Towards Enabling Meta-Learning from Target Models

Su Lu Han-Jia Ye Le Gan De-Chuan Zhan
State Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, 210023, China
{lusu,yehj}@lamda.nju.edu.cn, {ganl,zhandc}@nju.edu.cn

Abstract

Meta-learning can extract an inductive bias from previous learning experience and assist the training of new tasks. It is often realized through optimizing a meta-model with the evaluation loss 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 (S/Q protocol). Different from S/Q protocol, we can also evaluate a task-specific solver by comparing it to a target model T, which is the optimal model for this task or a model that behaves well enough on this task (S/T protocol). Although being short of research, S/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. We empirically verify the effectiveness of S/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 and target models via knowledge distillation.

1 Introduction

Meta-learning means improving performance measures over a family of tasks by their training experience [22]. It has been researched in various fields such as image classification [11, 16] and reinforcement learning [6, 14]. 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 because the evaluation result acts as the supervision signal for meta-model. Early meta-learning algorithms [19, 23] 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 [24, 28], which respectively export solver evaluation from the perspective of “data” and “model”. In this paper, we call these two methodologies S/Q protocol and S/T protocol. In S/Q protocol, S means support set and Q means query set. They

*De-Chuan Zhan is the corresponding author.
contain non-overlapping instances sampled from a same distribution. By training the solver on \( S \) and evaluating it on \( Q \), 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 \( T \). Assuming that \( T \) works well on a task, we can minimize the discrepancy between the trained solver and \( T \) to pull the solver closer to \( T \). Here \( T \) can be Bayesian optimal solution to a task or a model trained on a sufficiently informative dataset. Figure 1 gives an illustration of both \( S/Q \) protocol and \( S/T \) protocol.

Although appeared in the same year, \( S/Q \) protocol is more widely accepted by meta-learning society [4, 8, 13, 19] while the research about how to leverage target models remains immature. The main reason is the simplicity of \( S/Q \) and the computational hardness of \( S/T \). However, \( S/T \) 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 [2]. Thirdly, we can treat the target model as a teacher and incorporate a teacher-student framework like knowledge distillation [7] and curriculum learning [1] in meta-learning. Thus, it is necessary and meaningful to study \( S/T \) protocol in meta-learning.

This paper looks into \( S/T \) 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 \( S/T \) 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 studied several kinds of meta-knowledge like model initialization [3, 25], embedding network [21, 12, 9, 20, 4], external memory [19, 5], optimization strategy [15, 18], and data augmentation strategy [13]. Despite their diversity in meta-knowledge, most existing models are trained under \( S/Q \) protocol, and rely on a randomly sampled and possibly biased query set. Actually, most algorithms are protocol-agnostic, and both \( S/Q \) protocol and \( S/T \) protocol can be applied to them. Thus, our work on \( S/T \) is general, and it has a wide application field.

Learning from Target Models. The idea of learning from target models in meta-learning is first proposed by [28]. In [28], 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 [27, 31]. The most similar work to us is [31], which constructs target models with abundant instances and matches task-specific solvers and target models. However, they all assume that every single task has a target model, increasing both space and time complexity of $S/T$ protocol. To summarize, we claim that one key point in putting $S/T$ 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 $p(\tau)$ and reuses the learned meta-model on new tasks belonging to the same distribution. Each task $\tau$ has a task-specific support set $S = \{(x_i, y_i)\}_{i=1}^{|S|}$, and we can train on $S$ a solver $g : \mathcal{X} \rightarrow \mathcal{Y}$ parameterized by $\gamma$. Without loss of generality, a meta-model can be defined as $f : \mathbb{S} \rightarrow \mathbb{G}$ parameterized by $\theta_f$ 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 $S$ under the supervision of meta-knowledge $\theta_f$. Taking two well-known meta-learning algorithms, MAML [3] and ProtoNet [21], as examples, we have the following concrete forms of $f$:

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

$$f(S; \theta_f) = g \left( \cdot ; \theta_f - \eta \nabla_{\gamma} \sum_{(x_i, y_i) \in S} \ell(g(x_i; \gamma), y_i) \bigg|_{\gamma=\theta_f} \right)$$

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

$$f(S; \theta_f) = g \left( \cdot ; \theta_f, \{\phi_{\theta_f}(x_i) | (x_i, y_i) \in S\} \right)$$

$S/Q$ Protocol. How to evaluate the solver $g$ trained on $S$? The answer to this question differs conventional $S/Q$ protocol [24] from $S/T$ protocol. In $S/Q$ protocol, we sample another query set $Q = \{x_j, y_j\}_{j=1}^{|Q|}$ apart from $S$ for each task. Instances in $S$ and $Q$ are i.i.d. distributed and have a same label set, and we evaluate $g$ by its loss on $Q$. Since $S$ and $Q$ contain non-overlapping instances, loss on $Q$ is a more reliable supervision signal. $S/Q$ protocol can be formulated as Equ (2). Here $D^w$ is the meta-training set and we can sample meta-training tasks $\tau^w$ from it.

$$\min_f \sum_{\tau^w=(S^w, Q^w) \in D^w} \sum_{(x_j, y_j) \in Q^w} \ell(f(S^w)(x_j), y_j)$$

$S/T$ Protocol. Any sampled query set $Q$ can be biased and noisy, which may cause an inaccurate evaluation of the solver. An alternative is directly matching the task-specific solver $g = f(S)$ and a target model $T$ 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 (2), we have the following $S/T$ protocol Equ (3). Here $\mathcal{L} : \mathbb{G} 	imes \mathbb{G} \rightarrow \mathbb{R}_+$ is some loss function to measure the discrepancy between $g$ and target model $T$.

$$\min_f \sum_{\tau^w=(S^w, T^w) \in D^w} \mathcal{L}(f(S^w), T^w)$$

4 Effect of Target Model

We have introduced some basic concepts in meta-learning, and formulate $S/Q$ protocol and $S/T$ protocol in Section 3. 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 of S/T protocol.

### 4.1 Model Matching

In S/T protocol, one key point is how to match the solver $g$ and its target model $T$. In other words, we need to specify the concrete formulation of $\mathcal{L}(g, T)$. Generally, methods to match $g$ to $T$ can be classified into two categories. Firstly, we can directly match two models’ parameters or use another model to regress between two models’ parameters \(23\). For example, let $\gamma_g$ and $\gamma_T$ be the parameters of $g$ and $T$, we can set $\mathcal{L}(g, T) = \sum_{(x_i, y_i) \in S} f(g(x_i), y_i) + \lambda \|\gamma_g - \gamma_T\|^2$. Here $\lambda$ is a balancing hyperparameter. 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., $\mathcal{L}(g, T) = \sum_{(x_i, y_i) \in S} [(1 - \lambda)\ell(g(x_i), y_i) + \lambda D(T(x_i), g(x_i))].$ Here $D(\cdot, \cdot)$ is a function that measures the discrepancy between $T(x_i)$ and $g(x_i)$. If we instantiate $D(\cdot, \cdot)$ as KL divergence $KL(\cdot||\cdot)$ for classification problem, the aforementioned loss function is equivalent to that of knowledge distillation.

### 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 S/T protocol outperform S/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 $T(x) = a \sin(bx - c)$ 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 S/Q protocol, we additionally sample 30 query instances for each task. We then set $y = T(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.5)$ is 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 \(3\) and ProtoNet \(21\). MAML can be directly applied to 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 $\phi : \mathbb{R} \rightarrow \mathbb{R}^{100}$, with assistance of which the similarity-based regression model $g(\cdot : \phi(x_i)|(x_i, y_i) \in S)$ works well across all tasks. Here for any instance $(x, y)$, $\hat{y} = g(x) = \sum_{(x', y') \in S} w_{i}y_{i}$ and $w_{i} = \exp[\langle \phi(x_i), \phi(x)\rangle]$. A same embedding network is used in two algorithms. We train MAML and ProtoNet under S/Q protocol and S/T protocol. When using S/Q protocol, we minimize MSE loss on 30 query instances to optimize $\phi$. For S/T protocol, we match the solver and the target model in output space, and set $D(T(x_i), g(x_i)) = \|T(x_i) - g(x_i)\|^2$. Thus, the loss function under S/T protocol is $\mathcal{L}(g, T) = \sum_{(x_i, y_i) \in S} [(1 - \lambda)\ell(g(x_i), y_i)^2 + \lambda\|g(x_i) - T(x_i)\|^2]$. \(\lambda\) is a hyper-parameter. More implementation details can be found in the supplementary material.

**Superiority of S/T Protocol.** Table 1 shows the MSE of four models on meta-testing tasks. We can see that models trained under S/T protocol consistently outperform models trained under S/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 S/Q protocol and S/T protocol respectively. We can see that models trained under S/T protocol fit the target sinusoid regression problem well.
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\(^2\) that meta-learning loss under $S/T$ protocol $\left(1 - \lambda\right)\left\|g(x) - y\right\|^2 + \lambda\left\|g(x) - T(x)\right\|^2$ is an upper bound of $\left\|g(x) - (y - \lambda c)\right\|^2$, which is the standard MSE loss between the output of solver $g$ and cleaner label $y - \lambda c$ (raw label $y$ equals to $T(x) + c$). Therefore, the larger $\lambda$ is, the cleaner training labels are. Table 2 is an ablation study on hyper-parameter $\lambda$. As expected, both algorithms trained under $S/T$ achieve better performance with larger $\lambda$. These results demonstrate the superiority of $S/T$ 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 $S/T$ 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 [23] or building one global target model. In this paper, we study a more general 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 $S/Q$ 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 $S/T$ protocol outperform those trained under $S/Q$ protocol.

<table>
<thead>
<tr>
<th>Method</th>
<th>MAML $S/Q$</th>
<th>ProtoNet $S/Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE on $D^p$</td>
<td>4.933</td>
<td>4.706</td>
</tr>
<tr>
<td>MSE: MAML($S/T$)</td>
<td>3.220</td>
<td>3.419</td>
</tr>
<tr>
<td>MSE: ProtoNet($S/T$)</td>
<td>3.137</td>
<td>3.304</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>1</th>
<th>0.8</th>
<th>0.5</th>
<th>0.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE: MAML($S/T$)</td>
<td>3.220</td>
<td>3.419</td>
<td>3.621</td>
<td>3.833</td>
</tr>
<tr>
<td>MSE: ProtoNet($S/T$)</td>
<td>3.137</td>
<td>3.304</td>
<td>3.332</td>
<td>3.550</td>
</tr>
</tbody>
</table>

Figure 3: Visualization of a randomly sampled meta-testing task. Dotted lines are used for $S/Q$ protocol while dashed lines are used for $S/T$ protocol. We can see that models trained under $S/T$ 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.

\(^2\)We leave the proof to the supplementary material.
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. In few-shot learning, we have two datasets that contain non-overlapping classes, i.e., \( D^{tr} \) and \( D^{ts} \). \( D^{tr} \) is composed of seen classes while \( D^{ts} \) contains unseen classes. We can sample \( N \)-way \( K \)-shot \( \text{meta-training} \) tasks from \( D^{tr} \) to train the meta-model, and expect that the trained meta-model will also work well on \( D^{ts} \).

5.1 Task Hardness

Following the idea of constructing target models for hard tasks, we firstly investigate which tasks are hard in few-shot learning. We consider the relationship between classes as a key factor that determines the hardness of a classification task. Assuming that there are \( C^u \) classes in \( D^{tr} \), we first compute a similarity matrix \( F \in \mathbb{R}^{C^u \times C^u} \) whose element \( F_{uv} \) equals to the similarity between the \( u \)-th class centre and the \( v \)-th class center. In few-shot learning, pre-training the backbone network on \( D^{tr} \) has become a common practice \cite{26, 30}, and we can compute these class centres based on the pre-trained model \( \phi^{pt} \) as Eq \( (5) \) and Eq \( (6) \). In Eq \( (5) \), \( K_u \) is the number of instances of class \( u \) in \( D^{tr} \), and with a bit abuse of notation, we use \( y_i = u \) to select instances belonging to the \( u \)-th class.

\[
c_u = \frac{1}{K_u} \sum_{(x_i, y_i) \in D^{tr}, y_i = u} \phi^{pt}(x_i), \quad u \in [C^u]
\]  

(5)

\[
F_{uv} = \frac{c_u \cdot c_v}{||c_u|| \cdot ||c_v||}, \quad u, v \in [C^u]
\]  

(6)

With similarity matrix \( F \), we can take out the sub-similarity matrix of task \( \tau \) by slicing the rows and columns corresponding to classes contained in \( \tau \). The hardness of task \( \tau \) is defined as the sum of its sub-similarity matrix. The more similar classes in \( \tau \) are, the more difficult to differ them from each other. The hardness of every \text{meta-training} task can be evaluated with similarity matrix \( F \), and we compute \( F \) 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 \( \phi^{pt} \) is optimized using cross-entropy loss on the whole \text{meta-training} set, and can classify all classes in \( D^{tr} \). Since there are \( C^u \) classes in \( D^{tr} \), the output of \( \phi^{pt} \) is a \( C^u \)-dimensional vector. Given a specific \( N \)-way task \( \tau \), 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 \( D^{tr} \) that contains classes in \( \tau \) can give us a better target model for \( \tau \).

Evaluation on Auxiliary Dataset. To verify the reasonability of the heuristic task hardness metric and the effectiveness of the fine-tuning approach, we need another auxiliary dataset \( D^{au} \). \( D^{au} \) contains same classes as \( D^{tr} \), and we can evaluate the accuracy of constructed target models (trained on \( D^{tr} \)) on \( D^{au} \). We conduct an experiment on \text{miniImageNet} \cite{24} to check whether fine-tuned target models

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
Target Model & pre-train & fine-tune \\
\hline
Accuracy on \( D^{au} \) & 98.24 & 99.37 \\
\hline
\end{tabular}
\caption{Average accuracy on auxiliary dataset \( D^{au} \). Fine-tuned target models outperform a single pre-trained target model on randomly sampled tasks.}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure5.png}
\caption{Grouping of 1000 tasks according to their hardness. Both \( \phi^{pt} \) and \( \phi^{au} \) achieve lower accuracy on harder tasks, verifying the reasonability of our proposed hardness metric. Fine-tuned target models obtain a remarkable performance gain on hard tasks.}
\end{figure}

\footnote{An \( N \)-way \( K \)-shot task is a classification task with \( N \) classes and \( K \) instances in each class.}

6
are better than pre-trained target models. Firstly, we pre-train a ResNet-12 with a linear layer on the meta-training split of miniImageNet. After that, we randomly sample 1000 5-way tasks from \( D^\text{pt} \), and fine-tune the pre-trained backbone to obtain 1000 target models. For each task \( \tau \), we take out all instances in \( D^\text{su} \) that belong to classes in \( \tau \) to evaluate \( \phi^\text{pt} \) and \( \phi^\text{ft} \). Table 3 shows the average accuracy on auxiliary dataset \( D^\text{su} \). 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 already works well enough on these seen classes. This means it is not cost-effective to fine-tune a target model and unearth the potential of existing meta-learning algorithms.

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 for a small ratio of hardest tasks. This opens the door for future research of meta-learning.

Now we can summarize our \( S/T \) protocol for few-shot learning. Firstly, we pre-train \( \phi^\text{pt} \) on \( D^\text{pt} \), 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 \( D^\text{pt}_2 \) the set of tasks that have target models and \( D^\text{pt}_1 \) the set of tasks that do not have target models. For tasks in \( D^\text{pt}_1 \), we train task-specific solvers on their support sets, and then evaluate these solvers under \( S/T \) protocol. For tasks in \( D^\text{pt}_2 \), we simply use \( S/Q \) to compute query loss, as shown in Equ. 7.

\[
\min_f \sum_{(S^1_j, T^1_j) \in D^\text{pt}_1} \sum_{(x_i, y_i) \in S^1_j} [(1 - \lambda)\ell(f(S^j_1)(x_i), y_i) + \lambda \text{KL}(T^j_1(x_i) || f(S^j_1)(x_i))]
+ \sum_{(S^2_j, Q^2_j) \in D^\text{pt}_2} \sum_{(x_j, y_j) \in Q^2_j} \ell(f(S^j_2)(x_j), y_j)
\]

Different from \( S/Q \) protocol, \( S/T \) protocol does not rely on randomly sampled query sets, and target models usually offer more information than instances. Distillation term plays the role of regularizing, enforcing the solvers for hard tasks to be smooth (see next subsection). Although the idea of \( S/T \) 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 \( S/T \) 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 S/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 \( \mu \sim \mathcal{U}[−10, 10] \in \mathbb{R}^2 \) and covariance matrix \( \Sigma = \Sigma' \Sigma' \) where \( \Sigma' \sim \mathcal{U}[−2, 2] \in \mathbb{R}^{2 \times 2} \). 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 \([21]\) trained under S/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 \( c_n = \frac{1}{N} \sum_{(x, y) \in S \cap \mathcal{y}_n = n} \phi(x) \) be the support class center of the \( n \)-th class. Then for instance \( x \), the model will predict its \( N \)-dimensional label \( \hat{y} \) as \( \hat{y}_n = \frac{\exp\{\langle c_n, \phi(x) \rangle \}}{\sum_{i \in [N]} \exp\{\langle c_i, \phi(x) \rangle \}} \), \( n \in [N] \). As a comparison, we also train a ProtoNet under S/T protocol. Here the target model is constructed by fine-tuning the pre-trained global embedding network on specific tasks. To check whether S/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. Refer to supplementary material for more details.

Results and Discussions. Firstly, we report the meta-testing accuracy of different models in Table 4. Methods under S/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 S/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 T, i.e., for an instance \( x \), \( p(\hat{y} = n|x) \propto \frac{1}{2\pi} \left| \Sigma_n \right|^{-\frac{1}{2}} \exp \left\{ -\frac{1}{2}(x - \mu_n)^\top \Sigma_n^{-1}(x - \mu_n) \right\} \) where \( \mu_n \) and \( \Sigma_n \) 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 S/Q protocol and S/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 S/T protocol’s benefit: target models impose a regularization on task-specific solvers, making them more robust to noisy and biased instances. In fact, \([32]\) 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 S/T protocol can defend biased sampling to the maximum extent because of the strong supervision offered by target models.

Table 4: Average accuracy on meta-testing set. Models trained under S/T protocol outperform those trained under S/Q protocol even though there are only a few target models. \( \phi^a \) 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 S/T protocol is more evident because target models make task-specific solvers more robust.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>( \phi^a )</th>
<th>S/Q</th>
<th>S/T-5%</th>
<th>S/T-10%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC</td>
<td>82.33</td>
<td>87.90</td>
<td>90.32</td>
<td>92.87</td>
</tr>
<tr>
<td>ACC(&lt; 0.3)</td>
<td>77.41</td>
<td>81.25</td>
<td>87.66</td>
<td>90.14</td>
</tr>
<tr>
<td>ACC(&lt; 0.1)</td>
<td>65.57</td>
<td>70.10</td>
<td>79.22</td>
<td>84.02</td>
</tr>
</tbody>
</table>

4With a bit of abuse of notation, we use \( y_i = n \) to select instances belonging to the \( n \)-th class.
Table 5: Average test accuracy with 95% confidence intervals on meta-testing tasks of miniImageNet and tieredImageNet. All the methods use ResNet-12 as backbone network except MAML with * mark. The row with * mark uses a 4-layer ConvNet as backbone network, which is shallower than ResNet-12. Blue values are cited from existing papers while red values are reproduced by us. Best results are in bold. We can see that MAML and ProtoNet trained under S/T protocol outperform models trained under S/Q protocol even with a few target models. Specifically, ProtoNet trained under S/T protocol achieves state-of-the-art performance in most cases.

<table>
<thead>
<tr>
<th>Method</th>
<th>miniImageNet 5-way 1-shot</th>
<th>miniImageNet 5-way 5-shot</th>
<th>tieredImageNet 5-way 1-shot</th>
<th>tieredImageNet 5-way 5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepEMD [33]</td>
<td>65.91 ± 0.82</td>
<td>82.41 ± 0.56</td>
<td>71.16 ± 0.87</td>
<td>86.03 ± 0.58</td>
</tr>
<tr>
<td>FEAT [30]</td>
<td>66.78 ± 0.20</td>
<td>82.05 ± 0.14</td>
<td>70.80 ± 0.23</td>
<td>84.79 ± 0.16</td>
</tr>
<tr>
<td>FRN [29]</td>
<td>66.45 ± 0.19</td>
<td>82.83 ± 0.13</td>
<td>72.06 ± 0.22</td>
<td>86.89 ± 0.14</td>
</tr>
<tr>
<td>MAML (S/Q)* [3]</td>
<td>48.70 ± 1.84</td>
<td>63.11 ± 0.92</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MAML (S/Q, re-implement)</td>
<td>58.84 ± 0.25</td>
<td>74.62 ± 0.38</td>
<td>63.02 ± 0.30</td>
<td>67.26 ± 0.32</td>
</tr>
<tr>
<td>MAML (S/T-5%)</td>
<td>59.14 ± 0.33</td>
<td>75.77 ± 0.29</td>
<td>64.52 ± 0.30</td>
<td>68.39 ± 0.34</td>
</tr>
<tr>
<td>MAML (S/T-10%)</td>
<td>60.06 ± 0.35</td>
<td>76.34 ± 0.42</td>
<td>65.23 ± 0.45</td>
<td>70.02 ± 0.33</td>
</tr>
<tr>
<td>ProtoNet (S/Q) [21]</td>
<td>60.37 ± 0.83</td>
<td>78.02 ± 0.57</td>
<td>65.65 ± 0.92</td>
<td>83.40 ± 0.65</td>
</tr>
<tr>
<td>ProtoNet (S/Q, re-implement)</td>
<td>65.30 ± 0.30</td>
<td>79.93 ± 0.39</td>
<td>70.34 ± 0.45</td>
<td>84.68 ± 0.55</td>
</tr>
<tr>
<td>ProtoNet (S/T-5%)</td>
<td>67.35 ± 0.49</td>
<td>81.67 ± 0.62</td>
<td>71.25 ± 0.37</td>
<td>85.80 ± 0.31</td>
</tr>
<tr>
<td>ProtoNet (S/T-10%)</td>
<td><strong>68.03 ± 0.52</strong></td>
<td>82.53 ± 0.47</td>
<td><strong>72.41 ± 0.39</strong></td>
<td><strong>86.91 ± 0.47</strong></td>
</tr>
</tbody>
</table>

Table 6: Ablation study. “Random” means selecting tasks randomly rather than according to their hardness. “φpt” means using the pre-trained network as target model for all tasks. Best results are in bold. We can see that our proposed heuristic hardness metric and the fine-tuning strategy improve model performance.

<table>
<thead>
<tr>
<th>Model</th>
<th>miniImageNet 5-way 1-shot</th>
<th>miniImageNet 5-way 5-shot</th>
<th>tieredImageNet 5-way 1-shot</th>
<th>tieredImageNet 5-way 5-shot</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAML (S/Q)</td>
<td>58.84</td>
<td>74.62</td>
<td>63.02</td>
<td>67.26</td>
</tr>
<tr>
<td>MAML (S/T-10%-random)</td>
<td>59.66</td>
<td>74.90</td>
<td>65.11</td>
<td>68.63</td>
</tr>
<tr>
<td>MAML (S/T-10%-φpt)</td>
<td>59.35</td>
<td>75.88</td>
<td>64.78</td>
<td>69.26</td>
</tr>
<tr>
<td>MAML (S/T-10%-hardness-φpt)</td>
<td><strong>60.06</strong></td>
<td><strong>76.34</strong></td>
<td><strong>65.23</strong></td>
<td><strong>70.02</strong></td>
</tr>
<tr>
<td>ProtoNet (S/Q)</td>
<td>65.30</td>
<td>79.93</td>
<td>70.34</td>
<td>84.68</td>
</tr>
<tr>
<td>ProtoNet (S/T-10%-random)</td>
<td>66.72</td>
<td>81.05</td>
<td>71.22</td>
<td>85.37</td>
</tr>
<tr>
<td>ProtoNet (S/T-10%-φpt)</td>
<td>67.47</td>
<td>81.70</td>
<td>71.55</td>
<td>86.04</td>
</tr>
<tr>
<td>ProtoNet (S/T-10%-hardness-φpt)</td>
<td><strong>68.03</strong></td>
<td><strong>82.53</strong></td>
<td><strong>72.41</strong></td>
<td><strong>86.91</strong></td>
</tr>
</tbody>
</table>

5.4 Empirical Study: Benchmark Evaluation

In this part, we evaluate our S/T protocol on two benchmark datasets, i.e., miniImageNet [24] and tieredImageNet [17]. Refer to supplementary material for dataset details. We try to answer four questions: (1) Can we achieve SOTA performance with a classic meta-learning model trained under S/T protocol? (2) How does each component influence model’s performance? (3) How does the hyper-parameter λ influence model’s performance? (4) How much time does S/T protocol cost?

Algorithms. We implement two classic meta-learning algorithms, MAML and ProtoNet, under S/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 supplementary material.

Competitive Results against SOTA. We show in Table 5 that MAML or ProtoNet can be improved a lot when trained under S/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 by retraining it under S/T protocol without the few target models. This verifies the superiority of S/T protocol. In fact, S/T protocol is a generic training protocol that can be applied to any meta-learning algorithm, and we compare S/T protocol to more meta-learning algorithms in the supplementary material to show the effectiveness of our method.

*[1] Our code is available at https://github.com/njulus/ST*
Table 7: Time consumption for fine-tuning target models on miniImageNet.

<table>
<thead>
<tr>
<th>Number of Target Models</th>
<th>Time Consumption (min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>500 (5%)</td>
<td>224.3</td>
</tr>
<tr>
<td>1000 (10%)</td>
<td>420.6</td>
</tr>
<tr>
<td>2000 (20%)</td>
<td>851.7</td>
</tr>
</tbody>
</table>

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. Randomly sampling 10% tasks and constructing target models for these tasks improves model performance. The third row and the seventh row in Table 6 verify that learning from target models is beneficial even though the target models are not optimal. With only 10% locally fine-tuned target models and our heuristic hardness metric, we can achieve nearly state-of-the-art performance by ProtoNet.

Hyper-Parameter. We check the influence of hyper-parameter λ in Equ (7). We sample 5-way tasks from miniImageNet, and try different λ values. Figure 7 shows that larger λ tends to benefit model performance. We set λ to 0.8, a relatively large value, in most of experiments.

Time Consumption. In S/T protocol for few-shot learning, we need to construct target models through fine-tuning the globally pre-trained network. This will cost extra time to train a model. In this part, we try to answer the following question: how much time does S/T protocol cost in few-shot learning? We range the ratio of tasks that have target models in {5%, 10%, 20%}, and report the time consumption of fine-tuning target models on miniImageNet. Results are shown in Table 7. We run the experiment on an Nvidia GeForce RTX 2080ti GPU and Intel(R) Xeon(R) Silver 4110 CPU. We can see that about 4 hours are needed to fine-tune target models for 5% meta-training tasks, and time consumption for fine-tuning 2000 target models is still acceptable.

6 Conclusion

In this paper, we study S/T meta-learning protocol that evaluates a task-specific solver by comparing it to a target model. S/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 S/T protocol and effectiveness of our proposed method.

Acknowledgements

This work is supported by National Key R&D Program of China (2020AAA0109401), NSFC (61773198, 62006112), and NSF of Jiangsu Province (BK20200313).

References


