point of view, we generalize the single scaling parameter to a dimensional scaling vector which transforms the embedding space to fit the data.

If the dimension of the embedding space is too low, the data points cannot be projected to a linearly-separable space. Conversely, if the dimension is too high, there may have many redundant dimensions. The optimal number of dimensions is data-dependent and difficult to be selected as a hyperparameter before training. Here, we address this problem by learning a data-dependent dimensional scaling vector to modify the embedding space, i.e., learning different weights for each dimension to highlight the important dimensions and reduce the influence of the redundant ones. Figure 4.1 shows a two-dimensional example. It can be seen that the single scaling parameter α simply changes the scale of the embedding space, but the dimensional scaling $\alpha = (\alpha^1, \alpha^2)$ changes the relative locations of the query and the supports.

The proposed dimensional stochastic variational scaling method is similar to Algorithm 1, with the variational parameters $\mu = (\mu^1, \mu^2, \ldots, \mu^M)$ and $\sigma = (\sigma^1, \sigma^2, \ldots, \sigma^M)$. Accordingly, the metric scaling operation is changed to

$$d(x_{ij}, \mathbf{c}_k) = (\phi_w(x_{ij}) - \mathbf{c}_k)^T \begin{pmatrix} \alpha_i^1 & & \\ & \alpha_i^2 & \\ & & \ddots \\ & & & \alpha_i^M \end{pmatrix} (\phi_w(x_{ij}) - \mathbf{c}_k). \tag{4.9}$$

The gradients of the variational parameters are still easy to compute and the computational

cost can be ignored.

4.3.3 Amortized Variational Scaling

The proposed stochastic variational scaling methods above consider the metric scale as a global scalar or vector parameter, i.e., the entire meta-sample $\mathbf{S} = \{S_i\}_{i=1}^n$ shares the same embedding space. However, the tasks randomly sampled from the task distribution may have specific task-relevant feature representations [52, 61, 67]. To adapt the learned embeddings to the task-specific representations, we propose to apply amortized variational inference to learn the task-dependent dimensional scaling parameters.

For amortized variational inference, α is a local latent variable dependent on S instead of a global parameter. Similar to stochastic variational scaling, we apply the variational distribution $q_{\psi(\beta)}(\alpha|S)$ to approximate the posterior distribution $p_w(\alpha|S)$. In order to learn the dependence between α and S, amortized variational scaling learns a mapping approximated by a neural network G_β, from the task S_i to the distribution parameters $\{\mu_i, \sigma_i\}$ of α_i.

By leveraging the re-parameterization trick, we obtain the objective function of amortized variational scaling:

$$\mathcal{L}(\beta, w; \mathbf{S}) = -\sum_{i=1}^{n}\sum_{j=m+1}^{m+q} \log p_w(y_{ij}|x_{ij}, S_i^{tr}, g_{\mu_i,\sigma_i}(\epsilon_i)) + \log \frac{\sigma_0}{{\sigma_i}^2} + \frac{{\sigma_i}^2 + (\mu_i - \mu_0)^2}{2\sigma_0^2}, \quad (4.10)$$

where $g_{\mu_i,\sigma_i}(\epsilon_i) = \sigma_i\epsilon_i + \mu_i$. Note that the local parameters $\{\mu_i, \sigma_i\}$ are functions of β, i.e., $\{\mu_i, \sigma_i\} = G_\beta(S_i)$. We iteratively update β and w by minimizing the loss function (4.10) during meta-training. During meta-testing, for each task, the generator produces a variational distribution's parameters and we still use the mean vector as the metric scaling vector for inference.

Auxiliary loss. To learn the mapping G_β from a set S_i to the variational parameters of the local random variable α_i, we compute the mean vector of the embedded queries and the embedded supports as the task prototype to generate the variational parameters. A problem is that the embeddings are not ready to generate good scaling parameters during early epochs. Existing approaches including co-training [84] and pre-training [67] can alleviate this problem at the expense of computational efficiency. They pre-train or co-train an auxiliary

Algorithm 2 Dimensional Amortized Variational Scaling for Prototypical Networks

Input: Meta-sample $\{S_i\}_{i=1}^n$, learning rates l_w, l_β, prior μ_0, σ_0 and step size l_λ.
Randomly initialize β and w, $\lambda = 1$.
for i in $\{1, 2, \ldots, n\}$ **do**
  $\mathbf{C}_i = \frac{1}{m+q} \sum_{j=1}^{m+q} \phi_w(x_{ij})$ // Compute the task prototype.
  $\mu_i, \sigma_i = G_\beta(\mathbf{C}_i)$, $\epsilon_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\alpha_i = \sigma_i \odot \epsilon_i + \mu_i$ //Generate μ_i and σ_i for i^{th} task.
  for k in $\{1, 2, \ldots, K\}$ **do**
    $\mathbf{c}_k = \frac{1}{N} \sum_{z_{ij} \in S_i^{tr}, y_{ij}=k} \phi_w(x_{ij})$
    for j in $\{m+1, m+2, \ldots, m+q\}$ **do**
      $d(x_{ij}, \mathbf{c}_k) = (\phi_w(x_{ij}) - \mathbf{c}_k)^T \begin{pmatrix} \alpha_i^1 & & \\ & \alpha_i^2 & \\ & & \ddots \\ & & & \alpha_i^M \end{pmatrix} (\phi_w(x_{ij}) - \mathbf{c}_k)$.
    end for
  end for
  $w = w - l_w * \nabla_w \mathcal{L}_\lambda(\beta, w; S_i)$ //Update the model parameters w.
  $\beta = \beta - l_\beta * \nabla_\beta \mathcal{L}_\lambda(\beta, w; S_i)$ //Update the parameters β of the generator.

  if $\lambda \neq 0$ **then** $\lambda = \lambda - l_\lambda$
end for

supervised learning classifier in a traditional supervised manner over the meta-sample $\mathbf{S}$, and then apply the pre-trained embeddings to generate the task-specific parameters and fine-tune the embeddings during meta-training. Here, we propose an end-to-end algorithm which can improve training efficiency in comparison with pre-training or co-training. We optimize the following loss function (4.11) where an auxiliary weight λ is used instead of minimizing (4.10) in Algorithm 2 , i.e.,

$$\mathcal{L}_\lambda(\beta, w; \mathbf{S}) = (1 - \lambda)\mathcal{L}(\beta, w; \mathbf{S}) + \lambda \mathcal{L}(w; \mathbf{S}), \tag{4.11}$$

where $\mathcal{L}(w; \mathbf{S}) = -\sum_{i=1}^n \sum_{j=m+1}^{m+q} \log p_w(y_{ij}|x_{ij}, \mathbf{1})$, i.e., no scaling is used. Given a decay step size γ, λ starts from 1 and linearly decays to 0 as the number of epochs increases, i.e., $\lambda = \lambda - 1/\gamma$. During the first epochs, the weight of the gradients $\frac{\partial \mathcal{L}(w;\mathbf{S})}{\partial w}$ is high and the algorithm learns the embeddings of PN. As the training proceeds, β is updated to tune the learned embedding space. See the details in Algorithm 2.

4.4 Experiments

To evaluate our methods, we plug them into two popular algorithms, prototypical networks (PN) [100] and TADAM [84], implemented by both Conv-4 and ResNet-12 backbone networks.

To be elaborated later, Table 4.1 shows our main results in comparison to state-of-the-art meta-algorithms, where it can be seen that our dimensional stochastic variational scaling algorithm outperforms other methods substantially. For TADAM, we incorporate our methods into TADAM in conjunction with all the techniques proposed in their paper and still observe notable improvement.

Table 4.1: Test accuracies of 5-way classification tasks on *mini*ImageNet using Conv-4 and ResNet-12 respectively. * indicates results by our re-implementation.

***mini*ImageNet test accuracy**			
Backbones	**Model**	5-way 1-shot	5-way 5-shot
Conv-4	Matching networks [108]	43.56 ± 0.84	55.31 ± 0.73
	Relation Net [102]	50.44 ± 0.82	65.32 ± 0.70
	Meta-learner LSTM [93]	43.44 ± 0.77	60.60 ± 0.71
	MAML [30]	48.70 ± 1.84	63.11 ± 0.92
	LLAMA [44]	49.40 ± 1.83	–
	REPTILE [81]	49.97 ± 0.32	65.99 ± 0.58
	PLATIPUS [32]	50.13 ± 1.86	–
ResNet-12	adaResNet [79]	56.88 ± 0.62	71.94 ± 0.57
	SNAIL [75]	55.71 ± 0.99	68.88 ± 0.92
	TADAM [84]	58.50 ± 0.30	76.70 ± 0.30
	TADAM Euclidean + D-SVS (**ours**)	$\mathbf{60.16 \pm 0.47}$	$\mathbf{77.25 \pm 0.15}$
	PN Euclidean [100] *	53.89 ± 0.38	73.59 ± 0.48
	PN Cosine [100] *	52.31 ± 0.83	70.74 ± 0.24
	PN Euclidean + D-SVS (**ours**) *	$\mathbf{55.30 \pm 0.08}$	$\mathbf{74.93 \pm 0.31}$
	PN cosine + D-SVS (**ours**) *	$\mathbf{56.09 \pm 0.19}$	$\mathbf{74.46 \pm 0.17}$

4.4.1 Dataset and Experimental Setup

Dataset. The *mini*ImageNet [108] consists of 100 classes with 600 images per class. We follow the data split suggested by [93], where the dataset is separated into a training set with 64 classes, a testing set with 20 classes and a validation set with 16 classes.

Table 4.2: Results of prototypical networks (the first row) and prototypical networks with SVS, D-SVS and D-AVS respectively by our re-implementation using Conv-4.

	5-way 1-shot		5-way 5-shot	
	Euclidean	Cosine	Euclidean	Cosine
PN	44.15 ± 0.39	42.20 ± 0.66	65.49 ± 0.53	60.91 ± 0.50
PN + SVS	47.84 ± 0.16	48.43 ± 0.20	66.86 ± 0.06	67.02 ± 0.14
PN + D-SVS	49.01 ± 0.39	49.20 ± 0.05	67.40 ± 0.32	67.33 ± 0.23
PN + D-AVS	$\mathbf{49.10 \pm 0.14}$	$\mathbf{49.34 \pm 0.29}$	$\mathbf{68.04 \pm 0.16}$	$\mathbf{67.83 \pm 0.16}$

Model architecture. To evaluate our methods with different backbone networks, we re-implement PN with the Conv-4 architecture proposed by [100] and the ResNet-12 architecture adopted by [84], respectively. The Conv-4 backbone contains four convolutional blocks, where each block is sequentially composed of a 3×3 kernel convolution with 64 filters, a batch normalization layer, a ReLU nonlinear layer and a 2×2 max-pooling layer. The ResNet-12 architecture contains 4 Res blocks, where each block consists of 3 convolutional blocks followed by a 2×2 max-pooling layer.

Training details. We follow the episodic training strategy proposed in [108]. In each episode, K classes and N shots per class are selected from the training set, the validation set or the test set. For fair comparisons, the number of queries, the sampling strategy of queries, and the testing strategy are designed in line with PN or TADAM. For Conv-4, we use Adam optimizer with a learning rate of $1e-3$ without weight decay. The total number of training episodes is $20,000$ for Conv-4. For ResNet-12, we use SGD optimizer with momentum 0.9, weight decay $4e-4$ and $45,000$ episodes in total. The learning rate is initialized as 0.1 and decayed 90% at episode steps 15000, 30000 and 35000. Besides, we use gradient clipping when training ResNet-12. The reported results are the mean accuracies with 95% confidence intervals estimated by 5 runs.

We normalize the embeddings before computing the distances between them. As shown in Eq. (4.7) and (4.8), the gradient magnitude of variational metric scaling parameters is proportional to the norm of embeddings. Therefore, to foster the learning process of these

parameters, we adopt a separate learning rate l_ψ for all variational metric scaling parameters. We adopt the following sampling strategy for the proposed three approaches. For meta-training, we sample once per task from the variational distribution for the metric scaling parameter. For meta-test, we use the mean of the learned Gaussian distribution as the metric scaling parameter. The computational overhead is very small and can be ignored.

4.4.2 Evaluation

The effectiveness of our proposed methods is illustrated in Table 4.2 progressively, including stochastic variational scaling (SVS), dimensional stochastic variational scaling (D-SVS) and dimensional amortized variational scaling (D-AVS). On both 5-way 5-shot and 5-way 1-shot classification, noticeable improvement can be seen after incorporating SVS into PN. Compared to SVS, D-SVS is more effective, especially for 5-way 1-shot classification. D-AVS performs even better than D-SVS by considering task-relevant information.

Performance of SVS. We study the performance of SVS by incorporating it into PN. We consider both 5-way and 20-way training scenarios. The prior distribution of the metric scaling parameter is set as $p(\alpha) = \mathcal{N}(1, 1)$ and the variational parameters are initialized as $\mu_{init} = 100$, $\sigma_{init} = 0.2$. The learning rate is set to be $l_\psi = 1e - 4$.

Results in Table 4.3 show the effect of the metric scaling parameter (SVS). Particularly, significant improvement is observed for the case of PN with cosine similarity and for the case of 5-way 1-shot classification. Moreover, it can be seen that with metric scaling there is no clear difference between the performance of Euclidean distance and cosine similarity.

We also compare the performance of a fixed $\sigma = 0.2$ with a trainable σ. We add a shifted ReLU activation function ($x = \max\{1e - 2, x\}$) on the learned σ to ensure it being positive. Nevertheless, in our experiments, we observe that the training is very stable and the variance is always positive even without the ReLU activation function. We also find that there is no significant difference between the two settings. Hence, we treat σ as a fixed hyperparameter in other experiments.

Performance of D-SVS. We validate the effectiveness of D-SVS by incorporating it into PN and TADAM, with the results shown in Table 4.1 and Table 4.2. On 5-way-1-shot

Table 4.3: Results of prototypical networks and prototypical networks with SVS by our re-implementation using Conv-4.

	5-way 1-shot		5-way 5-shot	
	5-way training	20-way training	5-way training	20-way training
PN Euclidean	44.15 ± 0.39	48.05 ± 0.47	65.49 ± 0.53	67.32 ± 1.20
PN Cosine	42.20 ± 0.66	46.75 ± 0.18	60.91 ± 0.50	66.28 ± 0.14
PN Euclidean + SVS ($\sigma = 0.2$)	47.84 ± 0.16	51.15 ± 0.16	66.86 ± 0.06	68.00 ± 0.22
PN Cosine + SVS ($\sigma = 0.2$)	48.12 ± 0.13	51.74 ± 0.13	66.95 ± 0.78	67.88 ± 0.10
PN Euclidean + SVS (learned σ)	48.28 ± 0.14	51.36 ± 0.15	66.84 ± 0.30	67.80 ± 0.06
PN Cosine + SVS (learned σ)	48.43 ± 0.20	51.68 ± 0.18	67.02 ± 0.14	67.72 ± 0.16

Table 4.4: Ablation study of prototypical networks with D-AVS by our re-implementation using Conv-4.

		5-way 1-shot		5-way 5-shot	
Auxiliary training	Prior	Euclidean	Cosine	Euclidean	Cosine
		47.79 ± 0.10	47.45 ± 0.17	66.26 ± 0.48	66.03 ± 0.34
	✓	48.12 ± 0.55	47.49 ± 0.26	66.69 ± 0.25	66.43 ± 0.38
✓		48.56 ± 0.44	49.13 ± 0.32	67.11 ± 0.14	67.23 ± 0.19
✓	✓	$\mathbf{49.10 \pm 0.14}$	$\mathbf{49.34 \pm 0.29}$	$\mathbf{68.04 \pm 0.16}$	$\mathbf{67.83 \pm 0.16}$

classification, for PN, we observe about 4.90% and 1.41% absolute increase in test accuracy with Conv-4 and ResNet-12 respectively; for TADAM, 1.66% absolute increase in test accuracy is observed. The learning rate for D-SVS is set to be $l_\psi = 16$. Here, we use a large learning rate since the gradient magnitude of each dimension of the metric scaling vector is extremely small after normalizing the embeddings.

Figure 4.4 illustrates the distributions of the mean vector $\mu = (\mu^1, \mu^2, \ldots, \mu^M)$ during the meta-training procedure of 5-way 1-shot and 5-way 5-shot classification respectively. Darker colour means more frequent occurrence.

At step 0, all dimensions of μ are initialized as 100. They diverge as the meta-training proceeds, which shows that D-SVS successfully learns different scaling parameters for different

dimensions. It is also worth noting that for both tasks, the distribution of μ converges eventually (after $15k$ steps).

Performance of D-AVS. We evaluate the effectiveness of D-AVS by incorporating it into PN. We use a multi-layer perception (MLP) with one hidden layer as the generator G_β. The learning rate l_β is set to be $1e-3$. In Table 4.2, on both 5-way 1-shot and 5-way 5-shot classification, we observe about 1.0% absolute increase in test accuracy for dimensional amortized variational scaling (D-AVS) over SVS with a single scaling parameter. In our experiments, the hyperparameter γ is selected from the range of $[100, 150]$ with 200 training epochs in total.

4.4.3 Ablation study.

Ablation study of SVS: Comparison with a Special Case. Reference [91] proposed to train the single scaling parameter together with model parameters. Their method can be seen as a special case of our stochastic variational scaling method SVS under the conditions of $q_\psi(\alpha) = \mathcal{N}(\mu, 0)$, $\sigma_0 \to \infty$ and $l_\psi = l_w$. We compare our method with theirs by varying the initialization of μ (μ_{init}).

Noticeably, our method achieves absolute improvements of 2.86%, 1.77%, 0.73% and 2.5% for four different initializations respectively. As shown in Table 4.5, our method is stable w.r.t. the initialization of μ, which should be attributed to the prior information introduced in our Bayesian framework that may counteract the influence of initialization.

Table 4.5: Comparison of PN (Training together) and PN+SVS implemented by Conv-4 backbone.

Method \ μ_{init}	1	10	100	1000
PN+SVS	66.45	66.95	67.02	66.72
PN (Training together)	63.59	65.18	66.29	64.22

Ablation study of D-AVS. To assess the effects of the auxiliary training strategy and the prior information, we provide an ablation study as shown in Table 4.4. Without the

auxiliary training and the prior information, D-AVS degenerates to a task-relevant weight generating approach [61]. Noticeable performance drops can be observed after removing the two components. Removing either one of them also leads to performance drop, but not as significant as removing both. The empirical results confirm the necessity of the auxiliary training and a proper prior distribution for amortized variational metric scaling.

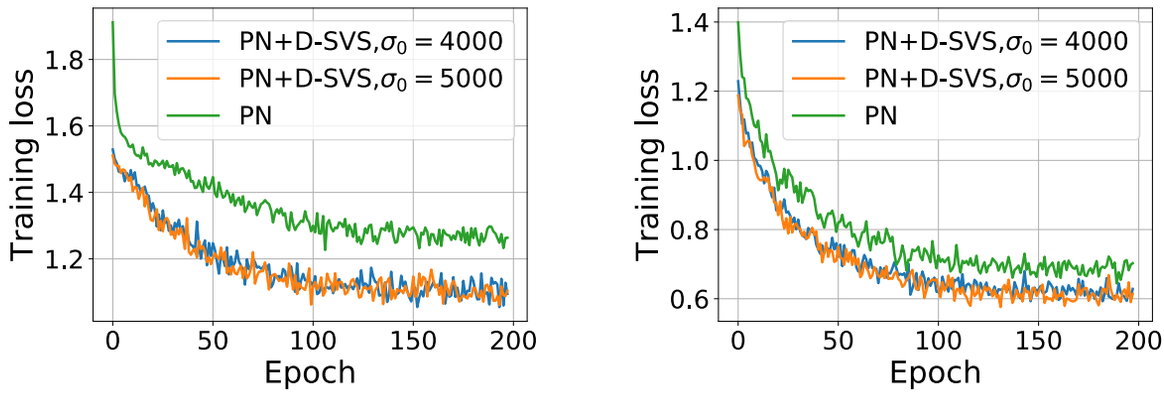
4.4.4 Robustness Study

We also design experiments to show: 1) The convergence speed of existing methods does not slow down after incorporating our methods; 2) Given the same prior distribution, the variational parameters converge to the same values in spite of different learning rates and initializations; 3) The variational metric scaling framework is not sensitive to the prior and initialization.

For the iterative update of the model parameters w and the variational parameters ψ, a natural question is whether it will slow down the convergence speed of the algorithm. Figure 4.2 shows the learning curves of PN and PN+D-SVS on both 5-way 1-shot and 5-way 5-shot classification. It can be seen that the incorporation of SVS does not reduce the convergence speed.

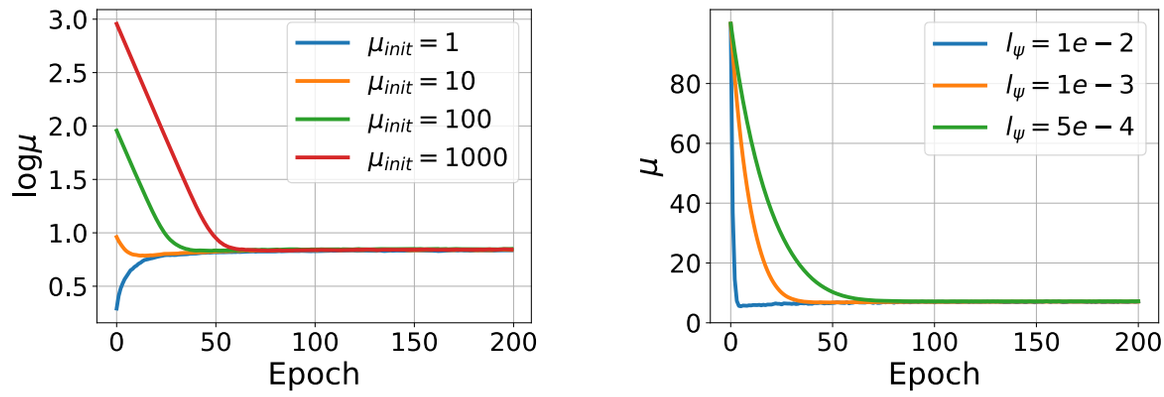
We plot the learning curves of the variational parameter μ w.r.t. different initializations and different learning rates l_ψ. Given the same prior distribution $\mu_0 = 1$, Fig. 4.3(a) shows that the variational parameter μ with different initializations will converge to the same value. Fig. 4.3(b) shows that μ is robust to different learning rates.

In Bayesian framework, the prior distribution has a significant impact on learning posterior distribution. For stochastic variational inference, initialization is another key factor for learning the variational parameters. Here, we conduct experiments of PN+SVS with different prior distributions and initializations. The results of 5-way 5-shot classification are summarized in Table 4.6. It can be observed that our method is not sensitive to the prior and initialization as long as either one of them is not too small.



(a) 5-way 1-shot (b) 5-way 5-shot

FIGURE 4.2: Learning curves of prototypical networks and prototypical networks with D-SVS.



(a) $\mu_0 = 1, l_\psi = 1e-3$ (b) $\mu_0 = 1, \mu_{init} = 100$

FIGURE 4.3: Learning curves of μ (a) for different initializations and (b) for different learning rates.

4.5 Related Work

Metric-based meta-learning. Reference [58] proposed the first metric-based meta-algorithm for few-shot learning, in which a siamese network [17] is trained with the triplet loss to compare the similarity between a query and supports in the embedding space. Matching networks [108] proposed the episodic training strategy and used the cross-entropy loss where the logits are the distances between a query and supports. Prototypical networks [100] improved Matching networks by computing the distances between a query and the prototype (mean of supports) of each class. Many metric-based meta-algorithms [84, 33, 102, 67] extended prototypical networks in different ways.

Table 4.6: Results of PN+SVS w.r.t. different initializations and priors implemented by Conv-4.

μ_0 \ μ_{init}	1	10	100	1000
1	60.25 ± 0.70	63.00 ± 0.34	66.86 ± 0.06	66.26 ± 0.24
10	63.95 ± 0.24	65.89 ± 0.32	66.79 ± 0.55	66.34 ± 0.22
100	65.94 ± 0.42	66.95 ± 0.27	67.02 ± 0.38	66.43 ± 0.10
1000	66.45 ± 0.17	66.66 ± 0.23	66.88 ± 0.16	66.72 ± 0.28

Some recent methods proposed to improve prototypical networks by extracting task-conditioning features. Reference [84] trained a network to generate task-conditioning parameters for batch normalization. Reference [67] extracted task-relevant features with a category traversal module. Our methods can be incorporated into these methods to improve their performance.

In addition, there are some works related to our proposed dimensional scaling methods. Reference [52] trained a meta-model to re-weight features obtained from the base feature extractor and applied it for few-shot object detection. Reference [61] proposed a generator to generate task-adaptive weights to re-weight the embeddings, which can be seen as a special case of our amortized variational scaling method.

Metric scaling. Cross-entropy loss is widely used in many machine learning problems, including metric-based meta-learning and metric learning [4, 91, 68, 111, 118, 4, 110]. In metric learning, the influence of metric scaling on the cross-entropy loss was first studied in [111] and [91]. They treated the metric scaling parameter as a trainable parameter updated with model parameters or a fixed hyperparameter. Reference [118] proposed a "heating-up" scaling strategy, where the metric scaling parameter decays manually during the training process. The scaling of logits in cross-entropy loss for model compression was also studied in [50], where it is called temperature scaling. The temperature scaling parameter has also been used in confidence calibration [45].

The effect of metric scaling for few-shot learning was first discussed in [100] and [84]. The former found that Euclidean distance outperforms cosine similarity significantly in prototypical networks, and the latter argued that the superiority of Euclidean distance could be offset by

imposing a proper metric scaling parameter on cosine similarity and using cross validation to select the parameter.

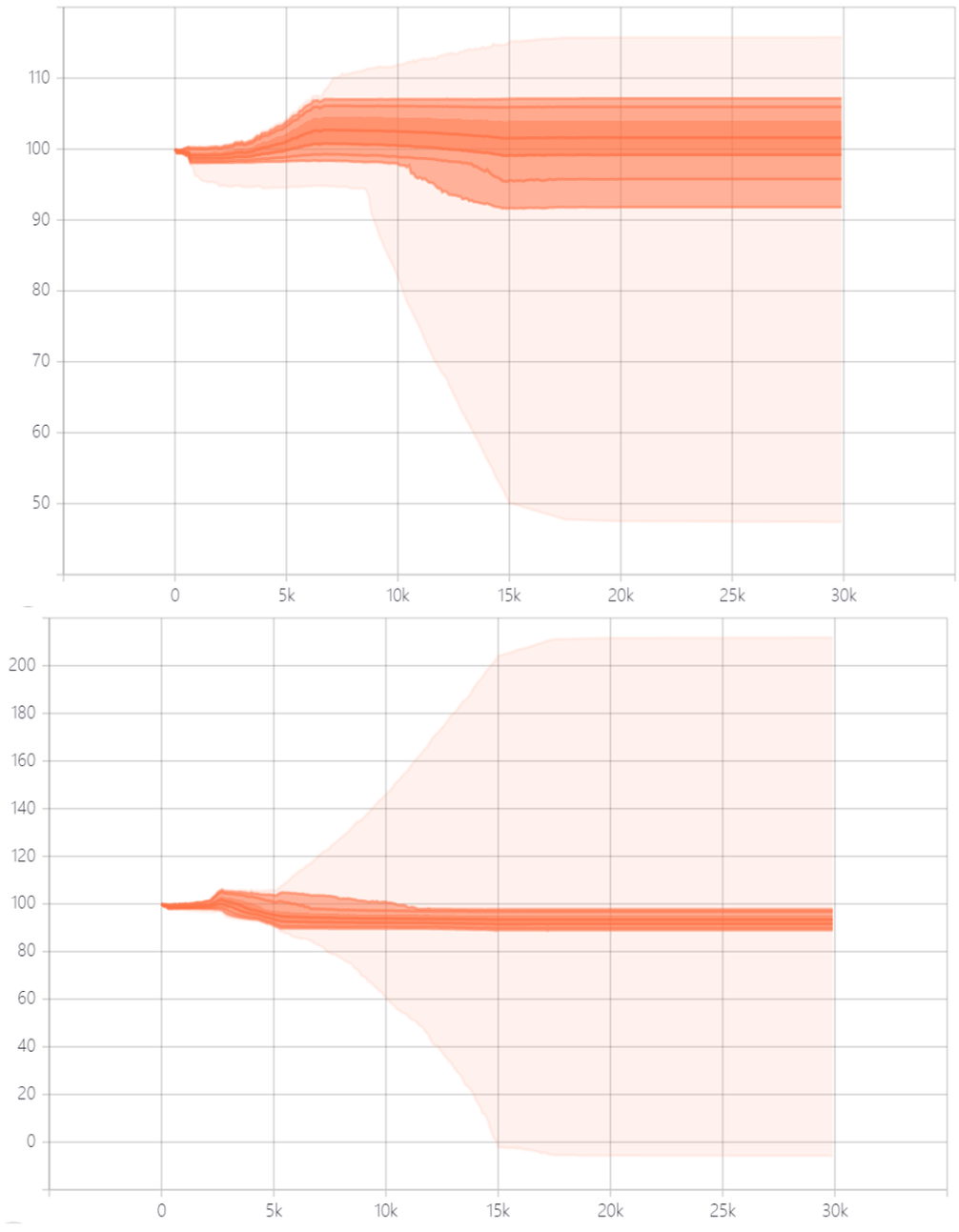


FIGURE 4.4: Distributions of the learned μ w.r.t. the number of training steps. The horizontal and vertical axes are the number of training steps and values of μ, respectively. The top is for 5-way 1-shot classification and the bottom is for 5-way 5-shot.

5

Cross-Domain Meta-Learning via Meta^2 Learning

5.1 Introduction

Meta learning has emerged as a dominating approach to solve few-shot learning tasks and shown encouraging performance. In practice, there usually exists a discrepancy between the test environment and the training ones. Especially, when annotated data in the target domain is difficult to acquire, it is crucial to leverage labeled data in the training domains for knowledge transfer. For example, the target task is to classify rare types of Fungi, where labeled images are scarce and hard to collect, but we may have access to ample training data in the domains of Birds and Flowers. The problem of generalizing to tasks in a new *domain* (task distribution) with unknown statistics [103], is far from being solved [16, 46].

To tackle this problem, a reasonable assumption is that there is a *domain-shared structure* that enables *cross-domain generalization*, and meanwhile each domain has its *domain-specific features*, which needs to be captured for *inner-domain learning*. [104] adopted this assumption and proposed to learn an embedding function to capture the general structure and a global feature-wise transformation to simulate various feature distributions within different domains for cross-domain few-shot classification. However, the global transformation may not be sufficient to capture the distinct features of various domains.

In this paper, we propose a hierarchical gradient-based meta learning framework, which

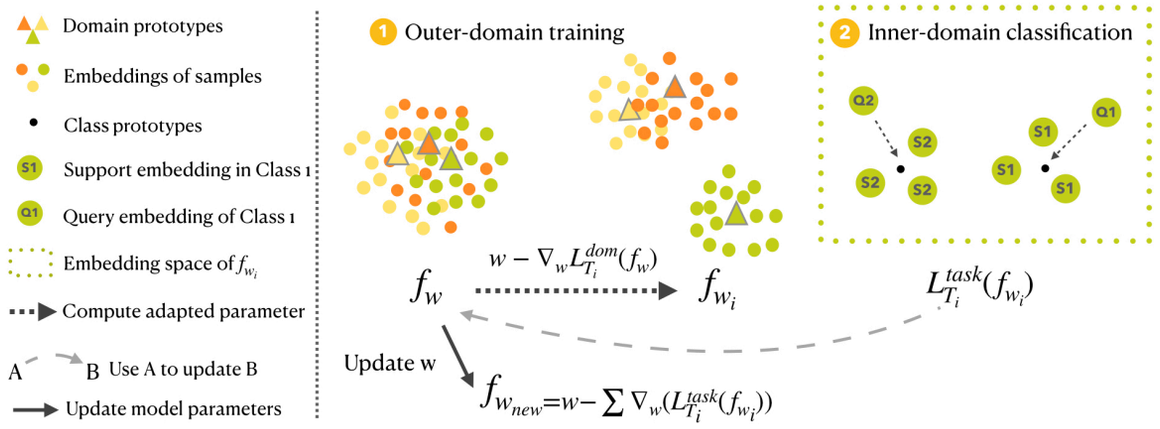


FIGURE 5.1: Metric-based M^{2}L for cross-domain few-shot classification.

aims to learn a domain-shared structure that can quickly adapt to a specific domain and a specific task progressively. In particular, at outer-domain level, the general feature extractor f_w is adapted to a *domain-specific* f_{w_i} by learning to distinguish samples of the current domain from those of other domains in the common embedding space (f_w). Within each domain, *standard few-shot learning* is performed in the embedding space (f_{w_i}) of each training domain i. The general model parameters w are then updated by simultaneously optimizing the loss of the query set in each domain, which capture the *domain-shared* structure for rapid generalization to new test domains. Meta learning can be performed in both outer-domain training and inner-domain few-shot learning, leading to a hierarchical meta learning framework, hence we name it meta^2 learning (M^{2}L).

M^{2}L is a general framework that can integrate any meta learning algorithm to solve cross-domain few-shot learning tasks. In this paper, we verify the effectiveness of M^{2}L by tackling the challenging cross-domain few-shot classification problem [16, 103, 104]. As M^{2}L is a high-order (>2) framework if gradient-based algorithms such as MAML [30] are used in outer-domain training or inner-domain learning, to avoid computing high-order derivatives in training, we simplify M^{2}L by employing metric-based algorithms for efficient outer-domain training and inner-domain classification, which leads to a second-order framework as illustrated in Figure 5.1. As summarized in Table 5.1, compared to other second-order methods such as MAML and ProtoMAML [103], the advantages of M^{2}L are: (1) It performs domain-level adaptation; and (2) It adopts metric-based inner-task algorithms for few-shot classification which typically

Table 5.1: Comparison of metric-based M^{2}L with other second-order methods; Right: Comparison of domain adaptation/generalization, few-shot learning and cross-domain few-shot learning. Note that the label spaces of tasks in training and testing are the same in domain adaptation/generalization but different in (cross-domain) few-shot learning.

	Adaptation to Specific Domain	Metric-Based Inner-Task Algorithm	Second-Order Optimization
MAML	×	×	✓
ProtoMAML	×	×	✓
M^{2}L (metric-based)	✓	✓	✓

generalize better [16]. Compared with standard metric-based methods such as ProtoNet [100] and MatchingNet [108], which use a same feature extractor for all domains, M^{2}L can adapt to individual domains and thus are less prone to overfitting to training domains.

M^{2}L can employ any metric-based algorithm as base model for inner-domain few-shot classification. We evaluate M^{2}L on **9** fine-grained benchmark datasets and experiment with **4** popular metric-based algorithms including MatchingNet, ProtoNet, RelationNet [102] and GNN [37] as the base model for inner-domain classification. The results show that M^{2}L can significantly improve their generalization ability to unseen domains for few-shot classification. M^{2}L also consistently outperforms MAML and ProtoMAML. The empirical study demonstrates the effectiveness of M^{2}L in learning a cross-domain common structure and efficiently adapting to a new domain.

5.2 Notations and Preliminaries

An N_w-way N_s-shot classification task is to recognize samples (*queries*) from N_w novel categories with only N_s training samples (*supports*) of each category. To overcome data limitation, meta learning algorithms learn from a bag of training tasks $\{T_t\}_{t=1}^{N_T}$ and train a base-learner which can adapt to an unseen task T sampled from novel categories. A training task T_t is comprised of a support set $\{\mathcal{X}_{T_t}^{\text{tr}}, \mathcal{Y}_{T_t}^{\text{tr}}\}$ and a query set $\{\mathcal{X}_{T_t}^{\text{ts}}, \mathcal{Y}_{T_t}^{\text{ts}}\}$ with N_q queries per category.

Cross-domain few-shot learning considers the scenario where test tasks and training tasks come from different domains (task distributions). We denote the datasets of training

domains by $\{Dom_i\}_{i=1}^{N_d}$ and the unseen test domain by Dom. For instance, the training tasks are sampled from the datasets of "fungi", "cars" and "birds", whereas the test tasks are sampled from the "flowers" dataset. For each episode in meta-training, we sample one task T_i from domain Dom_i. Note that we denote T_i instead of T_t to indicate that T_i is sampled from Dom_i.

Gradient-based meta learning algorithms such as MAML [30] learns a common initialization which can adapt to a novel task with a few gradient update steps. For each task T_t, the model parameters w are adapted to w_{T_t} by minimizing the error of the support set during *inner-task training*:

$$w_{T_t} = w - \nabla_w \mathcal{L}(\mathcal{X}_{T_t}^{\text{tr}}, \mathcal{Y}_{T_t}^{\text{tr}}; w). \tag{5.1}$$

Remark that, unless otherwise specified, *we set the step size as 1 for simplicity* in the following sections. The performance of the task-specific predictor $f_{w_{T_t}}$ is measured by the query set. As the adapted parameters w_{T_t} is a function of the model parameters w, the model parameters can be updated by minimizing the error of the query set w.r.t. w_{T_t}. The *meta update* rule is formulated as follows:

$$\begin{aligned} w &= w - \nabla_w \mathcal{L}(\mathcal{X}_{T_t}^{\text{ts}}, \mathcal{Y}_{T_t}^{\text{ts}}; w_{T_t}) \\ &= w - \nabla_{w_{T_t}} \mathcal{L}(\mathcal{X}_{T_t}^{\text{ts}}, \mathcal{Y}_{T_t}^{\text{ts}}; w_{T_t}) \cdot \nabla_w w_{T_t} \\ &= w - \nabla_{w_{T_t}} \mathcal{L}(\mathcal{X}_{T_t}^{\text{ts}}, \mathcal{Y}_{T_t}^{\text{ts}}; w_{T_t}) \cdot (\mathrm{I} - \nabla_w^2 \mathcal{L}(\mathcal{X}_{T_t}^{\text{tr}}, \mathcal{Y}_{T_t}^{\text{tr}}; w)). \end{aligned} \tag{5.2}$$

Metric-based meta learning methods learn a comparison algorithm implemented by a feature extractor f_w and a pre-defined metric function g. For each task, the embedding function maps the supports and queries into an embedding space. The metric function g is then utilized to measure the similarity between the embeddings, and each query is classified to the category of the most similar support, i.e., $\hat{\mathcal{Y}}_{T_t}^{\text{ts}} = g(f_w(\mathcal{X}_{T_t}^{\text{ts}}), f_w(\mathcal{X}_{T_t}^{\text{tr}}), \mathcal{Y}_{T_t}^{\text{tr}})$. With the predictions $\hat{\mathcal{Y}}_{T_t}^{\text{ts}}$ and the ground-truth labels $\mathcal{Y}_{T_t}^{\text{ts}}$ of the queries, meta training proceeds by minimizing the classification error, i.e.,

$$w = w - \nabla_w \mathcal{L}(\hat{\mathcal{Y}}_{T_t}^{\text{ts}}, \mathcal{Y}_{T_t}^{\text{ts}}; w), \tag{5.3}$$

where the loss function $\mathcal{L}$ can be cross-entropy loss [100, 108], mean-squared loss [102], etc. Different algorithms differ in the choice of the metric function g. For instance, ProtoNet [100]

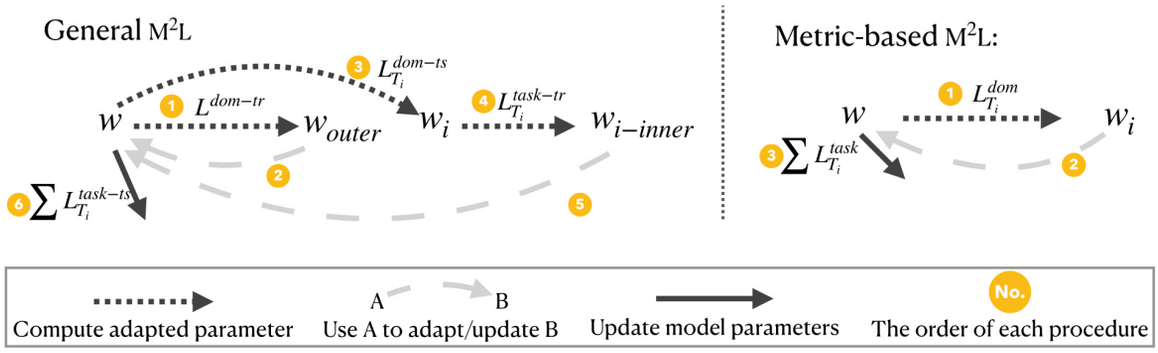


FIGURE 5.2: Parameter update in M^{2}L and metric-based M^{2}L.

uses a non-parametric Euclidean distance metric. Some other methods use a parameterized metric function, denoted by g_ϕ. For example, GNN [37] uses a graph neural network and RelationNet [102] adopts a CNN module, parameterized by ϕ to measure the similarity. In this case, ϕ can be seen as part of the model parameters and updated together with w as in Eq. (5.3).

Different from the gradient-based algorithms, the inner-task training of metric-based methods is to "memorize" the support set instead of adapting the model parameters with it. Hence, the meta update rule in Eq. (5.3) avoids the computation of second-order derivatives in gradient-based algorithms as in Eq. (5.2).

5.3 Meta^2 Learning (M^{2}L)

To extract domain-shared structures, we aim to learn a domain-shared model f_w which can adapt to a specific domain and a specific task progressively. To this end, we propose M^{2}L, a hierarchical meta-learning framework, which allows to implement meta-learning in both outer-domain training and inner-domain few-shot learning.

5.3.1 Outer-Domain Training

In the outer-domain loop, we adapt a domain-shared model f_w to a domain-specific one f_{w_i} by differentiating the training samples of domain Dom_i from those of other domains. Our key intuition is that if f_{w_i} can distinguish the training samples of the domain Dom_i from other domains, it must learn some domain-specific information of Dom_i making it different from other domains. We formulate this intuition into a domain-level classification task, i.e., N_d-way

1-shot classification problem, as will be explained below. Following [108], we adopt the episodic training strategy. In each episode, one task T_i is sampled from each domain Dom_i.

Domain Prototype. Given the embedding function of f_w, the supports and queries of each task are all mapped to a domain-shared embedding space. It is reasonable to assume that samples in each domain are close and those from different domains are distant, and a large discrepancy exists between different domains. In this sense, each domain (e.g., fine-grained datasets such as Birds, Flowers, Fungi etc.) can be represented by a domain prototype. For each domain Dom_i, we take the mean of the supports and queries of task T_i as the prototype $\mathbf{P}_w^i$ of domain Dom_i:

$$\mathbf{P}_w^i = \frac{1}{N_w \times (N_s + N_q)} \sum_{x \in \mathcal{X}_{T_i}^{\text{tr}} \bigcup \mathcal{X}_{T_i}^{\text{ts}}} f_w(x). \tag{5.4}$$

N_d-way 1-shot classification. With the domain prototypes $\{\mathbf{P}_w^i\}_{i=1}^{N_d}$, we formulate an N_d-way 1-shot classification task, where each domain Dom_i is considered a category ($i \in \{1, \ldots, N_d\}$), the domain prototype $\mathbf{P}_w^i$ is the *support* of Dom_i, and the *queries* of Dom_i are the labeled samples of task T_i (e.g., the supports of task T_i – $\mathcal{X}_{T_i}^{\text{tr}}$). The goal is to assign a query sample to the domain it belongs. By solving the N_d-way 1-shot classification task, i.e., classifying queries of Dom_i, the general model f_w can be adapted to the specific domain Dom_i.

Domain-level Adaptation. Any meta learning algorithm can be applied to solve the N_d-way 1-shot classification task. As shown in Figure 5.2, the first step is to adapt w to w_{outer} by training an N_d-way classifier with the domain prototypes:

$$\text{Step O-1: } w_{\text{outer}} = w - \nabla_w \mathcal{L}^{\text{dom-tr}}(\{\mathbf{P}_w^i\}_{i=1}^{N_d}; w), \tag{5.5}$$

where $\mathcal{L}^{\text{dom-tr}}$ denotes the training loss of separating the domain prototypes. Remark that we ignore the step size for simplicity in this section. Given the classifier $f_{w_{\text{outer}}}$, the general model can be adapted to a domain-specific model by minimizing the classification error of the queries of domain Dom_i:

$$\text{Step O-2: } w_i = w - \nabla_w \mathcal{L}_{T_i}^{\text{dom-ts}}(\mathcal{X}_{T_i}^{\text{tr}}; w_{\text{outer}}) = w - \nabla_{w_{\text{outer}}} \mathcal{L}_{T_i}^{\text{dom-ts}} \cdot (\mathrm{I} - \nabla_w^2 \mathcal{L}^{\text{dom-tr}}), \tag{5.6}$$

where $\mathcal{L}_{T_i}^{\text{dom-ts}}$ denotes the classification loss of the support samples in task T_i using the classifier trained by $\mathcal{L}^{\text{dom-tr}}$. After outer-domain training, we obtain N_d domain-specific models $\{f_{w_i}\}_{i=1}^{N_d}$ that can capture domain-specific structures.

5.3.2 Inner-Domain Few-Shot Learning

The domain-specific model f_{w_i} will be further adapted to solve the inner-domain few-shot learning task $T_i \in Dom_i$, whose support set is $\{\mathcal{X}_{T_i}^{\text{tr}}, \mathcal{Y}_{T_i}^{\text{tr}}\}$ and query set is $\{\mathcal{X}_{T_i}^{\text{ts}}, \mathcal{Y}_{T_i}^{\text{ts}}\}$. The domain-specific (but task-shared) model f_{w_i} is adapted to a task-specific model $f_{w_{i-\text{inner}}}$ using the support set:

$$\text{Step I-1: } w_{i\text{-inner}} = w_i - \nabla_{w_i} \mathcal{L}_{T_i}^{\text{task-tr}}(\mathcal{X}_{T_i}^{\text{tr}}, \mathcal{Y}_{T_i}^{\text{tr}}; w_i), \tag{5.7}$$

where $\mathcal{L}_{T_i}^{\text{task-tr}}$ refers to the training error of task T_i.

Meta Update of General M^{2}L. The task-specific model $f_{w_{i-\text{inner}}}$ can then be used to evaluate the query set of task T_i. The error of all the training tasks is then used to update the model parameters w. The goal is to obtain a general f_w which can perform well on a new task in a new domain after domain-level adaptation and task-level adaptation. The update rule is derived as:

$$\begin{aligned}
\text{Step I-2: } w &= w - \nabla_w \sum_{i=1}^{N_d} \mathcal{L}_{T_i}^{\text{task-ts}}(\mathcal{X}_{T_i}^{\text{ts}}, \mathcal{Y}_{T_i}^{\text{ts}}; w_{i\text{-inner}}), \\
&= w - \sum_{i=1}^{N_d} \nabla_{w_{i\text{-inner}}} \mathcal{L}_{T_i}^{\text{task-ts}} \cdot \nabla_{w_i} w_{i\text{-inner}} \cdot \nabla_w w_i, \\
&= w - \sum_{i=1}^{N_d} \nabla_{w_{i\text{-inner}}} \mathcal{L}_{T_i}^{\text{task-ts}} \cdot (\mathrm{I} - \nabla_{w_i}^2 \mathcal{L}_{T_i}^{\text{task-tr}}) \\
&\quad \cdot \big(\mathrm{I} - \nabla_{w_{\text{outer}}}^2 \mathcal{L}_{T_i}^{\text{dom-ts}} \cdot (\mathrm{I} - \nabla_w^2 \mathcal{L}^{\text{dom-tr}})^2 + \nabla_{w_{\text{outer}}} \mathcal{L}_{T_i}^{\text{dom-ts}} \cdot \nabla_w^3 \mathcal{L}^{\text{dom-tr}}\big),
\end{aligned} \tag{5.8}$$

where $\mathcal{L}_{T_i}^{\text{task-ts}}$ refers to the error of the task-specific model over the query set of T_i. The above update rule is general and compatible with any meta learning algorithm used in the outer-domain or inner-domain training of M^{2}L which can be applied to solve any cross-domain few-shot learning task. In the following section, we propose to apply M^{2}L in cross-domain

few-shot classification problem. As M^{2}L requires high-order (>2) gradient computation in Eq. 5.8, which is highly computationally expensive, we propose a simplied metric-based M^{2}L. As will be introduced in the next section, in metric-based M^{2}L, metric-based meta algorithms are used in both the outer-domain and inner-domain training without adapting the parameters in Step O-1 and Step I-1.

5.4 Metric-based Meta^2 Learning

In this section, as illustrated in Figure 5.2, we simplify M^{2}L by using metric-based meta algorithms in both outer-domain training and inner-domain classification to avoid the computation of high-order derivatives (Eq. (5.8)). Any metric-based method can be employed, and we take the *mean-centroid* classification algorithm used in ProtoNet [100] as an example for illustration purpose.

5.4.1 Metric-Based Outer-Domain Training

We adopt a mean-centroid classifier to solve the aforementioned N_d-way 1-shot classification task in outer-domain training. In particular, without using Step O-1 to obtain w_{outer}, w can be directly adapted to w_i for each domain Dom_i by fixing the prototypes $\{\mathbf{P}_w^j\}_{j\neq i}^{N_d}$ of other domains $\{Dom_j\}_{j\neq i}^{N_d}$ and minimizing the classification error of the supports $\mathcal{X}_{T_i}^{\text{tr}}$ w.r.t. Dom_i. The adapted parameters w_i are computed by

$$w_i = w - \alpha \nabla_w \mathcal{L}_{T_i}^{\text{dom}}(f_w), \tag{5.9}$$

where

$$\mathcal{L}_{T_i}^{\text{dom}}(f_w(\mathcal{X}_{T_i}^{\text{tr}}), \mathbf{P}_w^i, \{\mathbf{P}_w^j\}_{j\neq i}^{N_d}; w) = \frac{1}{N_s \times N_w} \sum_{x\in\mathcal{X}_{T_i}^{\text{tr}}} -\log\left(\frac{e^{-\|f_w(x)-\mathbf{P}_w^i\|_2^2}}{\sum_{1\leq j\leq N_d} e^{-\|f_w(x)-\mathbf{P}_w^j\|_2^2}}\right). \tag{5.10}$$

Note that the outer-training procedure with $\mathcal{L}^{\text{dom-tr}}$ and $\mathcal{L}_{T_i}^{\text{dom-ts}}$ is simplified to minimize $\mathcal{L}_{T_i}^{\text{dom}}$. As such, N_d domain-specific feature extractors $\{f_{w_i}\}_{i=1}^{N_d}$ can be obtained. Note that w_i is a function of w and only first-order gradient is involved in Eq. (5.9).

5.4.2 Metric-Based Inner-Domain Few-Shot Classification

Here, we use ProtoNet [100] as the inner-domain classification algorithm. It adopts a mean-centroid classifier with N_w centroids (class prototypes) $\{\mathbf{c}_{w_i}^j\}_{j=1}^{N_w}$, where $\mathbf{c}_{w_i}^j = \frac{1}{N_s}\sum_{x\in\mathcal{X}_{T_i}^{\text{tr}}, y=j} f_{w_i}(x)$ is the mean of the embedded supports of the j-th class in task T_i. A query is then classified to the category of the nearest class prototype. The classification loss of task T_i is computed over the query set:

$$\mathcal{L}_{T_i}^{\text{task}}(f_{w_i}) = -\frac{1}{N_q \times N_w}\sum_{x\in\mathcal{X}_{T_i}^{\text{ts}}} \log \frac{e^{-||f_{w_i}(x)-\mathbf{c}_{w_i}^y||^2}}{\sum_{j'}^{N_w} e^{-||f_{w_i}(x)-\mathbf{c}_{w_i}^{j'}||^2}}. \tag{5.11}$$

The classification loss $\mathcal{L}_{T_i}^{\text{task}}(f_{w_i})$ is then minimized to update the model parameters w.

Meta Update of Metric-based M^{2}L. The model parameters w are updated by optimizing the sum of the classification loss (Eq. (5.11)) of all the training tasks $\{T_i\}_{i=1}^{N_d}$, i.e.,

$$\begin{aligned} w &= w - \beta\sum_{i=1}^{N_d}\nabla_w \mathcal{L}_{T_i}^{\text{task}}(f_{w_i}) = w - \beta\sum_{i=1}^{N_d}\nabla_{w_i}\mathcal{L}_{T_i}^{\text{task}}(f_{w_i})\cdot\nabla_w w_i \\ &= w - \beta\sum_{i=1}^{N_d}\nabla_{w_i}\mathcal{L}_{T_i}^{\text{task}}(f_{w_i})\cdot(\mathrm{I} - \alpha\nabla_w^2\mathcal{L}_{T_i}^{\text{dom}}(f_w)), \end{aligned} \tag{5.12}$$

where $\nabla_w w_i$ is computed by Eq. (5.9) and Eq. (5.10). To update w, it only needs to compute the second-order gradient in Eq. (5.12), and first-order approximation is used as in [30] to speed up training.

5.4.3 Meta Test on a New Domain

After meta training, a general feature extractor f_w is learned, which is expected to capture domain-shared features and can be quickly adapted to a new test domain. For meta test, f_w is adapted to a novel domain similarly as in outer-domain training. Specifically, we randomly select N_d training tasks from the training domains (one task per domain) and save the domain prototypes $\{\mathbf{P}_w^i\}_{i=1}^{N_d}$ (N_d vectors). During meta test, f_w is adapted to a new domain with the saved prototypes and the supports of the prediction task in the new domain, i.e.,

$$w_{\text{new}} = w - \alpha\nabla_w\mathcal{L}_{T_{\text{new}}}^{\text{dom}}(f_w(\mathcal{X}_{T_{\text{new}}}^{\text{tr}}), \mathbf{P}_w^{\text{new}}, \{\mathbf{P}_w^i\}_{i=1}^{N_d}; w). \tag{5.13}$$

Table 5.2: Cross-domain few-shot classification under the leave-one-out setting. The results of baselines [16] and LFT [104] are produced by their *official* public codes. For *fairness*, M^{2}L is also implemented on the code of [104].

		5-way 5-shot				5-way 1-shot			
Baseline		Cars	Fungi	CUB	Flowers	Cars	Fungi	CUB	Flowers
RelationNet	-	39.87 ± 0.56	51.03 ± 0.58	57.30 ± 0.57	78.68 ± 0.64	30.63 ± 0.56	36.99 ± 0.59	39.96 ± 0.59	60.48 ± 0.81
	LFT	41.54 ± 0.60	51.33 ± 0.57	54.94 ± 0.59	77.43 ± 0.66	30.78 ± 0.54	36.55 ± 0.56	38.71 ± 0.57	59.35 ± 0.75
	M^{2}L	$\mathbf{43.38 \pm 0.50}$	$\mathbf{53.64 \pm 0.74}$	$\mathbf{59.38 \pm 0.57}$	$\mathbf{79.44 \pm 0.59}$	$\mathbf{32.16 \pm 0.66}$	$\mathbf{38.85 \pm 0.60}$	$\mathbf{43.06 \pm 0.63}$	$\mathbf{65.99 \pm 0.85}$
		+3.51	*+2.61*	*+2.08*	*+0.76*	*+1.53*	*+1.86*	*+3.10*	*+5.51*
MatchingNet	-	39.26 ± 0.56	53.95 ± 0.60	52.71 ± 0.60	75.34 ± 0.65	30.57 ± 0.52	40.10 ± 0.60	40.83 ± 0.55	61.68 ± 0.81
	LFT	39.16 ± 0.57	49.47 ± 0.58	55.24 ± 0.59	78.75 ± 0.62	31.42 ± 0.55	38.86 ± 0.56	39.30 ± 0.61	$61.91 \pm 0.8.$
	M^{2}L	$\mathbf{48.58 \pm 0.53}$	$\mathbf{58.18 \pm 0.80}$	$\mathbf{58.48 \pm 0.56}$	$\mathbf{83.64 \pm 0.56}$	$\underline{\mathbf{34.96 \pm 0.72}}$	39.71 ± 0.42	$\mathbf{45.29 \pm 0.63}$	$\mathbf{69.31 \pm 0.84}$
		+9.32	*+4.23*	*+5.77*	*+8.30*	*+4.39*	–	*+4.46*	*+7.63*
ProtoNet	-	48.17 ± 0.60	59.06 ± 0.61	63.91 ± 0.56	86.67 ± 0.51	32.58 ± 0.51	39.85 ± 0.57	42.06 ± 0.58	65.88 ± 0.79
	LFT	48.53 ± 0.57	59.55 ± 0.58	57.59 ± 0.58	81.90 ± 0.57	33.21 ± 0.54	38.77 ± 0.56	41.55 ± 0.59	59.71 ± 0.71
	M^{2}L	$\underline{\mathbf{50.99 \pm 0.52}}$	$\mathbf{62.02 \pm 0.59}$	63.71 ± 0.57	86.27 ± 0.55	$\mathbf{33.29 \pm 0.54}$	$\mathbf{40.51 \pm 0.69}$	$\mathbf{43.23 \pm 0.59}$	65.66 ± 0.80
		+2.82	*+2.96*	–	–	*+0.71*	*+0.66*	*+1.17*	–
GNN	-	47.33 ± 0.66	60.59 ± 0.69	66.25 ± 0.67	88.91 ± 0.52	33.31 ± 0.58	41.40 ± 0.67	44.77 ± 0.68	71.69 ± 0.87
	LFT	47.78 ± 0.68	61.65 ± 0.72	62.88 ± 0.68	89.20 ± 0.55	34.01 ± 0.61	42.66 ± 0.71	44.22 ± 0.71	71.76 ± 0.87
	M^{2}L	$\mathbf{49.64 \pm 0.42}$	$\underline{\mathbf{65.06 \pm 0.62}}$	$\underline{\mathbf{67.63 \pm 0.68}}$	$\underline{\mathbf{89.77 \pm 0.51}}$	$\mathbf{34.44 \pm 0.74}$	$\underline{\mathbf{43.94 \pm 0.48}}$	$\underline{\mathbf{45.32 \pm 0.67}}$	$\mathbf{72.04 \pm 0.44}$
		+2.31	*+4.47*	*+1.38*	*+0.86*	*+1.13*	*+2.54*	*+0.55*	*+0.35*
MAML	-	37.63 ± 0.51	42.92 ± 0.80	47.88 ± 0.72	74.83 ± 1.00	30.71 ± 0.43	35.97 ± 0.58	36.95 ± 0.66	49.71 ± 0.57
ProtoMAML	-	50.87 ± 0.44	62.97 ± 0.61	63.47 ± 0.34	87.34 ± 0.46	34.64 ± 0.19	42.11 ± 0.66	42.98 ± 0.40	$\underline{72.53 \pm 0.99}$

The loss $\mathcal{L}^{\text{dom}}_{T_{\text{new}}}(f_w)$ in Eq. (5.13) is the same as the one in Eq. (5.10) except that the number of domains is $N_d + 1$ instead of N_d. We make predictions on the queries with the adapted feature extractor $f_{w_{new}}$.

5.5 Experiments

We evaluate metric-based M^{2}L for cross-domain few-shot classification on 9 fine-grained datasets of different domains. To provide an extensive evaluation, we conduct experiments under two settings: **(1)** Following the leave-one-out setting in [104], we select a domain from multiple domains as the test domain and use the rest for training M^{2}L. **(2)** We train M^{2}L over a set of domains and test on another set of domains. All the reported results can be reproduced.

5.5.1 Datasets and Setup

Datasets. For the leave-one-out setting, we conduct experiments on four fine-grained datasets: Cars [59], CUB-200-2011 [113], Fungi [97], VGG Flower [82]. We pick one dataset as the test domain and the rest three datasets are used as training domains. We use the *mini*Imagenet dataset, which contains images of quite diverse classes, for pretraining and validation, following [104]. For the second setting, we use all these four datasets as training domains to train M^{2}L and test on another five new datasets including Vegetable [51], Pets [85], Food [62], Butterfly [15] and Plantae [105].

Baselines. We compare with popular baseline models including ProtoNet [100], GNN [37], RelationNet [102], MatchingNet [108], MAML [30] and ProtoMAML [103]. We instantiate our method M^{2}L and a recent method LFT [104] with the first four metric-based methods. The performance of the baselines, LFT and our M^{2}L are compared. The baselines are implemented with a standard ResNet-10 backbone [49] and trained by mixing up all the training datasets as a whole.

Implementation Details. Following [104], to improve the convergence rate, we use a pre-trained feature extractor trained over *mini*Imagenet in a supervised learning manner, which is directly downloaded from [104]. For *fairness*, all compared methods use the same pre-trained model and are trained with same implementation details. Adam is used as the optimizer with a fixed learning rate $\beta = 10^{-3}$, and 200 epoches are trained in total. For a 5-way 5-shot task, 5 classes are randomly selected from the training domain, and 5 supports and 16 queries are randomly sampled per category, following the practice in [16] and [104]. During meta test, the test performance is measured over 1000 tasks. For efficiency, we use a first-order approximation for second-order derivatives as in [30]. We set the step size as $\alpha = 0.1$ and the number of inner-domain update steps as 5.

5.5.2 Comparison with State-of-the-Art

For the leave-one-out setting, the results are shown in Table 5.2. We can make the following observations. **(1)** For both 5-way 5-shot tasks and 5-way 1-shot tasks, M^{2}L consistently and significantly outperforms the baseline models in most cases, which verifies the effectiveness

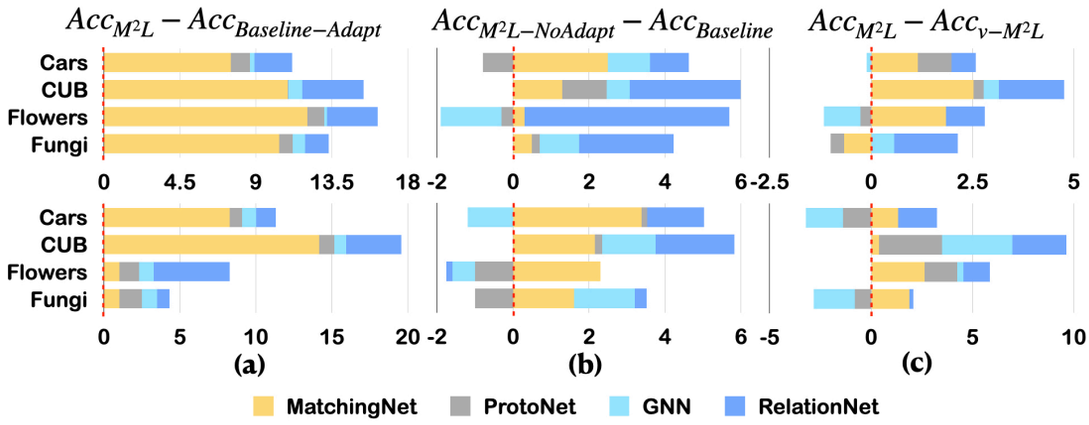


FIGURE 5.3: The performance gaps between (a) M^{2}L and baselines with outer-domain training in meta test, i.e., $\text{Acc}_{\text{M}^2\text{L}} - \text{Acc}_{\text{Baseline-Adapt}}$; (b) M^{2}L without outer-domain training in meta test and baselines, i.e., $\text{Acc}_{\text{M}^2\text{L-NoAdapt}} - \text{Acc}_{\text{Baseline}}$; and (c) M^{2}L and v-M^{2}L, i.e., $\text{Acc}_{\text{M}^2\text{L}} - \text{Acc}_{\text{v-M}^2\text{L}}$. Top: 5-way 5-shot. Bottom: 5-way 1-shot.

of M^{2}L. **(2)** M^{2}L outperforms LFT in *every* case and with a considerable margin in many cases. Overall, LFT does not have a clear advantage over the baseline models (worse than them in some cases), which shows that the global feature-wise transformation used by LFT fails to capture the domain-specific features for fine-grained classification. By contrast, M^{2}L with the devised outer-domain training mechanism can efficiently adapt to individual domains to extract domain-specific features. **(3)** Although both M^{2}L and MAML update the feature extractor in the outer-loop of training, M^{2}L outperforms MAML by a wide margin. The results verify that domain-level training plays a crucial role in cross-domain few-shot classification. **(4)** ProtoMAML achieves competitive performance in the cross-domain few-shot classification tasks by combining the complementary strengths of ProtoNet and MAML. However, M^{2}L still shows superior performance under 7 cases and is comparable to ProtoMAML under 1 case.

The results of the second setting are reported in Table 5.3. Recall that for this setting, M^{2}L is trained on 4 datasets and tested on another 5 datasets of different domains. Still, M^{2}L consistently improves upon the metric-based baseline models in most cases and outperforms LFT in *every* case. This further confirms that M^{2}L can learn a general feature extractor shared across the training domains and adapt to different unseen domains for fine-grained few-shot classification.

Table 5.3: Cross-domain few-shot classification by training on CUB, Cars, Fungi and Flowers and testing on other datasets.

5-way 5-shot		Plantae	Butterfly	Pets	Food	Vegetable
RelationNet	-	51.33 ± 0.60	70.93 ± 0.54	52.10 ± 0.45	44.42 ± 0.50	68.30 ± 0.54
	LFT	50.86 ± 0.62	70.38 ± 0.54	47.56 ± 0.45	43.20 ± 0.47	67.76 ± 0.57
	M^{2}L	$\mathbf{55.44 \pm 0.62}$	$\mathbf{75.97 \pm 0.51}$	$\mathbf{55.73 \pm 0.45}$	$\mathbf{47.42 \pm 0.37}$	$\mathbf{69.95 \pm 0.58}$
		+4.11	+5.04	+3.63	+3.00	+1.65
MatchingNet	-	54.94 ± 0.58	71.74 ± 0.54	56.40 ± 0.49	43.28 ± 0.50	70.63 ± 0.57
	LFT	52.31 ± 0.58	70.94 ± 0.54	54.71 ± 0.48	44.87 ± 0.50	68.84 ± 0.54
	M^{2}L	$\mathbf{58.85 \pm 0.56}$	$\mathbf{78.94 \pm 0.31}$	55.87 ± 0.46	$\mathbf{47.42 \pm 0.37}$	$\mathbf{71.09 \pm 0.53}$
		+3.91	+7.20	–	+4.14	+0.46
ProtoNet	-	59.30 ± 0.67	79.05 ± 0.47	67.60 ± 0.44	54.00 ± 0.53	74.04 ± 0.53
	LFT	55.50 ± 0.57	75.43 ± 0.51	65.80 ± 0.46	52.02 ± 0.51	71.65 ± 0.52
	M^{2}L	$\mathbf{60.47 \pm 0.60}$	78.64 ± 0.45	67.15 ± 0.45	$\mathbf{54.13 \pm 0.41}$	$\mathbf{75.27 \pm 0.52}$
		+1.17	–	–	+0.13	+1.23
GNN	-	61.53 ± 0.70	86.39 ± 0.47	67.55 ± 0.57	52.33 ± 0.60	79.88 ± 0.59
	LFT	62.09 ± 0.70	85.31 ± 0.48	66.91 ± 0.55	52.25 ± 0.58	80.26 ± 0.61
	M^{2}L	$\mathbf{64.48 \pm 0.71}$	$\mathbf{87.22 \pm 0.46}$	$\mathbf{70.25 \pm 0.52}$	$\mathbf{53.80 \pm 0.63}$	$\mathbf{80.62 \pm 0.58}$
		+2.95	+0.83	+2.70	+1.47	+0.74

5.5.3 Ablation Study

To have a closer look over M^{2}L, we conduct the following experiments: **(1)** To evaluate the generalization ability of the learned feature extractor of M^{2}L, we directly test the learned feature extractor of M^{2}L without doing outer-domain training in meta test and compare with the feature extractors learned by the baselines; **(2)** To evaluate the effectiveness of the meta training procedure of M^{2}L, we incorporate the outer-domain training mechanism into the baseline models and compare them with M^{2}L; **(3)** To further demonstrate the effectiveness of the proposed outer-domain training process in M^{2}L, we study a variant of M^{2}L (v-M^{2}L) that separates all the training domains simultaneously without adapting to individual domains.

(1) $\mathbf{Acc_{Baseline}}$ vs. $\mathbf{Acc_{M^2L\text{-}NoAdapt}}$. To verify that M^{2}L can learn a domain-shared feature extractor f_w that generalizes better to a novel domain than the baselines, we compare

$\text{Acc}_{\text{Baseline}}$ and $\text{Acc}_{\text{M}^2\text{L-NoAdapt}}$. In meta test, we directly test the feature extractors f_w learned by M^{2}L and the baselines without further adaptation. It can be observed in Figure 5.3 (a) that M^{2}L without adaptation achieves higher test accuracy than the baselines in most cases, showing that the learned feature extractor has better generalization ability.

(2) $\text{Acc}_{\text{Baseline-Adapt}}$ vs. $\text{Acc}_{\text{M}^2\text{L}}$. To demonstrate the effectiveness of the meta training procedure of M^{2}L, i.e., outer-domain training and second-order learning, in meta test, we adapt the learned feature extractors of the baselines to a new domain with the proposed outer-domain training mechanism and compare them with M^{2}L. As shown in Fig. 5.3 (b), $\text{Acc}_{\text{Baseline-Adapt}}$ is significantly lower than $\text{Acc}_{\text{M}^2\text{L}}$ in *every* case, which confirms the effectiveness of the meta training procedure of M^{2}L.

(3) M^{2}L vs. v-M^{2}L. To explore the effectiveness of the devised outer-domain training process in M^{2}L, we compare it with v-M^{2}L. Particularly, v-M^{2}L learns a *global* feature extractor f_{w_u} to discriminate all domains instead of a specific f_{w_i} for each domain Dom_i. As such, all the few-shot classification tasks of each training domain are performed in the embedding space of f_{w_u}. The update rules for the feature extractor f_{w_u} and the model parameters w are as follows:

$$w_u = w - \alpha \nabla_w \sum_{i=1}^{N_d} \mathcal{L}_{T_i}^{\text{dom}}(f_w(\mathcal{X}_{T_i}^{\text{tr}}), \{\mathbf{P}_w^i\}_{i=1}^{N_d}; w),$$

$$w = w - \beta \nabla_{w_u} \sum_{i=1}^{N_d} \mathcal{L}_{T_i}^{\text{task}}(\hat{\mathcal{Y}}_{T_i}^{\text{ts}}, \mathcal{Y}_{T_i}^{\text{ts}}; w_u) \cdot \nabla_w w_u.$$

After training, v-M^{2}L can adapt the feature extractor f_{w_u} to a new test domain similarly as in M^{2}L. It can be observed in Figure 5.3 (c) that M^{2}L significantly and consistently outperforms v-M^{2}L in most cases, demonstrating the advantage of the devised outer-domain training process of M^{2}L.

5.6 Related Work

Domain adaptation and generalization. Domain generalization and domain adaptation aim to solve domain shift where the training (source) domain and the test (target) domain have a discrepancy. Depending on whether the target domain is accessible in training, we

Table 5.4: Comparison of domain adaptation/generalization, few-shot learning, and cross-domain few-shot learning. Note that the label spaces of tasks in training and testing are the same in domain adaptation/generalization but different in (cross-domain) few-shot learning.

	Domain Shift	Different Label Space	Few Training Samples
Domain Adapt. / Gen.	✓	×	×
Few-Shot Learning	×	✓	✓
Cross-Domain FSL	✓	✓	✓

have domain adaptation [83, 70, 72, 71, 83, 40] and domain generalization [76, 9, 114, 65, 38, 76]. The target domain is accessible for domain adaptation, but not accessible for domain generalization. Although the target domain and the source domain have a domain shift in domain adaptation/generalization, the label space, i.e., the tasks used in training and testing are the same. For example, the task is to classify cats and dogs, and we may train on the front views of cats and dogs but test on the profiles of cats and dogs.

Cross-domain few-shot learning. Cross-domain few-shot learning [16, 103, 47, 104, 46, 101, 120, 116, 94] is undoubtedly a more general yet challenging problem than within-domain few-shot learning, since both the domains and the label spaces are different for training and testing. For instance, a base-learner may be trained to recognize animals but tested on plants. To our knowledge, the first work to consider cross-domain few-shot learning is [16], which proposed to train a base-learner on the *mini*Imagenet dataset and test it on the CUB dataset to evaluate the generalization ability of meta-learning algorithms. Recently, several works [103, 47, 104] independently studied cross-domain few-shot classification under the leave-one-out setting, i.e., leaving one dataset out for testing and using the rest for training. Moreover, several meta-datasets [103, 47] were introduced to standardize research of meta-learning across different domains. Very recently, [104] proposed a learnable feature-wise transformation to improve the generalization ability of a feature extractor with conditional batch normalization [18].

The key differences of the above three learning paradigms are summarized in Table 5.4.

6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have presented new insights into meta-learning, both theoretical and practical ones and proposed new state-of-the-art meta-algorithms based on the deeper insights. To start with, we presented new theoretical results on the stability and generalization of modern meta-learning algorithms with the support/query (S/Q) episodic training strategy. In addition, we have provided a comparison to the proposed leave-one-out training strategy for meta-learning. Our analysis provides a generalization guarantee for empirically successful modern meta-learning algorithms with S/Q episodic training, which is particularly meaningful in the few-shot learning paradigm. Apart from the general theoretical analysis of the modern meta-training strategy, we also pointed out that the optimization procedure of the existing meta-training paradigm highly relies on the differentiable analytical expression of the task-specific parameters w.r.t. the model parameters. Such dependency results in limited choices of the inner-task algorithms and limited expressiveness of the meta-algorithms. To remove such restrictions, we have proposed an adaptation-agnostic meta-training strategy for few-shot classification. Such strategy naturally leads to a flexible and efficient ensemble framework A2E which is general enough to combine any inner-task algorithm with a differentiable loss function. Empirical evidence shows that A2E can generalize well in different few-shot classification scenarios.

Further, we focus on the metric-based meta-learning which shows high generalization

ability and excellent performance over the few-shot classification problems. We proposed a generic variational metric scaling framework for metric-based meta-algorithms, under which three efficient end-to-end methods were developed. To learn a better embedding space to fit data distribution, we have considered the influence of metric scaling on the embedding space by taking into account data-dependent and task-dependent information progressively. Our methods are lightweight and can be easily plugged into existing metric-based meta-algorithms to improve their performance.

To solve a more challenging problem, i.e., cross-domain few-shot classification, we have presented a domain-oriented meta-learning framework called DOM which is simple, effective and flexible. It can employ any metric-based meta-learning method for within-domain few-shot classification while improving their generalization across domains via an interleaved gradient-based training. In our experiments on difficult cross-domain fine-grained few-shot classification tasks, consistent and considerable improvements of DOM over state-of-the-art meta-learning algorithms have been observed, which demonstrate the effectiveness of DOM.

6.2 Future Work

In the future, I will continue focusing on meta-learning including to provide deeper theoretical understanding of meta-learning, to design better meta-algorithms and to apply meta-learning methods to solve real-life challenges.

In Chapter 2, we have proposed a generalization bound of $O(n)$ for meta-learning with episodic support/query training strategy. This theoretical result, independent from the inner-task training sample size, indicates that the generalization gap converges despite of a small m. However, there still exists some limitations of such result since it is an upper bound of the generalization gap. An upper bound independent from m cannot demonstrate the generalization bound independent from m. There might exist more tighter generalization bounds, for instance, $O(1/\sqrt{n} + 1/(n * m))$. In the future, I will derive tighter generalization bounds for providing deeper understanding of the generalization ability of meta-learning. Besides, apart from the standard meta-learning, context-based meta-learning attracts a lot attention recently, especially in reinforcement learning [90]. In comparison with standard

meta-learning which captures the task-specific information by adapting model parameters, context meta-learning learns a shared predictor but captures the task-specific information by adapting input context vector. There lacks study of the theoretical property of context-based meta-learning, for instance, generalization ability [13] and model capacity [31]. In the future, I will focus on theoretically understanding context-based meta-learning and analyzing the differences between the context-based and standard meta-learning.

Based on the theoretical insights of meta-learning, we will focus on designing new training paradigms or meta-learning frameworks. Particularly, I will focus on more challenging and realistic scenarios. For instance, incremental few-shot classification under which the model is not only required to distinguish new categories but also should not forget the base classes during meta-training. This is a very challenging problem due to the inherent catastrophic forgetting question of deep neural network [57]. And I will also focus on another realistic and challenging imbalanced multi-task scenario where the training sample size of each task is very imbalanced. Most multi-task or meta-learning questions suffer from this problem, for instance, in low-resource machine translation, corpus is sufficient in English or Chinese but limited in Bulgarian. Moreover, I will try to apply meta-learning in various application scenarios, such as the aforementioned low-resource machine translation and reinforcement learning. It is widely known that reinforcement learning suffers from sample inefficient and sparse reward problems. How to make an agent adapt fast to a new environment is a crucial and hard problem. Applying meta learning to solve this problem is natural [15, 24].

To sum up, meta-learning aims to extract and transfer knowledge from prior experiences to adapt fast to a new environment. Such ability is worth exploring which can help us get closer to a general-purpose AI system.

Bibliography

[1] Noga Alon, Shai Ben-David, Nicolo Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *JACM*, 44(4):615–631, 1997.

[2] Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. *ICML*, 2018.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *NeurIPS*, 2016.

[4] Artem Babenko and Victor Lempitsky. Aggregating deep convolutional features for image retrieval. *arXiv preprint arXiv:1510.07493*, 2015.

[5] Jonathan Baxter. A model of inductive bias learning. *JAIR*, 12:149–198, 2000.

[6] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *ICLR*, 2019.

[7] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. *NeurIPS*, 2016.

[8] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *JACM*, 36(4):929–965, 1989.

[9] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *NeurIPS*, 2016.

[10] Olivier Bousquet and André Elisseeff. Stability and generalization. *JMLR*, 2(Mar):499–526, 2002.

[11] Leo Breiman et al. Heuristics of instability and stabilization in model selection. *The annals of statistics*, 24(6):2350–2383, 1996.

[12] Brian Bullins, Elad Hazan, Adam Kalai, and Roi Livni. Generalize across tasks: Efficient algorithms for linear representation learning. *ALT*, 2019.

[13] Jiaxin Chen, Xiao-Ming Wu, Yanke Li, Qimai Li, Li-Ming Zhan, and Fu-lai Chung. A closer look at the training strategy for modern meta-learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[14] Jiaxin Chen, Li-Ming Zhan, Xiao-Ming Wu, and Fu-lai Chung. Variational metric scaling for metric-based meta-learning. *AAAI*, 2020.

[15] Tianshui Chen, Wenxi Wu, Yuefang Gao, Le Dong, Xiaonan Luo, and Liang Lin. Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding. In *ACM Multimedia*, 2018.

[16] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *ICLR*, 2019.

[17] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[18] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *NeurIPS*, 2017.

[19] Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. *ICML*, 2019.

[20] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Incremental learning-to-learn with statistical guarantees. *UAI*, 2018.

[21] Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. *NeurIPS*, 2018.

[22] Giulia Denevi, Dimitris Stamos, Carlo Ciliberto, and Massimiliano Pontil. Online-within-online meta-learning. *NeurIPS*, 2019.

[23] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[24] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[25] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *CVPR*, pages 3723–3731, 2019.

[26] Andre Elisseeff, Theodoros Evgeniou, and Massimiliano Pontil. Stability of randomized learning algorithms. *JMLR*, 6(Jan):55–79, 2005.

[27] André Elisseeff, Massimiliano Pontil, et al. Leave-one-out error and stability of learning algorithms with applications. *NATO science series sub series iii computer and systems sciences*, 190:111–130, 2003.

[28] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. On the convergence theory of gradient-based model-agnostic meta-learning algorithms. *AISTATS*, 2020.

[29] Vitaly Feldman and Jan Vondrak. Generalization bounds for uniformly stable algorithms. In *NeurIPS*, 2018.

[30] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, 2017.

[31] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In *International Conference on Learning Representations*, 2018.

[32] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *NeurIPS*, 2018.

[33] Stanislav Fort. Gaussian prototypical networks for few-shot learning on omniglot. *arXiv preprint arXiv:1708.02735*, 2017.

[34] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimilano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. *ICML*, 2018.

[35] Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

[36] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[37] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *ICLR*, 2018.

[38] Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*, 2015.

[39] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, pages 4367–4375, 2018.

[40] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *ICML*, 2016.

[41] Wenbo Gong, Yingzhen Li, and José Miguel Hernández-Lobato. Meta-learning for stochastic gradient mcmc. *arXiv preprint arXiv:1806.04522*, 2018.

[42] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Versa: Versatile and efficient few-shot learning. In *Third workshop on Bayesian Deep Learning*, 2018.

[43] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *ICLR*, 2019.

[44] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. *ICLR*, 2018.

[45] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org, 2017.

[46] Yunhui Guo, Noel C Codella, Leonid Karlinsky, James V Codella, John R Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. *arXiv preprint*, 2020.

[47] Yunhui Guo, Noel CF Codella, Leonid Karlinsky, John R Smith, Tajana Rosing, and Rogerio Feris. A new benchmark for evaluation of cross-domain few-shot learning. *arXiv preprint arXiv:1912.07200*, 2019.

[48] Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *ICML*, 2016.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[51] Saihui Hou, Yushan Feng, and Zilei Wang. Vegfru: A domain-specific dataset for fine-grained visual categorization. In *ICCV*, 2017.

[52] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8420–8429, 2019.

[53] Mikhail Khodak, Maria-Florina Balcan, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. *ICML*, 2019.

[54] Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. *NeurIPS*, 2019.

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

[56] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2013.

[57] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[58] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. *ICML Deep Learning Workshop*, 2, 2015.

[59] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshops*, 2013.

[60] Ilja Kuzborskij and Christoph H Lampert. Data-dependent stability of stochastic gradient descent. *ICML*, 2018.

[61] Nan Lai, Meina Kan, Shiguang Shan, and Xilin Chen. Task-adaptive feature reweighting for few shot classification. In *Asian Conference on Computer Vision*, pages 649–662. Springer, 2018.

[62] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *CVPR*, 2018.

[63] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665, 2019.

[64] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, 2018.

[65] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[66] Fei-Fei Li, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

[67] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019.

[68] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017.

[69] Yaoyao Liu, Qianru Sun, An-An Liu, Yuting Su, Bernt Schiele, and Tat-Seng Chua. Lcc: Learning to customize and combine neural networks for few-shot learning. *arXiv preprint arXiv:1904.08479*, 2019.

[70] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[71] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, 2018.

[72] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

[73] Andreas Maurer. Algorithmic stability and meta-learning. *JMLR*, 6(Jun):967–994, 2005.

[74] Andreas Maurer. A second-order look at stability and generalization. In *COLT*, 2017.

[75] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *ICLR*, 2018.

[76] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *ICML*, 2013.

[77] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25(1-3):161–193, 2006.

[78] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *ICML*, 2017.

[79] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. *arXiv preprint arXiv:1712.09926*, 2017.

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

[81] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.

[82] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[83] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[84] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

[85] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *ICCV*, 2012.

[86] Anastasia Pentina and Christoph Lampert. A pac-bayesian bound for lifelong learning. *ICML*, 2014.

[87] Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. *NeurIPS*, 2015.

[88] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *ICLR*, 2020.

[89] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, pages 113–124, 2019.

[90] 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*, pages 5331–5340. PMLR, 2019.

[91] Rajeev Ranjan, Carlos D Castillo, and Rama Chellappa. L2-constrained softmax loss for discriminative face verification. *arXiv preprint arXiv:1703.09507*, 2017.

[92] Sachin Ravi and Alex Beatson. Amortized bayesian meta-learning. *ICLR*, 2018.

[93] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. *ICLR*, 2017.

[94] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *NeurIPS*, 2019.

[95] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

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

[97] Brigit Schroeder and Yin Cui. FGVCx fungi classification challenge 2018. `https://github.com/visipedia/fgvcx_fungi_comp`, 2013.

[98] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[99] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *JMLR*, 11(Oct):2635–2670, 2010.

[100] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *NeurIPS*, 2017.

[101] Jiamei Sun, Sebastian Lapuschkin, Wojciech Samek, Yunqing Zhao, Ngai-Man Cheung, and Alexander Binder. Explanation-guided training for cross-domain few-shot classification. *arXiv preprint arXiv:2007.08790*, 2020.

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

[103] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*, 2020.

[104] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. In *ICLR*, 2020.

[105] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018.

[106] Vladimir Vapnik. *Estimation of dependences based on empirical data*. Springer Science & Business Media, 2006.

[107] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 2013.

[108] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NeurIPS*, 2016.

[109] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[110] Weitao Wan, Yuanyi Zhong, Tianpeng Li, and Jiansheng Chen. Rethinking feature distribution for loss functions in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9117–9126, 2018.

[111] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: l 2 hypersphere embedding for face verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049. ACM, 2017.

[112] Yong Wang, Xiao-Ming Wu, Qimai Li, Jiatao Gu, Wangmeng Xiang, Lei Zhang, and Victor OK Li. Large margin few-shot learning. *arXiv preprint arXiv:1807.02872*, 2018.

[113] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. *Technical Report CNS-TR-2010-001*, 2010.

[114] Zheng Xu, Wen Li, Li Niu, and Dong Xu. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*, 2014.

[115] Yongxin Yang and Timothy Hospedales. Deep multi-task representation learning: A tensor factorisation approach. *arXiv preprint arXiv:1605.06391*, 2016.

[116] Jia-Fong Yeh, Hsin-Ying Lee, Bing-Chen Tsai, Yi-Rong Chen, Ping-Chia Huang, and Winston H Hsu. Large margin mechanism and pseudo query set on cross-domain few-shot learning. *arXiv preprint arXiv:2005.09218*, 2020.

[117] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *ICLR*, 2020.

[118] Xu Zhang, Felix Xinnan Yu, Svebor Karaman, Wei Zhang, and Shih-Fu Chang. Heated-up softmax embedding. *arXiv preprint arXiv:1809.04157*, 2018.

[119] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *ICML*, pages 7693–7702, 2019.

[120] Yixiong Zou, Shanghang Zhang, José MF Moura, JianPeng Yu, and Yonghong Tian. Revisiting mid-level patterns for distant-domain few-shot recognition. *arXiv preprint arXiv:2008.03128*, 2020.

Appendix A

Appendix

A.1 Proof of Theorem 2

Lemma 1 (McDiarmid)**.** *Let* $\mathbf{S}$ *and* $\mathbf{S}^{(i)}$ *defined as above,* $F : (\mathcal{Z}^{m+q})^n \to \mathbb{R}$ *be any measurable function for which there exists constant* $c_i (i = 1, \ldots, n)$ *such that,*

$$\sup_{\mathbf{S}\in(\mathcal{Z}^{m+q})^n, S_i'\in\mathcal{Z}^{m+q}} |F(\mathbf{S}) - F(\mathbf{S}^{(i)})| \leq c_i, \tag{A.1}$$

then

$$\mathbb{P}_{\mathbf{S}}[F(\mathbf{S}) - \mathbb{E}_{\mathbf{S}}[F(\mathbf{S})] \geq \epsilon] \leq e^{-2\epsilon^2/\sum_{i=1}^n c_i^2}. \tag{A.2}$$

Given Lemma 1, we can upper bound the outer-task gap for S/Q training.

Theorem 2. *For any task distribution* τ *and meta-sample* $\mathbf{S}$ *with* n *tasks, if a meta-algorithm* $\mathbf{A}$ *has uniform stability* β *w.r.t. a loss function* l *bounded by* M*, then the following statement holds with probability of at least* $1 - \delta$ *for any* $\delta \in (0, 1)$*:*

$$\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) \leq \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S}) + \epsilon(n, \beta), \tag{A.3}$$

where $\epsilon = 2\beta + (4n\beta + M)\sqrt{\frac{\ln(1/\delta)}{2n}}$.

Proof. Let $F(\mathbf{S}) = \mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) - \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S})$ and $F(\mathbf{S}^{(i)}) = \mathcal{R}(\mathbf{A}(\mathbf{S}^{(i)}), \tau) - \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}^{(i)}), \mathbf{S}^{(i)})$. We have

$$|F(\mathbf{S}) - F(\mathbf{S}^{(i)})| \leq |\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau) - \mathcal{R}(\mathbf{A}(\mathbf{S}^{(i)}), \tau)| + |\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}), \mathbf{S}) - \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}^{(i)}), \mathbf{S}^{(i)})|. \tag{A.4}$$

The first term in (A.4) can be written as

$$|\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)-\mathcal{R}(\mathbf{A}(\mathbf{S}^{(i)}),\tau)|\leq|\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)-\mathcal{R}(\mathbf{A}(\mathbf{S}^{\setminus i}),\tau)|+|\mathcal{R}(\mathbf{A}(\mathbf{S}^{(i)},\tau)-\mathcal{R}(\mathbf{A}(\mathbf{S}^{\setminus i}),\tau)|.$$

We can upper bound the first term in (A.4) by studying the variation when a sample set S_i of training task $\mathcal{D}_i$ is deleted,

$$\begin{aligned}
&|\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)-\mathcal{R}(\mathbf{A}(\mathbf{S}^{\setminus i}),\tau)|\\
\leq&\mathbb{E}_{\mathcal{D}\sim\tau}\mathbb{E}_{S^{tr}\sim\mathcal{D}^m}\mathbb{E}_{S^{ts}\sim\mathcal{D}^q}|\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}),S^{ts})-\hat{L}(\mathbf{A}(\mathbf{S}^{\setminus i})(S^{tr}),S^{ts})|\\
\leq&\sup_{\mathcal{D}\sim\tau,S^{tr}\sim\mathcal{D}^m,S^{ts}\sim\mathcal{D}^q}|\hat{L}(\mathbf{A}(\mathbf{S})(S^{tr}),S^{ts})-\hat{L}(\mathbf{A}(\mathbf{S}^{\setminus i})(S^{tr}),S^{ts})|\leq\beta.
\end{aligned}$$

Similarly, we have $|\mathcal{R}(\mathbf{A}(\mathbf{S}^{(i)}),\tau)-\mathcal{R}(\mathbf{A}(\mathbf{S}^{\setminus i}),\tau)|\leq\beta$. So the first term in (A.4) is upper bounded by 2β. The second factor in (A.4) can be guaranteed likewise as follows,

$$\begin{aligned}
&|\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}),\mathbf{S})-\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}^{(i)}),\mathbf{S}^{(i)})|\\
\leq&\frac{1}{n}\sum_{l\neq i}|\hat{L}(\mathbf{A}(\mathbf{S})(S_l^{tr}),S_l^{ts})-\hat{L}(\mathbf{A}(\mathbf{S}^{(i)})(S_l^{tr}),S_l^{ts})|\\
&+\frac{1}{n}|\hat{L}(\mathbf{A}(\mathbf{S})(S_i^{tr}),S_i^{ts})-\hat{L}(\mathbf{A}(\mathbf{S}^{(i)})(S_i'^{tr}),S_i'^{ts})|\\
\leq&2\beta+\frac{M}{n}.
\end{aligned}\tag{A.5}$$

Hence, $|F(\mathbf{S})-F(\mathbf{S}^{(i)})|$ satisfies the condition of Lemma 1 with $c_i=4\beta+\frac{M}{n}$. It remains to bound $\mathbb{E}_{\mathbf{S}}[F(\mathbf{S})]=\mathbb{E}_{\mathbf{S}}[\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)]-\mathbb{E}_{\mathbf{S}}[\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}),\mathbf{S})]$. The first term can be written as follows,

$$\mathbb{E}_{\mathbf{S}}[\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)]=\mathbb{E}_{\mathbf{S},S_i'^{tr},S_i'^{ts}}\hat{L}(\mathbf{A}(\mathbf{S})(S_i'^{tr}),S_i'^{ts}).$$

Similarly, the second term is,

$$\begin{aligned}
\mathbb{E}_{\mathbf{S}}[\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}),\mathbf{S})]=&\mathbb{E}_{\mathbf{S}}[\frac{1}{n}\sum_{i=1}^{n}\hat{L}(\mathbf{A}(\mathbf{S})(S_i^{tr}),S_i^{ts})]\\
=&\mathbb{E}_{\mathbf{S}}[\hat{L}(\mathbf{A}(\mathbf{S})(S_i^{tr}),S_i^{ts})]\\
=&\mathbb{E}_{\mathbf{S},S_i'^{tr},S_i'^{ts}}[\hat{L}(\mathbf{A}(\mathbf{S}^{(i)})(S_i'^{tr}),S_i'^{ts})].
\end{aligned}$$

Hence, $\mathbb{E}_{\mathbf{S}}[F(\mathbf{S})]$ is upper bounded by 2β,

$$\begin{aligned}
&\mathbb{E}_{\mathbf{S}}[\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau)]-\mathbb{E}_{\mathbf{S}}[\hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}),\mathbf{S})]\\
=&\mathbb{E}_{\mathbf{S},S_i'^{tr},S_i'^{ts}}[\hat{L}(\mathbf{A}(\mathbf{S})(S_i'^{tr}),S_i'^{ts})-\hat{L}(\mathbf{A}(\mathbf{S}^{(i)})(S_i'^{tr}),S_i'^{ts})]\leq2\beta.
\end{aligned}\tag{A.6}$$

Plugging the inequality (A.6) in Lemma 1, we obtain

$$\mathbb{P}_{\mathbf{S}}[\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau) - \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}),\mathbf{S}) \geq 2\beta + \epsilon] \leq e^{-2\epsilon^2/\sum_{i=1}^{n}(4\beta+\frac{M}{n})^2}. \tag{A.7}$$

Finally, setting the right side of (A.7) to δ, the following result holds with probability of $1-\delta$,

$$\mathcal{R}(\mathbf{A}(\mathbf{S}),\tau) \leq \hat{\mathcal{R}}(\mathbf{A}(\mathbf{S}),\mathbf{S}) + 2\beta + (4n\beta + M)\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

□

A.2 Proof of Theorem 3

For simplicity, we denote the training error of each task by $f(w,S)$. The following Lemma 2 and Lemma 3 are proposed in [48] to prove Theorem 3.

Lemma 2 ([48]). *Denote by $G_{f,\zeta}$ the gradient update rule with a loss function f and step size ζ. If f is α-smooth, then $G_{f,\zeta}$ is $(1+\alpha\zeta)$-expansive, i.e., $\forall v, w \in \mathcal{W}, \|G(v) - G(w)\| \leq (1+\alpha\zeta)\|v-w\|$. If f is η-Lipschitz continuous, then $G_{f,\zeta}$ is $(\zeta\eta)$-bounded, i.e., $\|v - G_{f,\zeta}(v)\| \leq \zeta\eta$.*

Based on Lemma 2, the following Lemma 3 states that given two arbitrary sequences of updates: $G_1, \ldots, G_T$ and $G_1, \ldots, G'_T$, if they have the same initialization: $w_0 = w'_0$, the gap between their outputs at each step t: $\delta_t = \|w_t - w'_t\|$ can be bounded.

Lemma 3 ([48]). *Fix any arbitrary sequences of updates $G_1, \ldots, G_T$ and $G'_1, \ldots, G'_T$. Let w_t and w'_t be the outputs of the step t and define $\delta_t = \|w_t - w'_t\|$. Assume $w_0 = w'_0$, we have*

$$\delta_{t+1} \leq \begin{cases} (1+\alpha\zeta)\delta_t & G_t = G'_t \text{ is } (1+\alpha\zeta)\text{-expansive} \\ \delta_t + 2\zeta\eta & G_t \text{ and } G'_t \text{ are } \zeta\eta\text{-bounded,} \\ & G_t \text{ is } (1+\alpha\zeta)\text{-expansive} \end{cases}$$

With Lemma 3, we can obtain the upper bound of β.

Theorem 3. *Assume that the loss function l is α-smooth, η-Lipschitz continuous w.r.t. input and bounded by $M > 0$. Suppose that a meta-algorithm $\mathbf{A}$ is implemented by a SGM after T steps with step size $\zeta_t \leq c/t$, where c is a constant and $t < T$, then $\mathbf{A}$ has randomized uniform stability*

$$\beta \leq \frac{1 + 1/\alpha c}{n-1}\left(\frac{M}{2c\eta^2}\right)^{\alpha c+1} T^{\frac{\alpha c}{\alpha c+1}}. \tag{A.8}$$

Proof. Let $\mathbf{S}$ and $\mathbf{S}^{\setminus i}$ be a meta-sample and the leave-one-out meta-sample. Denote by $G_1, \ldots, G_T$ and $G'_1, \ldots, G'_T$ two arbitrary sequences of updates induced by implementing SGM on $\mathbf{S}$ and $\mathbf{S}^{\setminus i}$ respectively. Let w_t and w'_t be the outputs of G_t and G'_t, where $t \in \{1, \ldots, T\}$. Let $\delta_t = \|w_t - w'_t\|$.

Denote by $I \in \{1, 2, \ldots, n\}$ the step index that the meta-algorithm $\mathbf{A}$ selects the deleted training set S_i for the first time. For any $t_0 \in \{1, \ldots, n\}$, if $t_0 < I$, we have $\mathbb{E}_{\mathbf{A}}[G_{t_0}] = \mathbb{E}_{\mathbf{A}}[G'_{t_0}]$ and $\mathbb{E}_{\mathbf{A}}[\delta_{t_0}] = 0$. As S_{i_t} is uniformly and randomly selected from the meta-sample, the probability $\mathbb{P}(\delta_{t_0} \neq 0) = \mathbb{P}(I < t_0) = \frac{t_0}{n}$. And since $f(w, S)$ is η-Lipschitz continuous, we can obtain

$$\begin{aligned}
&\mathbb{E}_{\mathbf{A}}[|f(w_T, S) - f(w'_T, S)|] \\
=&\mathbb{P}(\delta_{t_0} \neq 0)\mathbb{E}_{\mathbf{A}}[|f(w_T, S) - f(w'_T, S)||\delta_{t_0} \neq 0] + \mathbb{P}(\delta_{t_0} = 0)\mathbb{E}_{\mathbf{A}}[|f(w_T, S) - f(w'_T, S)||\delta_{t_0} = 0] \\
\leq&\frac{t_0}{n}M + \eta\mathbb{E}_{\mathbf{A}}[\delta_T|\delta_{t_0} = 0]
\end{aligned} \tag{A.9}$$

where M is the upper bound of the loss function.

Based on the fact $\mathbf{S}^{\setminus i} \subset \mathbf{S}$ and $\mathbf{S} \setminus \mathbf{S}^{\setminus i} = S_i$, it can be inferred that SGM selects S_i with probability $\frac{1}{n}$ and other meta samples with probability $1 - \frac{1}{n}$. Therefore, with probability $1 - \frac{1}{n}$, we have $\mathbb{E}_{\mathbf{A}}[G_t] = \mathbb{E}_{\mathbf{A}}[G'_t]$. According to Lemma 3, we get $\mathbb{E}_{\mathbf{A}}[\delta_{t+1}] \leq (1 + \alpha\zeta_t)\mathbb{E}_{\mathbf{A}}[\delta_t]$. Similarly, with probability $\frac{1}{n}$, we have $\mathbb{E}_{\mathbf{A}}[\delta_{t+1}] \leq \mathbb{E}_{\mathbf{A}}[\delta_t] + 2\zeta_t\eta$. We conclude

$$\mathbb{E}_{\mathbf{A}}[\delta_{t+1}|\delta_{t_0} = 0] \leq (1 - \frac{1}{n})(1 + \alpha\zeta_t)\mathbb{E}_{\mathbf{A}}[\delta_t|\delta_{t_0} = 0] + \frac{1}{n}\mathbb{E}_{\mathbf{A}}[\delta_t|\delta_{t_0} = 0] + \frac{2\zeta_t\eta}{n}. \tag{A.10}$$

Following the way of manipulating (A.10) proposed in [48], we get

$$\mathbb{E}_{\mathbf{A}}[|f(w_T, S) - f(w'_T, S)|] \leq \frac{t_0 M}{n} + \frac{2\eta^2}{\alpha(n-1)}(\frac{T}{t_0})^{\alpha c}. \tag{A.11}$$

Eventually, we can get the upper bound of β (A.8), through minimizing the right-hand side of inequality (A.11) w.r.t. t_0 approximately (consider $n \approx n - 1$). □

A.3 Leave-One-Out Training

A.3.1 Meta-Algorithms with LOO Episodic Training

As shown in Eq. (2.6), the generalization bound of meta-algorithms with LOO training relies on both the stability of meta-algorithms β and the stability of the inner-task algorithm $\tilde{\beta}$.

Note that the stability is algorithm-dependent. We have studied β by considering the specific training strategy of meta-algorithms, but the parameter $\tilde{\beta}$ relies on the specific inner-task algorithm. Hence, there exists no general $\tilde{\beta}$ for generic meta-algorithms. If the inner-task algorithm is stable, $\tilde{\beta}$ should depend on the sample size m and converges to 0 as $m \to \infty$. Here, take the prototypical networks as an example, whose inner-task algorithm is a metric-based classification algorithm, we show that its stability parameter $\tilde{\beta} \leq O(1/m)$ as follows.

Stability of Inner-Task Algorithm. If the loss function $l(w, z)$ is convex, the stability parameter $\tilde{\beta}$ can be derived straightforwardly [98]. However, for non-convex loss functions such as the cross-entropy loss used in [100], there is no known general result of the stability parameter, to our best knowledge. In this section, we derive the stability $\tilde{\beta}$ for the cross-entropy loss of prototypical networks.

Prototypical networks find the prototype (mean vector) of each class first and then classifying the query into the nearest prototype's class in the embedding space. The loss function of prototypical networks is defined as

$$l(w, z) = -\log \frac{e^{-d(\phi_w(x), \mathbf{c}_y)}}{\sum_{k=1}^{K} e^{-d(\phi_w(x), \mathbf{c}_k)}}, \tag{A.12}$$

where $z = (x, y)$ is a query. The prototype of class k is denoted by $\mathbf{c}_k = \frac{1}{N} \sum_{y_i=k} \phi_w(x_i)$, which is the mean vector of the embedded support samples belonging to class k. Note that $\mathbf{c}_y = \frac{1}{N-1} \sum_{y_i=y} \phi_w(x_i)$ for LOO loss.

In practice, we randomly select K classes and N samples in each class as the support set S_i^{tr} of a training task $\mathcal{D}_i$ for K-way N-shot learning. However, the data generating process (i.i.d.) cannot guarantee that the support set exactly contains K classes and N samples per class. Hence, we study the hypothesis stability (Definition 6) w.r.t. the expectation of the training set S^{tr}.

Definition 6 (Hypothesis stability [10]). *An algorithm A has hypothesis stability $\tilde{\beta}$ w.r.t. the loss function l if the following holds $\forall j \in \{1, \ldots, m\}$:*

$$\mathbb{E}_{S^{tr} \sim \mathcal{D}^m, z \sim \mathcal{D}}[|l(A(S^{tr}), z) - l(A(S^{tr\setminus j}), z)|] \leq \tilde{\beta}.$$

Based on the definition, the following gives the derivation of the stability parameter $\tilde{\beta}$.

Lemma 4. *Given a metric $d(\phi_w(x), \phi_w(x'))$ bounded by B, the inner-task algorithm of prototypical networks with loss function $l(w, z)$ in* (A.12) *has the uniform stability $\tilde{\beta} \leq O(1/m)$.*

Proof. Based on Definition 6, to obtain the hypothesis stability $\tilde{\beta}$, we upper bound the expectation of the variation $|l(A(S^{tr}), z) - l(A(S^{tr\setminus j}), z)|$ when deleting $\forall j \in \{1, 2, \ldots, m\}$ w.r.t. $S^{tr} \sim \mathcal{D}^m$ and $z \sim \mathcal{D}$. Given $\forall j \in \{1, 2, \ldots, m\}$, denote the class y_j of z_j as C_{y_j}. Considering two cases: (1) the query $z \notin C_{y_j}$ and (2) the query $z \in C_{y_j}$.

Case 1: The variation can be written as $|\log \sum_{k=1}^{K} e^{-d(\phi_w(x), \mathbf{c}_k)} - \log(\sum_{k=1, k\neq y_j}^{K} e^{-d(\phi_w(x), \mathbf{c}_k)} + e^{-d(\phi_w(x), \mathbf{c}_{y_j}^{\setminus j})})|(\star)$ where $\mathbf{c}_{y_j}^{\setminus j}$ is the prototype of the class C_{y_j} deleted z_j. Denote the bigger term in $(\star)$ as $\log a$ and the smaller one as $\log b$. Then $(\star)$ can be represented as $\log \frac{a}{b} = \log(1 + \frac{a-b}{b}) \leq \frac{a-b}{b} \leq \frac{a-b}{Ke^{-B}}$. Using the fact that e^{-x} is 1-Lipschitz continuous, we have $a - b \leq |d(\phi_w(x), \mathbf{c}_{y_j}) - d(\phi_w(x), \mathbf{c}_{y_j}^{\setminus j})|$. The training set S^{tr} is i.i.d. sampled from $\mathcal{D}^m$, assume that the size of the support samples belonging to C_{y_j} is κ. Then we know $\mathbf{c}_{y_j} = \frac{1}{\kappa} \sum_{y_l = y_j} x_j$ and $\mathbf{c}_{y_j}^{\setminus j} = \frac{1}{\kappa - 1} \sum_{y_l = y_j, l \neq j} x_j$, so the prototype $\mathbf{c}_{y_j}$ can be seen as a weighted average $\mathbf{c}_{y_j} = \frac{1}{\kappa}(x_j + (\kappa - 1)\mathbf{c}_{y_j}^{\setminus j})$. To obtain the upper bound, we consider the worst case under which the deleted sample x_j deviates from $\mathbf{c}_{y_j}^{\setminus j}$ most. There are two sub-cases: (1) $d(\phi_w(x), \phi_w(x_j)) = 0$, $d(\phi(x), \mathbf{c}_{y_j}^{\setminus j}) = B$ and (2) $d(\phi_w(x), \phi_w(x_j)) = B$, $d(\phi(x), \mathbf{c}_{y_j}^{\setminus j}) = 0$. For both sub-cases, the upper bound of the distance gap $a - b$ is $\frac{B}{\kappa}$ and the variation $(\star)$ is upper bounded by $\frac{B}{K\kappa e^{-B}}$.

Case 2: Similarly with Case 1, the upper bound of the variation is $\frac{B}{\kappa} + \frac{B}{K\kappa e^{-B}}$.

Using the fact that the query z is i.i.d. sampled from $\mathcal{D}$, the probabilities of Case 1, Case 2 are $\frac{K-1}{K}$ and $\frac{1}{K}$. Then we obtain $\mathbb{E}_z[|l(A(S^{tr}), z) - l(A(S^{tr\setminus j}), z)|] \leq (1 + \frac{B + Be^B}{K})\frac{B}{\kappa}$.

For the expectation w.r.t. the support set S^{tr} which is also i.i.d. sampled from $\mathcal{D}^m$. The size of class C_{y_j} follows a multinomial distribution $\kappa \sim B(m, \frac{1}{K})$. The expectation of $\frac{1}{\kappa+1}$ can be computed as $\mathbb{E}(\frac{1}{\kappa+1}) = \sum_{l=0}^{m} \frac{1}{l+1}\binom{m}{l}(p)^l(1-p)^{n-l} = \frac{1}{(m+1)p}\sum_{l=0}^{m}\binom{m+1}{l+1}p^{l+1}(1-p)^{m-l} = \frac{1-(1-p)^{m+1}}{(m+1)p}$ where $p = \frac{1}{K}$. Obviously, we have $\mathbb{E}(\frac{1}{\kappa}) \leq \mathbb{E}(\frac{2}{\kappa+1})$, so we get the hypothesis stability parameter $\tilde{\beta} = 2B(K + B + Be^B)\frac{1-((K-1)/K)^{m+1})}{m+1}$. Omitting the constants B and K,

$((K-1)/K)^{m+1} \to 0$ as $m \to \infty$. Therefore, we obtain the upper bound of $O(1/m)$ for the hypothesis stability $\tilde{\beta}$. □

Based on the above results, we obtain Theorem 5.

Theorem 5 (Generalization bound of prototypical networks with the LOO training)**.** *Suppose that a mapping $\phi_w(x)$ is Lipschitz continuous and smooth w.r.t. x, the metric $d(\phi_w(x), \phi_w(x'))$ is bounded s.t. the LOO loss function is bounded by M and a meta-algorithm $\mathbf{A}$ is implemented by episode. For any task distribution τ and meta-sample $\mathbf{S}$ consisting of n tasks and m support samples per task, the following holds with probability of at least $1-\delta$, $\forall \delta \in (0,1)$:*

$$\mathbb{E}_{\mathbf{A}}[\mathcal{R}(\mathbf{A}(\mathbf{S}), \tau)] \leq \mathbb{E}_{\mathbf{A}}[\hat{\mathcal{R}}_{loo}(\mathbf{A}(\mathbf{S}), \mathbf{S})] + 2\beta + (4n\beta + M)\sqrt{\frac{\ln(1/\delta)}{2n}} + \tilde{\beta}, \tag{A.13}$$

where $\beta \leq O(1/n)$ and $\tilde{\beta} \leq O(1/m)$.

Theorem 5 shows the generalization gap converges to 0 as $n \to \infty$ and $m \to \infty$. With LOO training, the increase of training samples in each task can also improve generalization. However, in few-shot learning, m is typically very small, so the generalization bound cannot converge as n grows and hence is less meaningful in this scenario.