Towards Enabling Meta-Learning from Target Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Meta-learning can extract an inductive bias from previous learning experience and assist the training processes of new tasks. It is often realized through optimizing a meta-model with the evaluation loss of a series of task-specific solvers. Most existing algorithms sample non-overlapping *support* sets and *query* sets to train and evaluate the solvers respectively due to simplicity ($\mathcal{S}/\mathcal{Q}$ protocol). However, another evaluation method that assesses the discrepancy between the solver and a target model is short of research ($\mathcal{S}/\mathcal{T}$ protocol). $\mathcal{S}/\mathcal{T}$ protocol has unique advantages such as offering more informative supervision, but it is computationally expensive. This paper looks into this special evaluation method and takes a step towards putting it into practice. We find that with a small ratio of tasks armed with target models, classic meta-learning algorithms can be improved a lot without consuming many resources. Furthermore, we empirically verify the effectiveness of $\mathcal{S}/\mathcal{T}$ protocol in a typical application of meta-learning, *i.e.*, few-shot learning. In detail, after constructing target models by fine-tuning the pre-trained network on those hard tasks, we match the task-specific solvers to target models via knowledge distillation. Experiments demonstrate the superiority of our proposal.

1 Introduction

Human beings are able to sum up experience from previous tasks and reuse it in new tasks, which is also a vital characteristic of ideal machine learning systems. Meta-learning means improving performance measures over a family of tasks by their training experience [24]. It has been researched in various fields such as image classification [34, 18], action recognition [2, 13], and reinforcement learning [7, 16]. By reusing transferable meta-knowledge extracted from previous tasks, we can learn new tasks with a higher efficiency or a shortage of data.

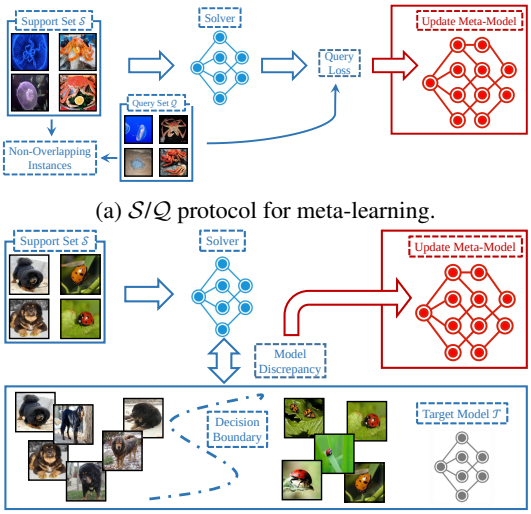
A typical meta-learning algorithm can be decomposed into two iterative phases. In the first phase, we train a solver of a task on its training set with assistance of meta-model. In the second phase, we optimize the solver's performance to update meta-model. One key factor in this procedure is the way to evaluate the solver since the evaluation result plays the role of supervision signal for meta-model. Early meta-learning algorithms [21, 25] directly use the solver's training loss as its performance metric, and optimize this metric over a distribution of tasks. Obviously, inner-task over-fitting may happen during the training of task-specific solvers, resulting in an inaccurate supervision signal for the meta-model. This drawback is even more amplified in applications where the training set of each task is limited such as few-shot learning and noisy learning.

Intuitively, assessment of solvers should be independent of their training sets. This principle draws forth two important meta-learning algorithms in 2016 [26, 30], which respectively export solver evaluation from the perspective of "data" and "model". In this paper, we call these two methodologies $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol. In $\mathcal{S}/\mathcal{Q}$ protocol, $\mathcal{S}$ means *support* set and $\mathcal{Q}$ means *query* set. They

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

contain non-overlapping instances sampled from a same distribution. By training the solver on $\mathcal{S}$ and evaluating it on $\mathcal{Q}$, we are able to obtain an approximate generalization error of the solver and eventually provide the meta-model with a reliable supervision signal. Another choice is to compare the trained solver with an ideal target model $\mathcal{T}$. Assuming that $\mathcal{T}$ works well on a task, we can minimize the discrepancy between the trained solver and $\mathcal{T}$ to pull the solver closer to $\mathcal{T}$. Here $\mathcal{T}$ can be Bayesian optimal solution to a task or a model trained on a sufficiently informative dataset. Figure 1 gives an illustration of both $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol.

Although appeared in the same year, $\mathcal{S}/\mathcal{Q}$ protocol is more widely accepted by meta-learning society [5, 9, 15, 36, 39, 11] while the research about how to leverage target models remains immature. The main reason is the simplicity of $\mathcal{S}/\mathcal{Q}$ and the computational hardness of $\mathcal{S}/\mathcal{T}$. However, $\mathcal{S}/\mathcal{T}$ protocol has some unique advantages. Firstly, it does not depend on possibly biased and noisy *query* sets. Secondly, by viewing *support* sets and their corresponding target models as (feature, label) samples, meta-learning is reduced to supervised learning and we can transfer insights from supervised learning to improve meta-learning [3]. Thirdly, we can treat the target model as a teacher and incorporate a teacher-student framework like knowledge distillation [8] and curriculum learning [1]. Thus, it is necessary to enable $\mathcal{S}/\mathcal{T}$ in meta-learning.



(a) $\mathcal{S}/\mathcal{Q}$ protocol for meta-learning.

(b) $\mathcal{S}/\mathcal{T}$ protocol for meta-learning.

Figure 1: Comparison between $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol. (a) In $\mathcal{S}/\mathcal{Q}$ protocol, each task contains a *support* set $\mathcal{S}$ and a *query* set $\mathcal{Q}$. We train a solver on $\mathcal{S}$ and evaluate it on $\mathcal{Q}$, and *query* loss is used to optimize meta-model. (b) In $\mathcal{S}/\mathcal{T}$ protocol, each task contains a *support* set $\mathcal{S}$ and a target model $\mathcal{T}$. After training a solver on $\mathcal{S}$, we directly minimize the discrepancy between it and $\mathcal{T}$.

This paper looks into $\mathcal{S}/\mathcal{T}$ protocol and takes a step towards enabling meta-learning from target models. We mainly answer two questions: (1) If we already have access to target models, how to learn from them? What are the benefits of learning from them? (2) In a real-world application, how to obtain target models efficiently and make $\mathcal{S}/\mathcal{T}$ protocol computationally tractable? For the first question, we propose to match the task-specific solver to the target model in output space. Learning from target models brings us more robust solvers. For the second question, we focus on a typical application scenario of meta-learning, *i.e.*, few-shot learning. We construct target models by fine-tuning the globally pre-trained network on those hard tasks to maintain efficiency.

2 Related Work

Meta-Learning. Meta-learning aims at extracting task-level experience (so-called meta-knowledge) from seen tasks, while generalizing the learned meta-knowledge to unseen tasks efficiently. Researchers have instantiated meta-knowledge as model initialization [4, 27], embedding network [23, 14, 10, 22, 5], external memory [21, 6, 38], optimization strategy [17, 12, 20], and data augmentation strategy [15]. Despite their diversity in meta-knowledge, most existing models are trained under $\mathcal{S}/\mathcal{Q}$ protocol, and rely on a randomly sampled and possibly biased *query* set. Actually, most algorithms are protocol-agnostic, and both $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol can be applied to them. Thus, our work on $\mathcal{S}/\mathcal{T}$ is general and meaningful because of its wide application field.

Learning from Target Models. The idea of learning from target models in meta-learning is first proposed by [30]. In [30], the authors constructed a model regression network that explicitly regresses between small-sample classifiers and target models in parameter space. Here both solvers and target models are limited to low-dimensional linear classifier, making it feasible to regress between them. From our perspective, matching two models' parameters is not practical when the dimension of parameters is too high. Thus, we match two models in output space in this paper. Similarly, there

are other papers focusing on meta-learning from target models [37, 29]. However, they all assume that every single task has a target model, increasing both space and time complexity of $\mathcal{S}/\mathcal{T}$ protocol. To summarize, we claim that one key point in putting $\mathcal{S}/\mathcal{T}$ protocol into practice is reducing the requirement for target models. In this paper, we focus on those hard tasks, and find that by learning from a small ratio of informative target models, classic meta-learning algorithms can be improved.

3 Preliminary

Meta-learning extracts high-level knowledge by a meta-model from *meta-training* tasks sampled from a task distribution $p(\boldsymbol{\tau})$ and reusing the learned meta-model on new tasks belonging to the same distribution. Each task $\boldsymbol{\tau}$ has a task-specific *support* set $\mathcal{S} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{|\mathcal{S}|}$, and we can train on $\mathcal{S}$ a solver $g : \mathbb{X} \to \mathbb{Y}$ parameterized by $\boldsymbol{\gamma}_g$. Without loss of generality, a meta-model can be defined as $f : \mathbb{S} \to \mathbb{G}$ parameterized by $\boldsymbol{\theta}_f$ that receives a *support* set as input and outputs a solver. Here $\mathbb{S}$ is the space of *support* sets and $\mathbb{G}$ is the space of solvers. In other words, f encodes the training process of g on $\mathcal{S}$ under the supervision of meta-knowledge $\boldsymbol{\theta}_f$. Taking two well-known meta-learning algorithms, MAML [4] and ProtoNet [23], as examples, we have the following concrete forms of f:

- MAML meta-learns a model initialization $\boldsymbol{\theta}_f$ and fine-tunes it on each $\mathcal{S}$ with one gradient descent step to obtain a task-specific solver g. It can be written as Equ (1). η is step size and $\ell : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}_+$ is some loss function.

$$f(\mathcal{S}; \boldsymbol{\theta}_f) = g\left(\cdot\,; \boldsymbol{\theta}_f - \eta \, \nabla_{\boldsymbol{\gamma}} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}} \ell\left(g(\mathbf{x}_i; \boldsymbol{\gamma}), \mathbf{y}_i\right) \Bigg|_{\boldsymbol{\gamma} = \boldsymbol{\theta}_f} \right) \tag{1}$$

- ProtoNet meta-learns an embedding function $\phi_{\boldsymbol{\theta}_f}$ parameterized by $\boldsymbol{\theta}_f$ and generates a lazy solver which classifies an instance to the category of its nearest class center. Here g is implicitly parameterized by both $\boldsymbol{\theta}_f$ and embedded *support* instances.

$$f(\mathcal{S}; \boldsymbol{\theta}_f) = g\left(\cdot\,; \boldsymbol{\theta}_f, \{\phi_{\boldsymbol{\theta}_f}(\mathbf{x}_i) | (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}\} \right) \tag{2}$$

$\mathcal{S}/\mathcal{Q}$ **Protocol.** How to evaluate the solver g trained on $\mathcal{S}$? The answer to this question differs conventional $\mathcal{S}/\mathcal{Q}$ protocol [26] from $\mathcal{S}/\mathcal{T}$ protocol. In $\mathcal{S}/\mathcal{Q}$ protocol, we sample another *query* set $\mathcal{Q} = \{\mathbf{x}_j, \mathbf{y}_j\}_{j=1}^{|\mathcal{Q}|}$ apart from $\mathcal{S}$ for each task. Instances in $\mathcal{S}$ and $\mathcal{Q}$ are *i.i.d.* distributed and have a same label set, and we evaluate g by its loss on $\mathcal{Q}$. Since $\mathcal{S}$ and $\mathcal{Q}$ have non-overlapping instances, the training signal on $\mathcal{Q}$ is more reliable. $\mathcal{S}/\mathcal{Q}$ protocol can be formulated as Equ (3). Here $\mathcal{D}^{\text{tr}}$ is the *meta-training* set and we can sample *meta-training* tasks $\boldsymbol{\tau}^{\text{tr}}$ from it. We use $\mathcal{D}^{\text{ts}}$ to represent *meta-testing* set hereinafter.

$$\min_f \sum_{\boldsymbol{\tau}^{\text{tr}} = (\mathcal{S}^{\text{tr}}, \mathcal{Q}^{\text{tr}}) \in \mathcal{D}^{\text{tr}}} \sum_{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{Q}^{\text{tr}}} \ell(f(\mathcal{S}^{\text{tr}})(\mathbf{x}_j), \mathbf{y}_j) \tag{3}$$

$\mathcal{S}/\mathcal{T}$ **Protocol.** Any sampled *query* set $\mathcal{Q}$ can be biased and noisy, which may cause an inaccurate evaluation of the solver. An alternative is directly matching the task-specific solver $g = f(\mathcal{S})$ to a target model $\mathcal{T}$ that works well on the corresponding task. By computing the distance from the solver to target model, we obtain a more robust training signal to update meta-model. By replacing the solver evaluation part in Equ (3), we have the following $\mathcal{S}/\mathcal{T}$ protocol Equ (4). Here $\mathcal{L} : \mathbb{G} \times \mathbb{G} \to \mathbb{R}_+$ is some loss function to measure the discrepancy between g and target model $\mathcal{T}$.

$$\min_f \sum_{\boldsymbol{\tau}^{\text{tr}} = (\mathcal{S}^{\text{tr}}, \mathcal{T}^{\text{tr}}) \in \mathcal{D}^{\text{tr}}} \mathcal{L}(f(\mathcal{S}^{\text{tr}}), \mathcal{T}^{\text{tr}}) \tag{4}$$

4 Effect of Target Model

We have introduced some basic concepts in meta-learning, and formulate $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol in last section. In this section, we assume that target models are available, and study how to utilize them to assist meta-learning. Firstly, we propose a model matching framework based on output comparison. Secondly, we verify the effectiveness of our proposal in a synthetic experiment. Moreover, we try to decrease the ratio of tasks that have target models, and show that it is possible to reduce the resource consumption in $\mathcal{S}/\mathcal{T}$ protocol.

4.1 Model Matching

In $\mathcal{S}/\mathcal{T}$ protocol, one key point is how to match the solver g to its target model $\mathcal{T}$. In other words, we need to specify the concrete formulation of $\mathcal{L}(g, \mathcal{T})$. Generally, methods to match g to $\mathcal{T}$ can be classified into two categories. Firstly, we can directly match two models' parameters or use another model to regress between two models' parameters [30]. For example, let $\boldsymbol{\gamma}_g$ and $\boldsymbol{\gamma}_{\mathcal{T}}$ be the parameters of g and $\mathcal{T}$, we can set $\mathcal{L}(g, \mathcal{T}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}^{\text{tr}}} \ell(g(\mathbf{x}_i), \mathbf{y}_i) + \lambda \|\boldsymbol{\gamma}_g - \boldsymbol{\gamma}_{\mathcal{T}}\|_2^2$. Here λ is a balancing hyper-parameter. This method may work well for low-dimensional parameters, but is not suitable for complex models like deep neural networks. A better alternative is to match two models in their output space, *i.e.*, $\mathcal{L}(g, \mathcal{T}) = \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S}^{\text{tr}}} [(1-\lambda)\ell(g(\mathbf{x}_i), \mathbf{y}_i) + \lambda \mathbf{D}(\mathcal{T}(\mathbf{x}_i), g(\mathbf{x}_i))]$. Here $\mathbf{D}(\cdot, \cdot)$ is a function that measures the discrepancy between $\mathcal{T}(\mathbf{x}_i)$ and $g(\mathbf{x}_i)$. If we instantiate $\mathbf{D}(\cdot, \cdot)$ as KL divergence $\mathbf{KL}(\cdot || \cdot)$ for classification problem, the aforementioned loss function is equivalent to that of knowledge distillation. Figure 2 is an illustration of approaches to matching a solver to a target model.

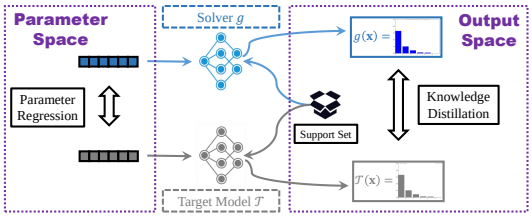


Figure 2: Two approaches to matching g to target model $\mathcal{T}$. Left: matching them in parameter space. Right: matching them in output space.

4.2 Empirical Study: Sinusoid Regression

In this part, we assume that target models are available, and evaluate the effectiveness of our proposed matching approach. We construct a synthetic regression problem, and try to answer the following questions: (1) Can $\mathcal{S}/\mathcal{T}$ protocol outperform $\mathcal{S}/\mathcal{Q}$ protocol when target models are available? (2) Is it possible to improve meta-learning with only a few target models?

Setting. Consider regression tasks $\mathcal{T}(x) = a \sin(bx - c)$ where a, b, and c are uniformly sampled from $[0.1, 5]$, $[0.5, 2]$, and $[0.5, 2\pi]$ respectively. For each task, we generate 10 *support* instances by uniformly sampling x in range $[-5, 5]$. For $\mathcal{S}/\mathcal{Q}$ protocol, we additionally sample 30 *query* instances for each task. We then set $y = \mathcal{T}(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.5)$ is a Gaussian noise. 10000 tasks are used for both *meta-training* and *meta-testing*. 500 tasks are used for *meta-validation*.

Algorithms. We consider two classic meta-learning algorithms, MAML [4] and ProtoNet [23]. MAML can be directly applied in a regression task, but ProtoNet is originally designed for classification. In this part, we modify ProtoNet slightly to fit regression problem. In detail, we try to meta-learn an embedding function $\phi : \mathbb{R} \to \mathbb{R}^{100}$, with assistance of which the similarity-based regression model $g(\cdot\ ; \{\phi(x_i) | (x_i, y_i) \in \mathcal{S}\})$ works well across all tasks. Here for any instance (x, y), $\hat{y} = g(x) = \sum_{(x_i, y_i) \in \mathcal{S}} w_i y_i$ and $w_i = \frac{\exp\{\langle \phi(x_i), \phi(x) \rangle\}}{\sum \exp\{\langle \phi(x_i), \phi(x) \rangle\}}$. A same embedding network is used in two algorithms. We train MAML and ProtoNet under $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol. When using $\mathcal{S}/\mathcal{Q}$ protocol, we minimize MSE loss on 30 *query* instances to optimize ϕ. For $\mathcal{S}/\mathcal{T}$ protocol, we match the solver to the target model in output space, and set $\mathbf{D}(\mathcal{T}(x_i), g(x_i)) = \|\mathcal{T}(x_i) - g(x_i)\|_2^2$. Thus, the loss function under $\mathcal{S}/\mathcal{T}$ protocol is $\mathcal{L}(g, \mathcal{T}) = \sum_{(x_i, y_i) \in \mathcal{S}^{\text{tr}}} \left[(1-\lambda)\|g(x_i) - y_i\|_2^2 + \lambda \|g(x_i) - \mathcal{T}(x_i)\|_2^2\right]$. λ is a hyper-parameter. More implementation details can be found in the supplement.

Superiority of $\mathcal{S}/\mathcal{T}$ Protocol. Table 1 shows the MSE of four models on *meta-testing* tasks. We can see that models trained under $\mathcal{S}/\mathcal{T}$ protocol consistently outperform models trained under $\mathcal{S}/\mathcal{Q}$ protocol. In Figure 3, we visualize a randomly chosen *meta-testing* task. Different colors are used for different meta-learning algorithms, and dotted lines and dashed lines are used for $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol respectively. We can see that models trained under $\mathcal{S}/\mathcal{T}$ protocol fit the target sinusoid curve better. It is meaningful to discuss why target models improve meta-learning algorithms. In this empirical study, distillation from target models can be interpreted as label denoising. In detail, we can prove[1] that meta-learning loss under $\mathcal{S}/\mathcal{T}$ protocol $(1-\lambda)\|g(x) - y\|_2^2 + \lambda \|g(x) - \mathcal{T}(x)\|_2^2$ is equivalent to $\|g(x) - (y - \lambda\epsilon)\|_2^2$, which is the standard MSE loss between the output of solver g and

[1] We leave the proof to the supplement.

cleaner label $y - \lambda\epsilon$ (raw label y equals to $\mathcal{T}(x) + \epsilon$). Therefore, the larger λ is, the cleaner training labels are. Table 2 is an ablation study on hyper-parameter λ. As expected, both algorithms trained under $\mathcal{S}/\mathcal{T}$ achieve better performance with larger λ. These results demonstrate the superiority of $\mathcal{S}/\mathcal{T}$ protocol when target models are available.

Reducing the Requirement for Target Models. Despite the satisfying results in the empirical study, it does not mean that we can apply $\mathcal{S}/\mathcal{T}$ protocol in real-world applications and necessarily obtain higher performance. Up till now, we have assumed that every single *meta-training* task has a target model. This assumption is too strong from two aspects. Firstly, we usually don't have ready-made target models, and constructing target models is not trivial. Secondly, even though we have designed a method to construct target models, it will cost too much time to construct a target model for every single *meta-training* task. Existing researches that focus on meta-learning from target models often bypass this dilemma by restricting the complexity of solvers and target models [30] or building one global target model [37]. In this paper, we study a more generic methodology - reducing the number of required target models. If we randomly choose a small subset of *meta-training* tasks, and only provide these tasks' target models, how will the model performance change? To answer the question, we first randomly sample subsets of tasks that have target models, and abandon target models for other tasks. In this case, the meta-learning loss of tasks without target models degenerates to $\mathcal{S}/\mathcal{Q}$ loss. By ranging the size of this subset, we can plot the performance curve of MAML and ProtoNet in Figure 4. Then, we heuristically select the hardest tasks from all *meta-training* tasks and only deploy target models for these tasks. In this regression problem, a sinusoid curve is defined as $a\sin(bx - c)$, and larger a or smaller b induce steeper curves. We simply consider these steep curves as hard tasks, and sort the hardness of all *meta-training* tasks according to $a - b$. Another two performance curves using this heuristic are also plotted in Figure 4. We can see that when using this naive heuristic, we can obtain an evident performance gain with only $500(5\%)$ target models. This finding inspires us to analyse the hardness of tasks in meta-learning, and confirms the possibility of learning from a few target models.

Table 1: Average test MSE of two meta-learning algorithms. Models trained under $\mathcal{S}/\mathcal{T}$ protocol outperform those trained under $\mathcal{S}/\mathcal{Q}$ protocol.

Method	MAML		ProtoNet	
	$\mathcal{S}/\mathcal{Q}$	$\mathcal{S}/\mathcal{T}$	$\mathcal{S}/\mathcal{Q}$	$\mathcal{S}/\mathcal{T}$
MSE on $\mathcal{D}^{\text{ts}}$	4.933	**3.621**	4.706	**3.332**

Table 2: Average test MSE of two meta-learning algorithms with different λ values. Larger λ offers cleaner labels, resulting in better models.

λ	1	0.8	0.5	0.2
MSE: MAML($\mathcal{S}/\mathcal{T}$)	**3.220**	3.419	3.621	3.833
MSE: ProtoNet($\mathcal{S}/\mathcal{T}$)	**3.137**	3.304	3.332	3.550

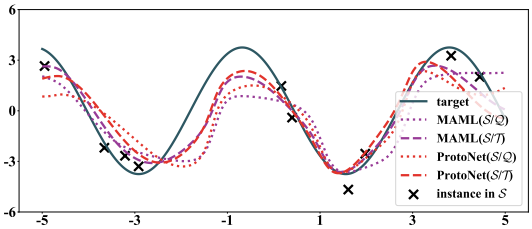


Figure 3: Visualization of a randomly sampled *meta-testing* task. Dotted lines are used for $\mathcal{S}/\mathcal{Q}$ protocol while dashed lines are used for $\mathcal{S}/\mathcal{T}$ protocol. We can see that models trained under $\mathcal{S}/\mathcal{T}$ protocol can fit the target sinusoid curve better.

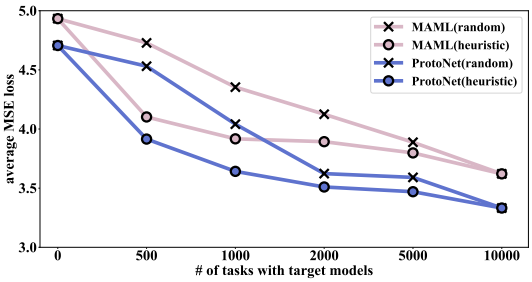


Figure 4: Change of MSE loss over number of *meta-training* tasks that have target models. By selecting hard tasks heuristically, we are able to obtain an evident performance gain with a small number of target models.

5 Application Case: Few-Shot Learning

Few-shot learning is a typical application of meta-learning. It aims at recognizing new categories with only a few labelled instances available. In few-shot learning, we have two datasets that contain non-overlapping classes, *i.e.*, $\mathcal{D}^{\text{tr}}$ and $\mathcal{D}^{\text{ts}}$. $\mathcal{D}^{\text{tr}}$ is composed of seen classes while $\mathcal{D}^{\text{ts}}$ contains unseen

classes. We can sample N-way K-shot[2] *meta-training* tasks from $\mathcal{D}^{\text{tr}}$ to train the meta-model, and expect the trained meta-model will also work well on $\mathcal{D}^{\text{ts}}$.

5.1 Task Hardness

Following the idea of constructing target models only for hard tasks, we firstly investigate which tasks are hard in few-shot learning. Inspired by [40], we consider the relationship between classes as a key factor that determines the hardness of a classification task. Assuming that there are C^{tr} classes in $\mathcal{D}^{\text{tr}}$, we first compute a similarity matrix $\mathbf{F} \in \mathbb{R}^{C^{\text{tr}} \times C^{\text{tr}}}$ whose element $\mathbf{F}_{uv}$ equals to the similarity between the u-th class centre and the v-th class center. In few-shot learning society, pre-training the backbone network on $\mathcal{D}^{\text{tr}}$ has become a common practice [28, 33, 34], and we can compute these class centres based on the pre-trained model ϕ^{pt} as Equ (5) and Equ (6). In Equ (5), K_u is the number of instances of class u in $\mathcal{D}^{\text{tr}}$, and with a bit abuse of notation, we use $\mathbf{y}_i = u$ to select instances belonging to the u-th class.

$$\mathbf{c}_u = \frac{1}{K_u} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}^{\text{tr}} \wedge \mathbf{y}_i = u} \phi^{\text{pt}}(\mathbf{x}_i), \quad u \in [C^{\text{tr}}] \tag{5}$$

$$\mathbf{F}_{uv} = \frac{\mathbf{c}_u \cdot \mathbf{c}_v}{\|\mathbf{c}_u\| \cdot \|\mathbf{c}_v\|}, \quad u, v \in [C^{\text{tr}}] \tag{6}$$

Table 3: Average accuracy on auxiliary dataset $\mathcal{D}^{\text{au}}$. Fine-tuned target models outperform a single pre-trained target model on randomly sampled tasks.

Target Model	pre-train	fine-tune
Accuracy on $\mathcal{D}^{\text{au}}$	98.24	**99.37**

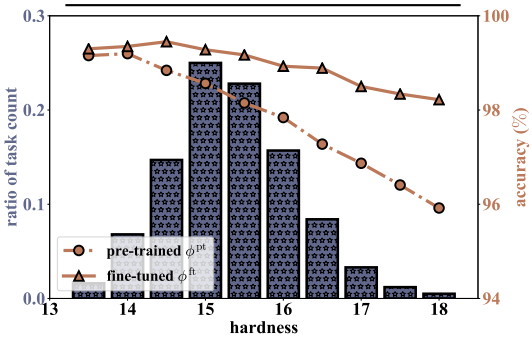


Figure 5: Grouping of 1000 randomly sampled tasks according to their hardness. Both ϕ^{pt} and ϕ^{ft} achieve lower accuracy on harder tasks, verifying the rationality of our proposed hardness metric. Fine-tuned target models obtain a remarkable performance gain on hard tasks.

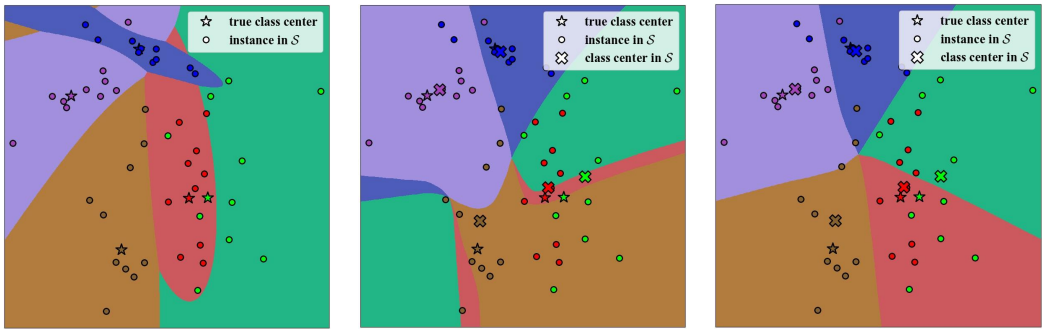
With similarity matrix $\mathbf{F}$, we can take out the sub-similarity matrix of task $\boldsymbol{\tau}$ by slicing the rows and columns corresponding to classes contained in $\boldsymbol{\tau}$. The hardness of task $\boldsymbol{\tau}$ is defined as the sum of its sub-similarity matrix. The more similar classes in $\boldsymbol{\tau}$ are, the more difficult to differ them from each other. The hardness of every *meta-training* task can be evaluated with similarity matrix $\mathbf{F}$, and we compute $\mathbf{F}$ only once.

5.2 Target Model Construction

As mentioned in last part, pre-training the backbone network on seen classes is a widely used technology in few-shot learning. The pre-trained network ϕ^{pt} is optimized using cross-entropy loss on the whole *meta-training* set, and can classify all classes in $\mathcal{D}^{\text{tr}}$. Since there are C^{tr} classes in $\mathcal{D}^{\text{tr}}$, the output of ϕ^{tr} is a C^{tr}-dimensional vector. Given a specific N-way task $\boldsymbol{\tau}$, a naive approach to obtain a target model is taking out N corresponding dimensions of the pre-trained model's output. However, using a single pre-trained model to assist the meta-learning of all tasks is sub-optimal. We claim that fine-tuning the pre-trained model on the subset of $\mathcal{D}^{\text{tr}}$ that contains classes in $\boldsymbol{\tau}$ can give us a better target model for $\boldsymbol{\tau}$.

Evaluation on Auxiliary Dataset. To verify the rationality of the heuristic task hardness metric and the effectiveness of the fine-tuning approach, we need another auxiliary dataset $\mathcal{D}^{\text{au}}$. $\mathcal{D}^{\text{au}}$ contains same classes as $\mathcal{D}^{\text{tr}}$, and we can evaluate the accuracy of constructed target models (trained on $\mathcal{D}^{\text{tr}}$) on $\mathcal{D}^{\text{au}}$. We conduct an experiment on *mini*ImageNet [26] to check whether fine-tuned target models are better than pre-trained target models. Firstly, we pre-train a ResNet-12 with a linear layer on the *meta-training* split of *mini*ImageNet. After that, we randomly sample 1000 5-way tasks from $\mathcal{D}^{\text{tr}}$, and fine-tune the pre-trained backbone to obtain 1000 target models. For each task $\boldsymbol{\tau}$, we take out all instances in $\mathcal{D}^{\text{au}}$ that belong to the classes in $\boldsymbol{\tau}$ to evaluate ϕ^{pt} and $\phi^{\text{ft}}_{\boldsymbol{\tau}}$. Table 3 shows the average accuracy on auxiliary dataset $\mathcal{D}^{\text{au}}$. We can see that fine-tuned target models achieve higher accuracy because they are task-specific, but the performance gain is marginal. The pre-trained model

[2] An N-way K-shot task is a classification task with N classes and K instances in each class.



(a) Bayesian optimal classifier. (b) ProtoNet trained under $\mathcal{S}/\mathcal{Q}$. (c) ProtoNet trained under $\mathcal{S}/\mathcal{T}$.

Figure 6: Decision boundaries of three different models in raw 2-d space. Different point colors represent different classes, and different background colors represent different classification regions. 5% tasks have target models. A 5-way 10-shot task is visualized. (a) Bayesian optimal model constructed with parameters $\{\boldsymbol{\mu}_n\}_{n=1}^N$ and $\{\boldsymbol{\Sigma}_n\}_{n=1}^N$. Although having the lowest misclassification error in expectation, it is not robust to noises since the decision boundary is very steep. (b) ProtoNet trained under $\mathcal{S}/\mathcal{Q}$ protocol. Decision boundary is still not regular. (c) ProtoNet trained under $\mathcal{S}/\mathcal{T}$ protocol. Decision boundary is very smooth due to the regularization effect of knowledge distillation. Models trained under $\mathcal{S}/\mathcal{T}$ protocol are more robust to noisy of biased instances.

already works well enough on these seen classes. This means it is not cost-effective to fine-tune a target model for every single *meta-training* task. In Figure 5, we divide these 1000 tasks into 10 bins according to their hardness. In each bin, we compute the average accuracy of ϕ^{pt} and ϕ^{ft}. Now we can draw two conclusions. Firstly, both ϕ^{pt} and ϕ^{ft} achieve lower accuracy on harder tasks, and this verifies the rationality of our proposed hardness metric. Secondly, the performance gain of fine-tuned target models are most remarkable on hard tasks, and this means fine-tuning target models for hard tasks can simultaneously save computing resources and improve the pre-trained target model.

Now we can summarize our $\mathcal{S}/\mathcal{T}$ protocol for few-shot learning. Firstly, we pre-train ϕ^{pt} on $\mathcal{D}^{\text{tr}}$, and then sample *meta-training* tasks from seen classes. Secondly, we sort the *meta-training* tasks according to their hardness, and fine-tune the pre-trained network to obtain local target models for a small ratio of hardest tasks. Denote by $\mathcal{D}_1^{\text{tr}}$ the set of tasks that have target models and denote by $\mathcal{D}_2^{\text{tr}}$ the set of tasks that do not have target models. For tasks in $\mathcal{D}_1^{\text{tr}}$, we train task-specific solvers on their *support* sets, and then evaluate these solvers under $\mathcal{S}/\mathcal{T}$ protocol. For tasks in $\mathcal{D}_2^{\text{tr}}$, we simply use $\mathcal{S}/\mathcal{Q}$ to compute *query* loss, as shown in Equ (7).

$$\begin{aligned}\min_f \sum_{(\mathcal{S}_1^{\text{tr}},\mathcal{T}_1^{\text{tr}})\in\mathcal{D}_1^{\text{tr}}} \sum_{(\mathbf{x}_i,\mathbf{y}_i)\in\mathcal{S}_1^{\text{tr}}} &\left[(1-\lambda)\ell(f(\mathcal{S}_1^{\text{tr}})(\mathbf{x}_i),\mathbf{y}_i)+\lambda\mathbf{KL}(\mathcal{T}_1^{\text{tr}}(\mathbf{x}_i)||f(\mathcal{S}_1^{\text{tr}})(\mathbf{x}_i))\right] \\ &+\sum_{(\mathcal{S}_2^{\text{tr}},\mathcal{Q}_2^{\text{tr}})\in\mathcal{D}_2^{\text{tr}}} \sum_{(\mathbf{x}_j,\mathbf{y}_j)\in\mathcal{Q}_2^{\text{tr}}} \ell(f(\mathcal{S}_2^{\text{tr}})(\mathbf{x}_j),\mathbf{y}_j)\end{aligned} \tag{7}$$

Different from $\mathcal{S}/\mathcal{Q}$ protocol, $\mathcal{S}/\mathcal{T}$ protocol does not rely on randomly sampled *query* sets, and target models usually offer more information than instances. Distillation term plays the role of regularization, enforcing the solvers for hard tasks to be smooth (see next subsection). Although the idea of $\mathcal{S}/\mathcal{T}$ protocol is proposed in 2016, it is not widely used due to its computational intractability. However, in this paper we propose an efficient method to construct target models, and only deploy target models for a small ratio of hard tasks. This opens the door for future research of $\mathcal{S}/\mathcal{T}$ protocol, and unearth the potential of existing meta-learning algorithms.

5.3 Empirical Study: Gaussian Classification

In this part, we test our proposed method on a synthetic classification dataset. The purposes of this empirical study are two-fold: (1) check whether $\mathcal{S}/\mathcal{T}$ protocol with only a few target models can improve classic meta-learning algorithm; (2) study why distillation from target models can help.

Setting. In this experiment, we randomly generate 100 2-d Gaussian distributions. There are 64 classes for *meta-training*, 16 classes for *meta-validation*, and 20 classes for *meta-testing*. We sample

100 instances for each class to form the whole dataset. For each class, we sample its mean vector $\boldsymbol{\mu} \sim \mathcal{U}^2[-10, 10] \in \mathbb{R}^2$ and covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}'^\top \boldsymbol{\Sigma}'$ where $\boldsymbol{\Sigma}' \sim \mathcal{U}^{2\times 2}[-2, 2] \in \mathbb{R}^{2\times 2}$. Here $\mathcal{U}$ means uniform distribution. We then sample 10000 5-way 10-shot tasks for both *meta-training* and *meta-testing*. After every 500 episodes, we sample 500 tasks for *meta-validation*.

Algorithms. In this part, we use a ProtoNet [23] trained under $\mathcal{S}/\mathcal{Q}$ protocol as our baseline. It meta-learns a shared embedding function $\phi : \mathbb{R}^2 \to \mathbb{R}^{100}$ across tasks, and classifies an instance into the category of its nearest *support* class center. To be specific, let $\mathbf{c}_n = \frac{1}{K} \sum_{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{S} \wedge \mathbf{y}_i = n} \phi(\mathbf{x}_i)$ be the *support* class center of the n-th class[3], then for instance $\mathbf{x}$, the model will predict its N-dimensional label $\hat{\mathbf{y}}$ as $\hat{y}_n = \frac{\exp\{\langle \mathbf{c}_n, \phi(\mathbf{x}) \rangle\}}{\sum \exp\{\langle \mathbf{c}_n, \phi(\mathbf{x}) \rangle\}}, n \in [N]$. As a comparison, we also train a ProtoNet under $\mathcal{S}/\mathcal{T}$ protocol. Here the target model is constructed by fine-tuning the pre-trained global embedding network on specific tasks. To check whether $\mathcal{S}/\mathcal{T}$ protocol can work with only a few target models, we set the ratio the tasks that have target models to 5% and 10%. As presented in last part, we sort all *meta-training* tasks according to their hardness and fine-tune the pre-trained backbone on those hardest tasks. More implementation details can be found in the supplement.

Results and Discussions. Firstly, we report the *meta-testing* accuracy of different models in Table 4. Methods under $\mathcal{S}/\mathcal{T}$ protocol outperform vanilla ProtoNet by a large margin. Even with only 5% target models, we can obtain a remarkable accuracy improvement. Then, we study why $\mathcal{S}/\mathcal{T}$ protocol can help ProtoNet learn better. In Figure 6, we visualize a 5-way 10-shot task and the decision regions of 3 models in raw 2-d space. Figure 6a is the Bayesian optimal classifier $\mathcal{T}$, *i.e.*, for an instance $\mathbf{x}$, $p(\hat{\mathbf{y}} = n|\mathbf{x}) \propto \frac{1}{2\pi} \frac{1}{|\boldsymbol{\Sigma}_n|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_n)^\top \boldsymbol{\Sigma}_n^{-1} (\mathbf{x} - \boldsymbol{\mu}_n)\right\}$ where $\boldsymbol{\mu}_n$ and $\boldsymbol{\Sigma}_n$ are the mean vector and covariance matrix of class n. Because different classes have different covariance matrices, the decision boundary of Bayesian classifier is very steep. Figure 6b and Figure 6c are results of ProtoNet trained under $\mathcal{S}/\mathcal{Q}$ protocol and $\mathcal{S}/\mathcal{T}$ protocol respectively. In Figure 6c, the decision boundary is smooth and regular, which is different from the previous two models. This result offers a natural interpretation of $\mathcal{S}/\mathcal{T}$ protocol's benefit: target models impose a regularization on task-specific solvers, making them more robust to noisy and biased instances. In fact, [35] also gives a similar conclusion: knowledge distillation can be seen as a special label smoothing and it can regularize model training. In order to more clearly verify this property, we sample biased tasks only containing instances that have low likelihoods (< 0.3 or < 0.1), and test different models on them. In the second row and third row of Table 4, we can see that $\mathcal{S}/\mathcal{T}$ protocol can defend biased sampling to the maximum extent.

Table 4: Average accuracy on *meta-testing* set. Models trained under $\mathcal{S}/\mathcal{T}$ protocol outperform that trained under $\mathcal{S}/\mathcal{Q}$ protocol even though there are only a few tasks having target models. ϕ^{pt} means directly using the pre-trained network to solve *meta-testing* tasks without *meta-training* phase. The second row and the third row represent biased sampling, where we only sample instances that have low likelihoods. When instances are biased, the superiority of $\mathcal{S}/\mathcal{T}$ protocol is more evident because target models make task-specific solvers more robust.

Protocol	ϕ^{pt}	$\mathcal{S}/\mathcal{Q}$	$\mathcal{S}/\mathcal{T}$-5%	$\mathcal{S}/\mathcal{T}$-10%
ACC	82.33	87.90	**90.32**	**92.87**
ACC(< 0.3)	77.41	81.25	**87.66**	**90.14**
ACC(< 0.1)	65.57	70.10	**79.22**	**84.02**

5.4 Empirical Study: Benchmark Evaluation

In this part, we evaluate our $\mathcal{S}/\mathcal{T}$ protocol on two benchmark datasets, *i.e.*, *miniImageNet* [26] and *tiered*ImageNet [19]. Refer to supplement for dataset details. We try to answer four questions: (1) Can we achieve SOTA performance with a classic meta-learning model trained under $\mathcal{S}/\mathcal{T}$ protocol? (2) How does each component influence model's performance? (3) How does the hyper-parameter λ influence model's performance? (4, in supplement) How much time does $\mathcal{S}/\mathcal{T}$ protocol cost?

Algorithms. We implement two classic meta-learning algorithms, MAML and ProtoNet, under $\mathcal{S}/\mathcal{T}$ protocol. We use ResNet-12 as the backbone network, which is pre-trained on the *meta-training* set. For a fair comparison, we only include other algorithms that also use ResNet-12 as backbone network in Table 5. More implementation details can be found in the supplement.

[3] With a bit abuse of notation, we use $\mathbf{y}_i = n$ to select instances belonging to the n-th class.

Table 5: Average test accuracy with 95% confidence intervals on *meta-testing* tasks of *mini*ImageNet and *tiered*ImageNet. All the methods use ResNet-12 as backbone network. **Black** values are cited from existing papers while **red** values are implemented by us. Best results are in bold. We can see that MAML and ProtoNet trained under $\mathcal{S}/\mathcal{T}$ protocol outperform models trained under $\mathcal{S}/\mathcal{Q}$ protocol even with a few target models. Specifically, ProtoNet trained under $\mathcal{S}/\mathcal{T}$ protocol achieves state-of-the-art performance in most cases.

Method	*mini*ImageNet		*tiered*ImageNet	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
DeepEMD [36]	65.91 ± 0.82	82.41 ± 0.56	71.16 ± 0.87	86.03 ± 0.58
FEAT [34]	66.78 ± 0.20	82.05 ± 0.14	70.80 ± 0.23	84.79 ± 0.16
FRN [31]	66.45 ± 0.19	$\mathbf{82.83 \pm 0.13}$	72.06 ± 0.22	86.89 ± 0.14
DMF [32]	67.76 ± 0.46	82.71 ± 0.31	71.89 ± 0.52	85.96 ± 0.35
MAML($\mathcal{S}/\mathcal{Q}$, re-implement)	63.39 ± 0.37	79.90 ± 0.25	67.41 ± 0.41	81.36 ± 0.29
MAML($\mathcal{S}/\mathcal{T}$-5%)	64.40 ± 0.42	80.33 ± 0.37	69.02 ± 0.30	82.99 ± 0.40
MAML($\mathcal{S}/\mathcal{T}$-10%)	64.71 ± 0.32	81.24 ± 0.62	70.13 ± 0.55	83.04 ± 0.25
ProtoNet($\mathcal{S}/\mathcal{Q}$) [23]	60.37 ± 0.83	78.02 ± 0.57	65.65 ± 0.92	83.40 ± 0.65
ProtoNet($\mathcal{S}/\mathcal{Q}$, re-implement)	65.30 ± 0.30	79.93 ± 0.39	70.34 ± 0.45	84.68 ± 0.55
ProtoNet($\mathcal{S}/\mathcal{T}$-5%)	67.35 ± 0.49	81.67 ± 0.62	71.25 ± 0.37	85.80 ± 0.31
ProtoNet($\mathcal{S}/\mathcal{T}$-10%)	$\mathbf{68.03 \pm 0.52}$	82.53 ± 0.47	$\mathbf{72.41 \pm 0.39}$	$\mathbf{86.91 \pm 0.47}$

Competitive Results against SOTA. We show in Table 5 that by MAML or ProtoNet can be improved a lot when trained under $\mathcal{S}/\mathcal{T}$ protocol with only 5% or 10% target models. Note that vanilla ProtoNet does not use pre-training trick, and we re-implement it with pre-training. ProtoNet is proposed in 2017, but we can obtain SOTA performance (2020, 2021) by retraining it under $\mathcal{S}/\mathcal{T}$ protocol with only a few target models. This verifies the superiority of $\mathcal{S}/\mathcal{T}$ protocol. In fact, $\mathcal{S}/\mathcal{T}$ protocol is a generic training protocol that can be applied to any meta-learning algorithm, and we apply $\mathcal{S}/\mathcal{T}$ protocol to more meta-learning algorithms in the supplement to show the effectiveness of our proposed method.

Ablation Study. In this part, we check the effectiveness of each component. Table 6 shows that our proposed hardness metric and fine-tuning strategy help to improve performance.

Hyper-Parameter. We check the influence of hyper-parameter λ in Equ (7). We sample 5-way tasks from *mini*ImageNet, and try different λ values. Figure 7 shows that larger λ tends to benefit model performance.

Table 6: Ablation study. "Random" means selecting tasks randomly rather than according to their hardness. "ϕ^{pt}" means using the pre-trained network as target model for all tasks. We can see that our proposed heuristic hardness metric and the fine-tuning strategy improve model performance.

Model	1-shot	5-shot
ProtoNet($\mathcal{S}/\mathcal{Q}$)	65.30	79.93
ProtoNet($\mathcal{S}/\mathcal{T}$-10%-random)	66.72	81.05
ProtoNet($\mathcal{S}/\mathcal{T}$-10%-ϕ^{pt})	67.47	81.70
ProtoNet($\mathcal{S}/\mathcal{T}$-10%-hardness-ϕ^{ft})	**68.03**	**82.53**

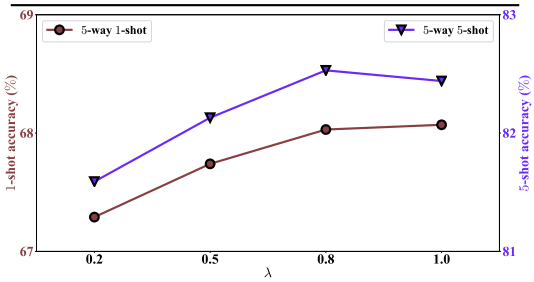


Figure 7: Performance change over λ. We can see that larger λ tends to benefit model accuracy. We set λ to 0.8, a relatively large value.

6 Conclusion

In this paper, we study $\mathcal{S}/\mathcal{T}$ meta-learning protocol that evaluates a task-specific solver by comparing it to a target model. $\mathcal{S}/\mathcal{T}$ protocol offers a more informative supervision signal for meta-learning, but is difficult to use in practice owing to its high computational cost. We find that by only deploying target models for those hardest tasks, we can improve existing meta-learning algorithms while maintaining efficiency. We propose a heuristic task hardness metric and a convenient target model construction method for few-shot learning. Experiments on synthetic datasets and benchmark datasets demonstrate the superiority of $\mathcal{S}/\mathcal{T}$ protocol and effectiveness of our proposal.

References

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, 2009.

[2] Kaidi Cao, Jingwei Ji, Zhangjie Cao, Chien-Yi Chang, and Juan Carlos Niebles. Few-shot video classification via temporal alignment. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 10618–10627, 2020.

[3] Wei-Lun Chao, Han-Jia Ye, De-Chuan Zhan, Mark Campbell, and Kilian Q Weinberger. Revisiting meta-learning as supervised learning. *CoRR*, abs/2002.00573, 2020.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.

[5] Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling meta-learning: Understanding feature representations for few-shot tasks. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3607–3616, 2020.

[6] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[7] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems 31*, pages 5302–5311, 2018.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[9] Ekaterina Iakovleva, Jakob Verbeek, and Karteek Alahari. Meta-learning with shared amortized variational inference. In *Proceedings of the 37th International Conference on Machine Learning*, pages 4572–4582, 2020.

[10] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *32nd International Conference on Machine Learning Workshop*, volume 2, 2015.

[11] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the 32nd IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[12] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *CoRR*, abs/1707.09835, 2017.

[13] Su Lu, Han-Jia Ye, and De-Chuan Zhan. Few-shot action recognition with compromised metric via optimal transport. *CoRR*, abs/2104.03737, 2021.

[14] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems 31*, pages 721–731. 2018.

[15] Seong-Jin Park, Seungju Han, Ji-Won Baek, Insoo Kim, Juhwan Song, Hae Beom Lee, Jae-Joon Han, and Sung Ju Hwang. Meta variance transfer: Learning to augment from the others. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7510–7520, 2020.

[16] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, pages 5331–5340, 2019.

[17] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[18] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *Proceedings of the 17th International Conference on Computer Vision*, pages 331–339, 2019.

[19] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and WRichard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

[20] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

[21] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1842–1850, 2016.

[22] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 4136–4145, 2020.
[23] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30*, pages 4077–4087. 2017.
[24] Sebastian Thrun and Lorien Pratt. *Learning to Learn*. Springer Science & Business Media, 2012.
[25] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.
[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638. 2016.
[27] Risto Vuorio, Shao-Hua Sun, Hexiang Hu, and Joseph J Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems 32*, pages 1–12. 2019.
[28] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *CoRR*, abs/1911.04623, 2019.
[29] Yu-Xiong Wang, Adrien Bardes, Ruslan Salakhutdinov, and Martial Hebert. Progressive knowledge distillation for generative modeling. 2019.
[30] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *Proceedings of the 14th European Conference on Computer Vision*, pages 616–634, 2016.
[31] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Fine-grained few-shot classification with feature map reconstruction networks. *CoRR*, abs/2012.01506, 2020.
[32] Chengming Xu, Chen Liu, Li Zhang, Chengjie Wang, Jilin Li, Feiyue Huang, Xiangyang Xue, and Yanwei Fu. Learning dynamic alignment via meta-filter for few-shot learning. *CoRR*, abs/2103.13582, 2021.
[33] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. *CoRR*, abs/2101.06395, 2021.
[34] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020.
[35] Li Yuan, Francis EH Tay, Guilin Li, Tao Wang, and Jiashi Feng. Revisiting knowledge distillation via label smoothing regularization. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 3903–3911, 2020.
[36] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 12203–12213, 2020.
[37] Min Zhang, Donglin Wang, and Sibo Gai. Knowledge distillation for model-agnostic meta-learning. In *Proceedings of the 24th European Conference on Artificial Intelligence*, pages 1355–1362. 2020.
[38] Xiantong Zhen, Yingjun Du, Huan Xiong, Qiang Qiu, Cees GM Snoek, and Ling Shao. Learning to learn variational semantic memory. *CoRR*, 2020.
[39] Xiantong Zhen, Haoliang Sun, Yingjun Du, Jun Xu, Yilong Yin, Ling Shao, and Cees Snoek. Learning to learn kernels with variational random features. In *Proceedings of the 37th International Conference on Machine Learning*, pages 11626–11636, 2020.
[40] Linjun Zhou, Peng Cui, Xu Jia, Shiqiang Yang, and Qi Tian. Learning to select base classes for few-shot classification. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 4624–4633, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See supplementary material.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We do not think our work has any potential negative societal impacts.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Table 5.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplementary material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5 and supplementary material.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]