Learning an Explicit Hyperparameter Prediction Policy Conditioned on Tasks

Jun Shu XJTUSHUJUN@GMAIL.COM
School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaan'xi Province, P. R. China
Pazhou Lab, Guangzhou, Guangdong Province, P. R. China

Deyu Meng* DYMENG@MAIL.XJTU.EDU.CN
School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaan'xi Province, P. R. China
Pazhou Lab, Guangzhou, Guangdong Province, P. R. China

Zongben Xu ZBXU@MAIL.XJTU.EDU.CN
School of Mathematics and Statistics and Ministry of Education Key Lab of Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, Shaan'xi Province, P. R. China
Pazhou Lab, Guangzhou, Guangdong Province, P. R. China

Editor:

Abstract

Meta learning has attracted much attention recently in machine learning community. Contrary to conventional machine learning aiming to learn inherent prediction rules to predict labels for new query data, meta learning aims to learn the learning methodology for machine learning from observed tasks, so as to generalize to new query tasks by leveraging the meta-learned learning methodology. In this study, we interpret such learning methodology as learning an explicit hyperparameter prediction policy shared by all training tasks. Specifically, this policy is represented as a parameterized function called meta-learner, mapping from a training/test task to its suitable hyperparameter setting, extracted from a pre-specified function set called meta learning machine. Such setting guarantees that the meta-learned learning methodology is able to flexibly fit diverse query tasks, instead of only obtaining fixed hyperparameters by many current meta learning methods, with less adaptability to query task's variations. Such understanding of meta learning also makes it easily succeed from traditional learning theory for analyzing its generalization bounds with general losses/tasks/models. The theory naturally leads to some feasible controlling strategies for ameliorating the quality of the extracted meta-learner, verified to be able to finely ameliorate its generalization capability in some typical meta learning applications, including few-shot regression, few-shot classification and domain generalization.

Keywords: Meta learning, statistical learning theory, few-shot learning, domain generalization, structural risk minimization

1. Introduction

The core goal of machine learning is to learn the inherent generalization rule underlying data from some empirical observations, so as to make label predictions for new query samples.

∗. Corresponding author

Table 1: Taxonomy of some typical recent literatures on meta learning based on their specified hyperparameters to learn.

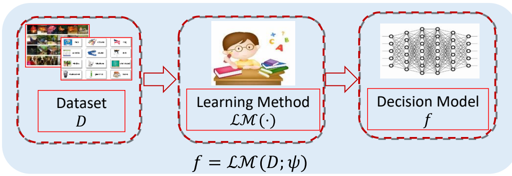
Taxonomy	Hyperparameters to learn in the learning process
Data collection	data simulator (Ruiz et al., 2018), dataset distillation (Wang et al., 2018), instance weights (Shu et al., 2019), label corrector (Wu et al., 2021; Zheng et al., 2021), exploration policy (Xu et al., 2018; Garcia and Thomas, 2019), noise generator (Madaan et al., 2020) data annotator (Konyushkova et al., 2017), data augmentation policy (Cubuk et al., 2019)
Model construct	neural architecture (Zoph and Le, 2017; Liu et al., 2019), activation function (Ramachandran et al., 2017), feature modulation function (Ryu et al., 2020; Wang et al., 2020a), neural modules (Alet et al., 2019) dropout (Lee et al., 2020), neural processes (Garnelo et al., 2018; Requeima et al., 2019), attention (Kanika et al., 2021), batch normalization (Bronskill et al., 2020; Yingjun et al., 2021)
Loss function preset	loss predictor (Houthooft et al., 2018; Huang et al., 2019; Gonzalez and Miikkulainen, 2020) metric (Sung et al., 2018; Lee and Choi, 2018), auxiliary loss (Li et al., 2019b; Veeriah et al., 2019), critic predictor (Sung et al., 2017), robust loss (Shu et al., 2020a,b), regularization (Balaji et al., 2018)
Algorithm design	gradient generator (Andrychowicz et al., 2016; Wichrowska et al., 2017; Ravi and Larochelle, 2017) initialization (Finn et al., 2017; Fakoor et al., 2020; Song et al., 2020), curvature (Park and Oliva, 2019), preconditioning matrix (Flennerhag et al., 2020), learning rate schedule (Li et al., 2017b; Shu et al., 2020c)

Such a generalization rule is generally modeled as a parameterized function (i.e., learner), and extracted from a pre-specified function set (i.e., learning machine) (Jordan and Mitchell, 2015). In the recent decade, deep learning approaches, equipped with highly parameterized learning machine (deep neural network architectures), have made great successes in a variety of fields (He et al., 2016; Silver et al., 2016; Devlin et al., 2019), strongly substantiating the validity of this learning framework.

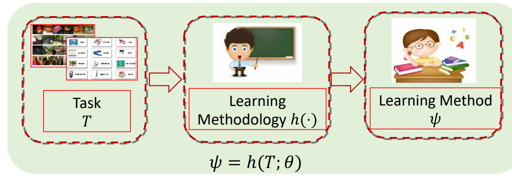
However, nowadays gradually more deficiencies have been emerging for this conventional machine learning framework. On the one hand, its success largely relies on vast quantities of pre-collected annotated data, and simultaneously huge computation resources. However, most applications in real world have intrinsically rare or expensive data, or limited computation resources. This inclines to largely degenerate the capability of conventional machine learning, especially deep learning, expected by general users. On the other hand, current deep learning methods are always designed with complicated architectures and possess huge amounts of hyperparameters, making them easily trapped into the overfitting issues, and possibly perform poorly on the test domain.

The above limitations can be mainly attributed to highly complicated hyperparametric configurations involved in almost every single component of the learning process, e.g., data screening, model constructing, loss function presetting and algorithm designing, etc. (see Section 3.1 and Table 1), for handling a practical learning task. If we take the entire learning process as an implicit function[1] mapping from an input training dataset to a decision model (i.e., a learner extracted from the learning machine), as shown in Fig. 1(a), these hyperparameters involved in the entire learning process then constitute

1. In many conventional literatures on learning theory, this implicit function is often called a learning algorithm (Maurer and Jaakkola, 2005; Chen et al., 2020). Yet it intrinsically represents the whole learning process from input data to output learner. In current machine learning, this process should contain more general learning component settings besides the pre-specified learning algorithm itself, like the designing of network architecture and loss function. To avoid possible clutters of readers, in this paper we call it as the function of learning method, or $\mathcal{LM}(\cdot)$ briefly.



(a) A learning method producing a decision model from an input dataset.



(b) A learning methodology producing a learning method from a query task.

Figure 1: (a) depicts the executive process when we deal with a machine learning problem. The learning method can be seen as an implicit function mapping from an input training dataset to a decision model. This function represents the entire learning process equipped with proper hyperparameter settings, guaranteeing the machine learning system to produce a decision function by executing it. (b) is the principle of how to determine the learning method. There exists a common learning methodology among various learning tasks, which can be seen as a function that maps an input task to the corresponding learning method. Once we achieve the function, it can allocate a proper learning method for a specific query task. The machine learning system can then automatically produce the decision model for the task by running learning pipeline (a).

the parameters of this function. Their proper settings essentially determine the ultimate capability, especially the generalization, of the extracted learner by this function through implementing the corresponding learning process equipped with these hyperparameters. It has been gradually more widely recognized nowadays that appropriately specifying these hyperparameters contained in the entire learning process has become significantly more difficult, time-consuming and laborious than directly running a well designed learning process itself.

Conventional machine learning literatures have raised many elegant approaches for such hyperparameter selection issue, like Akaike Information Criterion (AIC) (Akaike, 1974), Bayesian Information Criterion (BIC) (Schwarz et al., 1978), Minimum Description Length (MDL) (Barron and Cover, 1991), and validation set based approaches (Stone, 1974). Their availability, however, is mostly restricted to the problems with relatively small-scale hyperparametric structures. When facing complicated and massive hyperparametric configurations, especially those related to a deep neural network, these concise manners are generally incapable of taking effect. Instead, in most cases the task still highly relies on manual attempts by human experts to the problem. By deeply understanding the problem and accumulating heuristic experience of hyperparameter tuning, such human-designed manner does be able to make effect to specific tasks. However, when the task is with dynamic variations, the learning method is always required to be re-designed from scratch based on the new understandings of humans to the varying task. Especially, a learning method with carefully modulated hyperparameters might be excessively good to the investigated learning tasks, while hardly to be readily generalized to a new query task with insightful correlations but also evident variations, like data modalities, network architectures, loss formulations and utilized algorithms, with the trained tasks. Then such laborsome process for hyperparameter tuning has to be started again, which then inclines to result in most core issues encountered by current machine learning.

To alleviate such deficiencies of current machine learning, one natural idea is to obtain an explicit hyperparameter setting policy for predicting proper hyperparameters when adapted to new query tasks, as shown in Fig. 1(b). Specifically, equipped with such an explicit function, a novel machine learning system can automatically produce proper learning method for different tasks without need of much extra human intervention. More strictly speaking, what we want is a mapping from the learning task space to the hyperparameter space covering the whole learning process. From this perspective, learning an automatic hyperparameter setting policy conditioned on tasks is expected to get the "learning methodology" shared among various learning tasks, and directly use it to allocate learning method for new tasks. In this way, such a "learning methodology function" is hopeful to be employed to finely adapt varying query tasks with less computation/data costs, as well as fewer human interventions.

Meta learning, or learning to learn, provides a promising solution path for this "methodology learning" objective. The realization strategy is to learn the common hyperparameter specification principle among a set of training tasks, instead of training samples as conventional machine learning. The aim is to get the hyperparameter setting rule shared by training tasks, which is thus expected to finely generalize to new query tasks. In the recent years, many meta learning studies have been raised, constructed on specific problems, e.g., AutoML (Yao et al., 2018) and algorithm selection (Vanschoren, 2018), or particular applications, e.g., few-shot learning (Finn et al., 2017; Wang et al., 2020c; Shu et al., 2018), neural architecture search (NAS) (Elsken et al., 2019), or hyperparameter optimization (Franceschi et al., 2018). Typical researches along this line are listed in Table 1.

Recent studies on meta learning, however, still have two major issues. Firstly, many of current meta learning works put emphasis on learning fixed hyperparameters shared from the training tasks, and then directly applying them to adapt to new query tasks. Typical works include MAML (Finn et al., 2017) and its variants (Li et al., 2017b; Nichol et al., 2018; Antoniou et al., 2019; Finn et al., 2019), hyperparameter optimization (Franceschi et al., 2018), AutoML (Yao et al., 2018), and also the meta learning framework summarized in the recent excellent survey work in (Hospedales et al., 2020). Relying on the shared hyperparameters is challenging for complex task distributions (e.g., those with distribution shift), since different tasks may require to set substantially different hyperparameters. This makes it always infeasible to find a set of common hyperparameters performable for all tasks.

Secondly, although some solid theoretical justifications have been presented along this research line, most existing meta learning theories are developed under conventional machine learning framework and put emphasis on evaluating the generalization capability of the traditional learning model (i.e. the learner). This is somehow deviated from the current meta learning practice with innovatory support/query episodic training mode (Vinyals et al., 2016)[2]. Besides, most previous theoretical works seldom mention theory-inspired regularization strategies for guiding meta learning, thus are less functioned to feedback meta learning models for helping improve their practical generalization performance.

Against the above issues, in this work we treat meta learning as learning an explicit hyperparameter prediction policy conditioned on learning tasks. Especially, just succeeded

2. More details are presented in Section 4.

from the conventional machine learning framework, this policy is modelled as a parameterized function, called meta-learner, mapping from a learning task to its properly hyperparameter configuration, and extracted from a pre-specified function set, called meta learning machine. On the one hand, such meta-learner setting facilitates a better flexibility of the learned hyperparameters adaptable to diverse query tasks than fixed hyperparameters. Specifically, the extracted meta-learner can be readily used to produce proper hyperparameters conditioned on new query tasks. On the other hand, such meta learning with explicit hyperparameter prediction policy conditioned on learning tasks setting can be seen as a substantial but homologous extension from the conventional machine learning framework imposed on the pre-specified learning machine, and the corresponding statistical learning theory can be subsequently derived. Especially, similar to the structural risk minimization (SRM) principle in the conventional statistical learning theory (Vapnik, 1999; Shawe-Taylor et al., 1998), some beneficial theoretical results can then be achieved for revealing and controlling the intrinsic generalization capability of the preset meta learning machine. In summary, the main contributions of this work are as follows:

(1) We interpret meta learning as learning an explicit hyperparameter prediction policy conditioned on learning tasks. This formulation provides a general framework to understand meta learning, and reveals that the essential character of meta learning is to construct a meta-learner for simulating the learning methodology, and transferably use it to help fulfill new query tasks.

(2) We introduce a problem-agnostic definition of meta-learner to realize learning the hyperparameter prediction policy conditioned on learning tasks. The parameterized meta-learner can be easily integrated into the traditional machine learning framework to provide a fresh understanding and extension of the original machine learning framework. We further provide generalization bounds for the new framework with general losses, tasks, and models.

(3) We provide general-purpose bounds with decoupling the complexity of learning the task-specific learner from that of learning the task-transferrable meta-learner. The meta-learner is extracted from a pre-specified function set (i.e., meta learning machine), and the theoretical results facilitate some feasible controlling strategies on the meta-learner, capable of being easily embedded into current off-the-shelf machine learning programs and yet helping generally improve its generalization capability.

(4) We highlight the utility of our meta learning framework for obtaining the learning guarantees of some typical meta learning applications, including few-shot regression, few-shot classification and domain generalization. The theory-induced control effects of the meta-learner are empirically verified to be effective for stably improving its generalization capability on new query tasks.

The rest of the paper is organized as follows. Related works are introduced in Section 2, and the proposed meta learning framework is formulated in Section 3. In Section 4, we derive the generalization bounds of this meta learning manner during the meta-training and meta-test stages, respectively. We further instantiate our general theoretical framework on few-shot regression in Section 5, few-shot classification in Section 6, and domain generalization in Section 7. The conclusion and discussion are finally made.

2. Related Work

While there are a large number of meta learning literatures recently emerging in the field, most of them focused on practical feasible techniques against certain problems. In this section we mainly review the related studies considering the intrinsic understanding of the fundamental meta learning concepts as well as its basic learning theories. More related papers can be referred to in the recently proposed comprehensive surveys (Vanschoren, 2018; Hospedales et al., 2020).

Meta learning has a long history in psychology (Ward, 1937), and was described by (Maudsley, 1980) as "the process by which learners become aware of and increasingly in control of habits of perception, inquiry, learning, and growth that they have internalized". Then Schmidhuber (1987); Bengio et al. (1990) introduced it into computer science to train a meta-learner that learns how to update the parameters of the learner. Afterwards, this approach has been applied to learning to optimize deep networks (Andrychowicz et al., 2016). Besides, Vilalta and Drissi (2002) used meta learning to improve the learning bias dynamically through experience by the continuous accumulation of meta-knowledge. Lemke et al. (2015) further employed meta learning to extract meta-knowledge from different domains or problems. A recent survey paper of (Vanschoren, 2018) mainly attributed the capability of meta learning to its leveraging prior learning experience, and interpreted that it can learn new tasks more quickly. These literatures, however, have not presented a general mathematical formulation for meta learning, which could yet be useful to clearly help distinguish meta learning from previous related learning manners, like transfer learning and multi-task learning.

Very recently, Hospedales et al. (2020) proposed a comprehensive survey paper, introducing a taxonomy of meta learning along three independent axes, i.e., meta-representation, meta-optimizer, and meta-objective. However, as aforementioned, this paper summarized the task of meta learning as learning fixed hyperparameters instead of a hyperparameter-prediction-policy, making the learned methodology with insufficient flexibility adapt to new query tasks. Several recent works (Wang et al., 2020b; Denevi et al., 2020; Rusu et al., 2019; Vuorio et al., 2019; Yao et al., 2019; Wang et al., 2019) addressed the issue by conditioning hyperparameters on target tasks. While they mainly pay attention to specific initialization setting issue (Finn et al., 2017), our formulation considers the general hyperparameter setting problem related to the entire learning process for machine learning, and aims to develop learning theory for theoretically analyzing task-generalization insight involved in general meta learning problems.

As for the theoretical research on meta learning, it can date back to (Baxter, 2000), which provided generalization bounds for transfer learning using covering numbers. Afterwards, Maurer et al. (2016); Du et al. (2021); Tripuraneni et al. (2020) proposed a general framework for obtaining the generalization bound, and made analyses on the benefit of representation learning for reducing the sample complexity of the target task. Besides, the PAC-Bayes based (Pentina and Lampert, 2014, 2015; Amit and Meir, 2018) and stability theory based (Maurer and Jaakkola, 2005; Chen et al., 2020) generalization bounds have also been studied. Another recent line of theoretical work analyzed gradient-based meta learning methods (MAML) (Finn et al., 2017), and showed theoretical guarantees for convex losses by using tools from online convex optimization (Denevi et al., 2019a,b; Finn et al., 2019; Khodak et al., 2019; Balcan et al., 2019).

The generalization bounds we derive have several differences compared with the aforementioned theories. Specifically, previous works assume the traditional learning model to train the meta-learner, which is somehow deviated from the current meta learning practice with support/query episodic training mode such as MAML (Finn et al., 2017) and ProtoNet (Snell et al., 2017). Comparatively, we derive the bounds based on the commonly used support/query meta-training manner in meta learning practice. Besides, although previous works achieve solid theoretical justifications, they seldom mentioned theory-inspired regularization terms to conduct practically feasible controlling strategies for training the meta-learner, and thus might not be directly used as guidelines for improving the generalization performance of meta learning algorithms. In this paper, we try to bridge the gap between meta learning theory and its practical use, and employ the theory-induced regularization strategy for meta-learner to improve its generalization ability. The effectiveness of such strategy has been empirically verified with example applications on few-shot regression, few-shot classification and domain generalization, as detailed in Sections 5, 6 and 7, respectively.

3. Exploring a Task-Transferable Meta-learner for Meta-learning

For clarity, in the following we focus on supervised learning. We firstly recall the conventional machine learning framework, and then present the corresponding understanding for meta learning one, naturally succeeded from the former. A comparison with conventional multi-task/transfer learning manners is then presented.

Let's first introduce some necessary notations. $\mathbf{X}^\mathsf{T}$ denotes the transpose of a matrix $\mathbf{X}$. Let $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ be the data space, where $\mathcal{X} \subset \mathbb{R}^d$ and $\mathcal{Y} \subset \mathbb{R}$ (regression) or $\mathcal{Y} = \{0, 1, \cdots, K-1\}$ (multi-class classification) are the input and output spaces, respectively. We use the bracketed notation $[k] = \{1, 2, \cdots, k\}$ as shorthand for index sets. The norm $\|\cdot\|$ appearing on a vector or a matrix refers to its ℓ_2 norm or Frobenius norm.

3.1 Learning a Learner from Learning Machine

Machine learning (Jordan and Mitchell, 2015) aims to extract a learner (decision model) from a pre-specified learning machine based on a set of training observations. It generally includes the following ingredients.

Training dataset. It is usually composed of a finite paired dataset $D = \{(x_i, y_i), i \in [n]\}$ drawn i.i.d. from a task (probability distribution) μ, which simulates the input and output of the learner to be estimated.

Learner and learning machine. The learner corresponds to a mapping $f(x; \omega) : \mathcal{X} \to \mathcal{Y}$, which is requested to produce a prediction rule from $x \in \mathcal{X}$ to its label $y \in \mathcal{Y}$, and $\omega \in \Omega$ represents the parameters of f to be estimated. f is chosen from a preset learning machine (i.e., hypothesis space) $\mathcal{F}$, constituting a set of candidate learners.

Performance measure. The performance is measured by a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$, which reflects the extent of how well the learner fits training data. The expected risk $R_\mu(f)$ and empirical risk $\hat{R}_D(f)$ are respectively defined as:

$$R_\mu(f) = \mathbb{E}_{(x,y)\sim\mu}\ell(f(x), y), \quad \hat{R}_D(f) = \frac{1}{n}\sum_{i=1}^{n}\ell(f(x_i), y_i). \tag{1}$$

Optimization algorithm. The learning algorithm A is employed from optimization toolkits to extract a learner f from $\mathcal{F}$ guided by the performance measure. E.g., the commonly used algorithm for deep learning methods is SGD or Adam (Goodfellow et al., 2016).

The overall learning process of machine learning is shown in Fig.2(a). Actually, before executing machine learning for a given learning task in practice, it needs to firstly determine complicated hyperparametric configurations involved in all components of the learning process. Then the whole machine learning process can be executed to produce the decision model. If we take the entire learning process as an implicit function mapping from an input training dataset to a decision model, all involved hyperparameters then constitute the parameters of this function. We can represent this function as $\mathcal{LM}(D;\theta) : \mathcal{D} \to \mathcal{F}$. We denote ψ as all hyperparameters involved in the learning method[3]. For example, we can write $\psi = (\psi_D, \psi_f, \psi_\ell, \psi_A)$ to record hyperparameters in the learning configurations with respect to the data D, the learner f, the loss function ℓ and the optimization algorithm A, respectively. The ultimate capability, especially the generalization, of the decision model then essentially depends on the proper hyperparameter settings ψ.

After the learning process outputs the decision model, it can produce the label predictions for new query samples. Current great successes of deep learning have strongly substantiated the validity of this learning framework for summarizing the underlying label prediction rules from data. Assume the optimal learner underlying the task μ is $f^* = \arg\min_{f\in\mathcal{F}} R_\mu(f)$, and the estimated learner through minimizing the empirical risk from the training data D is $\hat{f} = \arg\min_{f\in\mathcal{F}} \hat{R}_D(f)$. Then we generally use $R_\mu(\hat{f}) - R_\mu(f^*)$ to measure the "closeness" between the estimated $\hat{f}$ and the optimal f^*, whose upper bound has been proved as follows (Mohri et al., 2018):

Theorem 1 *Let $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ be the data space, hypothesis (learning machine) $\mathcal{F}$ be the function class of the mapping $f : \mathcal{X} \to \mathcal{Y}$, $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, B]$ be the loss function. Assume that the loss $\ell(\cdot, y)$ is L-Lipschitz for any $y \in \mathcal{Y}$. Then for any $\delta > 0$, with probability at least $1 - \delta$, we have*

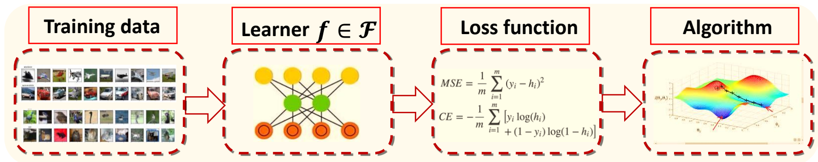
$$R_\mu(\hat{f}) - R_\mu(f^*) \leq 6L\mathcal{G}_n(\mathcal{F}) + 2B\sqrt{\frac{\ln(1/\delta)}{2n}}, \tag{2}$$

where $\mathcal{G}_n(\mathcal{F}) := \mathbb{E}_{D\sim\mu^n}[\hat{\mathcal{G}}_D(\mathcal{F})]$ is the Gaussian complexity, and $\hat{\mathcal{G}}_D(\mathcal{F})$ is the empirical Gaussian complexity of $\mathcal{F}$ w.r.t D defined as

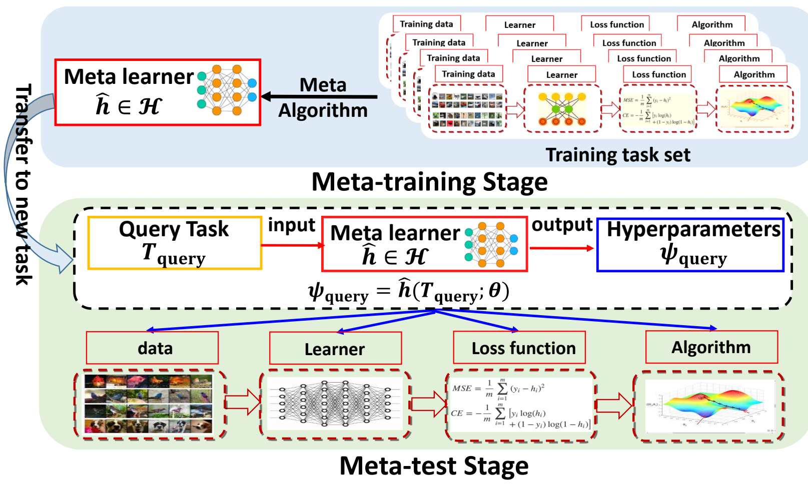
$$\hat{\mathcal{G}}_D(\mathcal{F}) := \mathbb{E}_{\mathbf{g}}\left[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} g_i f(x_i)\right], \; g_i \sim \mathcal{N}(0, 1)$$

where $\mathcal{N}(0, 1)$ is the Gaussian distribution with zero mean and unit variance.

3. In practice, the hyperparameters to be learned by meta learning generally only contain a certain subset of hyperparameters among all involved ones in the learning process. Some typical hyperparameters meta learning aims to learn are listed in Table 1.



(a) Machine learning framework.



(b) Task-Transferable Meta-learner for Meta Learning.

Figure 2: (a) A general machine learning framework. (b) The meta-training and meta-test stages of a meta learning framework.

Since $\mathcal{G}_n(\mathcal{F}) \sim \sqrt{C(\mathcal{F})/n}$ (Mohri et al., 2018), where $C(\cdot)$ measures the intrinsic complexity of the function class (e.g., VC dimension), Theorem 1 delivers the information that the distance between $\hat{f}$ and f^* will decrease as n becomes large for a given learning machine $\mathcal{F}$.

3.2 Learning a Meta-learner from Meta-learning Machine

Our aim is then to explore an explicit hyperparameter prediction policy for predicting proper hyperparametric settings when adapted to new query tasks. Equipped with such an explicit function, the novel machine learning system is expected to automatically produce proper hyperparameters in the learning method for different tasks without need of extra human intervention. Such an explicit function can be represented as $h(T;\theta) : \mathcal{T} \to \Psi$, called meta-learner, which maps from the learning task space $\mathcal{T}$ to the hyperparameter space Ψ. Then the goal of meta learning is to determine the parameter $\theta \in \Theta$ contained in the meta-learner h.

Different from the learner $f(x;\omega)$ in machine learning, attempting to extract the label prediction rule among data, the meta-learner $h(T;\theta)$ is requested to extract the hyperparameter prediction rule for learning process shared by training tasks, which is then expected to finely generalize to new query tasks to specify their hyperparameters for a learning method. To achieve such a meta-learner in practice, we usually assume access to the following ingredients corresponding to the conventional machine learning:

Training task set. It usually assumes access to a set of source tasks $\Gamma = \{\mu_t = \{\mu_t^s, \mu_t^q\}, t \in [T]\}$, to learn the meta-learner mapping, where μ_t denotes the probability distribution for the t-th task and i.i.d. sampled from an environment distribution η (Baxter, 2000). Here μ_t^s and μ_t^q denote the distributions sampling support/training set D_t^{tr} and query/validation dataset D_t^{val} for the task, respectively. This follows the support/query meta learning setting proposed by (Vinyals et al., 2016). Specifically, for the t-th task, it is composed of $D_t = (D_t^{tr}, D_t^{val})$, where $D_t^{tr} = \{(x_{ti}^{(s)}, y_{ti}^{(s)})\}_{i=1}^{m_t} \sim (\mu_t^s)^{m_t}, D_t^{val} = \{(x_{tj}^{(q)}, y_{tj}^{(q)})\}_{j=1}^{n_t} \sim (\mu_t^q)^{n_t}$, and m_t and n_t are the sizes of training set and validation set for the t-th task, respectively. We denote $\boldsymbol{D} = \{D_t\}_{t=1}^T$ as the entire training task dataset.

Meta-learner and meta learning machine. Formally, the meta-learner $h(T;\theta) : \mathcal{T} \to \Psi$ is required to extract the rule from $T \in \mathcal{T}$ to its hyperparameter setting $\psi \in \Psi$ for the learning method, and $\theta \in \Theta$ is the parameter contained in h to be estimated. h is selected from a pre-designed parameterized function set $\mathcal{H}$, called meta learning machine, constituting a function class of candidate meta-learners h.

Generally, the input of a meta-learner h is a representation conveyed by the task $T \in \mathcal{T}$. It can be some static information for task, e.g., directly obtained from the provided dataset D for the task, or dynamic knowledge extracted from the learning process for handling task T, e.g., gradient information (Andrychowicz et al., 2016), loss information (Shu et al., 2019), feature information (Li et al., 2019b), etc. Since the emphasis of this study is mainly on the mathematical formulation and the learning theory of meta learning, for simplicity and convenience, we just consider the static case as task representation in this paper. Thus we rewrite the meta-learner mapping as $h(D;\theta)$ in the following. For simplicity, we often briefly denote it as h.

Performance measure. Our goal is to learn a common hyperparameter prediction policy suitable for a family of tasks. For intuitive understanding, we rewrite the empirical risk $\hat{R}_D(f)$ as $\mathcal{L}(f,D) : \mathcal{F} \times \mathcal{D} \to \mathbb{R}^+$ to characterize the performance of the decision model on the task. Our aim is then to seek the best hyperparameter prediction policy h, which is employed to set the learning process on D^{tr}, so that the obtained learner can achieve optimal generalization performance on D^{val}. Specifically, the risk so described can be formally written as:

$$R_\eta(h) = \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D^{val}\sim(\mu^q)^n}\mathbb{E}_{D^{tr}\sim(\mu^s)^m}\mathcal{L}(\mathcal{LM}(D^{tr};h), D^{val}), \tag{3}$$

where $\mu = (\mu^s, \mu^q)$ is the task distribution sampled from an environment distribution η, and D^{tr} and D^{val} are training and validation sets i.i.d. generated from μ^s and μ^q, respectively.

In practice, we have access to a set of training task set $\boldsymbol{D}$, and the meta-learner is often learned by (approximately) minimizing the following expected empirical risk on the meta learning task:

$$\hat{R}_{\boldsymbol{D}}(h) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{LM}(D_t^{tr};h), D_t^{val}). \tag{4}$$

Optimization algorithm. The goal then is to achieve a meta-learner h from $\mathcal{H}$ by minimizing the empirical task average risk $\hat{R}_{\boldsymbol{D}}(h)$. There are multiple effective optimization algorithms designed for solving the problem, e.g., the bilevel optimization techniques (Finn et al., 2017; Shu et al., 2019; Hospedales et al., 2020).

Contrary to conventional machine learning whose performance can be directly computed on the given annotated data, meta learning algorithm is relatively hard to directly compute

Table 2: Comparison of the main notations used in conventional machine learning and the proposed meta learning frameworks.

Compared concepts	Machine Learning	Meta Learning
Source distribution	Task distribution μ	Environment η
Training instance	$D = \{(x_i, y_i)\}_{i=1}^n \sim \mu^n$	$\boldsymbol{D} = \{D_t = (D_t^{tr}, D_t^{val})\}_{t=1}^T, D_t^{tr} \sim (\mu_t^s)^{m_t}$, $D_t^{val} \sim (\mu_t^q)^{n_t}, \mu_t = (\mu_t^s, \mu_t^q) \sim \eta$
Test instance	$(x, y) \sim \mu$	$D_\mu^{tr} \sim (\mu^s)^{m_\mu}, (x, y) \in \mu^q, \mu = (\mu^s, \mu^q) \sim \eta$
Learning objective	Learner $f : \mathcal{X} \to \mathcal{Y}$	Meta-learner $h : \mathcal{T} \to \Psi$
Output of the (meta)-learner	$y = f(\mathbf{x}; \omega)$	$\psi = h(T; \theta)$
Performance measure	$R_\mu(f) = \mathbb{E}_{(x,y)\sim\mu}\ell(f(x), y)$	$R_\eta(h) = \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D^{val}\sim(\mu^q)^n}\mathbb{E}_{D^{tr}\sim(\mu^s)^m}$ $\mathcal{L}(\mathcal{LM}(D^{tr}; h), D^{val})$
Goal of the generalization	Predicting label y for query sample x	Predicting learning method ψ for query task T

the performance on given tasks. Generally, it needs a two-stage procedure to evaluate the performance of a meta-learner (Finn et al., 2017; Shu et al., 2019). Firstly, a meta-learner predicts the configuration of the hyperparameters after observing the training set D_t^{tr} in each task of the task set $\boldsymbol{D}$. Then the machine learning system equipped with such hyperparameters can be executed to automatically produce the proper decision model for D_t^{tr}. Then in the second stage, the task loss can be computed as the average over all errors calculated on the validation set $D_t^{val}, t \in [T]$ with respect to the corresponding decision model obtained from D_t^{tr}, reflecting the hyperparameter prediction capability of such meta-learner. This process actually is exactly expressed as the empirical task average risk $\hat{R}_{\boldsymbol{D}}(h)$ as formulated above. As shown in Fig.2(b), after achieving the meta-learner h, we can furtherly transfer it to new query tasks, which can help set the learning process (i.e., parameter of learning method) for new query tasks.

The above meta learning framework is evidently succeeded from but also with evident difference with conventional machine learning framework. To better clarify this point, we list in Table 2 the main notations of both frameworks to easily compare their main differences.

It should be noted that in the previous researches, meta learning is interpreted to improve performance by learning 'how to learn' (Thrun and Pratt, 1998). The main study of this work is expected to solidify such 'learning to learn' manner as learning an explicit hyperparameter prediction policy (i.e., a meta-learner) conditioned on learning tasks. From this view, a meta-learner helps formulate the learning method to tell the machine learning system how to automatically learn the decision model. Thus the insight of this formulation is to simulate humans to master the tricks for the learning methodology, and most importantly, then transfer it to help fulfill new tasks.

Besides, note that the optimization variable now is the parameters of the meta-learner, rather than the hyperparameters to be estimated. Actually, many of current meta learning algorithms directly learn the latter, e.g., weights initialization (Finn et al., 2017; Nichol et al., 2018; Rusu et al., 2019; Antoniou et al., 2019; Finn et al., 2019), learning rate (Li et al., 2017b; Baydin et al., 2019), and other hyperparameters (Franceschi et al., 2018). These works have been comprehensively introduced in the recent excellent survey paper (Hospedales et al., 2020). Relying on the shared and unique hyperparameters is challenging for adapting complex task distributions, especially those with distribution shift. As can be seen from the above understanding, this issue is addressed through developing an explicit hyperparameter prediction policy. With such an explicit function mapping, the meta-learned meta-learner is expected to be with a better adaptability for query task variations in setting proper configurations of hyperparameters than the fixed and shared ones. In fact, when we set the meta-learner as a constant function, these hyperparameter-fixed approaches can be categorized as a special case of this more general framework.

3.3 Comparison with Multi-task/Transfer Learning

Here we want to shortly clarify the main difference of the meta learning framework as aforementioned from conventional multi-task/transfer learning approaches. Although the latter also makes use of the intrinsic relationship among multiple tasks for improving the generalization performance for the learning results, they still fall into the scope of conventional machine learning, which considers the problem at the data/learner level. Specifically, most conventional multi-task/transfer learning works aim to improve the task learning performance by virtue of ameliorating the parameters of multiple learners imposed on different tasks by leveraging their underlying relevance. Meta-learning, however, considers the problem at the task/meta-learner level, and aims to learn the common hyperparameter setting policy imposed on the parameters of one single meta-learner functioned on different tasks. This way inclines to better improve its flexibility, available range and potential capability in practice than conventional learning manners. In particular, a feasible meta learning method allows the learners on different training/test tasks with very different forms, like variant input data modalities (Cubuk et al., 2019), model architectures (Zoph and Le, 2017), feature dimensions (Ryu et al., 2020), etc, and only requires them to possess certain shared common knowledge in learning method setting, which yet should be hardly handled by conventional multi-task/transfer learning manners. Such consideration in the learning-methodology perspective inclines to make this learning manner be with a more widely potential usage than many previous machine learning fashions.

4. Learning Theory

We use two phases to characterize the learning paradigm of the above introduced meta learning framework. The first is the meta-training phase, aiming to learn the hyperparameter prediction policy, and the second is the meta-test phase, purposing to generalize this policy for setting learning methods on new query tasks. We firstly present some preliminaries, and then present learning theory results for meta-training and meta-test phases, respectively.

4.1 Preliminaries

We use the Gaussian complexity to measure the complexity of a function class. For a generic vector-valued function class $\mathcal{E}$, containing a set of functions $e : \mathbb{R}^d \to \mathbb{R}^r$. Given a data set $\mathbf{U} = \{\mathbf{u}_i \in \mathbb{R}^d, i \in [N]\}$ i.i.d. drawn from a task μ, the empirical Gaussian complexity is then defined as (Bartlett and Mendelson, 2002):

$$\hat{\mathcal{G}}_{\mathbf{U}}(\mathcal{E}) = \mathbb{E}_{\mathbf{g}} \left[\sup_{e \in \mathcal{E}} \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{r} g_{ik} e_k(\mathbf{u}_i) \right], \ g_{ik} \sim \mathcal{N}(0,1),$$

Algorithm 1 Meta-Training: Learning the Methodology

Input: Training task set $\Gamma = \{D_t, t \in [T]\}$.

Do:

(1) Run algorithm to obtain $\hat{\mathbf{f}}^{(h)} = (\hat{f}_1^{(h)}, \cdots, \hat{f}_T^{(h)})$, where $\hat{f}_t^{(h)}$ is obtained by calculating $\mathcal{LM}(D_t^{tr}; h(D_t^{tr}))$.

(2) Put $\hat{\mathbf{f}}^{(h)}$ into Eq.(4), and then minimize the error $\hat{R}_{\boldsymbol{D}}(\hat{\mathbf{f}}^{(h)})$ to obtain $\hat{h}$.

Return: The meta-learner $\hat{h}$.

where $\mathbf{g} = \{g_{ik}\}_{i\in[N],k\in[r]}$, and $e_k(\cdot)$ is the k-th coordinate of the vector-valued function $e(\cdot)$. The corresponding population Gaussian complexity is defined as $\mathcal{G}_N(\mathcal{E}) = \mathbb{E}_{\mathbf{U}\sim\mu^N}[\hat{\mathcal{G}}_{\mathbf{U}}(\mathcal{E})]$, where the expectation is taken over the distribution of $\mathbf{U}$. In comparison, for empirical Rademacher complexity $\hat{\mathfrak{R}}_{\mathbf{U}}(\mathcal{E})$, g_{ik} are replaced by uniform $\{-1, 1\}$-distributed variables, and the corresponding population Rademacher complexity is $\mathfrak{R}_N(\mathcal{E}) = \mathbb{E}_{\mathbf{U}\sim\mu^N}[\hat{\mathfrak{R}}_{\mathbf{U}}(\mathcal{E})]$.

To prove the main learning theory results, we require the following assumptions, all being usually satisfied.

Assumption 1 (Bounded Inputs) *$\mathcal{X} \subset \mathcal{B}(0, R)$, for $R > 0$, where $\mathcal{B}(0, R) = \{x \in \mathbb{R}^d : \|x\| \leq R\}$.*

Assumption 2 (Bounded and Lipschitz Loss Function) *The loss function $\ell(\cdot,\cdot)$ is B-bounded, and $\ell(\cdot, y)$ be L-Lipschitz for any $y \in \mathcal{Y}$.*

4.2 Meta-Training Stage: Learning the Hyperparameter Prediction Policy

In the meta-training stage, the training task dataset $\boldsymbol{D} = \{D_t, t \in [T]\}$ is available for learning, where $D_t = (D_t^{tr}, D_t^{val})$, $D_t^{tr} \sim (\mu_t^s)^{m_t}, D_t^{val} \sim (\mu_t^q)^{n_t}, \mu_t = (\mu_t^s, \mu_t^q) \sim \eta$. We aim to learn the hyperparameter prediction policy from this task dataset. The overall learning process is listed in Algorithm 1. The quantity of interest in meta-training is the expectation of task average generalization error as follows:

$$R_{train}(h) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^{n_t}} \mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val}). \tag{5}$$

Denote the optimal meta-learner obtained by minimizing the theoretical risk $R_\eta(h)$ and empirical risk $\hat{R}_{\boldsymbol{D}}$ in meta learning as:

$$h^* = \arg\min_{h\in\mathcal{H}} R_\eta(h); \quad \hat{h} = \arg\min_{h\in\mathcal{H}} \hat{R}_{\boldsymbol{D}}(h), \tag{6}$$

respectively. where $R_\eta(h)$ and $\hat{R}_{\boldsymbol{D}}$ are defined in Eqs. (3) and(4), respectively. Then we can naturally use $R_{train}(\hat{h}) - R_{train}(h^*)$ to measure the "closeness" between the estimated $\hat{h}$ and the true underlying h^*, which captures the extent of how much the two meta-learners $\hat{h}$ and h^* differ in aggregating over the T training tasks.

We can then present a theoretical upper bound estimation for this "closeness" measure. Before showing the theorem, we first introduce an important notation, $d_{\mathcal{F}}(\mu_t^s, \mu_t^q)$, necessary for the theoretical result. $d_{\mathcal{F}}(\mu_t^s, \mu_t^q)$ denotes the discrepancy divergence (Ben-David et al.,

2010) between support and query data with respect to their sampled probability distributions μ_t^s and μ_t^q imposed on the hypothesis class $\mathcal{F}$:

$$d_{\mathcal{F}}(\mu_t^s, \mu_t^q) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{D_t^{val} \sim (\mu_t^q)^{n_t}} \mathcal{L}(f, D_t^{val}) - \mathbb{E}_{D_t^{tr} \sim (\mu_t^s)^{m_t}} \mathcal{L}(f, D_t^{tr}) \right|.$$

Theorem 2 *If Assumptions 1 and 2 hold, for any $\delta > 0$, with probability at least $1-\delta$, we have*

$$R_{train}(\hat{h}) - R_{train}(h^*) \leq 768L \log(4\sum_{t=1}^{T} n_t) \cdot L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) + \frac{6L}{T}\sum_{t=1}^{T} \hat{\mathcal{G}}_{D_t^{tr}}(\mathcal{F})$$
$$+ \frac{4}{T}\sum_{t=1}^{T} d_{\mathcal{F}}(\mu_t^s, \mu_t^q) + 6\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}} + \frac{6B}{T}\sum_{t=1}^{T}\sqrt{\frac{\log\frac{2}{\delta}}{m_t}} + \frac{12LDis(D^{val})}{(\sum_{t=1}^{T} n_t)^2}\sqrt{\sum_{t=1}^{T}\frac{1}{\beta_t T}},$$

where $Dis(D^{val}) = \sup_{h,h'} \rho_{D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')})$, $\rho_{D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')}) = \frac{1}{\sum_{t=1}^{T} n_t}\sum_{t=1}^{T}\sum_{j=1}^{n_t}(f_t^{(h)}(x_{tj}^{(q)}) - f_t^{(h')}(x_{tj}^{(q)}))^2$, $D^{val} = \{D_t^{val}\}_{t=1}^{T}$, $\mathbf{f}^{(h)} = (f_1^{(h)}, f_2^{(h)}, \cdots, f_T^{(h)})$, $f_t^{(h)} = \mathcal{LM}(D_t^{tr}; h)$, and $L(\mathcal{F})$ is the Lipschitz constant of $\mathbf{f}^{(h)}$ with respect to h, $\beta_t = \frac{n_t T}{\sum_{t=1}^{T} n_t}$.

The above bound for excess task average risk consists of three dominant terms. They can be interpreted as: the complexity of learning the meta-learner h (the first term), the complexity of learning the task-specific learners $\mathbf{f}$ (the second term), and the distribution shift between support and query sets (the third term). As aforementioned, it generally holds that $\hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) \sim \sqrt{C(\mathcal{H})/\sum_{t=1}^{T} n_t}$, and $\hat{\mathcal{G}}_{D_t^{tr}}(\mathcal{F}) \sim \sqrt{C(\mathcal{F})/m_t}$, where $C(\cdot)$ measures the intrinsic complexity of the function class (e.g., VC dimension). Thus the first term on the right hand side above is of the order $\tilde{\mathcal{O}}(1/\sqrt{\sum_{t=1}^{T} n_t})$, and the second term above is of the order $1/T\sum_{t=1}^{T}\mathcal{O}(1/\sqrt{m_t}) \leq \mathcal{O}(1/\sqrt{m}), m = \min\{m_1, \cdots, m_T\}$, where $\tilde{\mathcal{O}}$ denotes an expression that hides polylogarithmic factors in all problem parameters. Note that the leading term capturing the complexity of learning the meta-learner h decays in terms of the number of query samples ($\sum_{t=1}^{T} n_t$) among all training tasks. This implies that even with insufficient number of training tasks, improving the amount of query samples in the tasks can also be helpful to the final performance of the extracted meta-learner. Comparatively, such bounds deduced in some previous works, e.g., (Maurer et al., 2016), have mostly not involved this amount, and thus could not reflect such common sense fact that increasing the number of query samples in training tasks should be helpful for improving generalization of meta learning on new tasks. Actually, in many current meta learning applications, good performance can be obtained on the basis of not very large training task set (even only with one training task), e.g., hyperparameter learning (Franceschi et al., 2018), neural architecture search (NAS) (Elsken et al., 2019), etc. Therefore, our result could provide a more rational and comprehensive theoretical explanation for these methods.

Another point is necessary to be illustrated. Many of existing meta learning theories employ traditional empirical error to develop the error bounds (Maurer et al., 2016; Tripuraneni et al., 2020), whose training strategy is not episodic (i.e., support/query training strategy) (Vinyals et al., 2016). This will be somehow inapplicable for modern meta learning algorithm, e.g., gradient-based meta-algorithms (Finn et al., 2017; Franceschi et al., 2018). Besides, some recent theoretical investigations (Denevi et al., 2019b; Balcan et al., 2019) study model-agnostic meta-algorithm (Finn et al., 2017), and explore the convergence guarantees for gradient-based meta learning. Specifically, (Denevi et al., 2018, 2019a; Khodak et al., 2019) pay attention to specific meta-algorithms and propose the corresponding generalization guarantees. However, they study the case that the loss function is convex or the mapping is linear, which might be relatively hard to make analysis on deep neural network. Also, the training strategy studied is not episodic, which tends not to be easily used to train practical popular meta-algorithms. The most related work (Yin et al., 2020; Chen et al., 2020) considered the support/query episodic training strategy. Albeit beneficial to illustrate some generalization insight of meta learning, their theory has not been used to conduct some feasible regularization terms that can help improve the meta-learner. Different from the aforementioned works, we developed the theoretical results followed by the support/query training strategy. Besides, the theoretical guarantees contain the complexities of learning meta-learner h and learner $\mathbf{f}$, which can easily induce control effects of the meta-learner for improving generalization capability of meta-algorithms.

Particularly, it is seen that there exists a term $d_{\mathcal{F}}(\mu_t^s, \mu_t^q)$ in the error bound, which describes the distribution shift between training/suppot set and validation/query set among tasks. For applications without such domain shift, this term is zero and can be omitted. However, there are many meta learning applications with such support/query shift. A typical example is domain generalization, aiming to learn how to set a learning method implemented on the source domain data (i.e., support data), but able to yield a learner which could perform well on different target domain data (i.e., query data). To this purpose, various training tasks need to be collected, each containing the two simulated domains of data sets, and meta learning can then be executed to find such common learning methodology. Our theory can then be used to well explain the theoretical insight of meta learning in such cases.

4.3 Meta-Test Stage: Generalizing to New Query Tasks

In the meta-test stage, we aim to transfer the extracted meta-learner $\hat{h}$ to help set the learning method on new query tasks. As shown in Algorithm 2, in the meta-test stage, we have the training set $D_\mu^{tr} \sim (\mu^s)^{m_\mu}$, and for evaluating the performance we also assume to have a validation set $D_\mu^{val} \sim (\mu^q)^{n_\mu}$. The two sets are sampled from the query task $\mu = (\mu^s, \mu^q)$ drawn from the environment distribution η. Apply $\hat{h}$ obtained in the meta-training stage to set the learning process on D_μ^{tr} to obtain the decision model, and the test performance on D_μ^{val} reflects the generalization performance of the meta-learner $\hat{h}$ on the new task μ. In a nutshell, the test error $R_{test}(h)$ can be calculated as

$$R_{test}(h) = \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}}\mathcal{L}(\mathcal{LM}(D_\mu^{tr}; h)), D_\mu^{val}). \tag{7}$$

We further employ the following excess transfer risk to measure the "closeness" between the estimated $\hat{h}$ and the true underlying h^*, as defined in Eq. (6), in terms of helping fulfill new query task μ:

$$R_{test}(\hat{h}) - R_{test}(h^*).$$

Algorithm 2 Meta-Test: Generalization to New Query Tasks

Input: Query task $\mu = (\mu^s, \mu^q)$ drawn from environment η; meta-learned meta-learner $\hat{h}$.
Do:
 (1) Draw training set D_μ^{tr} from μ^s.
 (2) For given $\hat{h}$, calculate $\mathcal{LM}(D_\mu^{tr}; \hat{h})$ to obtain $\hat{f}_\mu$.
Return: The task-specific learner $\hat{f}_\mu$.

To build a bridge between the meta-training and meta-test processes, similar as recent works (Du et al., 2021; Tripuraneni et al., 2020), we make the following assumption:

Assumption 3 (Task diversity) *Given the meta learning machine $\mathcal{H}$, and $\hat{h}$ and h^* are defined in Eq. (6), it holds that*

$$R_\eta(\hat{h}) - R_\eta(h^*) \leq \alpha \left(R_{train}(\hat{h}) - R_{train}(h^*))\right) + \beta,$$

where $R_\eta(h)$ and $R_{train}(h)$ are defined in Eqs. (3) and (5), respectively, $\alpha \in \mathbb{R}$ represents the task diversity, and $\beta \in \mathbb{R}$ denotes the small additive error.

Actually, we do not have direct access to the underlying information of the meta-learner. One can only indirectly extract partial information from observed training tasks. It is anticipated that transferring the learning methodology will not be possible when the new query tasks are very different from those training ones. $R_\eta(\hat{h}) - R_\eta(h^*)$ presents to measure the closeness between the meta-trained meta-learner $\hat{h}$ and the underlying optimal meta-learner h^* in terms of an arbitrary new query task, and $R_{train}(\hat{h}) - R_{train}(h^*)$ defines this measure by virtue of the information extracted from training tasks. Thus task diversity essentially encodes the ratio of these two quantities, i.e., how well the training tasks can cover the space captured by the $\hat{h}$ needed to predict on new tasks. If all training tasks were quite similar, then it could only be expected that the meta-training stage can learn about a narrow slice of the learning methodology, which makes such task-transferring aim difficult. Generally, when the task diversity is large, then α is small; otherwise, large α implies relatively more similar training tasks.

We can then present the theoretical result for the meta-test phase of meta learning.

Theorem 3 *If Assumptions 1 - 3 hold, for any $\delta > 0$, with probability at least $1 - \delta$, we have*

$$\begin{aligned} R_{test}(\hat{h}) - R_{test}(h^*) \leq \alpha \left(R_{train}(\hat{h}) - R_{train}(h^*)\right) + \beta \\ + 6L\hat{\mathcal{G}}_{D_\mu^{tr}}(\mathcal{F}) + 2\mathbb{E}_{\mu\sim\eta} d_{\mathcal{F}}(\mu^s, \mu^q) + 6B\sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}}. \end{aligned} \tag{8}$$

Theorem 3 provides an excess transfer risk bound of the meta-learned $\hat{h}$ for query task μ with three dominant terms, i.e., one upper bounded by the task average excess risk in the meta-training stage, the complexity of learning the task-specific learner f_μ^* for new query task, as well as the distribution shift between training set and test set of query task.

In a word, the leading-order terms of the transfer risk for learning meta-learner h scales as $\tilde{\mathcal{O}}\left(\sqrt{C(\mathcal{H})/(\sum_{t=1}^{T} n_t)}+\sqrt{C(\mathcal{F})/m}+\sqrt{C(\mathcal{F})/m_\mu}\right), m=\min\{m_1,\cdots,m_T\}$. A naive algorithm which learns the new task in isolation, ignoring the training tasks, has an excess risk scaling $\mathcal{O}\left(\sqrt{(C(\mathcal{H})+C(\mathcal{F}))/m_\mu}\right)$. This theoretically explained the fact that when $\sum_{t=1}^{T} n_t$ are sufficiently large compared with m_μ (e.g., the setting of few-shot learning, m_μ is always relatively small), the performance of meta learning should be evidently better than the baseline of learning in isolation.

Since we introduce an explicit parameterized meta-learner to extract the hyperparameter prediction policy, it is easy to control and improve the learning of the meta-learner with proper regularization techniques, just like the SRM principle used in traditional machine learning (Vapnik, 2013), e.g., the large margin regularizer for learner. In the remaining section, we will instantiate our general meta learning framework for different meta learning applications, and substantiate how the generalization capability can be improved by theory-inspired regularization strategies.

5. Application I: Few-shot Regression

In this section, we instantiate the proposed meta learning framework for the few-shot regression model.

5.1 Basic Setting

Here we consider the few-shot regression setting where $\mathcal{X}=\mathbb{R}^d, \mathcal{Y}=\mathbb{R}$. Let $\|x\|\leq R, |y|\leq B, \forall x\in\mathcal{X}, y\in\mathcal{Y}$. The output of the meta-learner is the representation of each sample. The loss functions ℓ is chosen as the square loss $\ell(\mathbf{w}^\mathsf{T}h(x),y)=(y-\mathbf{w}^\mathsf{T}h(x))^2$. The task-specific learning machine $\mathcal{F}$ are chosen as linear regression function maps, and the meta learning machine $\mathcal{H}$ is parameterized as a depth-L vector-valued deep neural network (DNN) to extract the common representation for various regression tasks. Concretely, $h(x)$ can be written as:

$$h(x)=\phi_L(\mathbf{W}_L(\phi_{L-1}(\mathbf{W}_{L-1}\cdots\phi_1(\mathbf{W}_1 x)))), \tag{9}$$

where $\mathbf{W}_k, k\in[L]$ is the parameter matrix, and $\phi_k, k\in[L]$ is the activation function. Here we assume that the activation functions in all layers are element-wise 1-Lipschitz and zero-centered as assumed in (Golowich et al., 2018) which is typically satisfied, like the ReLU. Formally, $\mathcal{F}$ and $\mathcal{H}$ are writen as:

$$\begin{aligned}&\mathcal{F}=\{f|f(z)=\mathbf{w}^\mathsf{T}z, \mathbf{w}, z\in\mathbb{R}^{d_L}, \|\mathbf{w}\|\leq M\},\\&\mathcal{H}=\{h|h(x)\in\mathbb{R}^{d_L} \text{ as defined in } (9), x\in\mathcal{X}\}.\end{aligned} \tag{10}$$

Following the setting in (Finn et al., 2017), we assume that there are T tasks $\boldsymbol{D}=\{D_t\}_{t=1}^T$ available for learning, and $D_t=(D_t^{tr},D_t^{val})$, where $D_t^{tr}=\{z_{ti}^{(s)}=(x_{ti}^{(s)},y_{ti}^{(s)})\}_{i=1}^m, D_t^{val}=\{z_{tj}^{(q)}=(x_{tj}^{(q)},y_{tj}^{(q)})\}_{j=1}^n$. Here, the number of training/validation set samples for each task is identical. The few-shot regression model is then written as:

$$\begin{aligned} \boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \frac{1}{nT}\sum_{t=1}^{T}\sum_{j=1}^{n}\ell(\mathbf{w}_t^{*\mathsf{T}} h(x_{tj}^{(q)}), y_{tj}^{(q)}) \\ s.t.,\ \mathbf{w}_t^* = \arg\min_{\mathbf{w}_t}\frac{1}{m}\sum_{i=1}^{m}\ell(\mathbf{w}_t^T h(x_{ti}^{(s)}), y_{ti}^{(s)}), t\in[T], \end{aligned} \tag{11}$$

where $\boldsymbol{W} = \{\mathbf{W}_k, k\in[L]\}$ represents the collection of parameter matrices of h and $\mathbf{w}, h$ is chosen from $\mathcal{F}, \mathcal{H}$ as defined in Eq. (10). For notation convenience, we denote $\mathbf{P} = (\mathbf{w}_1,\cdots,\mathbf{w}_T)^{\mathsf{T}} \in \mathbb{R}^{T\times d_L}$, and $\mathbf{P}^* = (\mathbf{w}_1^*,\cdots,\mathbf{w}_T^*)^{\mathsf{T}}$ as its theoretical optimal solution.

5.2 Theoretical Analysis

In this part, we will instantiate Theorem 3 for few-shot regression model as defined in Eq. (11). The $Dis(D^{val})$ can be computed as

$$\begin{aligned} Dis(D^{val}) = \sup_{h,h'}\rho_{D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')}) \le 4\sup_{h, x\in\mathcal{X}}\|\mathbf{w}^{\mathsf{T}} h(x)\| \\ \le \sup_{h,x} 4M\|h(x)\| \le 4MD\cdot\prod_{k=1}^{L}\|\mathbf{W}_k\|, \end{aligned}$$

where $\rho_{D^{val}}(\cdot,\cdot)$ is defined in Theorem 2, and $\|h(x)\|$ is bounded by (Let $\mathbf{r}_k(\cdot)$ denote the vector-valued output of the k-layer for $k\in[L]$):

$$\begin{aligned} \|h(x)\| = \|\mathbf{r}_L(x)\| = \|\phi_L(\mathbf{W}_L\mathbf{r}_{L-1}(x))\| \\ \le \|\mathbf{W}_L\mathbf{r}_{L-1}(x)\| \le \|\mathbf{W}_L\|\|\mathbf{r}_{L-1}(x)\| \le D\prod_{k=1}^{L}\|\mathbf{W}_k\|. \end{aligned} \tag{12}$$

Now we can verify that Assumptions 1 - 3 holds. Observe that $\nabla_{\hat{y}}\ell(\hat{y}, y) = 2(\hat{y}-y) \le 2(B+M\|h(\mathbf{x})\|)$, and thus the loss function is Lipschitz continuous with $L = 2(B+M\|h(x)\|$, and $\|h(x)\|$ is bounded by Eq.(12). Besides, since $|\ell(\hat{y}, y)| \le B^2 + 2BM\|h(x)\| + M^2\|h(x)\|^2$, the loss function is bounded. The following result verifies that Assumption 3 holds,

Proposition 1 *Consider the few-shot regression model defined in Eq.(11), and the loss function $\ell(\cdot,\cdot)$ is chosen as the squared loss. The feature representation and the linear regression function are designed as in Eq.(10). Then Assumption 3 holds with $\alpha = \frac{M}{\sigma_{d_L}(\mathbf{K})}$ and $\beta = 0$, where $\mathbf{K} = \mathbf{P}^{*\mathsf{T}}\mathbf{P}^*/T$, $\sigma_{d_L}(\mathbf{K})$ denote the d_L-th singular value of matrix $\mathbf{K}$ at a decreasing order.*

It can be seen that α reflects the diversity of training task set. The larger the similarity of task-specific learners is (i.e., the smaller $\sigma_{d_L}(\mathbf{K})$ is), the large α value is. Namely, the larger the similarity of task-specific learners is, the harder the meta-learner could be transferablly used to new query tasks.

Now we can compute the leading-order terms in Eq.(8) for the parameterized meta-learner and learners as defined in Eq.(10). We firstly show the Rademacher complexity of $\mathcal{H}$ in the following theorem.

Theorem 4 (Theorem 1 in (Golowich et al., 2018)) *Let $\mathcal{H}$ be the class of real-valued DNN as defined in Eq.(9) while requiring $d_L = 1$ over $\mathcal{X} = \{\mathbf{x} : \|\mathbf{x}\| \leq R\}$, where each parameter matrix $\mathbf{W}_i$, $i \in [L]$ has Frobenius norm at most B_i, and the activation function $\phi_i, i \in [L]$ is 1-Lipschitz, with $\phi_i(0) = 0$, and applied element-wise. Then we have:*

$$\hat{\mathfrak{R}}_N(\mathcal{H}) \leq \frac{R\left(\sqrt{2\log(2)L}+1\right)\prod_{i=1}^{L} B_i}{\sqrt{N}}.$$

(1) For the Gaussian complexity of meta-learner h:

$$\begin{aligned}
\hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) &= \mathbb{E}\sup_{h\in\mathcal{H}} \frac{1}{nT}\sum_{t=1}^{T}\sum_{j=1}^{n}\sum_{k=1}^{d_L} g_{tjk} h_k(x_{tj}^{(q)}) \\
&\leq \sum_{k=1}^{d_L} \hat{\mathcal{G}}_{D^{val}}(h_k) \leq 2\sqrt{\log(nT)}\sum_{k=1}^{d_L}\hat{\mathfrak{R}}_{D^{val}}(h_k) \\
&\leq 2d_L\sqrt{\log(nT)} \cdot \frac{R\left(\sqrt{2\log(2)L}+1\right)\prod_{i=1}^{L} B_i}{\sqrt{nT}},
\end{aligned} \tag{13}$$

where the second inequality holds since $\hat{\mathcal{G}}_{\mathbf{X}}(\mathcal{H}) \leq 2\sqrt{\log(N)}\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{H})$ for any function class $\mathcal{H}$ when $\mathbf{X}$ has N datapoints (Ledoux and Talagrand, 2013).

(2) For the Gaussian complexity of the task-specific learner:

$$\begin{aligned}
\hat{\mathcal{G}}_{D_t^{val}}(\mathcal{F}) &= \mathbb{E}\sup_{\mathbf{w}\in\mathcal{F}} \frac{1}{m}\sum_{i=1}^{m} g_{ti}\mathbf{w}^{\mathsf{T}} h(x_{ti}^{(s)}) \\
&\leq \frac{\|\mathbf{w}\|}{m}\mathbb{E}\left\|\sum_{i=1}^{m} g_{ti} h(x_{ti}^{(s)})\right\| \leq \frac{\|\mathbf{w}\|}{m}\sqrt{\sum_{i=1}^{m}\left\|h(x_{ti}^{(s)})\right\|^2} \\
&\leq \frac{\|\mathbf{w}\|}{\sqrt{m}} \cdot \max_{x_{ti}^{(s)}\in D_t^{tr}} \|h(x_{ti}^{(s)})\|.
\end{aligned} \tag{14}$$

Thus, the transfer error defined in Eq.(8) now can be written as

$$\begin{aligned}
&R_{test}(\hat{f}_\mu, \hat{h}) - R_{test}(f_\mu^*, h^*) \\
&\leq \frac{M}{\sigma_{d_L}(\mathbf{K})}\left(768L\log(4nT)L(\mathcal{F})2d_L\sqrt{\log(nT)}\frac{R\left(\sqrt{2\log(2)L}+1\right)\prod_{i=1}^{L} B_i}{\sqrt{nT}}\right. \\
&\left.+\frac{6L\|\mathbf{w}\|}{\sqrt{m}T}\sum_{t=1}^{T}\max_{x_{ti}^{(s)}\in D_t^{val}}\|h(x_{ti}^{(s)})\| + 6B\sqrt{\frac{\log\frac{2}{\delta}}{2nT}} + 6B\sqrt{\frac{\log\frac{2}{\delta}}{m}} + \frac{48L\sup_{h,x} M\|h(x)\|}{n^2T^2}\right) \\
&+\frac{6L\|\mathbf{w}\|}{\sqrt{m_\mu}} \cdot \max_{x_i^{(s)}\in D_\mu^{val}}\|h(x_i^{(s)})\| + 6B\sqrt{\frac{\log\frac{2}{\delta}}{m_\mu}},
\end{aligned} \tag{15}$$

where $\sigma_1(\mathbf{X}), \cdots, \sigma_r(\mathbf{X})$ denote the singular values of a rank r matrix $\mathbf{X}$ at a decreasing order. Note that we assume that there exists no distribution shift between the training and validation sets, and thus the term $d_{\mathcal{F}}(D_t^{(tr)}, D_t^{(val)})$ is zero and can be omitted.

5.3 Theory-inspired Regularization

It can be seen that the complexity of the meta-learner is normal without extra term needed to be restricted, while the complexity of task-specific meta-learner has additional term $\max_{x_{ti}^{(s)} \in D_t^{val}} \|h(x_{ti}^{(s)})\|$, which is dependent on the output range of the meta-learner. The transfer error bound in Eq.(15) can then naturally inspire the following three controlling strategies for improving the generalization capability of the extracted meta-learner h.

(i) Control the output range of the meta-learner h. Conventional models usually set all activation functions of meta-learner easily as ReLU. If we revise the last activation function as Tanh (i.e., $\phi_L = \frac{e^z - e^{-z}}{e^z + e^{-z}}$), then we have

$$\|h(\mathbf{x})\| = \|\mathbf{r}_L(\mathbf{x})\| = \|\phi_L(\mathbf{W}_L \mathbf{r}_{L-1}(\mathbf{x}))\| \leq \sqrt{d_L}.$$

Generally, $\sqrt{d_L}$ is smaller than $D\prod_{k=1}^{L} \|\mathbf{W}_k\|_F$, and the complexity can thus be expected to substantially decrease, and the generalization of the calculated meta-learner is hopeful to be improved.

(ii) Minimize $\|\mathbf{w}\|$ of the learners. The terms $\frac{3L\|\mathbf{w}\|}{\sqrt{mT}} \sum_{t=1}^{T} \max_{x_{ti}^{(s)} \in D_t^{val}} \|h(x_{ti}^{(s)})\|$ and $\frac{3L\|\mathbf{w}\|}{\sqrt{m_\mu}}$ imply that minimizing the $\|\mathbf{w}\|$ also tends to decrease the transfer error in Eq.(15).

(iii) Maximize the $\sigma_{d_L}(\mathbf{K})$. The term $\frac{M}{\sigma_{d_L}(\mathbf{K})}$ accounts for the gravity of the meta-training error influencing the final transfer error in Eq.(15), and thus maxmizing the $\sigma_{d_L}(\mathbf{K})$ also inclines to reduce the transfer error.

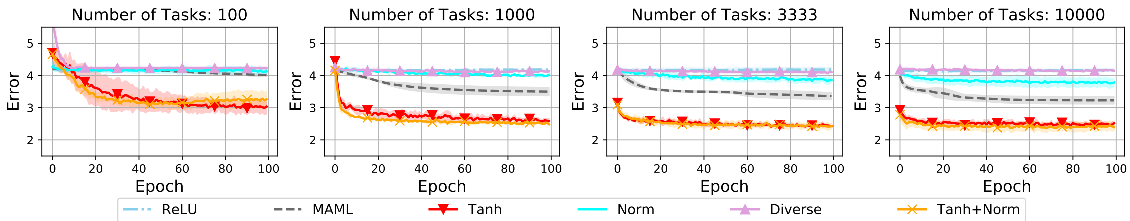
It should be emphasized that the above training strategy (i) corresponds to a regularized control manner imposed on the meta-learner, while the training strategies (ii) and (iii) put controls on the task-specific learners. And they could lead to different training behaviors as can be observed in the next subsection.

The training strategy (i) is easy to be implemented by directly replacing the last activation function of learners from conventional ReLU to Tanh function. The training strategy (ii) can be achieved by adding a $\|\mathbf{w}\|$ regularizer into the meta-training objective in Eq. (11) as:

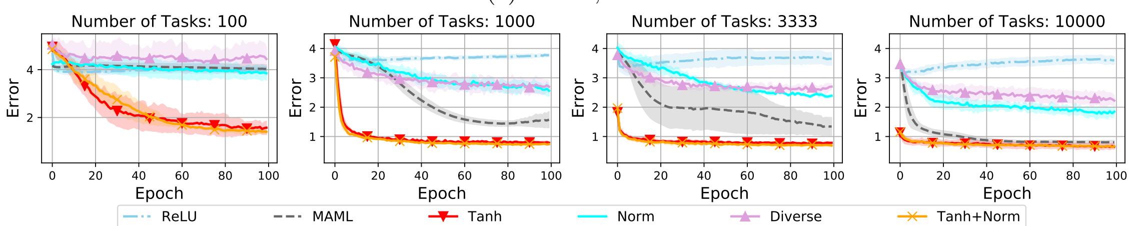
$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \frac{1}{nT} \sum_{t=1}^{T} \sum_{j=1}^{n} \ell(\mathbf{w}_t^{*\top} h(\mathbf{x}_{tj}^{(q)}), y_{tj}^{(q)})$$
$$s.t., \ \mathbf{w}_t^* = \arg\min_{\mathbf{w}_t} \frac{1}{m} \sum_{i=1}^{m} \ell(\mathbf{w}_t^\top h(\mathbf{x}_{ti}^{(s)}), y_{ti}^{(s)}) + \lambda_1 \|\mathbf{w}_t\|^2, t \in [T].$$

In the meta-test stage, it solves the following objective for a new query task given $\hat{h}$:

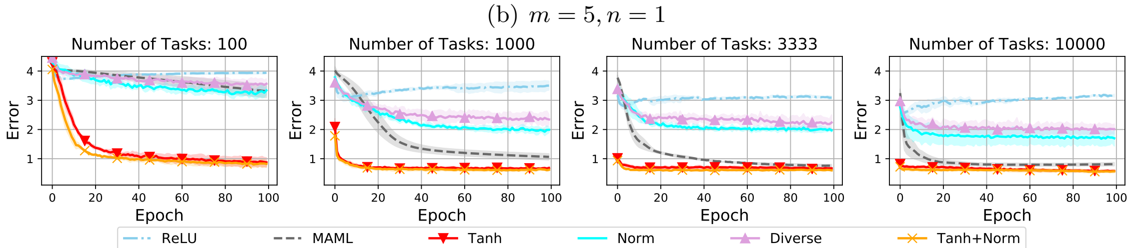
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{M_0} \sum_{i=1}^{M_0} \ell(\mathbf{w}^\top \hat{h}(\mathbf{x}_i^{(s)}), y_i^{(s)}) + \lambda_2 \|\mathbf{w}\|^2.$$



(a) $m = 1, n = 5$



(b) $m = 5, n = 1$



(c) $m = 5, n = 5$

Figure 3: Transfer error changing curves of different training strategies with $T = 100, 1000, 3333, 10000$ tasks and different m, n values. The variance of each curve over 3 independent runs has also depicted along the curve to show the stability of each method.

The training strategy (iii) can be realized by solving the following meta-training objective:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \frac{1}{nT} \sum_{t=1}^{T} \sum_{j=1}^{n} \ell(\mathbf{w}_t^{*\mathsf{T}} h(\mathbf{x}_{tj}^{(q)}), y_{tj}^{(q)})$$

$$s.t., \ \mathbf{P}^* = \arg\min_{\mathbf{P}} \frac{1}{mT} \sum_{t=1}^{T} \sum_{i=1}^{m} \ell(\mathbf{w}_t^{\mathsf{T}} h(\mathbf{x}_{ti}^{(s)}), y_{ti}^{(s)}) - \lambda_3 \sigma_{d_L}(\mathbf{P}^{\mathsf{T}}\mathbf{P}/T).$$

And the $\lambda_1, \lambda_2, \lambda_3$ are the hyperparamters of the above regularization problems.

5.4 Numerical Experiments

In this section, we test the effectiveness of the theory-induced training strategies on few-shot regression benchmark. We follow the experimental setting of MAML (Finn et al., 2017). Each task involves regression from the input to the output of a sine wave, where the amplitude and phase of the sinusoid are varied among tasks. Thus the problem aims to approximate a family of sine functions $f(x) = \alpha \sin(\beta x)$. The task distribution η 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 meta-training and meta-test tasks are randomly generated from the task distribution $p(\alpha, \beta)$. The function class $\mathcal{H}$ is set as an MLP with two hidden layers of size 40 with ReLU activation function, and $\mathcal{F}$ is a linear layer with bias False. Both the input and the output layers have dimensionality 1. The loss is the mean-squared error between the prediction $\mathbf{w}^\mathsf{T} h(x)$ and true value. The transfer error is averaged over 600 meta-test tasks with varying shots and queries. It uses episodic training strategy for meta-training, i.e., the meta-algorithm observes a set of training tasks sequentially and applies stochastic gradient descent with one task per batch.

We implement the MAML (Finn et al., 2017) as the baseline method in Eq.(11) solved by Bilevel Programming Franceschi et al. (2018). We denote the later by "ReLU", since the last activation function of the meta-learner is ReLU. The 'ReLU' and 'MAML' are implemented based on the code located at `https://github.com/jiaxinchen666/meta-theory` released by the paper (Chen et al., 2020). And 'Tanh', 'Norm', 'Diverse' and 'Tanh+Norm' denote the training strategies (i), (ii), (iii), and both (i), (ii) applied to the baseline method, respectively. We set $\lambda_1 = \lambda_2 = 1$ for training strategy (ii) and $\lambda_3 = 10$ for training strategy (iii). The task-specific optimizer is set as Adam optimizer with learning rate 0.01, and the meta-optimizer is set as Adam optimizer with learning rate 0.001. The other hyperparameters of task-specific optimizer and meta-optimizer are the default settings of Adam optimizer.

Fig.3 shows the transfer error of different training strategies with varied numbers of training tasks, support samples and query samples. It can be seen that the proposed training strategies can help consistently improve the performance of the baseline method on different meta-training tasks or support/query samples. As shown, training strategy (i) achieves an evidently larger improvement compared with (ii), (iii). Furthermore, we combine training strategies (i) and (ii), hoping to achieve further improvement compared with only one training strategy (i). However, there exists an unsubstantial improvement when using both training strategies (i) and (ii). This implies that a proper regularization technique for meta-learner can bring more essential improvements compared with the regularization techniques for task-specific learners. Specifically, when m is small ($m = 1$), the training strategies (ii), (iii) bring little improvement compared with 'ReLU', while training strategy (i) produces consistent improvement with different training tasks and m, n values.

Note that the training strategies (ii) and (iii) put controls on the task-specific learners, which are important to improve the performance for the given training tasks. While the training strategy (i) adds a regularized control imposed on the meta-learner, which is verified to be more important to improve the performance for transferring to the new query tasks. Such phenomenon is rational and has been observed comprehensively in our experiments. We thus will only study the control strategies for the meta-learner in the latter sections.

6. Application II: Few-shot Classification

In this section, we instantiate our meta learning framework for the few-shot classification problem.

6.1 Basic Setting

For the few-shot classification issue, usually one considers the K-way N-shot setting, in which each task contains NK examples from K classes with N examples for each class.

Thus the task-specific predictor machine $\mathcal{F}$ is often a K-class linear classifier. Let $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \{0, 1, \cdots, K-1\}$ and $\|x\| \leq R, \forall x \in \mathcal{X}$. The output of the meta-learner is the feature representation of each sample, and the meta learning machine $\mathcal{H}$ is parameterized as a depth-L vector-valued convolutional neural network (CNN) to extract the common feature representation for various classification tasks. Concretely, $h(x)$ can be written as:

$$h(x) = \phi_L(\mathbf{W}_L(\phi_{L-1}(\mathbf{W}_{L-1} \cdots \phi_1(\mathbf{W}_1 x)))), \tag{16}$$

where $\mathbf{W}_k, k \in [L]$ is the parameter matrix, and $\phi_k, k \in [L]$ is the 1-Lipschitz activation function. Formally, we additionally require both the norm of the weight matrix of each layer and the distance between the weights and the starting point weights are bounded, i.e.,

$$\|\mathbf{W}_j\|_F \leq B_j, \|\mathbf{W}_j^0\|_F \leq B_j, \|\mathbf{W}_j - \mathbf{W}_j^0\|_F \leq D_j, \quad j \in [L], \tag{17}$$

where $\mathbf{W}_j^0$ is the initialized parameter matrix of the model h. The distance to initialization has been observed to have substantial influence to generalization in deep learning (Bartlett et al., 2017; Neyshabur et al., 2019; Nagarajan and Kolter, 2019), and we introduce it here to develop the control strategy of the meta-learner. Specifically, $\mathcal{F}$ and $\mathcal{H}$ are chosen as:

$$\begin{aligned} &\mathcal{F} = \{f | f(z) = \boldsymbol{A}^\intercal z, \boldsymbol{A} \in \mathbb{R}^{d_L \times K}, \|\boldsymbol{A}\| \leq M\}, \\ &\mathcal{H} = \{h | h(x) \in \mathbb{R}^{d_L} \text{ as defined in (16)}, x \in \mathcal{X}\}. \end{aligned} \tag{18}$$

The conditional distribution of classification is

$$P(y = k | f(h(x))) = \frac{\exp(a_k)}{\sum_j \exp(a_j)} := \text{Softmax}(f(h(x))), a_k = (f(h(x)))_k, k \in [K]. \tag{19}$$

The loss functions ℓ is chosen as the cross-entropy loss $\ell(\boldsymbol{A}^\intercal h(x), y) = -\sum_{k=1}^K y_k \log(a_k)$, where y is the one-hot encoding of the ground-truth label, and a_k is defined as in Eq. (19).

Following the setting in (Finn et al., 2017; Lee et al., 2019), we assume that there are T tasks $\Gamma = \{D_t\}_{t=1}^T$ available for learning, and $D_t = (D_t^{tr}, D_t^{val})$, where $D_t^{tr} = \{z_{ti}^{(s)}\}_{i=1}^m, D_t^{val} = \{z_{tj}^{(q)}\}_{j=1}^n$. We set identical sample numbers for training/validation data sets for each task. The few-shot classification model is then written as

$$\begin{aligned} &\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \frac{1}{nT} \sum_{t=1}^T \sum_{j=1}^n \ell(\boldsymbol{A}_t^{*\intercal} h(x_{tj}^{(q)}), y_{tj}^{(q)}) \\ &s.t., \ \boldsymbol{A}_t^* = \arg\min_{\boldsymbol{A}_t} \frac{1}{m} \sum_{i=1}^m \ell(\boldsymbol{A}_t^T h(x_{ti}^{(s)}), y_{ti}^{(s)}), t \in [T], \end{aligned} \tag{20}$$

where $\boldsymbol{W} = \{\mathbf{W}_k, k \in [L]\}$ is the collection of the parameter matrices of h, and $\boldsymbol{A}, h$ is chosen from $\mathcal{F}, \mathcal{H}$ defined in Eq. (18). We denote $\mathbf{Q} = (\boldsymbol{A}_1, \cdots, \boldsymbol{A}_T)^\intercal \in \mathbb{R}^{T \times d_L \times K}$, and $\mathbf{Q}^* = (\boldsymbol{A}_1^*, \cdots, \boldsymbol{A}_T^*)^\intercal \in \mathbb{R}^{T \times d_L \times K}$ as its theoretical optimal solution.

6.2 Theoretical Analysis

Here, we will instantiate the general Theorem 3 for few-shot classification model defined in Eq. (20). The $Dis(D^{val})$ can be computed as

$$Dis(D^{val}) = \sup_{h,h'} \rho_{2,D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')}) \leq 4 \sup_{h, x \in \mathcal{X}} \|\boldsymbol{A}^\top h(x)\|_2$$
$$\leq \sup_{h,x} 4M\|h(x)\| \leq 4MD \cdot \prod_{k=1}^{L} \|\mathbf{W}_k\|_F.$$

Now we can verify that Assumptions 1 - 3 hold. The following proposition implies the Lipschitz continuity for the cross-entropy function.

Proposition 2 *The loss function $\ell(f(h(x)), y)$ is 1-Lipschitz with respect to $f(h(x))$, where ℓ is cross-entropy loss.*

According to proposition 2, we have

$$\ell(f(h(x)), y)| \leq \|f(h(x))\| = \|\boldsymbol{A}^\top h(x)\| \leq M\|h(x)\| \leq MD \prod_{k=1}^{L} \|\mathbf{W}_k\|_F,$$

and thus the loss function is bounded. The following result then verifies Assumption 3:

Proposition 3 *Consider the few-shot classification model defined in Eq.(20), and the loss function $\ell(\cdot, \cdot)$ chosen as the cross-entropy loss. The meta learning machine $\mathcal{H}$ and task-specific predictor machine are designed as in Eq.(18). Then the α, β in Assumption 3 are $\frac{M}{\sum_{k=1}^{K} \sigma_{d_L}((\mathbf{K})_k)}$ and 0, where $(\mathbf{K})_k = (\mathbf{Q}^*)_k^\top (\mathbf{Q}^*)_k / T$, and $(\cdot)_k$ denote the k-th element, $\sigma_{d_L}(\mathbf{K}_k)$ denote the d_L-th singular value of matrix $\mathbf{K}_k$ at a decreasing order*

It should be indicated that the value of α reflects the similarity of task-specific learners for given training task set. And minimizing α attempts to increase the diversity of learned task-specific learners, which can cover the space as much as possible captured by the h needed to be predicted on new tasks, and thus help improve the generalization ability of the extracted meta-learner for new query tasks.

Now we can compute the leading-order terms in Eq.(8) for the parameterized meta-learner and learners defined in Eq.(18). We firstly show the Rademacher complexity of $\mathcal{H}$ in the following theorem.

Theorem 5 (Theorem 2 in (Gouk et al., 2021)) *Let $\mathcal{H}$ be the class of real-valued DNN as defined in Eq.(16) and (17) over $\mathcal{X} = \{\mathbf{x} : \|\mathbf{x}\| \leq R\}$, where each parameter matrix $\mathbf{W}_i$, $i \in [L]$ has Frobenius norm at most B_i, and the activation function $\phi_i, i \in [L]$ is 1-Lipschitz, with $\phi_i(0) = 0$, and applied element-wise. Then we have:*

$$\hat{\mathfrak{R}}_N(\mathcal{H}) \leq \frac{2\sqrt{2} d_L R \sum_{j=1}^{L} \frac{D_j}{2B_j \prod_{i=1}^{j} \sqrt{c_i}} \prod_{j=1}^{L} 2B_j \sqrt{c_j}}{\sqrt{N}},$$

where c_i is the number of columns in $\mathbf{W}_i$.

(1) The Gaussian complexity of meta-learner h can be computed as

$$
\begin{aligned}
\hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) &= \mathbb{E} \sup_{h \in \mathcal{H}} \frac{1}{nT} \sum_{t=1}^{T} \sum_{j=1}^{n} \sum_{k=1}^{d_L} g_{tjk} h_k(x_{tj}^{(q)}) \\
&\leq \sum_{k=1}^{d_L} \hat{\mathcal{G}}_{D^{val}}(h_k) \leq 2\sqrt{\log(nT)} \sum_{k=1}^{d_L} \hat{\mathfrak{R}}_{D^{val}}(h_k) \\
&\leq 2\sqrt{\log(nT)} \cdot \frac{2\sqrt{2} d_L R \sum_{j=1}^{L} \frac{D_j}{2B_j \prod_{i=1}^{j} \sqrt{c_i}} \prod_{j=1}^{L} 2B_j \sqrt{c_j}}{\sqrt{nT}}.
\end{aligned}
$$

(2) The Gaussian complexity of the task-specific learner can be computed as:

$$
\begin{aligned}
\hat{\mathcal{G}}_{D_t^{val}}(\mathcal{F}) &= \mathbb{E} \sup_{\boldsymbol{A}_t \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} g_{tik} ((\boldsymbol{A}_t)_k^{\intercal} h(x_{ti}^{(s)})) \\
&= \frac{1}{m} \mathbb{E} \sup_{\boldsymbol{A}_t \in \mathcal{F}} \sum_{k=1}^{K} ((\boldsymbol{A}_t)_k^{\intercal} \sum_{i=1}^{m} g_{tik} h(x_{ti}^{(s)})) \\
&\leq \frac{M}{m} \mathbb{E} \sum_{k=1}^{K} \left\| \sum_{i=1}^{m} g_{tik} h(x_{ti}^{(s)})) \right\| \\
&= \frac{MK}{\sqrt{m}} \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left\| h(x_{ti}^{(s)}) \right\|_2^2} \\
&\leq \frac{MK}{\sqrt{m}} \cdot \max_{x_{ti}^{(s)} \in D_t^{tr}} \| h(x_{ti}^{(s)}) \|.
\end{aligned}
$$

Different from few-shot regression, there exists an additional multiplicative factor with term K, which is the number of classes. This implies the complexity of the task-specific learner increases as the number of classes increases, complying with general experience in common practice.

Thus, the transfer error defined in Eq.(8) now can be written as

$$
\begin{aligned}
& R_{test}(\hat{f}_\mu, \hat{h}) - R_{test}(f_\mu^*, h^*) \\
\leq & \frac{M}{\sum_{k=1}^{K} \sigma_{d_L}((\mathbf{K})_k)} \left(768 L \log(4nT) L(\mathcal{F}) \cdot \frac{2\sqrt{2} d_L R \sum_{j=1}^{L} \frac{D_j}{2B_j \prod_{i=1}^{j} \sqrt{c_i}} \prod_{j=1}^{L} 2B_j \sqrt{c_j}}{\sqrt{nT}} \right. \\
& \left. + \frac{6LMK}{\sqrt{m}T} \sum_{t=1}^{T} \max_{x_{ti}^{(s)} \in D_t^{val}} \| h(x_{ti}^{(s)}) \| + 6B \sqrt{\frac{\log \frac{2}{\delta}}{2nT}} + 6B \sqrt{\frac{\log \frac{2}{\delta}}{m}} + \frac{48 L \sup_{h,x} M \| h(x) \|}{n^2 T^2} \right) \\
& + \frac{6LMK}{\sqrt{m_\mu}} \cdot \max_{x_i^{(s)} \in D_\mu^{val}} \| h(x_i^{(s)}) \| + 6B \sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}}.
\end{aligned}
$$

Note that we assume there exists no distribution shift between the training and validation sets, and thus the term $d_{\mathcal{F}}(D_t^{(tr)}, D_t^{(val)})$ is zero and omitted.

6.3 Theory-Inspired Regularization

As aforementioned, we develop the following two controlling strategies for meta learner h to improve its methodology transferable generalization capability among tasks.

(i) Control the output range of the meta-learner h. All activation functions of CNN defined in Eq.(16) are usually assumed to be ReLU. We take the same control strategy as in Section 5.3, and revise the last activation function as Tanh (i.e., $\phi_L = \frac{e^z - e^{-z}}{e^z + e^{-z}}$).

(ii) Minimize the distance between the weights from the starting point weights in Eq.(17). This control strategy can decrease the complexity of meta-learner h.

The training strategy (ii) can be achieved by adding the penalty terms corresponding to each layer into the meta-training objective in Eq. (11),

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}} \frac{1}{nT} \sum_{t=1}^{T} \sum_{j=1}^{n} \ell(\boldsymbol{A}_t^{*\mathsf{T}} h(\mathbf{x}_{tj}^{(q)}), y_{tj}^{(q)}) + \lambda \sum_{j=1}^{L} \|\mathbf{W}_j - \mathbf{W}_j^0\|_F^2$$

$$s.t., \ \boldsymbol{A}_t^* = \arg\min_{\boldsymbol{A}_t} \frac{1}{m} \sum_{i=1}^{m} \ell(\boldsymbol{A}_t^{\mathsf{T}} h(\mathbf{x}_{ti}^{(s)}), y_{ti}^{(s)}), t \in [T],$$

where λ is the hyperparamters of the regularization. Note that this regularization scheme has been used to help improve the performance of transfer learning proposed by (Li et al., 2018b, 2019a). We firstly employ this regularization scheme for few-shot classification problems.

6.4 Numerical Experiments

In this section, we test the effectiveness of the theory-induced training strategies on few-shot classification benchmark.

6.4.1 Implementation details

We parameterize the meta-learner h as a ResNet-12 network and a standard 4-layer convolutional network followed by (Snell et al., 2017; Lee et al., 2019). Also, we use DropBlock regularization (Ghiasi et al., 2018) for the ResNet-12 network. The compared methods include ProtoNet, MetaOptNet-RR and MetaOptNet-SVM, whose task-specific learners are nearest-neighbor classifier (Snell et al., 2017), ridge regression classifier (Bertinetto et al., 2019) and SVM classifier (Lee et al., 2019), respectively. We denote "Tanh", "L^2-SP" and "Tanh + L^2-SP" as training strategies (i), (ii) and combining training strategy (i) and (ii).

We follow the setting in (Lee et al., 2019) to conduct the few-shot classification experiments. To optimize the meta-learners, we use SGD with Nesterov momentum of 0.9 and weight decay of 0.0005. Each mini-batch consists of 8 episodes. The model was meta-trained for 60 epochs, with each epoch consisting of 1000 episodes. The learning rate was initially set to 0.1, and then changed to 0.006, 0.0012, and 0.00024 at epochs 20, 40 and 50, respectively. During the meta-training stage, we adopt horizontal flip, random crop, and color (brightness, contrast, and saturation) jitter data augmentation techniques. We use 5-way classification in both meta-training and meta-test stages. Each class contains 6 test (query) samples during meta-training and 15 test samples during meta-testing. Our meta-trained model was chosen based on 5-way 5-shot test accuracy on the meta-validation set. Meanwhile, during meta-training, we set training shot to 15 for miniImageNet with ResNet-12; 5 for miniImageNet with 4-layer CNN; 10 for tieredImageNet; 5 for CIFAR-FS; and 15 for FC100. We set the hyperparameter λ as 0.1, and keep the default hyperparameter setting for the compared baselines in the original papers. Our implemention is based on the code provided on `https://github.com/kjunelee/MetaOptNet`.

6.4.2 Experiments on CIFAR derivatives

The **CIFAR-FS** dataset (Bertinetto et al., 2019) is a recently proposed few-shot image classification benchmark, consisting of all 100 classes from CIFAR-100. The classes are randomly split into 64, 16 and 20 for meta-training, meta-validation, and meta-testing, respectively. Each class contains 600 images of size 32×32.

The **FC100** dataset is another dataset derived from CIFAR-100 (Oreshkin et al., 2018), containing 100 classes which are grouped into 20 superclasses. These classes are partitioned into 60 classes from 12 superclasses for meta-training, 20 classes from 4 superclasses for meta-validation, and 20 classes from 4 superclasses for meta-testing. The goal is to minimize semantic overlap between classes. All images are also of size 32×32.

Results. Table 3 summarizes the results on the 5-way CIFAR-FS and FC100 classification tasks with different shots. It can be seen that our theory-induced training strategy does help improve generalization error from SOTA results in most cases. Specifically, we shows the results where we vary the meta-learner for two different embedding architectures. In both cases that the dimension of the feature representation is low (1600, a standard 4-layer convolutional network), and the dimension is much higher (16000, ResNet12), our proposed regularization schemes yield better few-shot accuracy than baselines. This implies that our proposed regularization scheme is model-agnostic, in the sense that it can be directly applied to regularize the learning for different meta-learners. Meanwhile, even on the harder FC100 dataset, our strategies can still help increase the accuracy for new query few-shot tasks, which highlights the advantage of our problem-agnostic regularization schemes.

6.4.3 Experiments on ImageNet derivatives

The **miniImageNet** dataset (Vinyals et al., 2016) is a standard benchmark for few-shot image classification benchmark, consisting of 100 randomly chosen classes from ILSVRC-2012 (Russakovsky et al., 2015). The meta-training, meta-validation, and meta-testing sets contain 64, 16 and 20 classes randomly split from 100 classes, respectively. Each class contains 600 images of size 84×84. We use the commonly-used split proposed by (Ravi and Larochelle, 2017).

The **tieredImageNet** benchmark (Ren et al., 2019) is a larger subset of ILSVRC-2012 (Russakovsky et al., 2015), composed of 608 classes grouped into 34 high-level categories. These categories are then split into 3 disjoint sets: 20 categories for meta-training, 6 for meta-validation, and 8 for meta-test. This corresponds to 351, 97 and 160 classes for meta-training, meta-validation, and meta-testing, respectively. This dataset aims to minimize the semantic similarity between the splits similar to FC100. All images are of size 84×84.

Results. Table 4 summarizes the results on the 5-way classification tasks with different shots on miniImageNet and tieredImageNet benchmarks. Compared with CIFAR derivatives dataset, this benchmark contains more classes and more natural images. It can be observed that our proposed regularization schemes can also help improve generalization error in most

Table 3: Average few-shot classification accuracies (%) with 95% confidence intervals on CIFAR-FS and FC-100 meta-test splits.

model	CIFAR-FS 5-way 1-shot	CIFAR-FS 5-way 5-shot	FC100 5-way 1-shot	FC100 5-way 5-shot
4-layer conv (feature dimension=1600)				
ProtoNet	59.89 ±0.70	80.14 ± 0.53	36.13 ± 0.53	50.69 ±0.55
ProtoNet(Tanh)	63.55 ± 0.75 4.66↑	80.30 ± 0.52 0.16↑	36.64 ± 0.55 0.51↑	50.53 ± 0.56 0.16↓
ProtoNet(L^2-SP)	60.71 ± 0.71 0.82↑	80.01 ± 0.53 0.13↓	36.52 ± 0.54 0.39↑	51.10 ± 0.56 0.01↑
ProtoNet(Tanh + L^2-SP)	63.58 ± 0.75 4.66↑	80.20 ± 0.52 0.06↑	36.74 ± 0.55 0.61↑	50.77 ± 0.54 0.08↑
MetaOptNet-RR	63.02 ± 0.71	79.60 ± 0.52	35.75 ± 0.52	51.20 ± 0.55
MetaOptNet-RR(Tanh)	63.45 ± 0.70 0.43↑	80.11 ± 0.53 0.51↑	38.61 ± 0.56 2.86↑	52.29 ± 0.56 1.09↑
MetaOptNet-RR(L^2-SP)	63.66 ± 0.72 0.64↑	79.49 ± 0.52 0.11↓	36.55 ± 0.52 0.80↑	51.27 ± 0.53 0.07↑
MetaOptNet-RR(Tanh + L^2-SP)	63.81 ± 0.71 0.69↑	80.44 ± 0.50 0.84↑	37.28 ± 0.53 1.53↑	52.31 ± 0.52 1.11↑
MetaOptNet-SVM	61.77 ± 0.73	79.08 ± 0.52	35.82 ± 0.54	50.60 ± 0.54
MetaOptNet-SVM(Tanh)	63.18 ± 0.71 1.41↑	79.87 ± 0.53 0.49↑	37.10 ± 0.58 1.28↑	51.62 ± 0.57 1.02↑
MetaOptNet-SVM(L^2-SP)	62.70 ± 0.71 0.93↑	79.95 ± 0.53 0.87↑	37.13 ± 0.68 1.31↑	51.42 ± 0.53 0.82↑
MetaOptNet-SVM(Tanh + L^2-SP)	63.24 ± 0.72 1.47↑	80.20 ± 0.51 1.12↑	37.29 ± 0.58 1.47↑	51.71 ± 0.55 1.11↑
ResNet-12 (feature dimension=16000)				
ProtoNet	68.00 ± 0.74	83.50 ± 0.51	38.43 ± 0.58	52.56 ± 0.56
ProtoNet(Tanh)	69.71 ± 0.76 1.71↑	83.62 ± 0.51 0.12↑	40.12 ± 0.58 1.69↑	55.02 ± 0.54 2.06↑
ProtoNet(L^2-SP)	69.01 ± 0.73 1.01↑	83.57 ± 0.51 0.07↑	39.16 ± 0.57 0.73↑	53.60 ± 0.56 1.04↑
ProtoNet(Tanh + L^2-SP)	71.22 ± 0.74 3.22↑	83.89 ± 0.51 0.39↑	40.70 ± 0.58 2.43↑	55.33 ± 0.54 2.77↑
MetaOptNet-RR	68.58 ± 0.73	84.75 ± 0.50	38.98 ± 0.56	54.46 ± 0.55
MetaOptNet-RR(Tanh)	71.64 ± 0.72 3.06↑	84.81 ± 0.49 0.06↑	40.13 ± 0.58 1.15↑	54.48 ± 0.54 0.02↑
MetaOptNet-RR(L^2-SP)	71.40 ± 0.70 2.82↑	84.91 ± 0.50 0.16↑	39.79 ± 0.57 0.81↑	54.80 ± 0.55 0.34↑
MetaOptNet-RR(Tanh + L^2-SP)	72.15 ± 0.72 3.57↑	85.03 ± 0.49 0.28↑	40.86 ± 0.57 1.88↑	55.58 ± 0.54 1.12↑
MetaOptNet-SVM	68.81 ± 0.74	83.80 ± 0.51	40.24 ± 0.58	54.71 ± 0.56
MetaOptNet-SVM(Tanh)	71.12 ± 0.71 2.31↑	85.19 ± 0.49 1.39↑	39.99 ± 0.57 0.25↓	53.92 ± 0.56 0.79↓
MetaOptNet-SVM(L^2-SP)	70.84 ± 0.72 2.03↑	84.22 ± 0.48 0.42↑	40.41 ± 0.57 0.17↑	55.53 ± 0.55 0.82↑
MetaOptNet-SVM(Tanh + L^2-SP)	71.52 ± 0.73 2.71↑	84.77 ± 0.48 0.97↑	40.83 ± 0.58 0.59↑	55.75 ± 0.56 1.04↑

cases from SOTA baselines method without adding these schemes with different embedding architectures of meta-learners or different semantic similarity between the splits.

It should be emphasized that we have just directly replaced the last activation function (ReLU) of feature extractor (meta-learner) as Tanh, or add the regularization scheme to the meta loss, and keep the original configurations without implementing any further fine-tuning executions and introuducing extra computation cost. This way allows us to clearly see the effect of proposed regularization schemes on the compared baselines. The empirical results verify that our theory-induced regularization schemes are rational and arguably simple to improve the baseline performance.

6.4.4 Cross-domain few-shot classification

To further justify the effectiveness of our proposed regularization schemes in producing robust features for unseen tasks, we further perform experiments under cross-domain few-shot classification settings, where the meta-test tasks are substantially different from meta-train tasks. We report the results in Table 5, using the same experiment settings, firstly introduced in (Chen et al., 2020), where miniImageNet is used as meta-train set and CUB dataset (Wah et al., 2011) as meta-test set.

Table 5 shows the cross-domain performance of the baselines and our proposed regularization schemes, which exhibits the similar tendency to few-shot classification results from Table 4. As is known, the CUB is a fine-grained classification dataset with more intra-class variations, and has a large domain shift to the miniImageNet dataset. Therefore, the extracted learning methodology (feature representation) from the meta-training dataset is mostly irrelevant to the meta-test datasets, which makes the learning tasks from new different domains relatively difficult, as suggested in (Baik et al., 2020). Even so, our proposed

Table 4: Average few-shot classification accuracies (%) with 95% confidence intervals on miniImageNet and tieredImageNet meta-test splits.

model	miniImageNet 5-way 1-shot	miniImageNet 5-way 5-shot	tieredImageNet 5-way 1-shot	tieredImageNet 5-way 5-shot
4-layer conv (feature dimension=1600)				
ProtoNet	47.69 ± 0.64	70.51 ± 0.52	50.10 ± 0.70	71.85 ± 0.57
ProtoNet(Tanh)	53.13 ± 0.64 5.44↑	71.22 ± 0.52 0.71↑	54.07 ± 0.69 3.93↑	71.05 ± 0.57 0.60↓
ProtoNet(L^2-SP)	48.15 ± 0.63 0.46↑	70.67 ± 0.52 0.16↑	50.84 ± 0.71 0.74↑	72.50 ± 0.50 0.64↑
ProtoNet(Tanh + L^2-SP)	54.43 ± 0.71 6.74↑	70.88 ± 0.52 0.37↑	54.06 ± 0.69 3.93↑	71.25 ± 0.57 0.40↓
MetaOptNet-RR	52.66 ± 0.63	69.80 ± 0.53	54.04 ± 0.68	72.09 ± 0.56
MetaOptNet-RR(Tanh)	52.70 ± 0.62 0.04↑	70.72 ± 0.52 0.98↑	54.23 ± 0.67 0.19↑	72.35 ± 0.55 0.26↑
MetaOptNet-RR(L^2-SP)	52.38 ± 0.62 0.28↓	68.95 ± 0.52 0.85↓	55.66 ± 0.68 1.62↑	72.15 ± 0.57 0.06↑
MetaOptNet-RR(Tanh + L^2-SP)	52.47 ± 0.62 0.19↓	70.17 ± 0.51 0.37↑	54.45 ± 0.69 0.41↑	72.40 ± 0.55 0.31↑
MetaOptNet-SVM	52.50 ± 0.62	69.66 ± 0.52	54.38 ± 0.70	71.48 ± 0.56
MetaOptNet-SVM(Tanh)	52.75 ± 0.62 0.25↑	70.79 ± 0.50 1.13↑	54.66 ± 0.69 0.28↑	71.57 ± 0.57 0.09↑
MetaOptNet-SVM(L^2-SP)	52.20 ± 0.60 0.30↓	69.04 ± 0.52 0.62↓	55.52 ± 0.71 1.14↑	71.49 ± 0.57 0.01↑
MetaOptNet-SVM(Tanh + L^2-SP)	52.54 ± 0.61 0.04↑	70.27 ± 0.52 0.61↑	54.66 ± 0.69 0.28↑	71.62 ± 0.50 0.14↑
ResNet-12 (feature dimension=16000)				
ProtoNet	56.13 ± 0.67	74.57 ± 0.53	63.02 ± 0.72	80.11 ± 0.56
ProtoNet(Tanh)	57.55 ± 0.67 1.42↑	74.62 ± 0.67 0.05↑	65.11 ± 0.73 2.09↑	79.69 ± 1.39 0.42↓
ProtoNet(L^2-SP)	56.51 ± 0.65 0.38↑	75.15 ± 0.53 0.58↑	63.48 ± 0.72 0.46↑	80.65 ± 0.54 0.54↑
ProtoNet(Tanh + L^2-SP)	58.10 ± 0.66 1.97↑	74.87 ± 0.51 0.30↑	65.68 ± 0.73 2.66↑	80.59 ± 0.51 0.48↑
MetaOptNet-RR	60.27 ± 0.67	76.19 ± 0.50	66.35 ± 0.71	81.25 ± 0.53
MetaOptNet-RR(Tanh)	60.44 ± 0.64 0.17↑	76.66 ± 0.47 0.47↑	66.97 ± 0.71 0.62↑	81.42 ± 0.52 0.17↑
MetaOptNet-RR(L^2-SP)	60.05 ± 0.65 0.22↓	77.02 ± 0.48 0.83↑	66.77 ± 0.71 0.42↑	81.48 ± 0.53 0.23↑
MetaOptNet-RR(Tanh + L^2-SP)	60.41 ± 0.65 0.14↑	76.98 ± 0.48 0.79↑	66.85 ± 0.71 0.50↑	81.49 ± 0.52 0.24↑
MetaOptNet-SVM	60.58 ± 0.66	75.99 ± 0.51	66.40 ± 0.72	81.18 ± 0.54
MetaOptNet-SVM(Tanh)	60.59 ± 0.65 0.01↑	77.05 ± 0.47 1.06↑	66.43 ± 0.70 0.03↑	81.30 ± 0.52 0.12↑
MetaOptNet-SVM(L^2-SP)	61.17 ± 0.64 0.59↑	77.40 ± 0.49 1.41↑	66.45 ± 0.71 0.05↑	81.52 ± 0.53 0.34↑
MetaOptNet-SVM(Tanh + L^2-SP)	60.99 ± 0.65 0.41↑	77.43 ± 0.48 1.44↑	66.49 ± 0.09 0.09↑	81.43 ± 0.52 0.25↑

regularization schemes can also evidently improves the performance of baselines. This further validates the effectiveness and validness of the developed theory and the theory-induced regularization schemes to learn a more robust feature extractor with better adaptability to new tasks.

7. Application III: Domain Generalization

In this section, we instantiate the proposed meta learning framework for the domain generalization problem.

7.1 Basic Setting and Theoretical Analysis

Here we consider the domain generalization (DG) setting. We present the generally used meta learning setting (Li et al., 2018a, 2019b) for heterogeneous DG. Usually, assuming that we have T domains (datasets) $\Gamma = \{D_1, D_2, \cdots, D_T\}$ for meta-training, each domain (task) has both meta-training and meta-validation data with $D_t = (D_t^{tr}, D_t^{val})$, where $D_t^{tr} = \{z_{ti}^{(s)}\}_{i=1}^{m_t}, D_t^{val} = \{z_{tj}^{(q)}\}_{j=1}^{n_t}$. Note that the label space is not shared between training/support and validation/query domains, but it is straightforwardly applicable to conventional (homogeneous) DG when assuming that the label space is shared between different domains.

There exist several differences between DG and few-shot classification problem. (1) The former often assumes that the meta-training and meta-validation sets of each domain (task) share the same label space, but have distribution shift. However, the latter often assumes that they are with the same distribution. (2) The latter pays more attention to

Table 5: Average few-shot classification accuracies (%) with 95% confidence intervals on 5-way cross-domain classification.

model	miniImageNet → CUB 1-shot	miniImageNet → CUB 5-shot
4-layer conv (feature dimension=1600)		
ProtoNet	36.20 ± 0.54	55.44 ± 0.52
ProtoNet(Tanh)	39.33 ± 0.55 3.13↑	56.30 ± 0.52 0.86↑
ProtoNet(L^2-SP)	36.40 ± 1.80 0.20↑	58.40 ± 0.52 2.96↑
ProtoNet(Tanh + L^2-SP)	39.23 ± 0.54 3.03↑	57.46 ± 0.53 2.02↑
MetaOptNet-RR	41.27 ± 0.55	58.15±0.52
MetaOptNet-RR(Tanh)	41.34 ± 0.56 0.07↑	59.82 ± 0.51 1.67↑
MetaOptNet-RR(L^2-SP)	41.23 ± 0.56 0.04↓	59.11 ± 0.51 0.96↑
MetaOptNet-RR(Tanh + L^2-SP)	42.13 ± 0.55 0.86↑	60.27 ± 0.52 2.12↑
MetaOptNet-SVM	41.47 ± 0.56	58.21 ± 0.53
MetaOptNet-SVM(Tanh)	41.89 ± 0.56 0.42↑	61.26 ± 0.52 3.05↑
MetaOptNet-SVM(L^2-SP)	41.00 ± 0.58 0.47↓	58.81 ± 0.51 0.60↑
MetaOptNet-SVM(Tanh + L^2-SP)	41.49 ± 0.54 0.02↑	59.44 ± 0.50 1.23↑
ResNet-12 (feature dimension=16000)		
ProtoNet	41.86 ± 0.59	58.41 ± 0.56
ProtoNet(Tanh)	42.23 ± 0.63 0.37↑	61.61 ± 0.56 3.20↑
ProtoNet(L^2-SP)	43.60 ± 0.60 1.74↑	62.59 ± 0.55 4.18↑
ProtoNet(Tanh + L^2-SP)	43.53 ± 0.63 1.67↑	60.03 ± 2.56 1.62↑
MetaOptNet-RR	44.43 ± 0.59	64.12 ± 0.51
MetaOptNet-RR(Tanh)	45.14 ± 0.58 0.71↑	64.23 ± 0.52 0.11↑
MetaOptNet-RR(L^2-SP)	45.00 ± 0.60 0.57↑	64.44 ± 0.53 0.32↑
MetaOptNet-RR(Tanh + L^2-SP)	45.24 ± 0.60 0.81↑	64.95 ± 0.50 0.83↑
MetaOptNet-SVM	44.60 ± 0.59	64.53 ± 0.53
MetaOptNet-SVM(Tanh)	45.09 ± 0.57 0.49↑	65.31 ± 0.52 0.78↑
MetaOptNet-SVM(L^2-SP)	44.68 ± 0.59 0.08↑	64.85 ± 0.53 0.32↑
MetaOptNet-SVM(Tanh + L^2-SP)	44.69 ± 0.46 0.09↑	64.90 ± 0.52 0.37↑

solving new query tasks with limited data, while the former not only deals with limited data in the meta-test domain, but also attempts to eliminate the distribution shift between meta-training and meta-test domain.

We take the same setting for $\mathcal{F}$ and $\mathcal{H}$ in few-shot classification, and borrow the theoretical analysis therein. Thus the transfer error defined in Eq.(8) can be written as

$$
\begin{aligned}
& R_{test}(\hat{f}_\mu, \hat{h}) - R_{test}(f_\mu^*, h^*) \\
\leq & \frac{M}{\sum_{k=1}^{K} \sigma_{d_L}((\mathbf{K})_k)} \Bigg(768L\log(4nT)L(\mathcal{F}) \cdot \frac{2\sqrt{2}d_L R \sum_{j=1}^{L} \frac{D_j}{2B_j \prod_{i=1}^{j}\sqrt{c_i}} \prod_{j=1}^{L} 2B_j\sqrt{c_j}}{\sqrt{nT}} \\
& + \frac{6LMK}{\sqrt{mT}} \sum_{t=1}^{T} \max_{x_{ti}^{(s)} \in D_t^{val}} \|h(x_{ti}^{(s)})\| + \frac{4}{T}\sum_{t=1}^{T} d_{\mathcal{F}}(D_t^{(tr)}, D_t^{(val)}) + 6B\sqrt{\frac{\log\frac{2}{\delta}}{2nT}} + 6B\sqrt{\frac{\log\frac{2}{\delta}}{m}} \\
& + \frac{48L\sup_{h,x} M\|h(x)\|}{n^2T^2} \Bigg) + \frac{6LMK}{\sqrt{m_\mu}} \cdot \max_{x_i^{(s)} \in D_\mu^{val}} \|h(x_i^{(s)})\| + \mathbb{E}_{\mu\sim\eta} d_{\mathcal{F}}(D_\mu^{(tr)}, D_\mu^{(val)}) + 6B\sqrt{\frac{\log\frac{2}{\delta}}{m_\mu}}.
\end{aligned}
$$

Note that there exists an additional distribution shift term $d_{\mathcal{F}}(D_t^{(tr)}, D_t^{(val)})$ in the error bound, characterizing the distribution shift between the meta-training and meta-validation domains, as well as the term $d_{\mathcal{F}}(D_\mu^{(tr)}, D_\mu^{(val)})$ characterizing the distribution shift between the training and test sets of meta-test set. It is easy to see that proposed regularization schemes in the few-shot classification can also be applied to DG, i.e., (i) Control the output range of the meta-learner h; (ii) Minimize the distance of the weights from the starting point weights as Eq.(17).

Considering the difference from few-shot classification, DG should require proper training strategy to eliminate the distribution shift. Recently, Li et al. (2019b) proposed a Feature-Critic Networks technique to produce an auxiliary loss function to guide learning on the meta-training set to produce more robust and effective feature extractor on the meta-validation set. From this view, Feature-Critic Networks attempts to simulate the distribution shift between the meta-training and meta-validation sets, i.e., $d_{\mathcal{F}}(D_t^{(tr)}, D_t^{(val)})$. Due to its rationality and efficacy, in the following part, we employ Feature-Critic Networks as the baseline method, and further verify the effectiveness of proposed regularization schemes for DG problem.

7.2 Numerical Experiments

In this section, we verify the effectiveness of the theory-induced training strategies on homogeneous and heterogeneous DG benchmark, respectively.

7.2.1 Homogeneous DG experiments

Dataset. The **PACS** dataset is a recent object recognition benchmark for domain generalisation (Li et al., 2017a). PACS contains 9991 images of size 224×224 from four different domains - Photo (P), Art painting (A), Cartoon (C) and Sketch (S). It has 7 categories across these domains: dog, elephant, giraffe, guitar, house, horse and person. It can be downloaded at `http://sketchx.eecs.qmul.ac.uk/`. We follow the standard protocol and perform leave-one-domain-out evaluation.

Implementation details. We parameterize the shared feature extrator (meta-learner) h as a pre-trained AlexNet (Krizhevsky et al., 2012) network followed by (Li et al., 2019b), and the task-specific learners as linear classifier. The competed methods include Reptile (Nichol et al., 2018), CrossGrad (Shankar et al., 2018), MetaReg (Balaji et al., 2018) and FC (Li et al., 2019b). We follow the experimental setting in (Li et al., 2019b). Specifically, the feature extrator and task-specific learners are trained with M-SGD optimizer (batch size/per meta-trian domain=32, batch size/per meta-test domain=16, lr=0.0005, weight decay=0.00005, momentum=0.9) for 45K iterations. The Feature-Critic (set embedding variant) is adopted as MLP type, and is trained with the M-SGD optimizer (lr=0.001, weight decay=0.00005, momentum=0.9). We denote "Tanh", "L^2-SP" and "Tanh + L^2-SP" as the training strategies (i), (ii) and combining training strategy (i) and (ii). Our implementation is based on the code provided on `https://github.com/liyiying/Feature_Critic/`.

Results. The comparison with state-of-the-art methods on PACS dataset is shown in Table 6. The results of Reptile, CrossGrad and MetaReg are directly taken from (Li et al., 2019b). We re-implement the result of AGG (Li et al., 2019b) and FC. Note that compared with those reported in (Li et al., 2019b), the depicted results of these two methods have

Table 6: Cross-domain recognition accuracy (%) on PACS using train split in (Li et al., 2017a) for training.

Target	Reptile	CrossGrad	MetaReg	AGG	FC	FC+Tanh	FC+L^2-SP	FC+Tanh+L^2-SP
A	63.4	61.0	63.5	62.2	63.0	**64.2**	63.4	63.5
C	67.5	67.2	69.5	66.2	67.2	**70.6**	68.2	68.3
P	**88.7**	87.6	87.4	87.0	87.9	87.5	**88.0**	87.6
S	55.9	55.9	59.1	54.9	59.5	60.9	**62.1**	61.4
Ave.	68.9	67.9	69.9	67.6	69.4	**70.8**	70.4	70.2

a slight drop on A, C, P domain, but a slight rise on S domain. These minor differences are due to our used higher Pytorch version. From the table, it can be easily observed that our proposed regularization schemes consistently outperform other competing methods. Especially, as compared with the SOTA method FC, aiming to learn representations that generalise to new domains through training a supervised loss and explicitly simulates domain shift, our proposed regularization scheme puts emphasis on decreasing the complexity of the feature extractor (meta-learner). Therefore, it can be combinationally used with FC to compensate its performance.

7.2.2 Heterogeneous DG experiments

Datasets. The **Visual Decathlon** (VD) dataset consists of ten well-known datasets from multiple visual domains (Rebuffi et al., 2017). **FGVC-Aircraft Benchmark** (Maji et al., 2013) contains 10,000 images of aircraft, with 100 images for each of 100 different aircraft model variants. **CIFAR100** (Krizhevsky et al., 2009) contains colour images for 100 object categories. **Daimler Mono Pedestrian Classification Benchmark (DPed)** (Munder and Gavrila, 2006) consists of 50,000 grayscale pedestrian and non-pedestrian images. **Describable Texture Dataset (DTD)** (Cimpoi et al., 2014) is a texture database, consisting of 5640 images, organized according to a list of 47 categories such as bubbly, cracked, marbled. **The German Traffic Sign Recognition (GTSR) Benchmark** (Stallkamp et al., 2012) contains cropped images for 43 common traffic sign categories in different image resolutions. **Flowers102** (Nilsback and Zisserman, 2008) is a fine-grained classification task which contains 102 flower categories from the UK. **ILSVRC12 (ImageNet)** (Russakovsky et al., 2015) is the largest dataset in our benchmark, containing 1000 categories and 1.2 million images. **Omniglot** (Lake et al., 2015) consists of 1623 different handwritten characters from 50 different alphabets. **The Street View House Numbers (SVHN)** (Netzer et al., 2011) is a real-world digit recognition dataset with around 70,000 images. **UCF101** (Soomro et al., 2012) is an action recognition dataset of realistic human action videos, collected from YouTube. It contains 13,320 videos for 101 action categories. Each video has been summarized into an image based on a ranking principle using the Dynamic Image encoding of (Bilen et al., 2016). The images have been pre-processed to the size of 72×72. Following the benchmark for DG in (Li et al., 2019b), we take the six larger datasets (CIFAR-100, DPed, GTSR, Omniglot, SVHN and ImageNet) as source domains and hold out the four smaller datasets (FGVC-Aircraft, DTD, Flowers102 and UCF101) as target domains.

Implementation details. We parameterize the shared feature extractor (meta-learner) h as a pre-trained ResNet-18 (He et al., 2016) network followed by (Li et al., 2019b), and the task-specific learners as the SVM classifier. The competing methods include Reptile

Table 7: Cross-domain recognition accuracy (%) of four held out target datasets on VD using train split in (Li et al., 2019b) for training.

Target	Reptile	CrossGrad	MR	MR-FL	AGG	FC	FC+Tanh	FC+L^2-SP	FC+Tanh+L^2-SP
FGVC-Aircraft	19.62	19.92	20.91	18.18	18.84	19.02	19.35	**19.80**	19.60
DTD	37.39	36.54	32.34	35.69	37.52	39.01	39.20	39.04	**40.37**
Flowers102	58.26	57.84	35.49	53.04	57.64	58.13	58.62	58.43	**60.78**
UCF101	49.85	45.80	47.34	48.10	47.88	51.12	51.38	51.48	**52.12**
Ave.	41.28	40.03	34.02	38.75	40.47	41.82	42.14	42.19	**43.22**

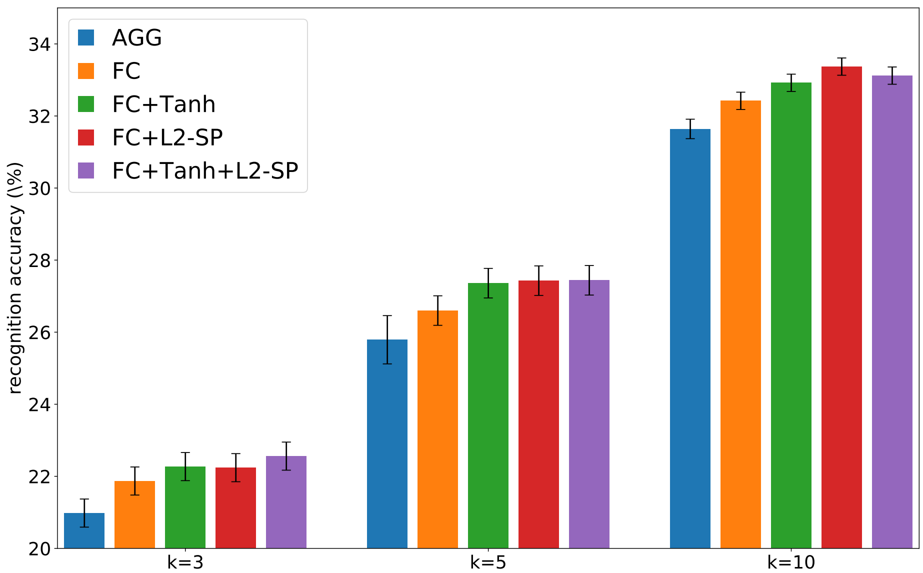


Figure 4: Recognition accuracies (%) of all competing methods averaged over 5 test runs on VD K-shot learning

(Nichol et al., 2018), CrossGrad (Shankar et al., 2018), MetaReg (Balaji et al., 2018; Li et al., 2019b) and FC (Li et al., 2019b). We also follow the experimental setting in (Li et al., 2019b), and train all components end-to-end using the AMSGrad (batch-size/per meta-train domain=64, batch-size/per meta-test domain=32, lr=0.0005, weight decay=0.0001) for 30k iterations where the learning rate decayed in 5K, 12K, 15K, 20K iterations by a factor 5, 10, 50, 100, respectively. Similar to MetaReg (Balaji et al., 2018), after the parameters are trained via meta learning, we fine-tune the network on all source datasets for the final 10k iterations. The Feature-Critic (set embedding variant) is adopted as the MLP type, and is trained with AMSGrad optimizer (lr=0.0001, weight decay=0.00001).

Results. Table 7 shows the classification accuracy on four hold-out target domain. The results of Reptile, CrossGrad, MR, MR-FL are directly taken from (Li et al., 2019b). We re-implemented the result of AGG (Li et al., 2019b) and FC. Different from homogeneous DG, heterogeneous DG assumes that the label spaces among different domains are different, and thus the task-specific learners can not be shared. Thus learning a robust off-the-shelf feature extractor is very important for final performance of heterogeneous DG. From the table, it can be seen that our proposed regularization schemes produce more robust and effective feature extractor for unseen domains compared with other competing methods. Especially, although our methods are easily rebuilt from FC, they can attain evident improvements. Furthermore, we repeat the evaluation assuming that few-shot (3-shot, 5-shot, 10-shot) samples of training splits are available for SVM training in the meta-test stage. Fig. 4 reports the target domain test accuracies under these settings. It can be easily observed that our proposed regularization schemes provide a consistent improvement over the baseline, and produce a superior off-the-shelf feature representation. It can thus be substantiated that the proposed regularization strategy is hopeful to generally improve the training quality of current meta learning methods to produce more robust and effective feature extractor.

8. Conclusion

This study has introduced an understanding and formulation of the meta learning framework, in which a meta-learner is extracted to get the hyperparameter prediction policy for machine learning over training task set. The meta-learner is represented as an explicit function mapping from task information to the hyperparameters involved in the learning process, facilitating the meta-learned meta-learner able to be readily transferred to new tasks to adaptively set their hyperparametric configurations. Besides, the corresponding learning theory is developed for this learning framework. Very similar to the SRM principle used to improve generalization capability of the extracted learner in conventional machine learning, this theory can help conduct some useful regularization strategies for ameliorating the generalization capability of the extracted meta-learner as a diverse-task-transferable learning methodology. We have further substantiated the beneficial effects brought by these regularization strategies in typical meta learning applications, including few-shot regression, few-shot classification, and domain generalization. Especially, the new regularization schemes can be easily embedded into the current meta learning programs by directly replacing the form of the output activation function in the learning model, or revising the parameter updating step for the learner or meta-learner from the unregularized solution to the regularized one. These regularized strategies are thus hopeful to be easily and extensively used in more comprehensively meta learning tasks.

In future research, we'll further ameliorate the presented meta learning theory, and especially try to deduce much tighter upper bound for evaluating the meta learning error to reveal more intrinsic generalization insight of this learning framework. Besides, we'll make endeavor to explore more helpful regularizer schemes useful for more comprehensive and diverse meta learning problems. Specifically, we'll continue to explore how to effectively design the structure and rectify the parameter learning of meta-learner by enforcing regularization on the model through certain regularization tricks, so as to improve its generalization capability among variant tasks. We believe this will be potentially beneficial to the meta learning field, just like the significance of the SRM principle in the conventional machine learning field.

Appendix A. Theoretical Tools

In this part, we will show some important auxiliary theoretical results and assumptions preparing for the proofs of the main theorems.

A.1 Some Properties of Gaussian Complexity

Here, we present some properties of Gaussian complexity, and the proofs can refer to the references therein.

Proposition 4 (Ledoux-Talagrand contraction principle (Ledoux and Talagrand, 2013)) *Let $\mathcal{X}$ be any set, $\mathcal{F}$ be a class of functions: $f : \mathcal{X} \to \mathbb{R}^d$. Then for N data points, $\mathbf{X} = (x_1, \cdots, x_N)^\mathsf{T}$, and a fixed, centered L-Lipschitz function $\phi : \mathbb{R}^d \to \mathbb{R}$, we have*

$$\hat{\mathcal{G}}_{\mathbf{X}}(\phi(\mathcal{F})) \leq L\hat{\mathcal{G}}_{\mathbf{X}}(\mathcal{F}); \quad \hat{\mathfrak{R}}_{\mathbf{X}}(\phi(\mathcal{F})) \leq L\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}),$$

where $\hat{\mathcal{G}}_{\mathbf{X}}(\mathcal{F})$ and $\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F})$ denote the empirical Gaussian complexity and empirical Rademacher complexity, respectively.

Proposition 5 (The relationship between Gaussian complexity and Rademacher complexity (Wainwright, 2019)) *The empirical Rademacher complexity can be lower/upper bounded by empirical Gaussian complexity as follows:*

$$\hat{\mathcal{G}}_{\mathbf{X}}(\mathcal{F}) \leq 2\sqrt{\log(N)} \cdot \hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}), \ \hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}) \leq \sqrt{\frac{\pi}{2}}\hat{\mathcal{G}}_{\mathbf{X}}(\mathcal{F}) \leq \frac{3}{2}\hat{\mathcal{G}}_{\mathbf{X}}(\mathcal{F}),$$

where $\mathbf{X} = (x_1, \cdots, x_N)^\mathsf{T}$.

A.2 Task-averaged Estimation Error

For the single-task setting, the uniform bounds on the estimation error based on the Rademacher complexity can be found in (Mohri et al., 2018). Here we show the estimation error for the multi-task setting.

Theorem 6 *Let $\mathcal{F}$ be a family of function mappings $f_t : \mathcal{X} \to [0, B]$, and let $\mu_1, \mu_2, \cdots, \mu_T$ be probability measures on $\mathcal{X}$ with $\mathbf{X} = (\mathbf{X}_1, \cdots, \mathbf{X}_T) \sim \prod_{t=1}^T (\mu_t)^{n_t}$, where $\mathbf{X}_t = (x_{t1}, \cdots, x_{t,n_t})$ for each $t \in [T]$. Let $\mathbf{f} = (f_1, \cdots, f_T)$, we define $\hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}] = \frac{1}{T}\sum_{t=1}^T \frac{1}{n_t}\sum_{j=1}^{n_t} f_t(x_{ti})$, and $\mathbb{E}[\mathbf{f}] = \mathbb{E}_{\mathbf{X}}[\hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}]]$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ we have*

$$\mathbb{E}[\mathbf{f}] \leq \hat{\mathbb{E}}_S[\mathbf{f}] + 2\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}^{\otimes T}) + 3\frac{B}{T}\sqrt{\sum_{t=1}^T \frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}},$$

where $\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}^{\otimes T}) = \mathbf{E}_\sigma[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \frac{1}{T}\sum_{t=1}^T \frac{1}{n_t}\sum_{j=1}^{n_t} \sigma_{tj} f_t(x_{ti})]$, and $\mathcal{F}^{\otimes T} = \underbrace{\mathcal{F} \times \mathcal{F} \cdots \mathcal{F}}_{T}$.

Proof We define the function $\Phi : \mathbf{X} \to \mathbb{R}$ as

$$\Phi(\mathbf{X}) = \sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} (\mathbb{E}[\mathbf{f}] - \hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}]).$$

Let $\mathbf{X}$ and $\mathbf{X}'$ be two samples differing by exactly one point, say x_{tj} in $\mathbf{X}$ and x'_{tj} in $\mathbf{X}'$. Then, since the difference of suprema does not exceed the supremum of the difference, we have

$$\begin{aligned}|\Phi(\mathbf{X}') - \Phi(\mathbf{X})| \leq & \left|\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} (\hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}] - \hat{\mathbb{E}}_{\mathbf{X}'}[\mathbf{f}])\right| \\ =& \frac{1}{Tn_t}\left|\sup_{f_t\in\mathcal{F}} f_t(\mathbf{x}'_{tj}) - f_t(\mathbf{x}_{tj})\right| \leq \frac{B}{Tn_t}.\end{aligned}$$

Then, based on the McDiarmid's inequality (Mohri et al., 2018), we have:

$$\begin{aligned}\mathbb{P}\left(\Phi(\mathbf{X}) - \mathbb{E}_{\mathbf{X}}[\Phi(\mathbf{X})] \geq \epsilon\right) \leq & \exp\left(\frac{-2\epsilon^2}{\sum_{t=1}^{T}\sum_{j=1}^{n_t}(\frac{B}{Tn_t})^2}\right) \\ =& \exp\left(\frac{-2T^2\epsilon^2}{B^2\sum_{t=1}^{T}\frac{1}{n_t}}\right).\end{aligned}$$

For $\delta > 0$, setting the right-hand side above to be $\delta/2$, with probability at least $1-\delta/2$, the following holds:

$$\Phi(\mathbf{X}) \leq \mathbb{E}_{\mathbf{X}}[\Phi(\mathbf{X})] + \frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}}.$$

We bound the expectation of the right-hand side as follows:

$$\mathbb{E}_{\mathbf{X}}[\Phi(\mathbf{X})] = \mathbb{E}_{\mathbf{X}}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \mathbb{E}[\mathbf{f}] - \hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}]\right] \tag{21}$$

$$=\mathbb{E}_{\mathbf{X}}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \mathbb{E}_{\mathbf{X}'}[\hat{\mathbb{E}}_{\mathbf{X}'}[\mathbf{f}] - \hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}]]\right] \tag{22}$$

$$\leq\mathbb{E}_{\mathbf{X},\mathbf{X}'}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \hat{\mathbb{E}}_{\mathbf{X}'}[\mathbf{f}] - \hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}]\right] \tag{23}$$

$$=\mathbb{E}_{\mathbf{X},\mathbf{X}'}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{j=1}^{n_t}(f_t(\mathbf{x}'_{tj}) - f_t(\mathbf{x}_{tj}))\right] \tag{24}$$

$$=\mathbb{E}_{\sigma,\mathbf{X},\mathbf{X}'}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{j=1}^{n_t}\sigma_{tj}(f_t(\mathbf{x}'_{tj}) - f_t(\mathbf{x}_{tj}))\right] \tag{25}$$

$$\leq\mathbb{E}_{\sigma,\mathbf{X}'}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{j=1}^{n_t}\sigma_{tj}f_t(\mathbf{x}'_{tj})\right] + \mathbb{E}_{\sigma,\mathbf{X}}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{j=1}^{n_t}-\sigma_{tj}f_t(\mathbf{x}_{tj})\right] \tag{26}$$

$$=2\mathbb{E}_{\sigma,\mathbf{X}}\left[\sup_{\mathbf{f}\in\mathcal{F}^{\otimes T}} \frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{j=1}^{n_t}\sigma_{tj}f_t(\mathbf{x}_{tj})\right] = 2\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}^{\otimes T}). \tag{27}$$

We introuce a ghost sample (denoted by S'), drawn from the same distribution as our original sample (denoted by S), and thus $\mathbb{E}[\mathbf{f}] = \mathbb{E}_{S'}[\hat{\mathbb{E}}_{S'}[\mathbf{f}]$ (Eq.(22)). The inequality (23) holds due to the sub-additivity of the supremum function. In Eq. (25), we introduce Rademacher variables σ_{tj}, which are uniformly distributed independent random variables taking values in $\{-1, 1\}$. Eq. (26) holds by the sub-additivity of the supremum function, and Eq. (27) stems from the definition of Rademacher complexity and the fact that the variables σ_{tj} and $-\sigma_{tj}$ are distributed in the same way.

By using again the McDiarmid's inequality, with probability $1 - \delta/2$ the following holds

$$\mathfrak{R}_{\mathbf{X}}(\mathcal{F}^{\otimes T}) \leq \hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}^{\otimes T}) + \frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}}.$$

Finally, we use the union bound to combine above inequalities, which yields that with probability at least $1 - \delta$ it holds:

$$\mathbb{E}[\mathbf{f}] \leq \hat{\mathbb{E}}_S[\mathbf{f}] + 2\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}^{\otimes T}) + 3\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}}. \tag{28}$$

■

Remark 1 *When $n_t = n$ for $t = 1, 2, \cdots, T$, the conclusion becomes*

$$\mathbb{E}[\mathbf{f}] \leq \hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}] + 2\hat{\mathfrak{R}}_{\mathbf{X}}(\mathcal{F}^{\otimes T}) + 3B\sqrt{\frac{\log\frac{2}{\delta}}{2nT}},$$

where $\hat{\mathbb{E}}_{\mathbf{X}}[\mathbf{f}] = \frac{1}{nT}\sum_{t=1}^{T}\sum_{j=1}^{n} f_t(\mathbf{x}_{ti})$. This recovers the result in Theorem 9 of (Maurer et al., 2016).

Remark 2 *When $T = 1$, it recovers the result of the single task case (Mohri et al., 2018).*

A.3 A Chain Rule for Gaussian Complexity $\mathcal{F}^{\otimes T}(\mathcal{H})$

Suppose we have the training task dataset $\boldsymbol{D} = \{D_t, t \in [T]\}$, with $D_t = (D_t^{tr}, D_t^{val})$, where $D_t^{tr} = \{(x_{ti}^{(s)}, y_{ti}^{(s)})\}_{i=1}^{m_t}, D_t^{val} = \{(x_{tj}^{(q)}, y_{tj}^{(q)})\}_{j=1}^{n_t}$. In this section, we aim to decouple the complexity of learning the class $\mathcal{F}^{\otimes T}(\mathcal{H})$ based on the chain rule tools (Maurer, 2016; Wainwright, 2019; Tripuraneni et al., 2020). We let $f_t^{(h)} = \mathcal{LM}(D_t^{tr}; h)$ in the following for simplicity.

The empirical Gaussian complexity of function class $\mathcal{F}^{\otimes T}(\mathcal{H})$ can be written as

$$\hat{\mathcal{G}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T}) = \mathbb{E}\sup_{f\in\mathcal{F}^{\otimes T}, h\in\mathcal{H}}\frac{1}{T}\sum_{t=1}^{T}\frac{1}{n_t}\sum_{j=1}^{n_t} g_{tj} f_t^{(h)}(x_{tj}^{(q)});\ g_{tj} \sim \mathcal{N}(0, 1), \tag{29}$$

where $D^{val} = \{D_t^{val}\}_{t=1}^{T}$. Generally, the Ledoux-Talagrand contraction principle in Proposition 4 can be applied to decoupling the complexity. However, we learn the meta-learner

h and the task-specific learners $f_t, t \in [T]$ simultaneously after observing the data, where learners are not fixed.

To present the result, we firstly list some important theorems for the proof.

Theorem 7 (Dudley's entropy integral bound (Wainwright, 2019)) *Let $\{\mathbf{X}_\vartheta, \vartheta \in \mathbb{T}\}$ be a zero-mean sub-Gaussian process with respect to induced pseudometric $\rho_{2,\mathbf{X}}$, and $Dis(\mathbf{X}) = \sup_{\vartheta,\vartheta' \in \mathbb{T}} \rho_{2,\mathbf{X}}(\vartheta, \vartheta')$. Then for any $\delta \in [0, E]$, we have*

$$E\left[\sup_{\vartheta,\vartheta' \in \mathbb{T}} (\mathbf{X}_\vartheta - \mathbf{X}_{\vartheta'})\right] \leq 2\mathbf{E}\left[\sup_{\gamma,\gamma' \in \mathbb{T}, \rho_{2,\mathbf{X}}(\gamma,\gamma') \leq \delta} (\mathbf{X}_\gamma - \mathbf{X}_{\gamma'})\right] + 32 \int_{\delta/4}^{E} \sqrt{\log N_{2,\mathbf{X}}(\mu, \rho_{2,\mathbf{X}}, \mathbb{T})}d\mu,$$

where $N_{2,\mathbf{X}}(\mu, \rho_{2,\mathbf{X}}, \mathbb{T})$ denotes the μ-covering number of $\mathbb{T}$ in the metric $\rho_{2,\mathbf{X}}$.

Theorem 8 (Sudakov minoration (Wainwright, 2019)) *Let $\{\mathbf{X}_\vartheta, \vartheta \in \mathbb{T}\}$ be a zero-mean sub-Gaussian process defined on the non-empty set $\mathbb{T}$. Then*

$$\mathbb{E}\left[\sup_{\vartheta \in \mathbb{T}} \mathbf{X}_\vartheta\right] \geq \sup_{\delta > 0} \frac{\delta}{2} \sqrt{\log M_{2,\mathbf{X}}(\delta, \rho_{2,\mathbf{X}}, \mathbb{T})},$$

where $M_{2,\mathbf{X}}(\delta, \rho_{2,\mathbf{X}}, \mathbb{T})$ is the δ-packing number of $\mathbb{T}$ in the metric $\rho_{2,\mathbf{X}}$.

Now, the following gives the decomposition theorm for Gaussian complexity $\hat{\mathcal{G}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T})$.

Theorem 9 *Let the function class $\mathcal{F}$ consist of functions that are ℓ_2-Lipschitz with constant $L(\mathcal{F})$. Define $Dis(D^{val}) = \sup_{h,h'} \rho_{2,D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')})$, $\mathbf{f}^{(h)} = (f_1^{(h)}, f_2^{(h)}, \cdots, f_T^{(h)})$, and then the empirical Gaussian complexity of function class $\mathcal{F}_{\mathcal{H}}^{\otimes T}$ satisfies*

$$\hat{\mathcal{G}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T}) \leq \frac{2Dis(D^{val})}{(\sum_{t=1}^T n_t)^2} \sqrt{\sum_{t=1}^T \frac{1}{\beta_t T}} + 128 \log(4 \sum_{t=1}^T n_t) \cdot L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H}),$$

where $\rho_{2,D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')}) = \frac{1}{\sum_{t=1}^T n_t} \sum_{i=1}^T \sum_{j=1}^{n_t} (f_t^{(h)}(\mathbf{z}_{tj}^{(q)}) - f_t^{(h')}(\mathbf{z}_{tj}^{(q)}))^2$, $\mathbf{f}^{(h)} = (f_1^{(h)}, f_2^{(h)}, \cdots, f_T^{(h)})$, $f_t^{(h)} = \mathcal{LM}(D_t^{tr}; h)$, and $L(\mathcal{F})$ is the Lipschitz constant of $f^{(h)}$ with respect to h.

Proof For ease of notation we define $N = nT = \sum_{t=1}^T n_t, n_t = \beta_t n$ in the following. We define the mean-zero stochastic process $Z_{\mathbf{f}^{(h)}} = \frac{1}{\sqrt{\sum_{t=1}^T n_t}} \sum_{i=1}^T \sum_{j=1}^{n_t} g_{tj}/\beta_t \cdot f_t^{(h)}(x_{tj}^{(q)})$ for a sequence of data points $x_{tj}^{(q)}$. Note that the process $Z_{\mathbf{f}^{(h)}}$ has sub-Gaussian increments, and then $Z_{\mathbf{f}^{(h)}} - Z_{\mathbf{f}^{(h')}}$ is a sub-Gaussian random variable with parameter $\rho_{2,D^{val}}(\mathbf{f}^{(h)}, \mathbf{f}^{(h')}) = \frac{1}{\sum_{t=1}^T n_t} \sum_{i=1}^T \sum_{j=1}^{n_t} (f_t^{(h)}(x_{tj}^{(q)}) - f_t^{(h)}(x_{tj}^{(q)}))^2$. Since $Z_{\mathbf{f}^{(h)}}$ is a mean-zero stochastic process, we have

$$\mathbb{E}[\sup_{\mathbf{f}^{(h)} \in \mathcal{F}_{\mathcal{H}}^{\otimes T}} Z_{\mathbf{f}^{(h)}}] = \mathbb{E}[\sup_{\mathbf{f}^{(h)} \in \mathcal{F}_{\mathcal{H}}^{\otimes T}} Z_{\mathbf{f}^{(h)}} - Z_{\mathbf{f}^{(h')}}] \leq \mathbb{E}[\sup_{\mathbf{f}^{(h)}, \mathbf{f}^{(h')} \in \mathcal{F}_{\mathcal{H}}^{\otimes T}} Z_{\mathbf{f}^{(h)}} - Z_{\mathbf{f}^{(h')}}].$$

According to the Dudley entropy integral bound (Theorem 7), we have

$$\begin{aligned}
&\mathbb{E}\left[\sup_{\mathbf{f}^{(h)},\mathbf{f}^{(h')}\in\mathcal{F}_{\mathcal{H}}^{\otimes T}} Z_{\mathbf{f}^{(h)}} - Z_{\mathbf{f}^{(h')}}\right] \\
\leq & 2\mathbb{E}\left[\sup_{\substack{\mathbf{f}^{(h)},\mathbf{f}^{(h')}\in\mathcal{F}_{\mathcal{H}}^{\otimes T}\\ \rho_{2,\mathbf{Z}}(\mathbf{f}^{(h)},\mathbf{f}^{(h')})\leq\delta}} Z_{\mathbf{f}^{(h)}} - Z_{\mathbf{f}^{(h')}}\right] + 32\int_{\delta/4}^{Dis(D^{val})}\sqrt{\log N_{D^{val}}(\mu;\rho_{2,D^{val}},\mathcal{F}_{\mathcal{H}}^{\otimes T})}d\mu.
\end{aligned}$$

The first term in the right hand can be computed as:

$$\begin{aligned}
&\mathbb{E}\sup_{\substack{\mathbf{f}^{(h)},\mathbf{f}^{(h')}\in\mathcal{F}_{\mathcal{H}}^{\otimes T}\\ \rho_{2,D^{val}}(\mathbf{f}^{(h)},\mathbf{f}^{(h')})\leq\delta}} Z_{\mathbf{f}^{(h)}} - Z_{\mathbf{f}^{(h')}} \\
=&\mathbb{E}\sup_{\substack{\mathbf{f}(h),\mathbf{f}(h')\in\mathcal{F}_{\mathcal{H}}^{\otimes T}\\ \rho_{2,D^{val}}(\mathbf{f}^{(h)},\mathbf{f}^{(h')})\leq\delta}} \frac{1}{\sqrt{\sum_{t=1}^T n_t}}\sum_{i=1}^T\sum_{j=1}^{n_t} g_{tj}/\beta_t\cdot(f_t^{(h)}(x_{tj}^{(q)}) - f_t^{(h')}(x_{tj}^{(q)})) \\
\leq&\mathbb{E}\sup_{\mathbf{v}:\|\mathbf{v}\|_2\leq\delta}\sqrt{\sum_{t=1}^T\sum_{j=1}^{n_t}(g_{ij}/\beta_t)^2\cdot\|\mathbf{v}\|} \\
\leq&\sqrt{\sum_{t=1}^T\sum_{j=1}^{n_t}\frac{1}{\beta_t^2}}\cdot\delta\leq\delta\sqrt{n\sum_{t=1}^T\frac{1}{\beta_t}}.
\end{aligned}$$

Let $C_{\mathcal{H}}$ be a covering of the function space $\mathcal{H}$ in the empirical ℓ_2-norm at scale ϵ_1 with respect to D^{val}. We will claim that $C_{\mathcal{F}_{\mathcal{H}}^{\otimes T}} = \cup_{h\in\mathcal{H}} C_{\mathcal{F}_h^{\otimes T}}$ is an $\epsilon_1\cdot L(\mathcal{F})$-covering for the function space $\mathcal{F}_{\mathcal{H}}^{\otimes T}$ in the empirical ℓ_2-norm. To see this, for any $h\in\mathcal{H},\mathbf{f}\in\mathcal{F}^{\otimes T}$, we let $h'\in C_{\mathcal{H}}$ be ϵ_1-close to h in $\mathcal{H}$. By construction we have $\mathbf{f}^{(h')}\in C_{\mathcal{F}_{\mathcal{H}}^{\otimes T}}$. The following inequality estabalishes the claim,

$$\begin{aligned}
\rho_{2,D^{val}}(\mathbf{f}^{(h)},\mathbf{f}^{(h')}) =& \frac{1}{\sum_{t=1}^T n_t}\sum_{t=1}^T\sum_{j=1}^{n_t}(f_t^{(h)}(x_{tj}^{(q)}) - f_t^{(h')}(x_{tj}^{(q)}))^2 \\
\leq& L(\mathcal{F})\cdot\frac{1}{\sum_{t=1}^T n_t}\sum_{t=1}^T\sum_{j=1}^{n_t}(h(D_t^{val}) - h'(D_t^{val}))^2 \\
=& L(\mathcal{F})\cdot\rho_{2,D^{val}}(h,h')\leq\epsilon_1\cdot L(\mathcal{F}),
\end{aligned}$$

where the first inequality holds since the Lipschitz propety of the learning method $\mathcal{LM}$ with respect to h. Now, the cardinaity of $C_{\mathcal{F}_{\mathcal{H}}^{\otimes T}}$ can be bounded as

$$|C_{\mathcal{F}_{\mathcal{H}}^{\otimes T}}| = \sum_{h\in C_{\mathcal{H}}}|C_{\mathcal{F}_h}^{\otimes T}|\leq|C_{\mathcal{H}}|\cdot\max_{h\in C_{\mathcal{H}}}|C_{\mathcal{F}_h}^{\otimes T}|.$$

Thus, it follows that

$$\log N_{2,D^{val}}(\epsilon_1 \cdot L(\mathcal{F}), \rho_{2,D^{val}}, C_{\mathcal{F}_h}^{\otimes T}) \leq \log N_{2,D^{val}}(\epsilon_1, \rho_{2,D^{val}}, \mathcal{H}).$$

We define $\epsilon_1 = \frac{\epsilon}{L(\mathcal{F})}$ to show that

$$\int_{\delta/4}^{Dis(D^{val})} \sqrt{\log N_{2,D^{val}}(\epsilon, \rho_{2,D^{val}}, C_{\mathcal{F}_{\mathcal{H}}^{\otimes T}})d\epsilon} \leq \int_{\delta/4}^{Dis(D^{val})} \sqrt{\log N_{2,D^{val}}(\epsilon/(L(\mathcal{F})), \rho_{2,D^{val}}, \mathcal{H})d\epsilon}.$$

Oberving that the covering number can be bounded by packing number $M(\epsilon, \rho_{2,D^{val}}, \mathcal{H})$, i.e., $N(\epsilon, \rho_{2,D^{val}}, \mathcal{H}) \leq M(\epsilon, \rho_{2,D^{val}}, \mathcal{H})$, we can then employ Sudakov minoration Theorem (Theorem 8) to upper bound the covering number by Gaussian complexity. For the covering number of $\mathcal{H}$, $N_{2,D^{val}}(\epsilon/(L(\mathcal{F})), \rho_{2,D^{val}}, \mathcal{H})$, we can apply the theorem with mean-zero Gaussian process $\frac{1}{\sqrt{\sum_{t=1}^T n_t}} \sum_{t=1}^T \sum_{j=1}^{n_t} g_{tj} \cdot h(D_t^{val})$, with $g_{tj} \sim \mathcal{N}(0,1)$,

$$\log N_{2,D^{val}}(\epsilon/(L(\mathcal{F})), \rho_{2,D^{val}}, \mathcal{H})d\epsilon \leq 4\left(\frac{\sqrt{\sum_{t=1}^T n_t}\hat{\mathcal{G}}_{D^{val}}(\mathcal{H})}{\epsilon/(L(\mathcal{F}))}\right)^2 = 4\left(\frac{\sqrt{\sum_{t=1}^T n_t}L(\mathcal{F})\hat{\mathcal{G}}_{D^{val}}(\mathcal{H})}{\epsilon}\right)^2.$$

Finally, we can show that

$$\begin{aligned}
&\hat{\mathcal{G}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T}) = \mathbb{E}[\frac{1}{\sqrt{nT}} \sup_{\mathbf{f} \in \mathcal{F}^{\otimes T}, h \in \mathcal{H}} Z_{\mathbf{f}(h)}] \\
&\leq \frac{1}{\sqrt{nT}}\left(2\delta\sqrt{n\sum_{t=1}^T \frac{1}{\beta_t}} + 64L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) \cdot \sqrt{\sum_{t=1}^T n_t}\int_{\delta/4}^{Dis(D^{val})} \frac{1}{\mu}d\mu\right) \\
&\leq 2\delta\sqrt{\sum_{t=1}^T \frac{1}{\beta_t T}} + 64\log(\frac{4Dis(D^{val})}{\delta})\left(L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H})\sqrt{\frac{\sum_{t=1}^T n_t}{nT}}\right) \\
&= 2\delta\sqrt{\sum_{t=1}^T \frac{1}{\beta_t T}} + 64\log(\frac{4Dis(D^{val})}{\delta}) \cdot L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H}).
\end{aligned}$$

Let $\delta = \frac{Dis(D^{val})}{(nT)^2}$, we have

$$\begin{aligned}
\hat{\mathcal{G}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T}) \leq& \frac{2Dis(D^{val})}{(nT)^2}\sqrt{\sum_{t=1}^T \frac{1}{\beta_t T}} + 128\log(4nT) \cdot L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) \\
=& \frac{2Dis(D^{val})}{(\sum_{t=1}^T n_t)^2}\sqrt{\sum_{t=1}^T \frac{1}{\beta_t T}} + 128\log(4\sum_{t=1}^T n_t) \cdot L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{D^{val}}(\mathcal{H}).
\end{aligned}$$

■

Appendix B. Proof of Theorem 2 : Excess Task Avergae Risk in Meta-Training Stage

Proof Recall that

$$R_\Gamma(h) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^{n_t}} \mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val}),$$
$$\hat{R}_{\boldsymbol{D}}(h) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val}),$$

and we denote $\tilde{R}_\Gamma(h), \overline{R}_\Gamma(h)$ as

$$\tilde{R}_\Gamma(h) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^{n_t}} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val})$$
$$\overline{R}_\Gamma(h) = \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val}).$$

For the task avergae excess risk $R_{train}(\hat{\mathbf{f}}, \hat{h}) - R_{train}(\mathbf{f}^*, h^*)$, we have the following decomposition

$$\begin{aligned}
&R_{train}(\hat{h}) - R_{train}(h^*)\\
=&R_\Gamma(\hat{h}) - R_\Gamma(h^*)\\
=&\underbrace{R_\Gamma(\hat{h}) - \overline{R}_\Gamma(\hat{h})}_{a} + \underbrace{\overline{R}_\Gamma(\hat{h}) - \hat{R}_\Gamma(\hat{h})}_{b} + \underbrace{\hat{R}_\Gamma(\hat{h}) - \hat{R}_\Gamma(h^*)}_{c} + \underbrace{\hat{R}_\Gamma(h^*) - \overline{R}_\Gamma(h^*)}_{d} + \underbrace{\overline{R}_\Gamma(h^*) - R_\Gamma(h^*)}_{e}.
\end{aligned}$$

For the terms (a) and (e), we have

$$\begin{aligned}
&(a) + (e)\\
\le&\sup_{h\in\mathcal{H}} 2\left|\frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^{n_t}} \mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val}) - \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}} \mathcal{L}(\mathcal{LM}(D_t^{tr}; h), D_t^{val})\right|\\
\le&4L\hat{\mathfrak{R}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T}) + 3\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}} \le 6L\hat{\mathcal{G}}_{D^{val}}(\mathcal{F}_{\mathcal{H}}^{\otimes T}) + 6\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}}\\
\le&768L\log(4\sum_{t=1}^{T} n_t)\cdot L(\mathcal{F})\cdot\hat{\mathcal{G}}_{D^{val}}(\mathcal{H}) + \frac{12LDis(D^{val})}{(\sum_{t=1}^{T} n_t)^2}\sqrt{\sum_{t=1}^{T}\frac{1}{\beta_t T}} + 6\frac{B}{T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_t}}\sqrt{\frac{\log\frac{2}{\delta}}{2}},
\end{aligned}$$

where $D^{val} = \{D_t^{val}\}_{t=1}^T$, and $|D_t^{val}| = n_t$, for $t \in [T]$. The second inequallity we use the task-averaged estimation error in Theorem 6 and the Ledoux-Talagrand contraction principle in Proposition 4. Proposition 5 is employed for the third inequallity. And the last inequallity holds for Theorem 9.

For the term (b), we have

$$
\begin{aligned}
&\overline{R}_\Gamma(\hat{h}) - \hat{R}_\Gamma(\hat{h}) \\
=&\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val}) - \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val}) \\
=&\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val}) - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{tr})}_{b1} \\
&+\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{tr}) - \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{tr})}_{b2} \\
&+\underbrace{\frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{tr}) - \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val})}_{b3}.
\end{aligned}
$$

For the terms ($b1$) and ($b3$), we have

$$
(b1) + (b3) \leq \frac{2}{T}\sum_{t=1}^{T} d_{\mathcal{F}}(\mu_t^s, \mu_t^q),
$$

where $d_{\mathcal{F}}(\mu_t^s, \mu_t^q)$ denotes the discrepancy divergence (Ben-David et al., 2010) between samples from the probability distributions μ_t^s and μ_t^q with respect to the hypothesis class $\mathcal{F}$,

$$
d_{\mathcal{F}}(\mu_t^s, \mu_t^q) = \sup_{f\in\mathcal{F}}\left|\mathbb{E}_{D_t^{val}\sim(\mu_t^q)^{n_t}}\mathcal{L}(f, D_t^{val}) - \mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}}\mathcal{L}(f, D_t^{tr})\right|.
$$

For the term ($b2$), we denote $\hat{f}_t^{(h^*)} = \mathcal{LM}(D_t^{tr}; h^*)$, and then we have

$$
\begin{aligned}
b2 &= \frac{1}{T}\sum_{t=1}^{T}\mathcal{L}(\hat{f}_t^{(h^*)}, D_t^{tr}) - \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^{m_t}}\mathcal{L}(\hat{f}_t^{(h^*)}, D_t^{tr}) \\
&\leq \frac{1}{T}\sum_{t=1}^{T}\left(2L\hat{\mathfrak{R}}_{D_t^{tr}}(\mathcal{F}) + 3B\sqrt{\frac{\log\frac{2}{\delta}}{m_t}}\right) \leq \frac{3L}{T}\sum_{t=1}^{T}\hat{\mathcal{G}}_{D_t^{tr}}(\mathcal{F}) + \frac{3B}{T}\sum_{t=1}^{T}\sqrt{\frac{\log\frac{2}{\delta}}{m_t}}.
\end{aligned}
$$

Thus we have

$$
b \leq \frac{2}{T}\sum_{t=1}^{T} d_{\mathcal{F}}(\mu_t^s, \mu_t^q) + \frac{3L}{T}\sum_{t=1}^{T}\hat{\mathcal{G}}_{D_t^{tr}}(\mathcal{F}) + \frac{3B}{T}\sum_{t=1}^{T}\sqrt{\frac{\log\frac{2}{\delta}}{m_t}}. \tag{30}
$$

Similar process can also be applied to the term (d), and (d) can be bounded by

$$
d \leq \frac{2}{T}\sum_{t=1}^{T} d_{\mathcal{F}}(\mu_t^s, \mu_t^q) + \frac{3L}{T}\sum_{t=1}^{T}\hat{\mathcal{G}}_{D_t^{tr}}(\mathcal{F}) + \frac{3B}{T}\sum_{t=1}^{T}\sqrt{\frac{\log\frac{2}{\delta}}{m_t}}. \tag{31}
$$

For the term (c), according to $\hat{h} = \arg\min_{h \in \mathcal{H}} \hat{R}_{\boldsymbol{D}}(h)$, we have $c \leq 0$.

Combining the above results from the terms (a) to (e), we have

$$R_{train}(\hat{h}) - R_{train}(h^*) \leq 768L \log(4 \sum_{t=1}^{T} n_t) \cdot L(\mathcal{F}) \cdot \hat{\mathcal{G}}_{\mathbf{\Gamma}^{(q)}}(\mathcal{H}) + \frac{6L}{T} \sum_{t=1}^{T} \hat{\mathcal{G}}_{D_t^{tr}}(\mathcal{F})$$

$$+ \frac{4}{T} \sum_{t=1}^{T} d_{\mathcal{F}}(D_t^{(tr)}, D_t^{(val)}) + 6\frac{B}{T} \sqrt{\sum_{t=1}^{T} \frac{1}{n_t}} \sqrt{\frac{\log \frac{2}{\delta}}{2}} + \frac{6B}{T} \sum_{t=1}^{T} \sqrt{\frac{\log \frac{2}{\delta}}{m_t}} + \frac{12LDis(D^{val})}{(\sum_{t=1}^{T} n_t)^2} \sqrt{\sum_{t=1}^{T} \frac{1}{\beta_t T}}.$$

■

Appendix C. Proof of Theorem 3: Excess Transfer Error in Meta-Test Stage

Proof Recall that

$$\begin{aligned}
&R_{test}(\hat{h}) - R_{test}(h^*) \\
&= \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h})), D_\mu^{val}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};h^*)), D_\mu^{val}) \\
&= \underbrace{\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{val}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{val})}_{a} \\
&+ \underbrace{\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{val}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};h^*), D_\mu^{val})}_{b} \\
&+ \underbrace{\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};h^*), D_\mu^{val}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};h^*)), D_\mu^{val})}_{c}.
\end{aligned}$$

We now bound the first term (a) by

$$\begin{aligned}
(a) =& \underbrace{\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h})), D_\mu^{val}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{tr'}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'})}_{a1} \\
&+ \underbrace{\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{tr'}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'}) - \mathbb{E}_{\mu\sim\eta}\mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'})}_{a2} \\
&+ \underbrace{\mathbb{E}_{\mu\sim\eta}\mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'})}_{a3} \\
&+ \underbrace{\mathbb{E}_{\eta}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'}) - \mathbb{E}_{\eta}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathbb{E}_{D_\mu^{tr'}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'})}_{a4} \\
&+ \underbrace{\mathbb{E}_{\eta}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathbb{E}_{D_\mu^{tr'}\sim(\mu^s)^{m_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{tr'}) - \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{val})}_{a5},
\end{aligned}$$

where $D_t^{tr'}$ is equivalent to D_t^{tr}.

For the terms $(a1)$ and $(a5)$, we have

$$(a1) + (a5) \leq 2\mathbb{E}_{\mu\sim\eta} d_{\mathcal{F}}(\mu^s, \mu^q),$$

where $d_{\mathcal{F}}(\mu^s, \mu^q)$ denotes the discrepancy divergence (Ben-David et al., 2010) between samples from the probability distributions μ^s and μ^q with respect to the hypothesis class $\mathcal{F}$,

$$d_{\mathcal{F}}(\mu^s, \mu^q) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{D_\mu^{val}\sim(\mu^q)^{n_\mu}} \mathcal{L}(f, D_t^{val}) - \mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^{m_\mu}} \mathcal{L}(f, D_t^{tr}) \right|.$$

For the term $(a2)$, we have

$$(a2) \leq 2L\hat{\mathfrak{R}}_{D_\mu^{tr}}(\mathcal{F}) + 3B\sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}} \leq 3L\hat{\mathcal{G}}_{D_\mu^{tr}}(\mathcal{F}) + 3B\sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}}.$$

Similarly, For the term $(a4)$, we have

$$(a4) \leq 3L\hat{\mathcal{G}}_{D_\mu^{tr}}(\mathcal{F}) + 3B\sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}}.$$

For the term $(a3)$, suppose that the outer loss can be upper bounded by the inner loss, and then we have $a3 < 0$ according to the definition of $\mathcal{LM}$ function, i.e., $\mathcal{LM}(D_\mu^{tr}; \hat{h}(D_\mu^{tr}))$ minimizes the $\mathcal{L}(f, D_\mu^{tr})$ equiped with hyperparameters $h(D_\mu^{tr})$.

Therefore, combining the above analysis, the term (a) can be upper bounded as

$$(a) \leq 2\mathbb{E}_{\mu\sim\eta} d_{\mathcal{F}}(\mu^s, \mu^q) + 6L\hat{\mathcal{G}}_{D_\mu^{tr}}(\mathcal{F}) + 6B\sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}}.$$

For the term (b), according to the task diversity definition (Assumption 3), we have

$$(b) = R_\eta(\hat{h}) - R_\eta(h^*) \leq \alpha\left(R_{train}(\hat{h}) - R_{train}(h^*))\right) + \beta.$$

For the term (c), according to definition of h^*, we have $(c) \leq 0$.

Combining above results from term (a) to (c), we have

$$\begin{aligned}
&R_{test}(\hat{h}) - R_{test}(h^*) \\
\leq&\alpha\left(R_{train}(\hat{\mathbf{f}}, \hat{h}) - R_{train}(\mathbf{f}^*, h^*)\right) + \beta + 6L\hat{\mathcal{G}}_{D_\mu^{tr}}(\mathcal{F}) + 2\mathbb{E}_{\mu\sim\eta} d_{\mathcal{F}}(\mu^s, \mu^q) + 6B\sqrt{\frac{\log \frac{2}{\delta}}{m_\mu}}.
\end{aligned}$$

■

Appendix D. Proof of the Proposition 1

Proof We denote

$$\tilde{\mathbf{w}} = \arg\min_{\mathbf{w}} \mathbb{E}_{(x^{(s)}, y^{(s)})\in\mu^s} \ell(\mathbf{w}^T \hat{h}(x^{(s)}), y^{(s)})$$

and

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}_{(x^{(s)}, y^{(s)})\in\mu^s} \ell(\mathbf{w}^T h^*(x^{(s)}), y^{(s)}).$$

Observe that

$$
\begin{aligned}
& R_\eta(\hat{h}) - R_\eta(h^*) \\
= & \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^n}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^m}\mathcal{L}(\mathcal{LM}(D_\mu^{tr};\hat{h}), D_\mu^{val}) - \\
& \qquad \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{D_\mu^{val}\sim(\mu^q)^n}\mathbb{E}_{D_\mu^{tr}\sim(\mu^s)^m}\mathcal{L}(\mathcal{LM}(D_\mu^{tr};h^*), D_\mu^{val}) \\
= & \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}\left\{\left|\tilde{\mathbf{w}}^\top\hat{h}(x) - \mathbf{w}^{*\top}h^*(x)\right|^2\right\}, \\
= & \sup_{\mathbf{w}_0}\inf_{\mathbf{w}}\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}\left\{\left|\mathbf{w}^\top\hat{h}(x) - \mathbf{w}_0^\top h^*(x)\right|^2\right\} \\
= & \sup_{\mathbf{w}_0}\inf_{\tilde{\mathbf{w}}}\left\{[\tilde{\mathbf{w}}^\top \quad -\mathbf{w}_0^\top]\Lambda\begin{bmatrix}\tilde{\mathbf{w}} \\ -\mathbf{w}_0\end{bmatrix}\right\},
\end{aligned}
$$

where Λ is defined as

$$
\Lambda(\hat{h}, h^*) = \begin{bmatrix}\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[\hat{h}(x)\hat{h}(x)^\top] & \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[\hat{h}(x)h^*(x)^\top] \\ \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[h^*(x)\hat{h}(x)^\top] & \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[h^*(x)h^*(x)^\top]\end{bmatrix} = \begin{bmatrix}\mathbf{G}_{\hat{h}\hat{h}} & \mathbf{G}_{\hat{h}h^*} \\ \mathbf{G}_{h^*\hat{h}} & \mathbf{G}_{h^*h^*}\end{bmatrix}.
$$

According to the partial minimization of a convex quadratic form (Boyd et al., 2004), we have

$$
\inf_{\tilde{\mathbf{w}}}\left\{[\tilde{\mathbf{w}}^\top \quad -\mathbf{w}_0^\top]\Lambda\begin{bmatrix}\tilde{\mathbf{w}} \\ -\mathbf{w}_0\end{bmatrix}\right\} = \mathbf{w}_0^\top\Lambda_S(\hat{h}, h^*)\mathbf{w}_0,
$$

where $\Lambda_S(\hat{h}, h^*) = \mathbf{G}_{h^*h^*} - \mathbf{G}_{h^*\hat{h}}(\mathbf{G}_{\hat{h}\hat{h}})^\dagger\mathbf{G}_{\hat{h}}h^*$ is the generalized Schur complement of the representation of h^* with respect to $\hat{h}$. Furthermore, according to the variational characterization of the singular value, we have

$$
\sup_{\mathbf{w}_0:\|\mathbf{w}_0\|\le M}\mathbf{w}_0^\top\Lambda_S\mathbf{w}_0 = M\sigma_1(\Lambda_S(\hat{h}, h^*)),
$$

where σ_1 denotes the maximal singular value.

Moreover, we denote

$$
\tilde{\mathbf{w}}_t = \arg\min_{\mathbf{w}_t}\mathbb{E}_{(x_t^{(s)}, y_t^{(s)})\in\mu_t^s}\ell(\mathbf{w}_t^T\hat{h}(x_t^{(s)}), y_t^{(s)}), t\in[T]
$$

and

$$
\mathbf{w}_t^* = \arg\min_{\mathbf{w}_t}\mathbb{E}_{(x_t^{(s)}, y_t^{(s)})\in\mu^s}\ell(\mathbf{w}_t^T h^*(x_t^{(s)}), y_t^{(s)}), t\in[T].
$$

Thus we have the following derivation:

$$
\begin{aligned}
& R_{train}(\hat{h}) - R_{train}(h^*) \\
= & \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^n}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^m}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val}) - \\
& \qquad \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^n}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^m}\mathcal{L}(\mathcal{LM}(D_t^{tr};h^*), D_t^{val}) \\
= & \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{\mu_t\sim\eta}\mathbb{E}_{x\sim\mu_t^q}\left\{\left|\tilde{\mathbf{w}}_t^{\mathsf{T}}\hat{h}(x) - \mathbf{w}_t^{*T}h^*(x)\right|^2\right\} \\
= & \frac{1}{T}\sum_{t=1}^{T} \inf_{\mathbf{w}_t}\mathbb{E}_{\mu_t\sim\eta}\mathbb{E}_{x\sim\mu_t^q}\left\{\left|\mathbf{w}_t^{\mathsf{T}}\hat{h}(x) - \mathbf{w}_t^{*T}h^*(x)\right|^2\right\} \\
= & \frac{1}{T}\sum_{t=1}^{T} \mathbf{w}_t^{*T}\Lambda_S(\hat{h},h^*)\mathbf{w}_t^* = tr(\Lambda_S(\hat{h},h^*)\mathbf{P}^{\mathsf{T}}\mathbf{P}/T) = tr(\Lambda_S(\hat{h},h^*)\mathbf{K}),
\end{aligned}
$$

where $\mathbf{K} = \mathbf{P}^{\mathsf{T}}\mathbf{P}/T$. Since $\Lambda_S \succeq 0$, and $\mathbf{K} \succeq 0$, through the Von-Neumann trace inequality, we have that

$$
\begin{aligned}
tr(\Lambda_S(\hat{h},h^*)\mathbf{K}) &\geq \sum_{i=1}^{d_L}\sigma_i(\Lambda_S(\hat{h},h^*))\sigma_{d_L-i+1}(\mathbf{K}) \geq \sum_{i=1}^{d_L}\sigma_i(\Lambda_S(\hat{h},h^*))\sigma_{d_L}(\mathbf{K}) \\
&= tr(\Lambda_S(\hat{h},h^*))\sigma_{d_L}(\mathbf{K}) \geq \sigma_1(\Lambda_S(\hat{h},h^*))\sigma_{d_L}(\mathbf{K}).
\end{aligned}
$$

Now, we can deduce that

$$
R_\eta(\hat{h}) - R_\eta(h^*) \leq \frac{M}{\sigma_{d_L}(\mathbf{K})}\left\{R_{train}(\hat{h}) - R_{train}(h^*)\right\}.
$$

Based on the above derivation, we can have the conclusion. ■

Appendix E. Proof of the Proposition 2

Proof Denote $z_i(x) = f(h(x))_i$, the loss function can be written as

$$
\ell(f(h(x)),y) = -\sum_i y_i \log\left(\frac{e^{z_i}}{\sum_k e^{z_k}}\right). \tag{32}
$$

Let $a_i = \frac{e^{z_i}}{\sum_k e^{z_k}}$,

$$
\frac{\partial \ell}{\partial z_i} = \sum_j\left(\frac{\partial \ell}{\partial a_j}\frac{\partial a_j}{\partial z_i}\right), \tag{33}
$$

where $\frac{\partial \ell}{\partial a_j} = -\frac{y_j}{a_j}$, for $\frac{\partial a_j}{\partial z_i}$, if $i = j$, $\frac{\partial a_j}{\partial z_i} = \frac{\partial(\frac{e^{z_i}}{\sum_k e^{z_k}})}{\partial z_i} = \frac{e^{z_i}\sum_k e^{z_k} - (e^{z_i})^2}{(\sum_k e^{z_k})^2} = a_i(1-a_i)$; otherwise, $i \neq j$, $\frac{\partial a_j}{\partial z_i} = \frac{\partial(\frac{e^{z_j}}{\sum_k e^{z_k}})}{\partial z_i} = \frac{-e^{z_i}e^{z_j}}{(\sum_k e^{z_k})^2} = -a_i a_j$. Therefore,

$$\begin{aligned}\frac{\partial \ell}{\partial z_i} &= -\sum_{j\neq i}(\frac{y_j}{a_j}(-a_i a_j)) - \frac{y_j}{a_j}(1-a_j)\\ &= a_i \sum_j y_j - y_i = a_i - y_i,\end{aligned}$$

since $a_i \in [0,1], y_i \in \{0,1\}$, we can obtain $\frac{\partial \ell}{\partial z_i} \in [-1,1]$. Therefore, we can demonstrate that the loss function is 1-Lipschitz with respect to $f(h(X))$. ∎

Appendix F. Proof of the Proposition 3

We first present some essential theoretical results preparing for proving the Proposition 3.

Lemma 1 *Consider the following generalized linear model*

$$p(\boldsymbol{y}|\boldsymbol{\eta}) = h(\boldsymbol{y})\exp(\boldsymbol{\eta}^T t(\boldsymbol{y}) - a(\boldsymbol{\eta})), \tag{34}$$

Then we have

$$(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c}_1)}{2}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}) \leq KL(p(\boldsymbol{y}|\boldsymbol{\eta})|p(\boldsymbol{y}|\hat{\boldsymbol{\eta}})) \leq (\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c}_2)}{2}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}), \tag{35}$$

where $\boldsymbol{c}_1 = \inf_{\boldsymbol{c}\in[\hat{\boldsymbol{\eta}},\boldsymbol{\eta}]} a''(\boldsymbol{c}), \boldsymbol{c}_2 = \sup_{\boldsymbol{c}\in[\hat{\boldsymbol{\eta}},\boldsymbol{\eta}]} a''(\boldsymbol{c})$.

Proof Observe that

$$\begin{aligned}KL(p(\boldsymbol{y}|\boldsymbol{\eta})|p(\boldsymbol{y}|\hat{\boldsymbol{\eta}})) &= \int (p(\boldsymbol{y}|\boldsymbol{\eta})\log\frac{(p(\boldsymbol{y}|\boldsymbol{\eta})}{(p(\boldsymbol{y}|\hat{\boldsymbol{\eta}})}d\boldsymbol{y}\\ =&\int p(\boldsymbol{y}|\boldsymbol{\eta})^\mathsf{T}[t(\boldsymbol{y})(\boldsymbol{\eta} - \hat{\boldsymbol{\eta}}) + a(\hat{\boldsymbol{\eta}}) - a(\boldsymbol{\eta})]d\boldsymbol{y}\\ =&a'(\boldsymbol{\eta})^\mathsf{T}(\boldsymbol{\eta} - \hat{\boldsymbol{\eta}}) + a(\hat{\boldsymbol{\eta}}) - a(\boldsymbol{\eta}),\end{aligned}$$

where $a'(\boldsymbol{\eta}) = \int t(\boldsymbol{y})^\mathsf{T} p(\boldsymbol{y}|\boldsymbol{\eta})d\boldsymbol{y}$. Based on the Taylor's theorem we have that

$$a(\hat{\boldsymbol{\eta}}) = a(\boldsymbol{\eta}) + a'(\boldsymbol{\eta})^\mathsf{T}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}) + (\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c})}{2}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}),$$

where $\boldsymbol{c} \in [\hat{\boldsymbol{\eta}}, \boldsymbol{\eta}]$. Therefore, we can obtain

$$KL(p(\boldsymbol{y}|\boldsymbol{\eta})|p(\boldsymbol{y}|\hat{\boldsymbol{\eta}})) = (\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c})}{2}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}).$$

We can then obtain

$$(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c}_1)}{2}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}) \leq KL(p(\boldsymbol{y}|\boldsymbol{\eta})|p(\boldsymbol{y}|\hat{\boldsymbol{\eta}})) \leq (\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c}_2)}{2}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}),$$

where $\boldsymbol{c}_1 = \inf_{\boldsymbol{c}\in[\hat{\boldsymbol{\eta}},\boldsymbol{\eta}]}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T} a''(\boldsymbol{c})(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta}), \boldsymbol{c}_2 = \sup_{\boldsymbol{c}\in[\hat{\boldsymbol{\eta}},\boldsymbol{\eta}]}(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})^\mathsf{T} a''(\boldsymbol{c})(\hat{\boldsymbol{\eta}} - \boldsymbol{\eta})$. ∎

Remark 3 If the data generating model satisfies the conditional likelihood defined in Eq. (19), for the cross-entropy loss we have

$$
\begin{aligned}
&\mathbb{E}_{(x,y)\in D_t^{val}}[\ell(f(h(x)),y)-\ell(f(h(x)),y)]\\
=&\mathbb{E}_x[KL(\text{Multi}(\text{Softmax}(f(h(x))))|KL(\text{Multi}(\text{Softmax}(f(h(x))))],
\end{aligned}
$$

where Multi denote the multinomial distribution.

When consider the multinomial distribution, the generalized linear model in Eq. (34) satisfies that $h(\boldsymbol{y})=1, t(\boldsymbol{y})=\boldsymbol{y}, a(\boldsymbol{\eta})=\log(\sum_{k=1}^K\exp(\eta_k)), \boldsymbol{\eta}=(\eta_1,\cdots,\eta_K)^\mathsf{T}$. Then, we have

$$
a'(\boldsymbol{\eta})=\left(\frac{\exp(\eta_1)}{\sum_{k=1}^K\exp(\eta_k)},\cdots,\frac{\exp(\eta_K)}{\sum_{k=1}^K\exp(\eta_k)}\right)^\mathsf{T},
$$

$$
a''(\boldsymbol{\eta})=\begin{pmatrix}
\frac{\exp(\eta_1)}{S}-\frac{\exp(\eta_1)^2}{S^2} & -\frac{\exp(\eta_1)\exp(\eta_2)}{S^2} & \cdots & -\frac{\exp(\eta_1)\exp(\eta_{K-1})}{S^2}\\
-\frac{\exp(\eta_2)\exp(\eta_1)}{S^2} & \frac{\exp(\eta_2)}{S}-\frac{\exp(\eta_2)^2}{S^2} & \cdots & -\frac{\exp(\eta_2)\exp(\eta_K)}{S^2}\\
\vdots & \vdots & \vdots & \vdots\\
-\frac{\exp(\eta_K)\exp(\eta_1)}{S^2} & -\frac{\exp(\eta_K)\exp(\eta_2)}{S^2} & \cdots & \frac{\exp(\eta_K)}{S}-\frac{\exp(\eta_K)^2}{S^2}
\end{pmatrix},
$$

where $S=[\sum_{k=1}^K\exp(\eta_k)]$. We denote $p_i=\exp(\eta_i)/S$, Then we have

$$
\begin{aligned}
&\mathbb{E}_{(x,y)\in D_t^{val}}[\ell(f(h(x)),y)-\ell(f(h(x)),y)]=(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta})^\mathsf{T}\frac{a''(\boldsymbol{c})}{2}(\hat{\boldsymbol{\eta}}-\boldsymbol{\eta})\\
=&\sum_{k=1}^K\left[p_k-p_k^2\right]\eta_k^2-\sum_{k=1}^K\sum_{j=1,j\neq k}^K p_kp_j\eta_k\eta_j\\
\geq&\min_{k,j\in[K],k\neq j}\{p_k-p_k^2,p_kp_j\}\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}\|_2^2:=C\|\hat{\boldsymbol{\eta}}-\boldsymbol{\eta}\|_2^2.
\end{aligned}
$$

Proof of Proposition 3

Proof We denote

$$
\tilde{\boldsymbol{A}}=\arg\min_{\boldsymbol{A}}\mathbb{E}_{(x^{(s)},y^{(s)})\sim\mu^s}\ell(\boldsymbol{A}^\mathsf{T}\hat{h}(x^{(s)}),y^{(s)})
$$

and

$$
\boldsymbol{A}^*=\arg\min_{\boldsymbol{A}}\mathbb{E}_{(x^{(s)},y^{(s)})\sim\mu^s}\ell(\boldsymbol{A}^\mathsf{T}h^*(x^{(s)}),y^{(s)}).
$$

Observe that

$$
\begin{aligned}
& R_\eta(\hat{h}) - R_\eta(h^*) \\
=& \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^n}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^m}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val}) - \\
& \qquad \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^n}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^m}\mathcal{L}(\mathcal{LM}(D_t^{tr};h^*), D_t^{val}) \\
=& \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{(x,y)\sim\mu^q}\left\{\ell(\tilde{\boldsymbol{A}}^{\mathsf{T}}\hat{h}(x), y) - \ell(\boldsymbol{A}^{*\mathsf{T}}h^*(x), y)\right\} \\
\le& \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}\|\tilde{\boldsymbol{A}}^{\mathsf{T}}\hat{h}(x) - \boldsymbol{A}^{*\mathsf{T}}h^*(x)\|_2^2 \\
=& \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}\left\{\sum_{k=1}^{K}\left|(\tilde{\boldsymbol{A}})_k^{\mathsf{T}}\hat{h}(x) - (\boldsymbol{A})_k^{*\mathsf{T}}h^*(x)\right|^2\right\} \\
=& \sup_{\boldsymbol{A}'}\inf_{\boldsymbol{A}}\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}\left\{\sum_{k=1}^{K}\left|(\boldsymbol{A})_k^{\mathsf{T}}\hat{h}(x) - (\boldsymbol{A})_k^{'\mathsf{T}}h^*(x)\right|^2\right\} \\
=& \sum_{k=1}^{K}\left\{\sup_{(\boldsymbol{A})_k'}\inf_{(\boldsymbol{A})_k}[(\boldsymbol{A})_k^{\mathsf{T}} \; -(\boldsymbol{A})_k^{'\mathsf{T}}]\Lambda\begin{bmatrix}(\boldsymbol{A})_k \\ -(\boldsymbol{A})_k'\end{bmatrix}\right\},
\end{aligned}
$$

where Λ is defined as

$$
\Lambda(\hat{h}, h^*) = \begin{bmatrix}\mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[\hat{h}(x)\hat{h}(x)^{\mathsf{T}}] & \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[\hat{h}(x)h^*(x)^{\mathsf{T}}] \\ \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[h^*(x)\hat{h}(x)^{\mathsf{T}}] & \mathbb{E}_{\mu\sim\eta}\mathbb{E}_{x\sim\mu^q}[h^*(x)h^*(x)^{\mathsf{T}}]\end{bmatrix} = \begin{bmatrix}\mathbf{G}_{\hat{h}\hat{h}} & \mathbf{G}_{\hat{h}h^*} \\ \mathbf{G}_{h^*\hat{h}} & \mathbf{G}_{h^*h^*}\end{bmatrix}.
$$

The first inequallity holds since the 1-Lipschitz continuity of the cross-entropy loss. According to the partial minimization of a convex quadratic form (Boyd et al., 2004), we have

$$
\inf_{(\boldsymbol{A})_k}\left\{[(\boldsymbol{A})_k^{\mathsf{T}} \; -(\boldsymbol{A})_k^{'\mathsf{T}}]\Lambda\begin{bmatrix}(\boldsymbol{A})_k \\ -(\boldsymbol{A})_k'\end{bmatrix}\right\} = (\boldsymbol{A})_k^{'\mathsf{T}}\Lambda_S(\hat{h}, h^*)(\boldsymbol{A})_k',
$$

where $\Lambda_S(\hat{h}, h^*) = \mathbf{G}_{h^*h^*} - \mathbf{G}_{h^*\hat{h}}(\mathbf{G}_{\hat{h}\hat{h}})^{\dagger}\mathbf{G}_{\hat{h}}h^*$ is the generalized Schur complement of the representation of h^* with respect to $\hat{h}$. Furthermore, according to the variational characterization of the singular value, we have

$$
\sum_{k=1}^{K}\sup_{(\boldsymbol{A})_k^{'\mathsf{T}}:\|(\boldsymbol{A})_k^{'\mathsf{T}}\|\le M_k}(\boldsymbol{A})_k^{'\mathsf{T}}\Lambda_S(\hat{h}, h^*)(\boldsymbol{A})_k' = \sum_{k=1}^{K}M_k\sigma_1(\Lambda_S(\hat{h}, h^*)) \le M\sigma_1(\Lambda_S(\hat{h}, h^*)),
$$

where σ_1 denotes the maximal singular value.

Moreover, we denote

$$
\tilde{\boldsymbol{A}}_t = \arg\min_{\boldsymbol{A}_t}\mathbb{E}_{(x_t^{(s)}, y_t^{(s)})\sim\mu_t^s}\ell(\boldsymbol{A}_t^{\mathsf{T}}\hat{h}(x_t^{(s)}), y_t^{(s)}), t\in[T]
$$

and

$$
\boldsymbol{A}_t^* = \arg\min_{\boldsymbol{A}_t}\mathbb{E}_{(x_t^{(s)}, y_t^{(s)})\sim\mu_t^s}\ell(\boldsymbol{A}_t^{\mathsf{T}}h^*(x_t^{(s)}), y_t^{(s)}), t\in[T].
$$

Thus we have the following derivation

$$
\begin{aligned}
& R_{train}(\hat{h}) - R_{train}(h^*) \\
=& \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^n}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^m}\mathcal{L}(\mathcal{LM}(D_t^{tr};\hat{h}), D_t^{val}) - \\
& \qquad\qquad \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{D_t^{val}\sim(\mu_t^q)^n}\mathbb{E}_{D_t^{tr}\sim(\mu_t^s)^m}\mathcal{L}(\mathcal{LM}(D_t^{tr};h^*), D_t^{val}) \\
=& \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{(x,y)\sim\mu_t^q}\left\{\ell(\tilde{\boldsymbol{A}}_t^{\mathsf{T}}\hat{h}(x), y) - \ell(\boldsymbol{A}_t^{*\mathsf{T}}h^*(x), y)\right\} \\
\geq& \frac{C}{T}\sum_{t=1}^{T} \mathbb{E}_{\mu_t\sim\eta}\mathbb{E}_{x\sim\mu_t^q}\left\{\|\tilde{\boldsymbol{A}}_t^{\mathsf{T}}\hat{h}(x) - \boldsymbol{A}_t^{*\mathsf{T}}h^*(x)\|_2^2\right\} \\
=& \frac{C}{T}\sum_{t=1}^{T} \inf_{\boldsymbol{A}_t} \mathbb{E}_{\mu_t\sim\eta}\mathbb{E}_{x\sim\mu_t^q}\left\{\|\tilde{\boldsymbol{A}}_t^{\mathsf{T}}\hat{h}(x) - \boldsymbol{A}_t^{*\mathsf{T}}h^*(x)\|_2^2\right\} \\
=& \frac{C}{T}\sum_{t=1}^{T} \inf_{\boldsymbol{A}_t} \mathbb{E}_{\mu_t\sim\eta}\mathbb{E}_{x\sim\mu_t^q}\left\{\sum_{k=1}^{K}|(\tilde{\boldsymbol{A}}_t)_k^{\mathsf{T}}\hat{h}(x) - (\boldsymbol{A}_t)_k^{*\mathsf{T}}h^*(x)|^2\right\} \\
=& \frac{C}{T}\sum_{t=1}^{T}\sum_{k=1}^{K}(\boldsymbol{A}_t)_k^{*\mathsf{T}}\Lambda_S(\hat{h}, h^*)(\boldsymbol{A}_t)_k^* = \sum_{k=1}^{K} tr(\Lambda_S(\hat{h}, h^*)(\mathbf{Q})_k^{\mathsf{T}}(\mathbf{Q})_k/T) = \sum_{k=1}^{K} tr(\Lambda_S(\hat{h}, h^*)(\mathbf{K})_k),
\end{aligned}
$$

where $(\mathbf{K})_k = (\mathbf{Q})_k^{\mathsf{T}}(\mathbf{Q})_k/T$. Since $\Lambda_S \succeq 0$, and $(\mathbf{K})_k \succeq 0$, through the Von-Neumann trace inequality, we have that

$$
\begin{aligned}
\sum_{k=1}^{K} tr(\Lambda_S(\hat{h}, h^*)(\mathbf{K})_k) &\geq \sum_{k=1}^{K}\sum_{i=1}^{d_L}\sigma_i(\Lambda_S(\hat{h}, h^*))\sigma_{d_L-i+1}((\mathbf{K})_{\mathbf{k}}) \geq \sum_{k=1}^{K}\sum_{i=1}^{d_L}\sigma_i(\Lambda_S(\hat{h}, h^*))\sigma_{d_L}((\mathbf{K})_k) \\
&= \sum_{k=1}^{K} tr(\Lambda_S(\hat{h}, h^*))\sigma_{d_L}((\mathbf{K})_k) \geq \sigma_1(\Lambda_S(\hat{h}, h^*))\sum_{k=1}^{K}\sigma_{d_L}((\mathbf{K})_k).
\end{aligned}
$$

Now, we can deduce that

$$
R_\eta(\hat{h}) - R_\eta(h^*) \leq \frac{M}{\sum_{k=1}^{K}\sigma_{d_L}((\mathbf{K})_k)}\left\{R_{train}(\hat{h}) - R_{train}(h^*)\right\}.
$$

Based on the above derivation, the conclusion can then obtained. ■

References

Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

Ferran Alet, Erica Weng, Tomás Lozano-Pérez, and Leslie Kaelbling. Neural relational inference with fast modular meta-learning. In *NeurIPS*, 2019.

Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended pac-bayes theory. In *ICML*, pages 205–214, 2018.

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. In *NeurIPS*, 2016.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *ICLR*, 2019.

Sungyong Baik, Seokil Hong, and Kyoung Mu Lee. Learning to forget for meta-learning. In *CVPR*, 2020.

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In *NeurIPS*, 2018.

Maria-Florina Balcan, Mikhail Khodak, and Ameet Talwalkar. Provable guarantees for gradient-based meta-learning. In *ICML*, pages 424–433, 2019.

Andrew R Barron and Thomas M Cover. Minimum complexity density estimation. *IEEE transactions on information theory*, 37(4):1034–1054, 1991.

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

Peter L Bartlett, Dylan J Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. volume 2017, 2017.

Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *ICML*, 2019.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. 1990.

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

Hakan Bilen, Basura Fernando, Efstratios Gavves, Andrea Vedaldi, and Stephen Gould. Dynamic image networks for action recognition. In *CVPR*, 2016.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

John Bronskill, Jonathan Gordon, James Requeima, Sebastian Nowozin, and Richard Turner. Tasknorm: Rethinking batch normalization for meta-learning. In *ICML*, pages 1153–1164, 2020.

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. In *NeurIPS*, volume 33, 2020.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPP*, 2014.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. In *CVPR*, 2019.

Giulia Denevi, Carlo Ciliberto, Dimitris Stamos, and Massimiliano Pontil. Learning to learn around a common mean. In *NeurIPS*, volume 31, 2018.

Giulia Denevi, Carlo Ciliberto, Riccardo Grazzi, and Massimiliano Pontil. Learning-to-learn stochastic gradient descent with biased regularization. In *ICML*, pages 1566–1575, 2019a.

Giulia Denevi, Dimitris Stamos, Carlo Ciliberto, and Massimiliano Pontil. Online-within-online meta-learning. In *NeurIPS*, volume 32, pages 1–11, 2019b.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *ACL*, 2019.

Simon S Du, Wei Hu, Sham M Kakade, Jason D Lee, and Qi Lei. Few-shot learning via learning the representation, provably. In *ICLR*, 2021.

Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, et al. Neural architecture search: A survey. *JMLR*, 20(55):1–21, 2019.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-q-learning. In *ICLR*, 2020.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135, 2017.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *ICML*, pages 1920–1930, 2019.

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

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *ICML*, pages 1568–1577, 2018.

Francisco Garcia and Philip S Thomas. A meta-mdp approach to exploration for lifelong reinforcement learning. In *NeurIPS*, volume 32, pages 5691–5700, 2019.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and SM Ali Eslami. Conditional neural processes. In *ICML*, pages 1704–1713. PMLR, 2018.

Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. In *NeurIPS*, 2018.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference On Learning Theory*, pages 297–299, 2018.

Santiago Gonzalez and Risto Miikkulainen. Improved training speed, accuracy, and data utilization through loss function optimization. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

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

Henry Gouk, Hospedales Subhransu, and Massimiliano Pontil. Distance-based regularisation of deep networks for fine-tuning. In *ICLR*, 2021.

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

Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

Rein Houthooft, Richard Y Chen, Phillip Isola, Bradly C Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. In *NeurIPS*, 2018.

Chen Huang, Shuangfei Zhai, Walter Talbott, Miguel Bautista Martin, Shih-Yu Sun, Carlos Guestrin, and Josh Susskind. Addressing the loss-metric mismatch with adaptive loss alignment. In *ICML*, pages 2891–2900, 2019.

Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

Madan Kanika, Ke Nan Rosemary, Goyal Anirudh, Schölkopf Bernhard, and Bengio Yoshua. Meta attention networks: Meta-learning attention to modulate information between recurrent independent mechanisms. In *ICLR*, 2021.

Mikhail Khodak, Maria-Florina F Balcan, and Ameet S Talwalkar. Adaptive gradient-based meta-learning methods. *NeurIPS*, 32:5917–5928, 2019.

Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning active learning from data. In *NeurIPS*, 2017.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

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

Michel Ledoux and Michel Talagrand. *Probability in Banach Spaces: isoperimetry and processes*. Springer Science & Business Media, 2013.

Hae Beom Lee, Taewook Nam, Eunho Yang, and Sung Ju Hwang. Meta dropout: Learning to perturb latent features for generalization. In *ICLR*, 2020.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, pages 2927–2936, 2018.

Christiane Lemke, Marcin Budka, and Bogdan Gabrys. Metalearning: a survey of trends and technologies. *Artificial intelligence review*, 44(1):117–130, 2015.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, pages 5542–5550, 2017a.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018a.

Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. In *ICLR*, 2019a.

Xuhong Li, Grandvalet Yves, and Davoine Franck. Explicit inductive bias for transfer learning with convolutional networks. In *ICML*, 2018b.

Yiying Li, Yongxin Yang, Wei Zhou, and Timothy Hospedales. Feature-critic networks for heterogeneous domain generalization. In *ICML*, 2019b.

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

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *ICLR*, 2019.

Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Learning to generate noise for robustness against multiple perturbations. *arXiv preprint arXiv:2006.12135*, 2020.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Donald B Maudsley. A theory of meta-learning and principles of facilitation: An organismic perspective. 1980.

Andreas Maurer. A chain rule for the expected suprema of gaussian processes. *Theoretical Computer Science*, 650:109–122, 2016.

Andreas Maurer and Tommi Jaakkola. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6(6), 2005.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Stefan Munder and Dariu M Gavrila. An experimental study on pedestrian classification. *IEEE transactions on pattern analysis and machine intelligence*, 28(11):1863–1868, 2006.

Vaishnavh Nagarajan and J Zico Kolter. Generalization in deep networks: The role of distance from initialization. *arXiv preprint arXiv:1901.01672*, 2019.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *ICLR*, 2019.

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

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008.

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

Eunbyung Park and Junier B Oliva. Meta-curvature. In *NeurIPS*, 2019.

Anastasia Pentina and Christoph Lampert. A pac-bayesian bound for lifelong learning. In *ICML*, pages 991–999, 2014.

Anastasia Pentina and Christoph H Lampert. Lifelong learning with non-iid tasks. In *NeurIPS*, volume 28, pages 1540–1548, 2015.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

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

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, pages 506–516, 2017.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2019.

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.

Nataniel Ruiz, Samuel Schulter, and Manmohan Chandraker. Learning to simulate. In *ICLR*, 2018.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

Jeong Un Ryu, JaeWoong Shin, Hae Beom Lee, and Sung Ju Hwang. Metaperturb: Transferable regularizer for heterogeneous tasks and architectures. *NeurIPS*, 33, 2020.

Jürgen Schmidhuber. *Evolutionary Principles in Self-referential Learning: On Learning how to Learn: the Meta-meta-meta...-hook*. 1987.

Gideon Schwarz et al. Estimating the dimension of a model. *Annals of statistics*, 6(2): 461–464, 1978.

Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *ICLR*, 2018.

John Shawe-Taylor, Peter L Bartlett, Robert C Williamson, and Martin Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44(5):1926–1940, 1998.

Jun Shu, Zongben Xu, and Deyu Meng. Small sample learning in big data era. *arXiv preprint arXiv:1808.04572*, 2018.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019.

Jun Shu, Qian Zhao, Keyu Chen, Zongben Xu, and Deyu Meng. Learning adaptive loss for robust learning with noisy labels. *arXiv preprint arXiv:2002.06482*, 2020a.

Jun Shu, Qian Zhao, Zengben Xu, and Deyu Meng. Meta transition adaptation for robust deep learning with noisy labels. *arXiv preprint arXiv:2006.05697*, 2020b.

Jun Shu, Yanwen Zhu, Qian Zhao, Deyu Meng, and Zongben Xu. Meta-lr-schedule-net: Learned lr schedules that scale and generalize. *arXiv preprint arXiv:2007.14546*, 2020c.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, pages 4080–4090, 2017.

Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. In *ICLR*, 2020.

Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

Flood Sung, Li Zhang, Tao Xiang, Timothy Hospedales, and Yongxin Yang. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv preprint arXiv:1706.09529*, 2017.

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

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. pages 3–17. Springer, 1998.

Nilesh Tripuraneni, Michael Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. In *NeurIPS*, volume 33, 2020.

Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

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

Vladimir N Vapnik. An overview of statistical learning theory. *IEEE transactions on neural networks*, 10(5):988–999, 1999.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Janarthanan Rajendran, Richard L Lewis, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. In *NeurIPS*, 2019.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, pages 3637–3645, 2016.

Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *NeurIPS*, 2019.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. *Technical report*, 2011.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge University Press, 2019.

Renzhen Wang, Kaiqin Hu, Yanwen Zhu, Jun Shu, Qian Zhao, and Deyu Meng. Meta feature modulator for long-tailed recognition. *arXiv preprint arXiv:2008.03428*, 2020a.

Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. Structured prediction for conditional meta-learning. In *NeurIPS*, 2020b.

Tongzhou Wang, Jun-Yan Zhu, Antonio Torralba, and Alexei A Efros. Dataset distillation. *arXiv preprint arXiv:1811.10959*, 2018.

Xin Wang, Fisher Yu, Ruth Wang, Trevor Darrell, and Joseph E Gonzalez. Tafe-net: Task-aware feature embeddings for low shot learning. In *CVPR*, 2019.

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

Lewis B Ward. Reminiscence and rote learning. *Psychological Monographs*, 49(4):i, 1937.

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *ICML*, 2017.

Yichen Wu, Jun Shu, Qi Xie, Qian Zhao, and Deyu Meng. Learning to purify noisy labels via meta soft label corrector. In *AAAI*, 2021.

Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore with meta-policy gradient. In *ICML*, 2018.

Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In *ICML*, 2019.

Quanming Yao, Mengshuo Wang, Yuqiang Chen, Wenyuan Dai, Yu-Feng Li, Wei-Wei Tu, Qiang Yang, and Yang Yu. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

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

Du Yingjun, Zhen Xiantong, Shao Ling, and Snoek Cees G. M. Metanorm: Learning to normalize few-shot batches across domains. In *ICLR*, 2021.

Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. Meta label correction for noisy label learning. In *AAAI*, 2021.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017.