Few-Shot Learning of Compact Models via Task-Specific Meta Distillation

Yong Wu[1], Shekhor Chanda[2], Mehrdad Hosseinzadeh[3], Zhi Liu[1*], Yang Wang[4*]

[1]Shanghai University, [2]University of Manitoba, [3] Huawei Technologies Canada, [4]Concordia University

yong_wu@shu.edu.cn, chandas@myumanitoba.ca, mehrdad.hosseinzadeh@live.com

liuzhi@staff.shu.edu.cn, yang.wang@concordia.ca

Abstract

We consider a new problem of few-shot learning of compact models. Meta-learning is a popular approach for few-shot learning. Previous work in meta-learning typically assumes that the model architecture during meta-training is the same as the model architecture used for final deployment. In this paper, we challenge this basic assumption. For final deployment, we often need the model to be small. But small models usually do not have enough capacity to effectively adapt to new tasks. In the mean time, we often have access to the large dataset and extensive computing power during meta-training since meta-training is typically performed on a server. In this paper, we propose task-specific meta distillation *that simultaneously learns two models in meta-learning: a large teacher model and a small student model. These two models are jointly learned during meta-training. Given a new task during meta-testing, the teacher model is first adapted to this task, then the adapted teacher model is used to guide the adaptation of the student model. The adapted student model is used for final deployment. We demonstrate the effectiveness of our approach in few-shot image classification using model-agnostic meta-learning (MAML). Our proposed method outperforms other alternatives on several benchmark datasets.*

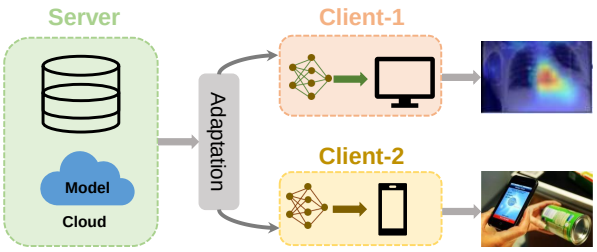

Figure 1: **Motivation of our problem**. Consider the image classification problem with a server (*e.g.* cloud vendor) and many clients. The cloud vendor has extensive computing power and training images of many classes. Each client may want to solve a client-specific image classification problem with some new classes not covered by the cloud vendor. For example, one client wants to classify different medical images, while another client wants to classify various merchandise in a particular store. Each client only has few-shot data. The cloud vendor can train some model on the server side. Due to privacy concerns, clients do not want to upload their own data to the cloud vendor. Instead, each client adapts the model from the cloud vendor to his/her specific image classification task on the client side. We would like the final model for deployment on the client side to be small.

1. Introduction

Meta-learning techniques, such as model-agnostic meta-learning (MAML) [9], have been successfully applied in many computer vision and machine learning tasks, such as few-shot learning [9], domain adaptation [24], domain generalization [25], etc. Meta-learning consists of a meta-training stage and a meta-testing stage. During meta-training, a global model is learned from a set of tasks. For example, in the case of few-shot learning (FSL), each task is a few-shot classification problem. During meta-testing, the learned global model can be adapted to a new few-shot classification problem with only few labeled examples. Meta-training is typically done on a central server where it is reasonable to assume access to extensive computational resources. In contrast, meta-testing is presumably done by an end-user or client who may not have the same computational resources as the central server, especially if the application of the client needs to run on low-powered edge devices. Existing meta-learning literature has largely overlooked this gap of computational resources. Existing meta-learning (or more broadly, few-shot learning in general) approaches typically assume that the architecture of the model

*Corresponding authors: Zhi Liu and Yang Wang.

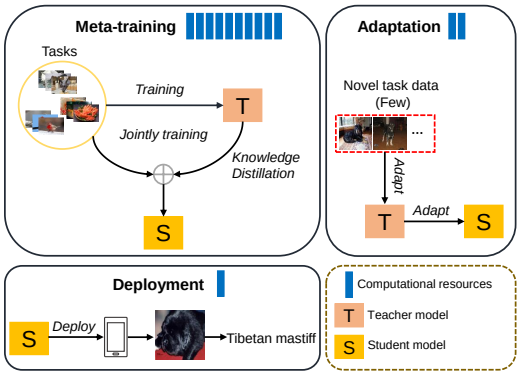

Figure 2: **Key idea of our approach**. On the server side, we jointly learn a large teacher model and a small student model in a meta-training framework. At the client side, the client first performs an adaptation stage. During this stage, the teacher model is first adapted to the task, then the adapted teacher model is used to guide the adaptation of the student model via distillation. The adapted student model is then used for final deployment. Different stages (meta-training, adaptation, deployment) of this pipeline involve different levels of computational resources.

during meta-training is the same as the one used by the client for the final deployment. In this paper, we challenge this basic assumption of existing meta-learning solutions. We propose a new problem setting that takes into account of different levels of available computational resources during meta-training and meta-testing.

Our problem setting is motivated by a practical scenario (shown in Fig. 1) consisting of a server and many clients. For example, the server can be a cloud vendor that provides pretrained image classification models (possibly via web API). On the server side, the cloud vendor may have a large image dataset with many object classes. The cloud vendor typically has access to significant computational resources to train very large models. We also have clients who are interested in solving some application-specific image classification problems. Each client may only be interested in recognizing a handful of object classes that are potentially not covered by the training dataset from the server side. For example, one client might be a medical doctor interested in recognizing different tumors in medical images, while another client might be a retail owner interested in classifying different merchandise in a store. Because of the cost of acquiring labeled images, each client may only have a small number of labeled examples for the target application. Due to privacy concerns, clients may not want to send their data to the cloud vendor. In this case, a natural solution is for a client to re-use a pretrained model provided by the cloud vendor and perform few-shot learning to adapt the pretrained model to the new object classes for the target application.

At first glance, the scenario in Fig. 1 is a classic meta-learning problem. The cloud vendor can perform meta-training on the server to obtain a global model. On the client side, the client performs two steps. The first step (called *adaptation*) is to adapt the global model from the server side to the target application. For example, the adaptation step in MAML performs a few gradient updates on the few-shot data. After the adaptation, the second step (called *deployment*) is to deploy the adapted model for the end application. The combination of these two steps (adaptation and deployment) is commonly known as "meta-testing" in meta-learning. In this paper, we make a distinction between adaptation and deployment since this distinction is important for motivating our problem setting. Our key observation is that the available computing resources are vastly different in these different stages. The meta-training stage is done on a server or the cloud with significant computing resources. The adaptation stage is often done on a client's local machine with moderate computing powers (e.g., desktop or laptop). For the deployment, we may only have access to very limited computing power if the model is deployed on an edge device. If we want to use classic meta-learning in this case, we have to choose a small model architecture to make sure that the final model can be deployed on the edge device. Unfortunately, previous work [27, 34] has shown that a small model may not have enough capacity to consume the information from a large amount of data available during meta-training, so the learned model may not effectively adapt to a new task.

In this paper, we propose a new approach called *task-specific meta distillation* to solve this problem. The key idea of our approach is illustrated in Fig. 2. During meta-training, we simultaneously learn a large teacher model and a small student model. Since the teacher model has a larger capacity, it can better adapt to a new task. During meta-training, these two models are jointly learned in a way that the teacher model can effectively guide the adaptation of the student model. During the adaptation step of meta-testing, we first adapt the teacher model to the target task, then use the adapted teacher to guide the adaptation of the student model via knowledge distillation. Finally, the *adapted* student model is used for the final *deployment*. In this paper, we apply our proposed approach to improve few-shot image classification with MAML. But our technique is generally applicable in other meta-learning tasks beyond few-shot learning.

The contributions of this work are manifold. First, previous work in meta-learning has largely overlooked the issue of the computational resource gap at different stages of meta-learning. This issue poses challenges in the real-world adoption of meta-learning applications. In this pa-

per, we consider the problem of few-shot learning of compact models to address this gap. Second, we propose a meta-learning approach that jointly learns a large teacher model and a small student model. During meta-testing, the adapted teacher model is used to distill task-specific knowledge for adapting the student model. The adapted student is used for final deployment. Finally, we apply the proposed approach to improve MAML-based few-shot image classification. Our proposed method significantly outperforms vanilla MAML. Although we focus on few-shot learning with MAML in this paper, the proposed method is generally applicable to other meta-learning tasks.

2. Related Work

In this section, we review several lines of research relevant to our work.

Knowledge distillation. Knowledge Distillation (KD) [38, 41, 51, 12, 40, 46, 23, 1, 47, 39, 15, 26, 32] is a widely used technique for model compression. KD aims to transfer the knowledge from a large model (called *teacher*) to a small model (called *student*). Most KD methods optimize a loss function that captures some form of dis-similarity between the teacher model and the student model, so that the student model is learned to mimic the teacher model. Hinton et al., [14] introduce a KD method by defining a divergence loss between the soft outputs of the teacher model and the student model. Li et al., [26] propose a similar approach. In addition to soft outputs, there have been KD methods based on various other information, such as intermediate-layer features [41], attention maps [51], hidden neuron activations [12], relationship of samples [38, 40, 47], mutual information [1], correlation and higher-order dependencies [46], etc.

Few-shot learning and meta-learning. The goal of few-shot learning (FSL) is to quickly learn new concepts from only a small number of training data. For example, in few-shot image classification, we are presented with some new classes that never appear during training. Our goal is to learn a model that can recognize those new classes when we only have very few training examples for each class. Meta-learning [44, 9, 45, 48, 7, 22, 8, 17, 49, 3] has been a popular approach for few-shot image classification. In meta-learning, the model is learned from a set of *tasks* during training, where each task is a few-shot image classification problem. For a new few-shot image classification task, the model is adapted to this new task using few-shot training examples of the task. Model-agnostic meta-learning (MAML) [9] learns a good initialization of the model, so that the model can adapt to a new few-shot classification task with only a few gradient updates. Prototypical Network [44] learns a metric space so that classification can be achieved by computing distances to class prototypes. Matching Network [48] learns a model to map a small labeled support set and an unlabeled example to its label. Relation network [45] uses a similar idea. MetaOptNet [22] learns representations with a discriminatively trained linear predictor. MeTAL [3] learns to adapt to various tasks via a loss function instead of hand-designing an auxiliary loss.

Meta-learning for knowledge distillation. Recently, there has been work [54, 37, 52, 16, 28, 2, 30, 37, 43, 29, 31, 53] on combining knowledge distillation and meta-learning. Most of these works focus on using meta-learning ideas to improve knowledge distillation. MetaDistill [54] uses meta-learning to learn a better teacher model that is more effective to transfer knowledge to a student model. Instead of fixing the teacher, the method uses the feedback from the student model to improve the teacher model. Meta-KD [37] uses a meta-teacher model to capture transferable knowledge across domains, then uses it to transfer knowledge to students. MetaDistille [30] generates better soft targets by using a label generator to fuse the feature maps. It uses meta-learning to optimize the label generator. Jang et al., [16] use meta-learning to learn what knowledge to transfer from the teacher model to the student model. Meta-DMoE [53] incorporates Mixture-of-Experts (MoE) as teachers to address multiple-source domains shift. The above-mentioned works improve knowledge distillation using meta-learning, but they do not consider few-shot learning. There are some recent works [2, 28, 43] on learning the student model with few-shot examples using KD. But they only consider using few-shot examples of *known* classes for learning the student model. They do not consider the few-shot learning of new classes. Lim et al., [29] propose to improve Efficient-PrototypicalNet performance with self knowledge distillation. Different from KD, the teacher and student models for self knowledge distillation have same network structure. Liu and Wang [31] propose a model that learns representation through online self-distillation. They introduce a special data augmentation-CutMix [50] to improve few-shot learning performance.

To the best of our knowledge, ours is the first work on using KD with meta-learning for few-shot learning.

3. Preliminary and Background

In this section, we briefly introduce some background knowledge and terminology relevant to our work.

Knowledge distillation. The goal of knowledge distillation is to transfer the knowledge of a large teacher model f_ψ parameterized by ψ when training a small student model g_θ parameterized by θ. Given a labeled dataset D and a pretrained teacher model f_ψ, we can learn the student model g_θ by optimizing the following loss function:

$$\min_\theta \mathcal{L}_S(\theta; D) + \mathcal{L}_{KD}(\psi, \theta; D) \tag{1}$$

where $\mathcal{L}_S(\cdot)$ is the standard classification loss (e.g., cross-entropy) of the student model on D and $\mathcal{L}_{KD}(\cdot)$ is a knowl-

edge distillation (KD) loss. The KD loss is used to transfer the knowledge from the teacher to the student. It is typically defined as some form of dis-similarity between the teacher model and the student model. The original work [14] on KD defines this similarity using the KL divergence between the outputs of the teacher model and the student model. This KD loss is applicable only when the teacher model and the student model have the same label space (i.e., the set of class labels in classification). In the literature, there are other KD losses using other information to measure this dis-similarity, including feature maps [41], mutual relations of data examples [38], attention distribution [51], etc. These KD losses are applicable even when the teacher model and the student model predict different sets of class labels. Our approach is a general framework and can be used together with any KD loss.

MAML for few-shot learning. Meta-learning is widely used for solving few-shot learning problems [44, 9, 48]. Our proposed method is built upon the model-agnostic meta-learning (MAML) [9] which is one of the most popular meta-learning approaches. MAML consists of meta-training and meta-testing. In meta-training, MAML learns a model f_θ parameterized by θ from a set of tasks. In few-shot classification, each training task $\mathcal{T}$ corresponds to a few-shot classification problem. Let $p(\mathcal{T})$ be a distribution over tasks and $\mathcal{T}_i \sim p(\mathcal{T})$ be a sampled task during meta-training. The task $\mathcal{T}_i$ has its own training set D_i^{tr} (also known as the *support set*) and validation set D_i^{val} (also known as the *query set*). The support set only contains a small number of training examples. Given the model parameter θ, MAML obtains a task-adapted model parameter θ_i' by taking a few gradient updates using the loss on D_i^{tr}

$$\theta_i' \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}\left(f_\theta; D_i^{tr}\right) \tag{2}$$

where α is the learning rate. Eq. 2 corresponds to the *inner update* in MAML.

In the *outer update* of MAML, we update the model parameters θ based on the loss of the task-adapted model on D_i^{val} over training tasks

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}; D_i^{val}\right) \tag{3}$$

where β is the learning rate. In Eq. 3, $\mathcal{L}_{\mathcal{T}_i}$ is based on the task-adapted model θ_i', but the gradient update is performed on the model parameter θ.

The essence of MAML is to learn the initial model parameter θ, so that it can effectively adapt to a new task given a small number of training examples of that task.

4. Our Approach

In this section, we elaborate our proposed approach. An overview of our approach is shown in Fig. 3.

Task-specific meta distillation. Let $f_\psi(\cdot)$ be the teacher model parameterized by ψ and $g_\theta(\cdot)$ be the student model parameterized by θ. Given a task $\mathcal{T}_i = (D_i^{tr}, D_i^{val})$ where D_i^{tr} is the support set and D_i^{val} is the query set of this task, we use $\mathcal{L}_T(\psi; D_i^{tr})$ to denote the classification cross-entropy loss of the teacher model f_ψ on D_i^{tr}, i.e.,

$$\mathcal{L}_T(\psi; D_i^{tr}) = \sum_{(x_j, y_j) \in D_i^{tr}} \ell_{CE}(f_\psi(x_j), y_j) \tag{4}$$

where $\ell_{CE}(\cdot)$ is the cross-entropy loss between the predicted class and the ground-truth. We similarly define the classification loss $\mathcal{L}_S(\theta; D_i^{tr})$ of the student model g_θ on D_i^{tr} as:

$$\mathcal{L}_S(\theta; D_i^{tr}) = \sum_{(x_j, y_j) \in D_i^{tr}} \ell_{CE}(g_\theta(x_j), y_j) \tag{5}$$

In task-specific meta distillation (TSMD), our goal is to adapt (ψ, θ) to the task $\mathcal{T}_i$ by performing a small number of gradient updates to change from (ψ, θ) to (ψ_i', θ_i') as follows. First we update the teacher model ψ to a task-adapted model ψ_i' according to the classification loss of the teacher model:

$$\psi_i' \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_T(\psi; D_i^{tr}) \tag{6}$$

where α is the learning rate. We then update the student model θ to a task-adapted model θ_i' by transferring the knowledge from the *adapted* teacher model ψ_i'. The motivation is that since the teacher model ψ has a higher capacity, the adapted teacher model ψ_i' is likely to provide useful knowledge for guiding the student model update (see Fig. 4). The update of the student model can be written as:

$$\theta_i' \leftarrow \theta - \lambda \nabla_\theta \Big(\mathcal{L}_S(\theta; D_i^{tr}) + \mathcal{L}_{KD}(\psi_i', \theta; D_i^{tr}) \Big) \tag{7}$$

where λ is the learning rate. In Eq. 7, $\mathcal{L}_{KD}(\psi_i', \theta; D_i^{tr})$ is a KD loss between the updated teacher model ψ_i' and the student model θ. Our proposed method can work with any well-defined KD loss.

It is worth noting the key difference between TSMD and standard KD. In standard KD, we assume that the teacher model is effective and we train the student model to emulate the teacher. In TSMD, we do not necessarily assume the teacher model to be effective for a new task. Instead, we assume that the teacher model has the capacity to successfully *adapt* to a new task given a small amount of data. Then the student model can emulate the *adapted teacher* model (not the original teacher model).

Meta-training. The updated teacher ψ_i' and student θ_i' have been specifically tailored to the task $\mathcal{T}_i$. Intuitively, we would like them to perform well on the query set D_i^{val} of

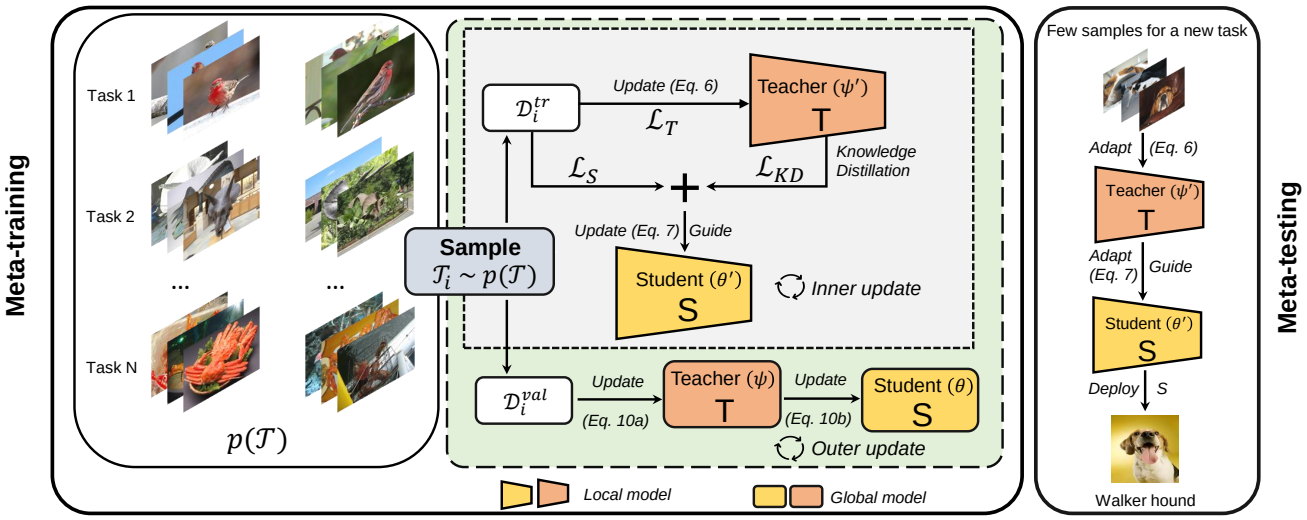

Figure 3: **Overview of our approach**. Similar to MAML [9], our approach has a meta-training stage and a meta-testing stage. During meta-training, we jointly learn a teacher model ψ and a student model θ. In each iteration of meta-training, we sample a batch of tasks, where each task $\mathcal{T}_i$ is a few-shot classification problem with its own support set $\mathcal{D}_i^{tr}$ and query set $\mathcal{D}_i^{val}$. In the inner update of meta-training, we obtain an updated task-specific teacher ψ_i' via gradient update on $\mathcal{D}_i^{tr}$. We then use ψ_i' to guide the adaptation of the task-specific student θ_i'. In the outer loop, the global models ψ and θ are updated based on the performance of the task-specific models on $\mathcal{D}_i^{val}$ across sampled tasks. During meta-testing, we use a procedure similar to the inner update of meta-training to obtain the adapted model for the new task.

this task. We measure their performance on D_i^{val} as:

$$\mathcal{L}_T(\psi_i'; D_i^{val}) = \sum_{(x_j, y_j) \in D_i^{val}} \ell_{CE}(f_{\psi_i'}(x_j), y_j) \quad (8a)$$

$$\mathcal{L}_S(\theta_i'; D_i^{val}) = \sum_{(x_j, y_j) \in D_i^{val}} \ell_{CE}(g_{\theta_i'}(x_j), y_j) \quad (8b)$$

The goal of meta-learning is to learn the initial models (ψ, θ), so that after model update using the support set (Eq. 6 and Eq. 7) of a particular task, the task-adapted models (ψ_i', θ_i') will minimize the loss defined in Eq. 8 on the corresponding query set across all tasks. The meta-objective can be defined as:

$$\min_{\psi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_T(\psi_i'; D_i^{val}), \quad \min_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_S(\theta_i'; D_i^{val}) \quad (9)$$

The meta-objective in Eq. 9 involves summing over all meta-training tasks. In practice, we sample a mini-batch of tasks in each iteration during meta-training. Note that we do not need any KD loss in the meta-objective in Eq. 9. The KD loss is only used in the inner update of meta-training.

The initial models (ψ, θ) are learned by optimizing the meta-objective (Eq. 9) using stochastic gradient descent as:

$$\psi \leftarrow \psi - \beta \nabla_{\psi} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_T(\psi_i'; D_i^{val}) \quad (10a)$$

$$\theta \leftarrow \theta - \eta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_S(\theta_i'; D_i^{val}) \quad (10b)$$

where β and η are learning rates. Note that the losses ($\mathcal{L}_T(\cdot)$ and $\mathcal{L}_S(\cdot)$) in Eq. 10 are computed using the adapted models (ψ_i', θ_i'), but the SGD updates are performed over the model parameters (ψ, θ).

The high-level intuition of meta-training is to learn the teacher model ψ and the student model θ jointly, so that the teacher model can effectively adapt to a new task, and the student can effectively adapt to the same new task using the distilled knowledge of the task-adapted teacher model. Algorithm 1 summarizes the meta-training process.

Meta-testing. After meta-training, we obtain the model parameters (ψ, θ). During meta-testing, we have a new task $\mathcal{T} = (D^{tr}, D^{val})$ with the support set D^{tr} and the query set D^{val}. We simply use Eq. 6 and Eq. 7 to obtain the task-specific parameters (ψ', θ'). We then use the updated student model θ' for predictions on D^{val}. Note that we do not use the updated teacher model ψ' during inference, since our assumption is that the teacher model is too large to be deployed for the end application.

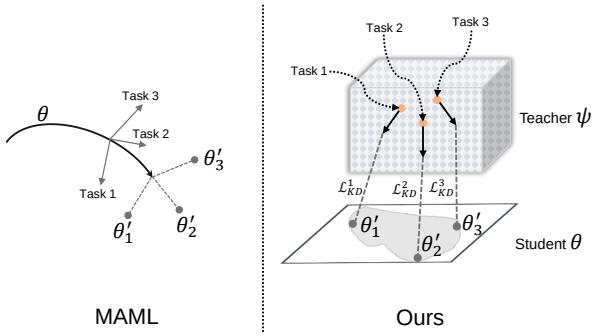

Figure 4: **Intuition of task-specific knowledge distillation.** In standard MAML, the adaptation of the model to a task only depends on the support set of the task. In our approach, we first adapt the teacher model using the support set. Since the teacher model has a higher capacity, the adapted teacher model can more effectively capture task-specific knowledge. The adapted teacher model can help guide the adaptation of the student model via the KD loss.

One might argue that since we do not use the adapted teacher model ψ' for the final inference, it is perhaps not necessary to update the teacher model in the meta update (Eq. 10a). But without Eq. 10a, the teacher model will stay the same throughout meta-training. This is clearly not desirable, especially if the teacher model is randomly initialized. In practice, we have found that it is crucial to update the teacher model using Eq. 10a to make sure that the teacher model is improved during meta-training.

Remarks. So far, we have assumed that the teacher model is updated during meta-training and meta-testing. In some applications, it might be difficult to update the teacher model. This can happen if the teacher model is a very large pretrained model (e.g., BEiT [4], GPT-3 [6]). These large models require significant resources and expertise to train. Once they are trained, it might be difficult to perform any update on these models. In some cases, the pretrained model is only provided as a callable API and the model itself is not directly accessible. Clearly we cannot update the pretrained model in these cases. In the meantime, these pretrained large models contain a significant amount of useful knowledge, so it is desirable to use them as fixed teacher models (even if these models cannot be updated) in our proposed framework. To handle this case, we simply need to slightly modify our method by omitting the update of the teacher model in Eq. 6 and Eq. 10a.

5. Experiments

In this section, we evaluate the proposed approach in MAML-based few-shot image classification on several benchmark datasets.

Algorithm 1 Meta-Training with Task-Specific Knowledge Distillation

Require: Distribution $p(\mathcal{T})$ over tasks $\mathcal{T}$
Require: Learning rates α, β, λ and η
Initialize teacher model ψ and student model θ
while not done **do**
  Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Let D_i^{tr} be the support set of $\mathcal{T}_i$ and D_i^{val} be the query set of $\mathcal{T}_i$
    Obtain adapted teacher ψ'_i via Eq. 6 on D_i^{tr}
    Obtain adapted student θ'_i via Eq. 7 on D_i^{tr}
    Evaluate meta-objective in Eq. 8 on D_i^{val}
  end for
  Update model parameters (ψ, θ) via Eq. 10
end while

5.1. Datasets and Implementation

Datasets. We evaluate our approach on the following standard benchmark datasets commonly used in few-shot image classification.

- *mini-ImageNet* [48]: This is a standard benchmark for few-shot image classification. It consists of 100 randomly chosen classes from ILSVRC-2012 [42]. These classes are randomly split into 64, 16 and 20 classes for meta-training, meta-validation and meta-testing, respectively. Each class contains 600 images of size 84×84.
- *FC100* [36]: This dataset is derived from CIFAR-100 [21]. It contains 100 classes which are grouped into 20 superclasses. These classes are split into 60 classes for meta-training, 20 classes for meta-validation, and 20 classes for meta-testing. Each class contains 600 images.
- *CIFAR-FS* [5]: This dataset contains 100 classes from CIFAR-100 [21]. The classes are randomly split into 64, 16 and 20 for meta-training, meta-validation and meta-testing, respectively. Each class contains 600 images.
- *FGVC-aircraft* [33]: This dataset contains 10,200 images of aircraft with 100 images for each of 102 different aircraft classes. The classes are randomly split into 50, 25 and 25 for meta-training, meta-validation and meta-testing, respectively. We resize images to 84×84.
- *CUB200* [13]: This dataset contains 200 classes and 11,788 images in total. Following the evaluation protocol [13], we randomly split the dataset into 100, 50 and 50 for meta-training, meta-validation and meta-testing, respectively. We resize images to 84×84.
- *Stanford dogs* [19]: This dataset contains images of 120 breeds of dogs. This dataset has been built using images and annotations from ImageNet for the task

	FC100		CIFAR-FS		mini-ImageNet	
Method	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Student	35.97±1.80	48.68±0.91	56.10±1.80	72.10±0.96	49.53±1.90	62.61±0.91
Fixed teacher	37.10±1.61	49.19±0.92	56.86±1.82	72.90±0.85	51.27±2.02	63.45±0.91
Ours	**38.33 ±1.78**	**50.71±1.05**	**57.50±1.73**	**73.23±0.93**	**51.45±1.80**	**63.80±0.85**
Oracle	44.61±1.01	59.60±0.70	72.32±0.99	85.80±0.77	55.60±0.96	66.80±0.64
	FGVC-aircraft		**CUB200**		**Stanford dogs**	
Method	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Student	48.87±1.71	64.27±0.87	52.83±1.46	67.82±0.69	37.43±1.90	52.05±1.03
Fixed teacher	51.12±1.71	64.97±0.85	53.60±1.52	68.01±0.96	37.99±1.89	52.94±0.78
Ours	**53.67±1.71**	**66.64±0.94**	**56.13±1.76**	**69.22±0.97**	**38.67±1.80**	**53.89±0.89**
Oracle	61.10±1.80	88.20±0.74	66.30±0.99	84.53±0.52	58.32±1.49	75.30±0.69

Table 1: **Experimental results**. We evaluate 1-shot and 5-shot classification on 6 benchmark datasets (mini-ImageNet, FC100, CIFAR-FS, FGVC-aircraft, CUB200 and Stanford dogs). On each dataset, We report the mean of 800 randomly generated test episodes as well as the 95% confidence intervals. Using a fixed teacher in our approach gives better performance than training the student with MAML. Our full approach achieves the best performance on all datasets.

of fine-grained image categorization. we randomly split the dataset into 60, 30 and 30 for meta-training, meta-validation and meta-testing, respectively. We also resize images to 84×84.

Implementation details. We use ResNet-50 [11] as the teacher network and a simple four-layer ConvNet (Conv-4) defined in [9] as the student network. For mini-ImageNet, FC100, and CIFAR-FS datasets, we first utilize a meta-training set to train the teacher with 160 epochs for each dataset. Then we use the parameters to initialize the teacher model ψ. For FGVC-aircraft, CUB200, and Stanford dogs datasets, we use ResNet-50 as the teacher which is initialized with ImageNet [42] pretrained weights. All methods use the Adam [20] optimizer with initial learning rate `1e-5` and `1e-3` for teacher and student, respectively. We apply standard data augmentation including random crop, random flip, and color jitter. We train our model for about 600 epochs on mini-ImageNet, and train our model with about 300 epochs for other datasets. In our experiments, we use the cross-entropy loss for classification. For transferring knowledge, we use the relation knowledge distillation (RKD) [38] loss when the teacher model is fixed, and use KL-divergence based KD loss [14] when the teacher model can be updated. We set the number of inner update in meta-training to 5. We will release our code upon publication.

5.2. Baseline and Oracle Methods

Since this paper addresses a new problem, there is no previous work that we can directly compare with. Nevertheless, we define several baseline methods and one oracle method for comparison.

MAML with student network (Student). In this baseline, we directly use MAML [9] to train the student network which uses a Conv-4 architecture.

Ours with fixed ResNet-50 as teacher (Fixed teacher). In this baseline, we use ResNet-50 [11] as the teacher model and use Conv-4 as the student model. For mini-ImageNet, FC100, and CIFAR-FS, we first train the teacher model with the meta-training set for each dataset. For FGVC-aircraft, CUB200, and Stanford dogs, we directly use the teacher model pretrained on ImageNet [42]. Then we train the student model using our approach while keeping the teacher model fixed. We use the relation knowledge distillation (RKD) loss [38] in this case.

Oracle. In our work, we use ResNet-50 [11] as the teacher network. We consider an oracle method which uses ResNet-50 with MAML. This method provides an upper bound on the performance of our approach.

	1-shot		5-shot	
Dataset	ResNet-50	BEiT	ResNet-50	BEiT
FC100	37.10±1.61	**37.90±1.42**	49.19±0.92	**50.22±0.69**
CIFAS-FS	56.86±1.82	**57.12±1.69**	**72.90±0.85**	72.62±0.87
mini-ImageNet	51.27±2.02	**51.69±1.08**	63.45±0.91	**63.80±0.81**
FGVC-aircraft	51.12±1.71	**51.30±2.00**	64.97±0.85	**65.02±0.95**
CUB200	53.60±1.52	**54.22±1.62**	68.01±0.96	**68.40±0.65**
Stanford dogs	37.99±1.89	**38.79±1.76**	52.94±0.78	**53.57±0.77**

Table 2: **Pretrained model as teacher**. We compare using an off-the-shelf pretrained model (*BEiT*) as the fixed teacher model instead of *ResNet-50* on different datasets. Using BEiT gives improved performance.

5.3. Main Results and Analysis

From the Table 1, we compare our approach with both baseline and oracle methods on six different datasets. Moreover, we compare to other state-of-the-art methods on mini-ImageNet, CIFAR-FS and FC100. The results are shown in Table 3.

As shown in Table 1, we can make several important observations. First of all, there is a significant gap between "Student" and "Oracle". Since both are trained with MAML and the only difference between them is the backbone, this performance gap confirms our hypothesis that MAML with a high capacity model can better adapt to new tasks. This also justifies "Oracle" being an upper bound on the performance. But of course, the model of "Oracle" might be too large to be deployed, e.g., when the application runs on edge devices. This is the key motivation behind our work. Second, even with a fixed teacher, our approach (i.e., "Fixed teacher") outperforms "Student". This demonstrates the value of distilling the knowledge from the teacher model even when the teacher model is not updated during meta-training. The proposed approach (i.e. "Ours") outperforms our approach (i.e. "Fixed teacher"). This demonstrates the effectiveness of adapting teacher during adaptation stage. Our final model ("Ours") gives the best performance. This shows the benefit of jointly learning the teacher model and the student model together in a unified framework. Besides,

Model	backbone	mini-ImageNet		CIFAR-FS		FC100	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML [9]	32-32-32-32	49.53 ± 1.90	62.61 ± 0.91	56.10 ± 1.80	72.10 ± 0.96	35.97 ± 1.80	48.68 ± 0.91
ProtoNet [44]	64-64-64-64	49.4 ± 0.8	68.2± 0.7	55.5 ± 0.7	72.0 ± 0.6	35.3 ± 0.6	48.6 ± 0.6
RelationNet [45]	64-96-128-256	49.3 ± 0.9	66.6± 0.7	55.0 ± 1.0	69.3 ± 0.8	-	-
MeTAL [3]	32-32-32-32	**52.63 ± 0.37**	**70.52± 0.39**	56.85 ± 0.29	73.10± 0.36	**39.32 ±0.33**	50.36 ± 0.30
Ours (fixed teacher)	32-32-32-32	51.27 ± 2.02	63.45± 0.91	56.86 ± 1.82	72.90± 0.85	37.10 ±1.61	49.19 ± 0.92
Ours	32-32-32-32	51.45 ± 1.80	63.80± 0.85	**57.50 ± 1.73**	**73.23± 0.93**	38.33 ±1.78	**50.71 ± 1.05**

Table 3: **Comparison results.** We compare our approach to other state-of-the-arts on mini-ImageNet, CIFAR-FS and FC100.

Method	1-shot	5-shot
Student	46.59±0.32	62.10±0.41
Fixed teacher	48.16±0.28	63.22±0.33
Ours	**49.59±0.28**	**64.35±0.39**
Oracle	55.12±0.36	66.30±0.44

Table 4: **Experimental results**. We evaluate 1-shot and 5-shot classification on mini-ImageNet with Reptile method.

we obtain the similar tendencies between 1-shot and 5-shot.

Table 3 shows the results compared with other state-of-the-art methods with respect to a standard 4-layer convolutional network. We can observe our proposed approach achieves good results compared with other methods, such as MeTAL [3] and RelationNet [45]. MeTAL takes more benefits from a transductive setting that all query samples are available at once. We think sometimes this setting is not practical in real-world applications.

5.4. Ablation Studies

We perform various ablation studies to gain further insights of our approach.

Pretrained model as teacher. For results in Sec. 5.3, the teacher model is ResNet-50. An interesting question is whether it is possible to use some other large-scale pretrained model as the teacher. This is motivated by the current trend in the AI community to pretrain increasingly larger models (e.g., GPT-3 [6], BEiT [4], BERT [18] and SEER [10]) from very large-scale data via self-supervised learning. Due to the sheer size of these models, very few organizations in the world have the resources to train these models. However, once these models are trained, they can be very useful for downstream tasks since these models contain rich knowledge learned from large-scale data. An intriguing question is whether it is possible to use these large pretrained models in meta-learning. Due to the large size of these models, so far there is little work on using these models in meta-learning.

Our proposed approach can be easily modified to use a large off-the-shelf pretrained model as the teacher model. To demonstrate this, we choose a self-supervised model (BEiT) [4] as a fixed teacher model in our framework. The results are shown in Table 2. Here we compare using fixed BEiT versus fixed ResNet-50 as the teacher model in our approach. We can see that using pretrained BEiT performs better in most cases. Due to the model size, it is difficult to directly use BEiT as the backbone architecture in MAML. Our approach provides an alternative effective way of using large pretrained models in MAML.

Reptile-based meta-learning. Our proposed approach is based on MAML [9]. In the literature, there are other different variants of meta-learning, e.g. Reptile [35]. Table 4 show results on mini-ImageNet with Reptile instead of MAML. Again, our approach outperforms other baselines. This shows that our approach is agnostic to the specific choice of the meta-learning method.

For more ablation studies, please refer to the supplementary document.

6. Conclusions

We have introduced a new problem called task-specific knowledge distillation in meta-learning. Unlike traditional meta-learning where the same model architecture is used in meta-training and meta-testing, our approach jointly learns a teacher model and a student model in the meta-learning framework. The adapted teacher model is then used to distill task-specific knowledge to guide the adaptation of the student model. We have demonstrated our approach in few-shot image classification with MAML. Our proposed method significantly outperforms vanilla MAML. Our proposed method is general and can be used to improve many meta-learning applications.

Limitations. Since our approach needs to maintain both teacher and student models, the training is computationally more expensive than learning the student model alone with MAML. In this paper, we have only considered some simple KD methods. As future work, we would like to explore more advanced KD methods in our approach.

Acknowledgements. This work is partly supported by the National Natural Science Foundation of China under Grant 62171269, the China Scholarship Council under Grant 202006890081 and a grant from NSERC.

References

[1] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D. Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[2] Haoli Bai, Jiaxiang Wu, Irwin King, and Michael Lyu. Few shot network compression via cross distillation. In *AAAI Conference on Artificial Intelligence*, 2020.

[3] Sungyong Baik, Janghoon Choi, Heewon Kim, Dohee Cho, Jaesik Min, and Kyoung Mu Lee. Meta-learning with task-adaptive loss function for few-shot learning. In *IEEE/CVF International Conference on Computer Vision*, pages 9465–9474, 2021.

[4] Hangbo Bao, Li Dong, and Furu Wei. BEiT: BERT pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

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

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[7] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

[8] Zhixiang Chi, Li Gu, Huan Liu, Yang Wang, Yuanhao Yu, and Jin Tang. Metafscil: A meta-learning approach for few-shot class incremental learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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

[10] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, and Piotr Bojanowski. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[12] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI Conference on Artificial Intelligence*, 2019.

[13] Nathan Hilliard, Lawrence Phillips, Scott Howland, Artëm Yankov, Courtney D. Corley, and Nathan O. Hodas. Few-shot learning with metric-agnostic conditional embeddings. *arXiv preprint arXiv:1802.04376*, 2018.

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

[15] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv preprint arXiv:1707.01219*, 2017.

[16] Yunhun Jang, Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Learning what and where to transfer. In *International Conference on Machine Learning*, 2019.

[17] Huaizu Jiang, Xiaojian Ma, Weili Nie, Zhiding Yu, Yuke Zhu, and Anima Anandkumar. Bongard-hoi: Benchmarking few-shot visual reasoning for human-object interactions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[18] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[19] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

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

[21] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

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

[23] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. Self-supervised knowledge distillation using singular value decomposition. In *European Conference on Computer Vision*, 2018.

[24] Da Li and Timothy Hospedales. Online meta-learning for multi-source and semi-supervised domain adaptation. In *European Conference on Computer Vision*, 2020.

[25] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI Conference on Artificial Intelligence*, 2018.

[26] Jinyu Li, Rui Zhao, Jui-Ting Huang, and Yifan Gong. Learning small-size dnn with output-distribution-based criteria. In *Interspeech*, 2014.

[27] Peng Li, Changyong Shu, Yuan Xie, Yanyun Qu, and Hui Kong. Hierarchical knowledge squeezed adversarial network compression. In *AAAI Conference on Artificial Intelligence*, 2020.

[28] Tianhong Li, Jianguo Li, Zhuang Liu, and Changshui Zhang. Few sample knowledge distillation for efficient network compression. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[29] Jit Yan Lim, Kian Ming Lim, Shih Yin Ooi, and Chin Poo Lee. Efficient-prototypicalnet with self knowledge distillation for few-shot learning. *Neurocomputing*, 2021.

[30] Benlin Liu, Yongming Rao, Jiwen Lu, Jie Zhou, and Cho jui Hsieh. Metadistiller: Network self-boosting via meta-learned top-down distillation. In *European Conference on Computer Vision*, 2020.

[31] Sihan Liu and Yue Wang. Few-shot learning with online self-distillation. In *IEEE/CVF International Conference on Computer Vision Workshops*, 2021.

[32] Yufan Liu, Jiajiong Cao, Bing Li, Chunfeng Yuan, Weiming Hu, Yangxi Li, and Yunqiang Duan. Knowledge distillation via instance relationship graph. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

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

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

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

[36] Boris N. Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in Neural Information Processing Systems*, 2018.

[37] Haojie Pan, Chengyu Wang, Minghui Qiu, Yichang Zhang, Yaliang Li, and Jun Huang. Meta-KD: A meta knowledge distillation framework for language model compression across domains. In *Annual Meeting of the Association for Computational Linguistics*, 2021.

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

[39] Nikolaos Passalis, Maria Tzelepi, and Anastasios Tefas. Probabilistic knowledge transfer for lightweight deep representation learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[40] Baoyun Peng, Xiao Jin, Dongsheng Li, Shunfeng Zhou, Yichao Wu, Jiaheng Liu, Zhaoning Zhang, and Yu Liu. Correlation congruence for knowledge distillation. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[41] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. In *International Conference on Learning Representations*, 2015.

[42] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2015.

[43] Chengchao Shen, Xinchao Wang, Youtan Yin, Jie Song, Sihui Luo, and Mingli Song. Progressive network grafting for few-shot knowledge distillation. In *AAAI Conference on Artificial Intelligence*, 2021.

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

[45] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[46] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. In *International Conference on Learning Representations*, 2020.

[47] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *IEEE/CVF International Conference on Computer Vision*, 2019.

[48] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems*, 2016.

[49] Vibashan VS, Domenick Poster, Suya You, Shuowen Hu, and Vishal M. Patel. Meta-uda: Unsupervised domain adaptive thermal object detection using meta-learning. In *IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022.

[50] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.

[51] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.

[52] Min Zhang, Donglin Wang, and Sibo Gai. Knowledge distillation for model-agnostic meta-learning. In *European Conference on Artificial Intelligence*, 2020.

[53] Tao Zhong, Zhixiang Chi, Li Gu, Yang Wang, Yuanhao Yu, and Jin Tang. Meta-dmoe: Adapting to domain shift by meta-distillation from mixture-of-experts. In *Neural Information Processing System*, 2022.

[54] Wangchunshu Zhou, Canwen Xu, and Julian J. McAuley. Meta learning for knowledge distillation. *arXiv preprint arXiv:2106.04570*, 2021.