Task-aware Adaptive Learning for Cross-domain Few-shot Learning

Yurong Guo[1], Ruoyi Du[1], Yuan Dong[1], Timothy Hospedales[2], Yi-Zhe Song[3], Zhanyu Ma[1]*
[1]Beijing University of Posts and Telecommunications, China
[2]University of Edinburgh, UK [3]University of Surrey, UK
{guoyurong, duruoyi, yuandong, mazhanyu}@bupt.edu.cn,
t.hospedales@ed.ac.uk, y.song@surrey.ac.uk

Abstract

Although existing few-shot learning works yield promising results for in-domain queries, they still suffer from weak cross-domain generalization. Limited support data requires effective knowledge transfer, but domain-shift makes this harder. Towards this emerging challenge, researchers improved adaptation by introducing task-specific parameters, which are directly optimized and estimated for each task. However, adding a fixed number of additional parameters fails to consider the diverse domain shifts between target tasks and the source domain, limiting efficacy. In this paper, we first observe the dependence of task-specific parameter configuration on the target task. Abundant task-specific parameters may over-fit, and insufficient task-specific parameters may result in under-adaptation – but the optimal task-specific configuration varies for different test tasks. Based on these findings, we propose the Task-aware Adaptive Network (TA^2-Net), which is trained by reinforcement learning to adaptively estimate the optimal task-specific parameter configuration for each test task. It learns, for example, that tasks with significant domain-shift usually have a larger need for task-specific parameters for adaptation. We evaluate our model on Meta-dataset. Empirical results show that our model outperforms existing state-of-the-art methods. Our code is available at https://github.com/PRIS-CV/TA2-Net.

1. Introduction

Traditional deep learning models [42, 36, 12, 17] show excellent generalization performance when training on a large number of labeled samples. However, both abundant samples and reliable annotations are not always available in realistic applications, *e.g.*, rare disease diagnosis, and fine-grained recognition. Few-shot learning [32, 33, 23, 4, 10], inspired by the fact that humans can quickly learn new knowledge, aims to adapt the model to new classes by only a few labeled samples for each one.

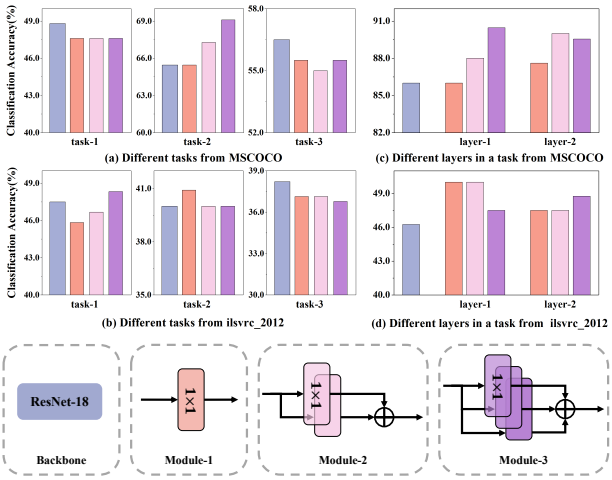


Figure 1. (a) and (b) reveal the accuracy of varying test tasks when inference was performed in different models, in which TA-Modules are attached to each layer in the backbone network. (c) and (d) show the accuracy when TA-Modules are attached to different layers in the backbone network. Four dashed boxes show the backbone and several TA-Modules.

Recently, meta-learning-based few-shot learning methods have made great strides in the setting where the train and test tasks are sampled from the same domain [49, 22]. However, more and more studies have demonstrated that these existing works fail to generalize well to novel classes that are heterogeneous with the source domain [45, 9, 14]. This limitation is attributable to the fact that most previous few-shot learning models focus solely on how to quickly adapt to a set of novel classes with only a few labeled samples per class but less effort has been made to understand and address the domain shift problem between the target (test) task and training domain. Recently, to address this problem, a series of research referred to as cross-domain few-shot learning are proposed [45, 14, 35, 3, 7, 20, 26, 6].

Generally, these existing methods address the domain

*Corresponding Author

shift problem by introducing a set of task-specific parameters to enable the feature extractor or classifier to adapt to the feature distributions of new tasks. According to the parameter generation method, these methods can be loosely divided into the following two main streams. One kind of approach is to use an auxiliary network [35, 3, 43] to generate task-specific parameters. The auxiliary network is first meta-trained with multiple tasks from the source domain, and then it utilizes the support set to estimate task-specific parameters for the target task. Other methods [21] directly attach a set of task-specific parameters to the pre-trained model on the source domain and then optimize them on a few labeled samples of the target task to well generalize the knowledge to new classes. Although cross-domain few-shot learning performance has improved, the fixed number of task-specific parameters may be too rigid to account for diverse tasks, resulting in less generality.

In this paper, from the perspective of parameter size, we studied the performance of test tasks when inference was performed in models equipped with varying sizes of task-specific parameters. Specifically, following the trend of introducing directly optimizing task-specific parameters [21], we designed several Task-specific Adapters (TA-Modules) with increasing parameters, which are respectively attached to the *conv* of pre-trained backbone network ($e.g.$, ResNet-18) to construct cross-domain models. As shown in Figure 1, the optimal task-specific parameters policy varies depending on the target task. More specifically, for different test tasks, the optimal task-specific parameters required in different layers of the backbone network for effectively adapting pre-learned knowledge to the target task's feature distribution differ. Abundant task-specific parameters may over-fit on target tasks, whereas few task-specific parameters will show the under-adaptation problem. In order to effectively direct the feature extractor to adapt feature distributions, we must carefully design task-specific parameters in each layer for the target task.

Motivated by the findings, for improving the recognition performance of cross-domain few-shot learning, we propose the Task-aware Adaptive Network (TA^2-Net) to learn the optimal task-specific parameters policy for target tasks adaptively. As shown in Figure 2, our TA^2-Net model consists of an Action Generation Network ("Agent") trained by Reinforcement Learning to generate "actions" – adaptive Task-specific Adapter execution decisions aimed at the target task, and an Adapted Network built on the "actions" to infer the target task, meanwhile, which can be viewed as the "Environment" to provide a "reward" for the Action Generation Network. The TA^2-Net can learn optimally adapted network graphs aimed at the target task to transfer pre-learned knowledge efficiently.

In summary, our contributions are:

(i) We propose the Task-aware Adaptive Network (TA^2-Net), which can adaptively learn optimal task-specific parameters policy for each target task.

(ii) We evaluate our model on the Meta-dataset. Empirical results show that our model obviously outperforms existing state-of-the-art methods.

(iii) We further analyze parameter distribution and discover that domains with significant distribution shift (compared with the source domain) usually have a larger demand for task-specific parameters to effectively adapt pre-learned knowledge. Furthermore, layers in deep blocks necessitate more task-specific parameters than those in shallow blocks for learning task-specific features.

2. Related Work

Cross-domain Few-shot Learning Existing cross-domain few-shot learning methods can be coarsely grouped into two categories: feature-selection-based methods and adaptation-based methods.

Several feature-selection-based methods, including SUR [7], URT [26], and URL [20], aim to achieve generalization by adaptively integrating feature representations from multiple training domains. SUR [7] and URT [26] rely on attention mechanisms to select appropriate domain-specific representations for a given few-shot learning task. While these methods have shown good performance, they require multiple forward networks during inference time. In contrast, Li et al. [20] proposed URL, an efficient method that distills knowledge from multiple domain networks to a single feature extractor. However, supervised training of multiple domains can result in supervision collapse, leading to the loss of information required for transfer to new tasks or domains. To address this issue, Doersch et al. [6] proposed using self-supervised learning to encourage the learning of general features from multiple training domains.

The goal of adaptation-based methods is to adapt a feature extractor and classifier learned from a source domain to a target task through task-specific parameters. CNAPS [35] employed an adaptation network to generate task-specific parameters for each hidden layer of the feature extractor and classifier to transfer knowledge. Simple CNAPS [3] improved upon this approach by replacing the parametric classifier with a Mahalanobis distance-based distance metric, which increased classification accuracy while reducing the number of parameters. Triantafillou et al.[43] proposed the FLUTE, which learns domain-specific parameters using FiLM layers called universal templates. Li et al.[21] have demonstrated that attaching task-specific weights to a pre-trained model and optimizing them from scratch directly on a small support set can enable good generalization to the target task. However, the shifts of feature distribution between different target tasks and the pre-learned domain vary, leading to the change of task-specific parameters demand. In this paper, we focus on adaptively generating task-specific

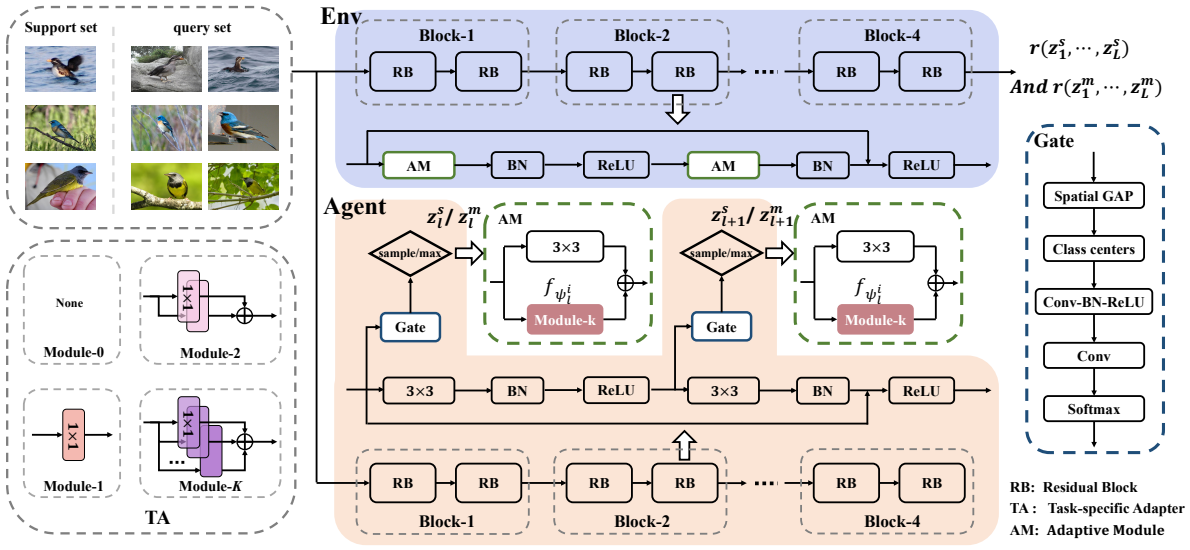


Figure 2. Task-aware Adaptive Network (TA^2-Net). The "Agent" is the Action Generation Network, which predicts the "action" of Task-specific Adapter execution and interacts with the "Environment" (Env, the Adapted Network) providing the "reward" for the Agent. "none" represent that the model does not need the Task-specific Adapter to adapt feature.

graphs for each target task, to adapt the prior knowledge to the target task well.

Reinforcement Learning Reinforcement learning [1, 8] is primarily used to solve the sequence decision problem, in which the reward value obtained from agent-environment interaction is used as a feedback signal to continuously optimize agent strategies. There are three main approaches to solving Reinforcement Learning (RL) problems: methods based on value functions [30, 46, 48], methods based on policy search [41, 40], and the actor-critic approach [15, 39]. The deep reinforcement learning algorithm based on value functions approximates the value function or action-value function with a deep neural network and updates it with the temporal difference (TD) learning or Q-learning method. Among them, Deep Q-Network (DQN) [30] algorithm is representative. Furthermore, a series of variants have been proposed, including double DQN (DDQN) [46], which optimizes the problem of overestimation about Q value in the DQN algorithm, and Dueling DQN [48], which improves DQN from the network structure and can estimate Q function more accurately. Policy-based methods [41, 40] directly search for an optimal policy without maintaining a value function model. Parameters of the chosen policy are updated using either gradient-based or gradient-free optimization to maximize the expected return. The actor-critic approach [15] employs both value functions and policy search to trade-off the variance reduction of policy gradients and bias introduction from value function methods, in which the "actor" (policy) learns by using feedback from the "critic" (value function) [1]. In this paper, we employ policy-gradient-based Reinforcement Learning to train the Action Generation Network to adaptively generate optimal actions – Task-specific Adapter execution decisions for each target task.

3. Method

In this section, we describe the problem setting and introduce our method.

3.1. Preliminary

Few-shot task Few-shot learning classification aims to classify a large number of samples with only a small number of labeled ones in each class. Labeled samples are generally referred to as the support set $S = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{N_S}$, while classified ones are known as the query set $Q = \{(\boldsymbol{x_i})\}_{i=1}^{N_Q}$. y_i is the label of the sample $\boldsymbol{x_i}$. A few-shot task is created by combining the support and query sets.

Cross-domain few-shot learning Compared with traditional few-shot learning, cross-domain few-shot learning is more challenging, since not only the labeled samples in the target task D_t are few, but also the distribution of the target domain and the base domain D_b may be highly heterogeneous. Therefore, we must strike a balance between model capacity and the reliability of the adaptation while avoiding over-fitting to the few labeled samples or under-adaptation for the target task.

3.2. Task-aware Adaptive Network (TA^2-Net)

Inspired by the observations that varying tasks require variant task-specific parameters for effective adaptation, we design the TA^2-Net adaptively learning the optimally

adapted network graph for each target task. As shown in Figure 2, the TA^2-Net consists of Action Generation Network and Adapted Network. To directly optimize the Action Generation Network to generate optimal task-specific parameters for the Adapted Network, we cast the model in Reinforcement Learning. In particular, Action Generation Network can be viewed as an "Agent" that interacts with the 'Environment" (Adapted Network). The Action Generation Network will result in the "actions" that are the prediction of Task-specific Adapter execution for the target task. Based on the generated "actions", we can build the unique Adapted Network, which will provide a "reward" for the agent – we denote this reward by r. We will introduce the Action Generation Network, Adapted Network, and their optimization process.

3.2.1 Action Generation Network

Our focus in the Action Generation Network is to generate adaptive Task-specific Adapter execution decisions for each layer of the Adapted Network aimed at the target task. As shown in Figure 2, the Gate Network is integrated into each layer of the feature encoder $f_\phi(\cdot)$ pre-trained by the seen domain to generate the discrete probability for K kinds of Task-specific Adapters. Concretely, the discrete probability in l-th layer can be formulated as $\left\{z_l^i\right\}_{i=1}^K$. Let $\boldsymbol{h_{l-1}} \in R^{N_S \times C \times W \times H}$ denotes the the input feature of l-th feature encoder. In the Gate Network, the input feature is first fed to the Global Average Pooling (GAP) to compress the spatial dimension. Formally, the output $\boldsymbol{u_{l-1}}$ can be achieved as

$$\boldsymbol{u_{l-1}} = GAP\left(\boldsymbol{h_{l-1}}\right), \tag{1}$$

where $\boldsymbol{u_{l-1}}$ can be written as $\boldsymbol{u_{l-1}} = \left[\boldsymbol{u_{l-1}^1}, \boldsymbol{u_{l-1}^2}, \cdots, \boldsymbol{u_{l-1}^{N_S}}\right]$. $\boldsymbol{u_{l-1}^i}$ represents the channel-level feature embedding of support sample i and $\boldsymbol{u_{l-1}^i} \in R^C$. Then, let $\boldsymbol{v_{l-1}} = \left[\boldsymbol{v_{l-1}^1}, \boldsymbol{v_{l-1}^2}, \cdots, \boldsymbol{v_{l-1}^N}\right]$ denotes the class prototypes in the target task, where $\boldsymbol{v_{l-1}^n}$ refers to the class center of class n and N denotes the number of classes in the target task. Formally, the class center for class n can be obtained by computing the mean vector of all embedded features in the class as

$$\boldsymbol{v_{l-1}^n} = \frac{1}{|S_n|} \sum_{i \in S_n} \boldsymbol{u_{l-1}^i}, \tag{2}$$

where S_n denotes the set of examples labeled with class n. The output $\boldsymbol{v_{l-1}}$ is a collection of all class descriptors with statistics that express the entire target task. To capture the dependencies between features, we add a simple non-linear function of two fully-connected layers coupled with a ReLU activation function. The output of this operation is a vector containing unnormalized scores for executing the K kinds of Task-specific Adapters.

$$\bar{\boldsymbol{z}}_{\boldsymbol{l}} = \mathbf{W_l^2} \sigma\left(\mathbf{W_l^1} \boldsymbol{v_{l-1}}\right), \tag{3}$$

where $\bar{\boldsymbol{z}}_{\boldsymbol{l}} \in R^K$. σ refers to the *ReLU* activation function. $\mathbf{W_l^1} \in R^{d \times C}$, $\mathbf{W_l^2} \in R^{K \times d}$ and d is the dimension of the hidden layer. We can write $\bar{\boldsymbol{z}}_{\boldsymbol{l}}$ as $\bar{\boldsymbol{z}}_{\boldsymbol{l}} = \left[\bar{z}_l^1, \bar{z}_l^2, \cdots, \bar{z}_l^K\right]$. Then, the Gate outputs a distribution over the l-th Task-specific Adapter execution decisions $\boldsymbol{z_l}$ using the Softmax function

$$\boldsymbol{z_l} = softmax(\bar{\boldsymbol{z}}_{\boldsymbol{l}}), \tag{4}$$

where $\boldsymbol{z_l} = \left[z_l^1, z_l^2, \cdots, z_l^K\right]$. z_l^k denotes the possibility of selecting k-th Task-specific Adapter in the l-th layer. And $\boldsymbol{z} = [\boldsymbol{z_1}, \boldsymbol{z_2}, \cdots, \boldsymbol{z_L}]$, L is the number of layers in the backbone.

Training The generation of Task-specific Adapter execution decisions is cast as the Reinforcement Learning problem. The Action Generation Network aims to decide the most appropriate Task-specific Adapters for conducting few-shot learning on the current tasks. We utilize the predicted result of the Adapted Network constructed based on the current "actions" as the rewards for optimizing the Action Generation Network. Inspired by the self-critical sequence training approach [34], the goal of training is to minimize the negative expected reward

$$L(\theta) = -(r(\boldsymbol{z^s}) - r(\boldsymbol{z^m})) \log(\boldsymbol{z^s}), \tag{5}$$

where θ is the parameters of the Gate Network. $\theta = \left\{\mathbf{W_l^1}, \mathbf{W_l^2}\right\}$ and $l \in \{1, 2, \cdots, L\}$. $\boldsymbol{z^s} = [z_1^s, ..., z_L^s]$ and z_l^s is the possibility of action randomly sampled from the K actions – Task-specific Adapters, at the layer l. $r(\boldsymbol{z^s})$ is the reward obtained by the Adapted Network, which is constructed by the actions with $\boldsymbol{z^s}$ discrete possibility. $\boldsymbol{z^m} = [z_1^m, ..., z_L^m]$, and

$$z_l^m = argmax\left(\boldsymbol{z_l}\right). \tag{6}$$

where $argmax(\cdot)$ returns the possibility of action equipped with maximum probability over the output distribution $\boldsymbol{z_l}$ of the Gate in l-th layer. $r(\boldsymbol{z^m})$ again is the reference reward obtained in Adapted Network equipped with actions with $\boldsymbol{z^m}$ discrete possibility. Accordingly, TA modules from the model that return a higher reward than $\boldsymbol{z^m}$ will be "pushed up", or increased in probability, while TA modules that result in a lower reward will be suppressed.

3.2.2 Adapted Network

As shown in Figure 2, the network architecture of the Adapted Network consists of a series of residual blocks, which are formed by the Adaptive Module (AM) followed by the BN layer and ReLU function. In the Adaptive Module, a 3×3 *conv* is paralleled with the Task-specific Adapter,

Algorithm 1 Task-aware Adaptive Learning

Train the backbone network $f_\phi(\cdot)$ on the source domain;
Build Action Generation Network M_{act} on the backbone network; Initialization the parameters θ of Gate Networks;
while training **do**
  Random sample task T_i in the source domain; $T_i = S \bigcup Q, S = \{(x_i, y_i)\}_{i=1}^{N_S}, Q = \{(x_i)\}_{i=1}^{N_Q}$;
  $\boldsymbol{z} = f_{\phi,\theta}(x), x \in S, \boldsymbol{z} = [\boldsymbol{z_1}, \boldsymbol{z_2}, \cdots, \boldsymbol{z_L}]$;
  $\boldsymbol{z^s} = [Sample(\boldsymbol{z_1}), \cdots, Sample(\boldsymbol{z_L})]$; $\boldsymbol{z^m} = [argmax(\boldsymbol{z_1}), \cdots, argmax(\boldsymbol{z_L})]$;
  Build Adapted Network M^1_{adapt}, M^2_{adapt} based on the backbone network according to actions $\boldsymbol{z^s}$, $\boldsymbol{z^m}$ respectively;
  Initialization ψ, π in M^1_{adapt} and M^2_{adapt};
  Optimize ψ, π over S based on $L_{adapt}(\psi, \pi) = \frac{1}{N_S}\sum_{(x,y)\in S} l(f_{\phi,\psi,\pi}(\boldsymbol{x}), y)$;
  Predict over Q; $r(\boldsymbol{z^s}) = N^1_{adapt}/N_Q$, $r(\boldsymbol{z^m}) = N^2_{adapt}/N_Q$;
  Optimize the parameters θ of M_{act} based on $L(\theta) = -(r(\boldsymbol{z^s}) - r(\boldsymbol{z^m}))\log(\boldsymbol{z^s})$;
end while

Output: Action Generation Network M_{act}

which is the action predicted by the corresponding Gate Network of the Action Generation Network.

Let $f_{\phi_l}(\cdot)$ denote the 3×3 *conv* in the l-th Adaptive Module, which is pre-trained by the seen domain. With $\boldsymbol{h_{l-1}} \in R^{N_S \times C \times W \times H}$ as the input feature of l-th Adaptive Module, the output is $\boldsymbol{h_l} = f_{\phi_l}(\boldsymbol{h_{l-1}})$ when no Task-specific Adapter is selected according to the decision generated by the Action Generation Network, that is the feature does not need adaptation. When the k-th Task-specific Adapter is chosen, which is constructed by paralleling k task-specific layers in our design, the feature learned by the Task-specific Adapter will be combined with the feature extracted from the 3×3 *conv* as

$$\boldsymbol{h_l} = f_{\phi_l}(\boldsymbol{h_{l-1}}) + \frac{1}{k}\sum_{i=1}^{k} f_{\psi_l^i}(\boldsymbol{h_{l-1}}), \tag{7}$$

where $f_{\psi_l^i}(\cdot)$ denotes the i-th task-specific layer in the Task-specific Adapter, which is 1×1 *conv* in the experiments.

Training Upon the generated "actions" with the discrete possibility $\boldsymbol{z^s}$ and $\boldsymbol{z^m}$ in the Action Generation Network, we will construct two Adapted Networks M^1_{adapt} and M^2_{adapt}. In training, the Adapted Networks are trained with the cross-entropy loss over the support samples:

$$L_{adapt}(\psi, \pi) = \frac{1}{N_S}\sum_{(x,y)\in S} l(f_{\phi,\psi,\pi}(\boldsymbol{x}), y), \tag{8}$$

where $f_{\psi,\pi}(\cdot)$ denotes the output softmax probability vector of the linear classifier with parameters π, in which dimensionality equals the number of categories in the support set S. Then, the reward for the agent is achieved by predicting the Adapted Network over the query samples Q:

$$r(\boldsymbol{z^s}) = \frac{N^1_{adapt}}{N_Q} \quad and \quad r(\boldsymbol{z^m}) = \frac{N^2_{adapt}}{N_Q}. \tag{9}$$

where N^1_{adapt} and N^2_{adapt} denote the number of query samples correctly predicted by Adapted Networks M^1_{adapt} and M^2_{adapt} respectively. We show the Task-aware Adaptive Learning pseudo code in Algorithm 1.

3.2.3 Testing

Note that the support samples of testing will first go through the Action Generation Network to generate optimal Task-specific Adapter executions for the target task, which are the action with maximum probability. And then, based on the executions, the Adapted Network will be built and fine-tuned over the support set. Finally, the query samples will be predicted by the Adapted Network.

4. Experiments

In this part, we first introduce the experimental setup and then compare our method with the state-of-the-art methods. Furthermore, we provide a series of analyses of our method.

4.1. Experimental setup

Dataset. Meta-Dataset [44] is the standard benchmark for evaluating the performance of few-shot classification. It consists of 13 image datasets: ILSVRC 2012 [36], Omniglot [18], FGVC-Aircraft [29], CUB-200-2011 [47], Describable Textures [5], Quick Draw [11], FGVCx Fungi [38], VGG Flower [31], Traffic Signs [13], MSCOCO [24], MNIST [19], CIFAR-10 [16], and CIFAR-100 [16]. For all datasets in the experiments, we used the standard split procedure as [44].

Implementation details. In the experiments, we considered two training settings: multi-domain learning and single-domain setting. In the multi-domain setting, the first eight datasets (ILSVRC 2012, Omniglot, Aircraft, CUB-200-2011, Textures, Quick Draw, Fungi, and VGG Flower) of the Meta-dataset are used to train the feature extractor. We followed the method in [20] to obtain the pre-trained model. And in the single-domain setting, only the ILSVRC 2012 dataset participated in the training of the feature extractor. As in [20], we adopted the ResNet-18 as the backbone no matter in the multi-domain setting or the single-domain setting. In the training, the TA^2-Net is trained using the Adam optimizer. The hyper-parameters K are set as $K = 3$. We also provide the ablation study in Section 4.4.

Table 1. Comparison to state-of-the-art methods **in the multi-domain setting**. Mean accuracy, 95% confidence interval are reported. The best results are marked in **bold**.

Test Dataset	CNAPS [35]	Simple CNAPS [3]	Transductive CNAPS [2]	SUR [7]	URT [26]	FLUTE [43]	tri-M [27]	URL [20]	TSA [21]	**Ours**
ImageNet	50.8 ± 1.1	58.4 ± 1.1	57.9 ± 1.1	56.2 ± 1.0	56.8 ± 1.1	58.6 ± 1.0	51.8 ± 1.1	58.8 ± 1.1	59.5 ± 1.0	$\mathbf{59.6 \pm 1.0}$
Omniglot	91.7 ± 0.5	91.6 ± 0.6	94.3 ± 0.4	94.1 ± 0.4	94.2 ± 0.4	92.0 ± 0.6	93.2 ± 0.5	94.5 ± 0.4	94.9 ± 0.4	$\mathbf{95.5 \pm 0.4}$
Aircraft	83.7 ± 0.6	82.0 ± 0.7	84.7 ± 0.5	85.5 ± 0.5	85.8 ± 0.5	82.8 ± 0.7	87.2 ± 0.5	89.4 ± 0.4	89.9 ± 0.4	$\mathbf{90.5 \pm 0.4}$
Birds	73.6 ± 0.9	74.8 ± 0.9	78.8 ± 0.7	71.0 ± 1.0	76.2 ± 0.8	75.3 ± 0.8	79.2 ± 0.8	80.7 ± 0.8	81.1 ± 0.8	$\mathbf{81.4 \pm 0.8}$
Textures	59.5 ± 0.7	68.8 ± 0.9	66.2 ± 0.8	71.0 ± 0.8	71.6 ± 0.7	71.2 ± 0.8	68.8 ± 0.8	77.2 ± 0.7	$\mathbf{77.5 \pm 0.7}$	77.4 ± 0.7
Quick Draw	74.7 ± 0.8	76.5 ± 0.8	77.9 ± 0.6	81.8 ± 0.6	82.4 ± 0.6	77.3 ± 0.7	79.5 ± 0.7	$\mathbf{82.5 \pm 0.6}$	81.7 ± 0.6	$\mathbf{82.5 \pm 0.6}$
Fungi	50.2 ± 1.1	46.6 ± 1.0	48.9 ± 1.2	64.3 ± 0.9	64.0 ± 1.0	48.5 ± 1.0	58.1 ± 1.1	$\mathbf{68.1 \pm 0.9}$	66.3 ± 0.8	66.3 ± 0.9
VGG Flower	88.9 ± 0.5	90.5 ± 0.5	92.3 ± 0.4	82.9 ± 0.8	87.9 ± 0.6	90.5 ± 0.5	91.6 ± 0.6	92.0 ± 0.5	92.2 ± 0.5	$\mathbf{92.6 \pm 0.4}$
Traffic Sign	56.5 ± 1.1	57.2 ± 1.0	59.7 ± 1.1	51.0 ± 1.1	48.2 ± 1.1	63.0 ± 1.0	58.4 ± 1.1	63.3 ± 1.1	82.8 ± 1.0	$\mathbf{87.4 \pm 0.8}$
MSCOCO	39.4 ± 1.0	48.9 ± 1.1	42.5 ± 1.1	52.0 ± 1.1	51.5 ± 1.1	52.8 ± 1.1	50.0 ± 1.0	57.3 ± 1.0	57.6 ± 1.0	$\mathbf{57.9 \pm 0.9}$
MNIST	–	94.6 ± 0.4	94.7 ± 0.3	94.3 ± 0.4	90.6 ± 0.5	96.2 ± 0.3	95.6 ± 0.5	94.7 ± 0.4	$\mathbf{96.7 \pm 0.4}$	$\mathbf{97.0 \pm 0.4}$
CIFAR-10	–	74.9 ± 0.7	73.6 ± 0.7	66.5 ± 0.9	67.0 ± 0.8	75.4 ± 0.8	78.6 ± 0.7	74.2 ± 0.8	$\mathbf{82.9 \pm 0.7}$	82.1 ± 0.8
CIFAR-100	–	61.3 ± 1.1	61.8 ± 1.0	56.9 ± 1.1	57.3 ± 1.0	62.0 ± 1.0	67.1 ± 1.0	63.5 ± 1.0	70.4 ± 0.9	$\mathbf{70.9 \pm 0.9}$
Average Seen	71.6	73.7	75.1	75.9	77.4	74.5	76.2	80.4	80.4	**80.7**
Average Unseen	–	67.4	66.5	64.1	62.9	69.9	69.9	70.6	78.1	**79.1**
Average All	–	71.2	71.8	71.4	71.8	72.7	73.8	76.6	79.5	**80.1**
Average Rank	–	7.6	6.2	7.2	6.5	5.8	5.5	3.2	2.2	**1.2**

4.2. Comparison to state-of-the-art methods

Following the experiment set in [21], we sampled tasks with varying numbers of ways and shots in the inference and conducted experiments in the multi-domain setting and single-domain setting.

Multi-domain setting. Here, the results of our method on 13 datasets of the Meta-Dataset are compared with the state-of-the-art methods. We also report average accuracy over seen domains, and unseen domains, as well as the average rank as in [21]. From the results in Table 1, our method obtains state-of-the-art results on 10 out of 13 datasets and achieves the best average accuracy on seen and unseen domains. It is worth noting that our method obtains significantly better average results than the second-best approach on the unseen datasets ($+1.0\%$). The unseen datasets are heterogeneous with respect to the training domain, with a significant domain gap between them. In contrast to adopting the fixed number of task-specific parameters for all target tasks, *e.g.* CNAPS, Simple CNAPS, FLUTE, and TSA, the TA^2-Net can adaptively learn the optimal task-specific parameter policies for unseen target tasks to efficiently transfer the pre-trained knowledge to the target feature distribution.

Single-domain setting. We also evaluate our method in the single-domain setting, which is more challenging due to optimizing the Action Generation Network only on the ImageNet dataset. Results are shown in Table 2. First, compared with other single-domain training methods, our method achieves significant superiority. For example, TA^2-Net outperforms the most competing method (TSA [21]) by $+2.9\%$, $+0.8\%$, $+1.6\%$, and $+1\%$ on Omniglot, Quick Draw, Traffic Signs and MNIST, respectively. Second, it is worth noting that, for the average accuracy on the unseen datasets, ours outperforms the second-best approach with a clear margin ($+3.1\%$). And we clearly observe that TA^2-Net achieves competitive accuracy on the ImageNet, compared with the best performance. Despite being trained on just one dataset, the Action Generation Network exhibits good generalization, producing the optimal task-specific parameter policies for the unseen domain in an excellently adaptive manner.

4.3. Further Analysis

Varying-way Five-shot setting. The number of support samples ('shot') in the standard Meta-Dataset evaluation setting is variable and can range up to 100. In this section, we evaluate our method in the 5-shot setting which is more challenging due to fewer labeled samples and can better reflect the progressiveness of the model. Table 3 shows the results compared with other state-of-the-art methods. As can be seen, ours achieves better average accuracy on both seen and unseen domains. More specifically, our method outperforms other methods on the vast majority of domains (8 out of 13) and obtains the best average rank. Furthermore, the average accuracy on unseen datasets is 1.2% higher than the second-best method.

Table 2. Comparison to state-of-the-art methods **in the single-domain setting**. Mean accuracy, 95% confidence interval are reported. The best results are marked in **bold**.

Test Dataset	ProtoNet [44]	fo-Proto -MAML [44]	ALFA+fo-Proto -MAML [44]	BOHB [37]	FLUTE [43]	TSA [21]	Ours
ImageNet	50.5 ± 1.1	49.5 ± 1.1	52.8 ± 1.1	51.9 ± 1.1	46.9 ± 1.1	**59.5 ± 1.1**	59.3 ± 1.1
Omniglot	60.0 ± 1.4	63.4 ± 1.3	61.9 ± 1.5	67.6 ± 1.2	61.6 ± 1.4	78.2 ± 1.2	**81.1 ± 1.1**
Aircraft	53.1 ± 1.0	56.0 ± 1.0	63.4 ± 1.1	54.1 ± 0.9	48.5 ± 1.0	72.2 ± 1.0	**72.6 ± 0.9**
Birds	68.8 ± 1.0	68.7 ± 1.0	69.8 ± 1.1	70.7 ± 0.9	47.9 ± 1.0	74.9 ± 0.9	**75.1 ± 0.9**
Textures	66.6 ± 0.8	66.5 ± 0.8	70.8 ± 0.9	68.3 ± 0.8	63.8 ± 0.8	**77.3 ± 0.7**	76.8 ± 0.8
Quick Draw	49.0 ± 1.1	51.5 ± 1.0	59.2 ± 1.2	50.3 ± 1.0	57.5 ± 1.0	67.6 ± 0.9	**68.4 ± 0.9**
Fungi	39.7 ± 1.1	40.0 ± 1.1	41.5 ± 1.2	41.4 ± 1.1	31.8 ± 1.0	44.7 ± 1.0	**45.3 ± 1.0**
VGG Flower	85.3 ± 0.8	87.2 ± 0.7	86.0 ± 0.8	87.3 ± 0.6	80.1 ± 0.9	90.9 ± 0.6	**91.0 ± 0.6**
Traffic Sign	47.1 ± 1.1	48.8 ± 1.1	60.8 ± 1.3	51.8 ± 1.0	46.5 ± 1.1	82.5 ± 0.8	**84.1 ± 0.7**
MSCOCO	41.0 ± 1.1	43.7 ± 1.1	48.1 ± 1.1	48.0 ± 1.0	41.4 ± 1.0	**59.0 ± 1.0**	58.0 ± 1.0
MNIST	–	–	–	–	80.8 ± 0.8	93.9 ± 0.6	**94.9 ± 0.5**
CIFAR-10	–	–	–	–	65.4 ± 0.8	**82.1 ± 0.7**	82.0 ± 0.7
CIFAR-100	–	–	–	–	52.7 ± 1.1	70.7 ± 0.9	**70.8 ± 0.9**
Average Seen	50.5	49.5	52.8	51.9	46.9	**59.5**	59.3
Average Unseen	56.7	58.4	62.4	60.0	53.2	71.9	**75.0**
Average All	56.1	57.5	61.4	59.2	52.6	70.7	**73.8**
Average Rank	5.9	5.2	3.4	4.1	5.6	1.7	**1.3**

Table 3. Results for **Varying-Way Five-Shot setting**. Mean accuracy, 95% confidence interval are reported. The best results are marked in **bold**.

Test Dataset	Simple CNAPS [3]	SUR [7]	URT [26]	URL [20]	TSA [21]	Ours
ImageNet	47.2 ± 1.0	46.7 ± 1.0	48.6 ± 1.0	**49.4 ± 1.0**	48.3 ± 1.0	49.3 ± 1.0
Omniglot	95.1 ± 0.3	95.8 ± 0.3	96.0 ± 0.3	96.0 ± 0.3	**96.8 ± 0.3**	96.6 ± 0.2
Aircraft	74.6 ± 0.6	82.1 ± 0.6	81.2 ± 0.6	84.8 ± 0.5	85.5 ± 0.5	**85.9 ± 0.4**
Birds	69.6 ± 0.7	62.8 ± 0.9	71.2 ± 0.7	76.0 ± 0.6	76.6 ± 0.6	**77.3 ± 0.6**
Textures	57.5 ± 0.7	60.2 ± 0.7	65.2 ± 0.7	**69.1 ± 0.6**	68.3 ± 0.7	68.3 ± 0.6
Quick Draw	70.9 ± 0.6	79.0 ± 0.5	**79.2 ± 0.5**	78.2 ± 0.5	77.9 ± 0.6	78.5 ± 0.5
Fungi	50.3 ± 1.0	66.5 ± 0.8	66.9 ± 0.9	70.0 ± 0.8	**70.4 ± 0.8**	70.3 ± 0.8
VGG Flower	86.5 ± 0.4	76.9 ± 0.6	82.4 ± 0.5	89.3 ± 0.4	89.5 ± 0.4	**90.0 ± 0.4**
Traffic Sign	55.2 ± 0.8	44.9 ± 0.9	45.1 ± 0.9	57.5 ± 0.8	72.3 ± 0.6	**76.7 ± 0.5**
MSCOCO	49.2 ± 0.8	48.1 ± 0.9	52.3 ± 0.9	56.1 ± 0.8	**56.0 ± 0.8**	**56.0 ± 0.8**
MNIST	88.9 ± 0.4	90.1 ± 0.4	86.5 ± 0.5	89.7 ± 0.4	92.5 ± 0.4	**93.3 ± 0.3**
CIFAR-10	66.1 ± 0.7	50.3 ± 1.0	61.4 ± 0.7	66.0 ± 0.7	72.0 ± 0.7	**73.1 ± 0.7**
CIFAR-100	53.8 ± 0.9	46.4 ± 0.9	52.5 ± 0.9	57.0 ± 0.9	**64.1 ± 0.8**	**64.1 ± 0.8**
Average Seen	69.0	71.2	73.8	76.6	76.7	**77.0**
Average Unseen	62.6	56.0	59.6	65.2	71.4	**72.6**
Average All	66.5	65.4	68.3	72.2	74.6	**75.3**
Average Rank	5.2	5	4.1	2.8	2.2	**1.5**

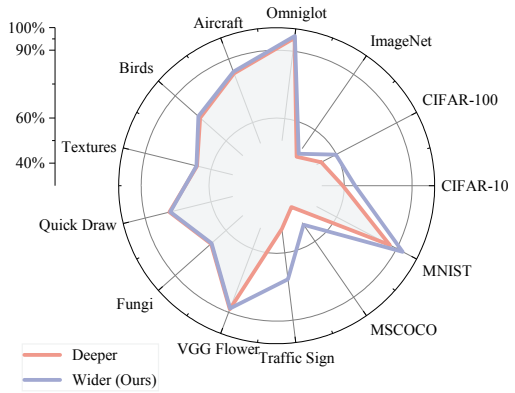


Figure 3. Performance of widening or deepening the Task-specific Adapter (TA) in the multi-domain setting. The axes indicate the accuracy of methods on a particular dataset. Each color corresponds to a different method.

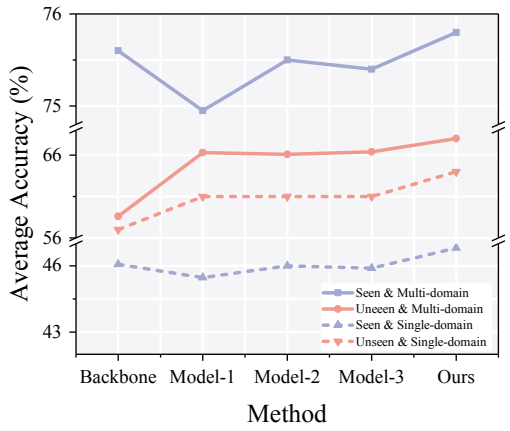
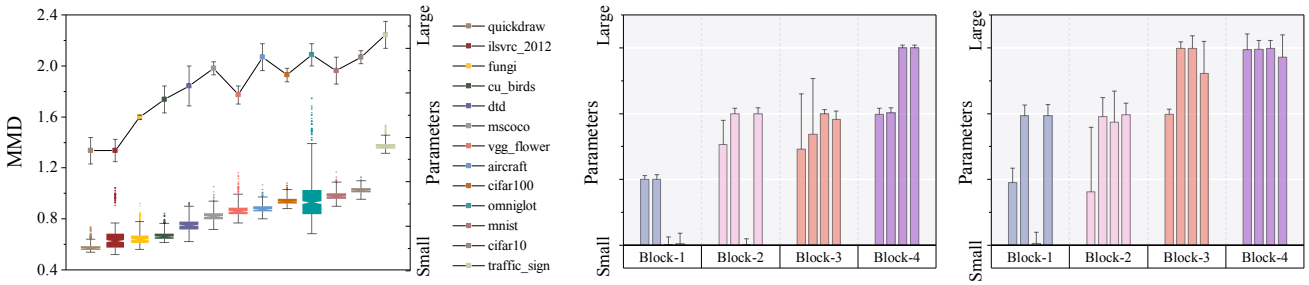


Figure 4. Effectiveness of adaptive learning.

Wider or deeper in the Task-specific Adapter (TA)? To explore the method for improving the adaptation of the model, we increase the parameters of the Task-specific Adapter (TA) in wide or depth respectively. Specifically, two types of TAs are designed: (a) 1×1 *Conv* in parallel, (b) 1×1 *Conv* in series. As shown in Figure 3, increasing the width of the TA allows the model to achieve better performance than deepening. In particular, among the 13 datasets, 5 datasets (Traffic Signs, MSCOCO, MNIST, CIFAR-10, and CIFAR-100) unseen in the training obviously benefited more from widening the Task-specific Adapter (TA). All of the above indicates that the best way to improve adaptability is to increase the width of the Task-specific Adapter (TA).

Effectiveness of adaptive learning. In this section, we attempt to provide a more in-depth discussion of the efficacy of fixed networks as well as the proposed adaptive learned network. Figure 4 depicts the performance over the Meta-dataset in the multi-domain and single-domain setting, with Model-$1/2/3$ constructed by attaching Module-$1/2/3$ respectively as shown in Figure 1 to the backbone. It can be observed that, in comparison to the backbone network, the average accuracy of seen domains in both the multi-domain and single-domain settings decreased when using the fixed Task-specific Adapter to adaptation, but our method performed better. Furthermore, average accuracy on unseen domains has consistently improved overall in the fixed Task-specific Adapter and can be improved further through adaptive learning. This discovery inspires us to recognize that simply designing a fixed Task-specific Adapter does not work and forces us to recognize the necessity of adaptively learning optimal task-specific parameters policy for each target task.

Effectiveness of Reinforcement Learning. Our TA^2-Net model adopts the Reinforcement Learning algorithm combined with Episodic Training to adaptively learn opti-



(a) MMD and TA parameters of different datasets in the Meta-Dataset

(b) TA parameters distribution in different layers for the Fungi dataset

(c) TA parameters distribution in different layers for the CIFAR-10 dataset

Figure 5. (a): The Line chart shows the mean and standard deviation of TA parameters adaptively learned by the TA^2-Net in target tasks sampled in different datasets. And the box plot shows the 25% quantiles of the MMD in different datasets while the notches indicate the median. (b) and (c): TA parameters distribution in different blocks of the task-aware adapted network for the Fungi and CIFAR-10 dataset. Sub-bars in the block denote different layers within the Block.

mal task-specific parameters policy for target tasks. To explore the effectiveness of Reinforcement Learning, we designed the Simple TA^2-Net trained by Episodic Training, which only includes the Adapted Network attaching a Gate Network in each layer for generating "decisions" – Chosen the Task-specific Adapter. As shown in Table 4, the TA^2-Net optimized with the Reinforcement Learning outperforms the Simple TA^2-Net, which shows that Reinforcement Learning can better learn the TA decisions for target tasks.

Parameters distribution. This is an intuitive guess that the greater the difference (domain shift) between the source domain and the target domain features, the greater the demand for TA parameters. In this section, we first observe the relationship between the TA parameters and the domain shift. As widely used in previous works [28, 25], we adopt the Maximum Mean Discrepancy (MMD) to quantify the domain shift between the target task features and the source domain features extracted by the pre-trained backbone (in the multi-domain setting). The larger MMD indicates a larger domain shift between the test task and the source domain. As shown in Figure 5 (a), the Line chart and box plot shows the TA parameters and MMD distribution in different datasets, respectively. We can find that MMD for each dataset has a narrow distribution. And, on the whole, datasets with significant MMD have a larger demand for the TA parameters. Furthermore, linear regression and Pearson correlation analysis revealed that MMD is positively correlated with TA parameters, with the *Pearson correlation coefficient* $R = 0.9186$. It is attributed to the greater heterogeneity, more task-specific parameters are required to effectively adapt pre-learned knowledge to the feature distribution of target tasks. We further explore the TA parameters distribution in different layers of the task-aware adapted network learned by our proposed TA^2-Net. As shown in Figure 5 (b) and (c), layers in the deep blocks have more task-specific parameters for learning the task-specific features than those in the shallow blocks as demonstrated by both Fungi and CIFAR-10, which may be because the shallow layer contains more general information while the deep layer contains more domain-specific information [50].

Table 4. Effectiveness of Reinforcement Learning in the multi-domain setting.

Model	Reinforcement Learning	Seen Domain	Unseen Domain	All
Simple TA^2-Net	✗	75.5	67.9	72.6
TA^2-Net	✓	77.0	72.6	75.3

Table 5. Mean accuracy (%) for different number of layers K in the Task-specific Adapter.

	$K=1$	$K=2$	$K=3$	$K=4$
Seen Domain	75.4	75.2	75.3	75.1
Unseen Domain	66.6	70.3	70.5	68.2
All	72.0	73.3	73.5	72.5

4.4. Ablation Study

Maximum width K of the Task-specific Adapter. Here, we discuss the maximum width K of the Task-specific Adapter, which represents the upper bound of the task-specific parameters. Experiments are conducted on the Meta-dataset in the varying-way five-shot setting. Table 5 shows the performances of $K \in [1, 2, 3, 4]$. We can observe that, as K increases, the average classification accuracy on seen domains lightly decreases. For unseen domains, the average classification accuracy increases at first and then decreases as K increases, with $K = 3$ producing the best results. Compared with seen domains, unseen domains are heterogeneous with the source domain and require more task-specific parameters to transfer the pre-learned knowledge. The average performance becomes poorer when using $K = 4$, which may be due to too much TA, the model

overfitting, or the search space of the model structure expanding, increasing the optimization difficulty of reinforcement learning. Considering the performance of various target tasks and the calculating costs, all experiments in this paper adopt $K = 3$.

5. Conclusion

In this paper, for studying cross-domain few-shot learning problems, we explored the performance of test tasks when inference was performed in models equipped with a variant number of task-specific parameters. The exploration revealed that target tasks require variant task-specific parameters for outstanding adaptation. Motivated by this, we proposed a Task-aware Adaptive Network (TA^2-Net), which can adaptively learn optimal task-specific parameters policy for each target task for exhaustively improving the recognition performance of cross-domain few-shot learning. By conducting extensive experiments, we proved that our TA^2-Net could achieve the best results on Meta-dataset.

Acknowledgment

This work was supported in part by Beijing Natural Science Foundation Project No. Z200002, in part by National Natural Science Foundation of China (NSFC) No. U19B2036, 62225601, in part by the Program for Youth Innovative Research Team of BUPT No. 2023QNTD02, in part by scholarships from China Scholarship Council (CSC) under Grant CSC No. 202206470055, in part by BUPT Excellent Ph.D. Students Foundation No. CX2022152.

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. 3

[2] Peyman Bateni, Jarred Barber, Jan-Willem van de Meent, and Frank Wood. Enhancing few-shot image classification with unlabelled examples. In *WACV*, pages 2796–2805, 2022. 6

[3] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *CVPR*, pages 14493–14502, 2020. 1, 2, 6, 7

[4] Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019. 1

[5] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014. 5

[6] Carl Doersch, Ankush Gupta, and Andrew Zisserman. CrossTransformers: spatially-aware few-shot transfer. *NeurIPS*, 33:21981–21993, 2020. 1, 2

[7] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a multi-domain representation for few-shot classification. In *ECCV*, pages 769–786, 2020. 1, 2, 6, 7

[8] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022. 3

[9] Yunhui Guo, Noel C Codella, Leonid Karlinsky, James V Codella, John R Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A broader study of cross-domain few-shot learning. In *ECCV*, pages 124–141, 2020. 1

[10] Yurong Guo, Ruoyi Du, Xiaoxu Li, Jiyang Xie, Zhanyu Ma, and Yuan Dong. Learning calibrated class centers for few-shot classification by pair-wise similarity. *IEEE TIP*, 31:4543–4555, 2022. 1

[11] David Ha and Douglas Eck. A Neural Representation of Sketch Drawings. *arXiv preprint arXiv:1704.03477*, 2017. 5

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *IJCNN*, pages 1–8, 2013. 5

[14] Yanxu Hu and Andy J. Ma. Adversarial feature augmentation for cross-domain few-shot classification. In *ECCV*, pages 20–37, 2022. 1

[15] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *NeurIPS*, 12, 1999. 3

[16] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 25, 2012. 1

[18] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 5

[19] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. 5

[20] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Universal representation learning from multiple domains for few-shot classification. In *ICCV*, pages 9526–9535, 2021. 1, 2, 5, 6, 7

[21] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Cross-domain few-shot learning with task-specific adapters. In *CVPR*, pages 7161–7170, 2022. 2, 6, 7

[22] Xiaoxu Li, Jijie Wu, Zhuo Sun, Zhanyu Ma, Jie Cao, and Jing-Hao Xue. BSNet: Bi-similarity network for few-shot fine-grained image classification. *IEEE TIP*, 30:1318–1331, 2020. 1

[23] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017. 1

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014. 5

[25] Ge Liu, Linglan Zhao, and Xiangzhong Fang. PDA: Proxy-based domain adaptation for few-shot image recognition. *Image and Vision Computing*, 110:104164, 2021. 8

[26] Lu Liu, William Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle. A universal representation transformer layer for few-shot image classification. In *ICLR*, 2021. 1, 2, 6, 7

[27] Yanbin Liu, Juho Lee, Linchao Zhu, Ling Chen, Humphrey Shi, and Yi Yang. A multi-mode modulator for multi-domain few-shot classification. In *ICCV*, pages 8453–8462, 2021. 6

[28] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105, 2015. 8

[29] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 5

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 3

[31] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. 5

[32] Hang Qi, Matthew Brown, and David G. Lowe. Low-Shot learning with imprinted weights. In *CVPR*, pages 5822–5830, 2018. 1

[33] Sachin Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017. 1

[34] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *CVPR*, pages 7008–7024, 2017. 4

[35] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. *NeurIPS*, 32, 2019. 1, 2, 6

[36] Olga Russakovsky, J. Deng, Hao Su, J. Krause, S. Satheesh, S. Ma, Zhiheng Huang, A. Karpathy, A. Khosla, Michael S. Bernstein, A. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015. 1, 5

[37] Tonmoy Saikia, Thomas Brox, and Cordelia Schmid. Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*, 2020. 7

[38] Brigit Schroeder and Yin Cui. FGVCx Fungi Classification Challenge 2018. `github.com/visipedia/fgvcx_fungi_comp`, 2018. 5

[39] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015. 3

[40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 3

[41] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pages 387–395, 2014. 3

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[43] Eleni Triantafillou, Hugo Larochelle, Richard Zemel, and Vincent Dumoulin. Learning a universal template for few-shot dataset generalization. In *ICML*, pages 10424–10433, 2021. 2, 6, 7

[44] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019. 5, 7

[45] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. In *ICLR*, 2020. 1

[46] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 3

[47] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 5

[48] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, pages 1995–2003, 2016. 3

[49] Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *CVPR*, pages 8012–8021, 2021. 1

[50] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *NeurIPS*, 27, 2014. 8