Cross-domain Few-shot Learning with Task-specific Adapters

Anonymous CVPR submission

Paper ID 38

Abstract

In this paper, we look at the problem of cross-domain few-shot classification that aims to learn a classifier from previously unseen classes and domains with few labeled samples. Recent approaches broadly solve this problem by parameterizing their few-shot classifiers with task-agnostic and task-specific weights where the former is typically learned on a large training set and the latter is dynamically predicted through an auxiliary network conditioned on a small support set. In this work, we focus on the estimation of the latter, and propose to learn task-specific weights from scratch directly on a small support set, in contrast to dynamically estimating them. In particular, through systematic analysis, we show that task-specific weights through parametric adapters in matrix form with residual connections to multiple intermediate layers of a backbone network significantly improves the performance of the state-of-the-art models in the Meta-Dataset benchmark with minor additional cost.

1. Introduction

Deep learning methods have seen remarkable progress in various fields where large quantities of data and compute power are available. However, the ability of deep networks to learn new concepts from small data remains limited. Few-shot classification [17,24] is inspired from this limitation and aims at learning a model that can be efficiently adapted to recognize unseen classes from few samples. In particular, the standard setting for learning few-shot classifiers involves two stages: (i) learning a model, typically from a large training set, (ii) adapting this model to learn new classes from a given small support set. These two stages are called meta-training and meta-testing respectively. The adapted model is finally evaluated on a query set where the task is to assign each query sample to one of the classes in the support set.

Early methods [13, 25, 27, 32, 34, 38] pose the few-shot classification problem in a learning-to-learn formulation by training a deep network over a distribution of related tasks, which are sampled from the training set, and transfer this experience to improve its performance for learning new classes.

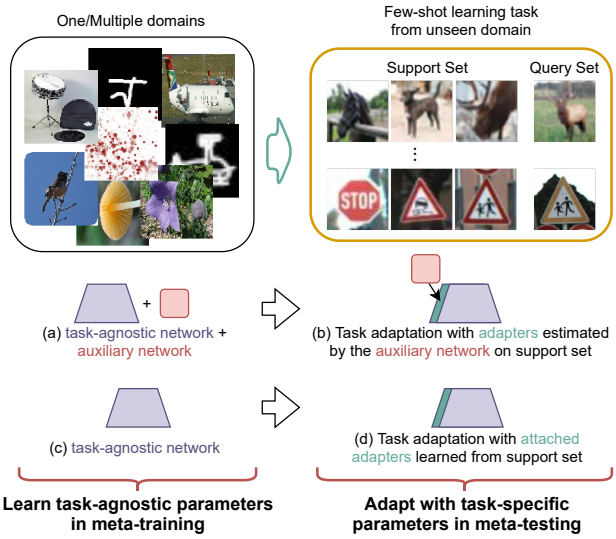

Figure 1. **Cross-domain Few-shot Learning** considers to learn a model from one or multiple domains to generalize to unseen domains with few samples. Prior works often learn a task-agnostic model with an auxiliary network during meta-training (a) and a set of adapters are generated by the auxiliary network to adapt to the given support set (b). While in this work, we propose to attach adapters directly to a pretrained task-agnostic model (c), which can be estimated from scratch during meta-testing (d). We also propose different architecture topologies of adapters and their efficient approximations.

Concretely, Vinyals *et al*. [38] learn a feature encoder that is conditioned on the support set in meta-training and does not require any further training in meta-test thanks to its non-parametric classifier. Ravi and Larochelle [27] take the idea of learning a feature encoder in meta-train further by also learning an update rule through a LSTM that produces the updates for a classifier in meta-test. Finn *et al*. [13] pose the task as a meta-learning problem and learn the parameters of a deep network in meta-training such that a network initialized with the learned parameters can efficiently finetuned on a new task. We refer to [15,39] for comprehensive review of early works.

Despite the significant progress, the scope of the early

methods has been limited to a restrictive setting where training and test samples come from a single domain (or data distribution) such as Omniglot [18], miniImageNet [38] and tieredImageNet [30]. They perform poorly in the more challenging cross-domain few-shot tasks, where test data is sampled from an unknown or previously unseen domain [37]. This setting poses an additional learning challenge, not only requires leveraging the limited information from the small support set for learning the target task but also *selectively transferring relevant knowledge* from previously seen domains to the target task.

Broadly, recent approaches address this challenge by parameterizing deep networks with a large set of task-agnostic and a small set of task-specific weights that encode generic representations valid for multiple tasks and private representations are specific to the target task respectively. While the task-agnostic weights are learned over multiple tasks, typically, from a large dataset in meta-training, the task-specific weights are estimated from a given small support set (*e.g.* 5 images per category) [2, 12, 19–21, 31, 36]. In the literature, the task-agnostic weights are used to parameterize a single network that is trained on large data from one domain [2, 11, 31] or on multiple domains [20], or to be distributed over multiple networks, each trained on a different domain [12, 21, 36] [1]. The task-specific weights are utilized to parameterize a linear classifier [19] and a pre-classifier feature mapping [20].

Recently, inspired from [26], *task-specific adapters* [2, 31], small capacity transformations that are applied to multiple layers of a deep network, have been successfully used to steer the few-shot classifiers to new tasks and domains. Their weights are often estimated dynamically through an auxiliary network conditioned on the support set [2, 21, 31, 36] (see Fig. 1.(a,b)), in a similar spirit to [3, 16]. As the auxiliary network is trained on multiple tasks in meta-training, the premise of estimating the task-specific adapter weights with it is based on the principle of transfer learning such that it can transfer the knowledge from the previous tasks to better estimate them for unseen tasks. However, learning an accurate auxiliary network is a challenging task due to two reasons. First, it has to generalize to previously unseen tasks and especially to new significantly different domains. Second, learning to predict high-dimensional weights where each corresponds to a dimension of a highly nonlinear feature space is a difficult learning problem too.

Motivated by this shortcoming, as shown in Fig. 1, we propose to employ a set of light-weight task-specific adapters along with the task-agnostic weights for adapting the few-shot classifier to the tasks from unseen domains. Unlike the prior work, we learn the weights of these adapters from scratch by directly optimizing them on a small support set (see Fig. 1.(c,d)). Moreover, we systematically study various combinations of several design choices for task-specific adaptation, which have not been explored before, including adapter connection types (serial or residual), parameterizations (matrix and its decomposed variations, channelwise operations) and estimation of task-specific parameters. Extensive experiments demonstrate that attaching parameteric adapters in matrix form to convolutional layers with residual connections significantly boosts the state-of-the-art performance in most domains, especially resulting in superior performance in unseen domains on Meta-Dataset with negligible increase in computations.

More related work. Here we provide more detailed discussion of the most related work. Both CNAPS [31] and Simple CNAPS [2] employ task-specific adapters via FiLM layers (which uses a channelwise affine transformation and connected to the backbone in a serial way) [26] to adapt their feature extractors to the target task and estimate them via an auxiliary network. Compared to them, we propose learning residual adapters in matrix form directly on the support set. SUR [12] and URT [21] learn an attention mechanism to select/fuse features from multiple domain-specific models in meta-train respectively. As we build on a single multi-domain feature extractor, our method does not require such attention but we attach task-specific adapters to the feature extractor to adapt the features to unseen tasks. URL [20] learns a pre-classifier feature mapping to adapt the feature from a single task-agnostic model learned from multiple domains for unseen tasks. While we build on their feature extractor and pre-classifier alignment, the pre-classifier alignment provides very limited capacity for task adaptation, which we address by adapting the feature extractor with adapters at multiple layers. FLUTE [36] follows a hybrid three step approach that first learns the parameters of domain-specific FiLM layers so called templates, employs an auxiliary network to initialize the parameters of a new FiLM layer for unseen task by combining the templates and finetunes them on the small support set. Different from FLUTE, our method learns such adaptation in a single step by learning residual adapters in meta-test.

There are also methods (*e.g.* [11, 33]) that do not fit into task-agnostic and task-specific parameterization grouping. BOHB [33] proposes to use multi-domain data as validation objective for hyper-parameter optimization such that the feature learned on ImageNet with the optimized hyper-parameter generalizes well to multi-domain. CTX [11] proposes to learn spatial correspondences from ImageNet and evaluates on the remaining domains as unseen domains. We also compare our method to them in the setting where we use a standard single domain learning network learned from ImageNet and adapt its representations through residual adapters.

[1] Note that the task-agnostic weights can also be finetuned on the target task (*e.g.* [6, 10]).

2. Method

Few-shot classification aims at learning to classify samples of new categories efficiently from few samples only. Each few-shot learning task consists of a support set $\mathcal{S} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{|\mathcal{S}|}$ with $|\mathcal{S}|$ sample and label pairs respectively and a query set $\mathcal{Q} = \{(\boldsymbol{x}_j)\}_{j=1}^{|\mathcal{Q}|}$ with $|\mathcal{Q}|$ samples to be classified. The goal is to learn a classifier on $\mathcal{S}$ that accurately predicts the labels of $\mathcal{Q}$. Note that this paper focuses on few-shot image classification problem, *i.e.* $\boldsymbol{x}$ and y denote an image and its label.

As in [12, 20, 21], we solve this problem in two steps involving i) representation learning where we learn a task-agnostic feature extractor f from a large dataset $\mathcal{D}_b$, ii) task adaptation where we adapt the task-agnostic representations through various task-specific weights to the target tasks $(\mathcal{S}, \mathcal{Q})$ that are sampled from another large dataset $\mathcal{D}_t$ by taking the subsets of the dataset to build $\mathcal{S}$ and $\mathcal{Q}$. Note that $\mathcal{D}_b$ and $\mathcal{D}_t$ contain mutually exclusive classes.

2.1. Task-agnostic representation learning

Learning task-agnostic or universal representations [4] has been key to the success of cross-domain generalization. Representations learned from a large diverse dataset such as ImageNet [9] can be considered as universal and successfully transferred to tasks in different domains with minor adaptations [12, 21, 28]. We denote this setting as single domain learning (SDL).

More powerful and diverse representations can be obtained by training a single network over multiple domains. Let $\mathcal{D}_b = \{\mathcal{D}_k\}_{k=1}^{K}$ consists of K subdatasets, each sampled from a different domain. The vanilla multi-domain learning (MDL) strategy jointly optimizes network parameters over the images from all K subdatasets:

$$\min_{\phi,\psi_k} \sum_{k=1}^{K} \frac{1}{|\mathcal{D}_k|} \sum_{\boldsymbol{x},y \in \mathcal{D}_k} \ell(g_{\psi_k} \circ f_\phi(\boldsymbol{x}), y), \tag{1}$$

where ℓ is cross-entropy loss, f is feature extractor that takes an image as input and outputs a d dimensional feature. f is parameterized by ϕ which is shared across K domains. g_{ψ_k} is the classifier for domain k and parameterized by ψ_k which is discarded in meta-test. We denote this setting as MDL. The challenge in MDL is to allow efficiently sharing the knowledge across the domains while preventing negative transfer between them and also carefully balancing the individual loss functions ([7]). URL [20], a variant of MDL, mitigates these challenges by first training individual domain-specific networks offline and then distilling their knowledge into a *single* multi-domain network. We refer to [20] for more details.

Another way of obtaining multi-domain representations is to employ multiple domain-specific feature extractors, one for each domain, and adaptively "fuse" their features for each task [12, 22, 36]. While these methods are effective, they require computing features for each image through multiple feature extractors and are thus computationally expensive. Due to its simplicity and effectiveness, we conduct experiments with the feature extractor of URL [20] along with the SDL one.

2.2. Task-specific weight learning

A good task-agnostic feature extractor f_ϕ is expected to produce representations that generalize to many previously unseen tasks and domains. However this gets more challenging when there is a large domain gap between the training set $\mathcal{D}_b$ and test set $\mathcal{D}_t$ which requires further adaptation to the target task. In this work, we propose to incorporate additional capacity to the task-agnostic feature extractor by adding task-specific weights to adapt the representations to the target task by using the support set. Specifically, we directly attach task-specific weights to a learned task-agnostic model, and estimate them from scratch given the support set. We denote the task-specific weights with ϑ and task-adapted classifier with $p_{(\phi,\vartheta)}$ that outputs a softmax probability vector whose dimensionality equals to the number of categories in the support set $\mathcal{S}$.

To obtain the task-specific weights, we freeze the task-agnostic weights ϕ and minimize the cross-entropy loss ℓ over the support samples in meta-test w.r.t. the task-specific weights ϑ [12, 20, 35]:

$$\min_{\vartheta} \frac{1}{|\mathcal{S}|} \sum_{(\boldsymbol{x},y) \in \mathcal{S}} \ell(p_{(\phi,\vartheta)}(\boldsymbol{x}), y), \tag{2}$$

where $\mathcal{S}$ is sampled from the test set $\mathcal{D}_t$. Most previous works freeze the task-agnostic weights but estimate the task-specific weights through an auxiliary network (or a task encoder) [2, 20, 31, 36], where inaccurate prediction of parameters can lead to noisy adaptation and wrong prediction.

2.3. Task-specific adapter parameterization (ϑ)

Task adaptation techniques can be broadly grouped into two categories that aims to adapt the feature extractor or classifier to a given target task. We use α and β to denote task-specific weights for adapting the feature extractor and classifier respectively where $\vartheta = \{\alpha, \beta\}$.

Feature extractor adaptation. A simple method to adapt f_ϕ is finetuning its parameters on the support set [6, 10]. However, this strategy tends to suffer from the unproportionate optimization, *i.e.* updating very high-dimensional weights from a small number of support samples. In this paper, we propose to attach task-specific adapters directly to the existing task-agnostic model, *e.g.* we attach the adapters to each module of a ResNet backbone in Fig. 2 (a), and the adapters can be efficiently learned/estimated from few samples. Concretely, let f_{ϕ_l} denote the l-th layer of the feature

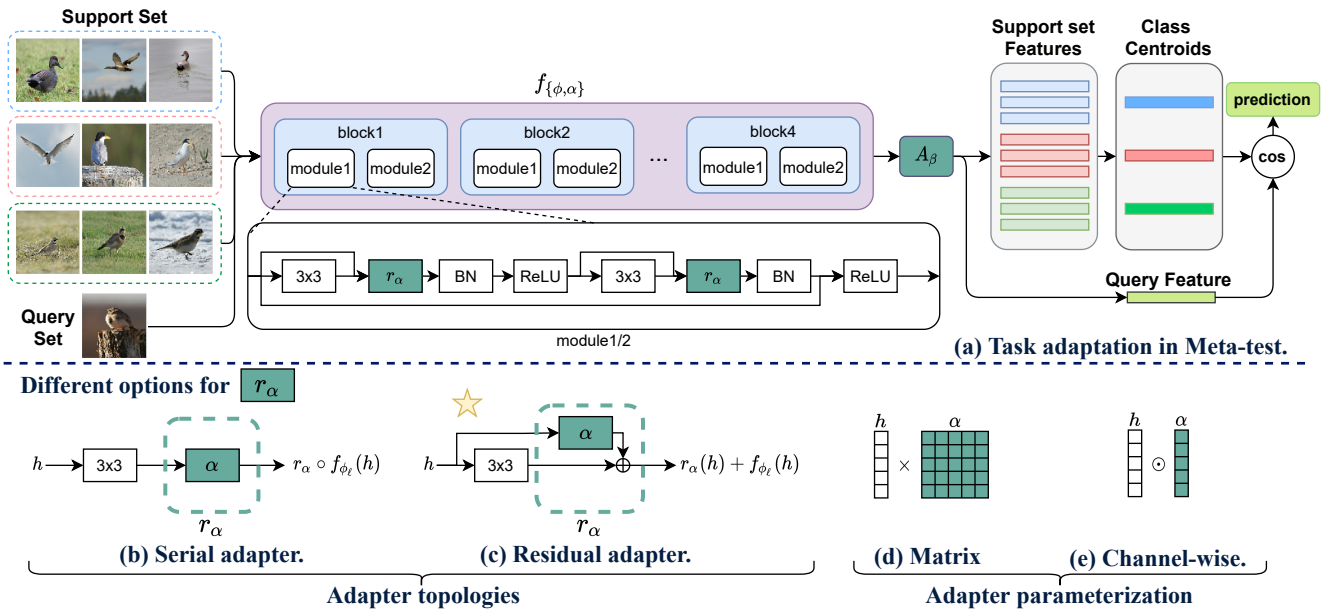

Figure 2. Illustration of our task adaptation for cross-domain few-shot learning. In meta-test stage (a), our method first attaches a parametric transformation r_α to each layer, where α can be constructed by (b) a serial or (c) a residual topology. They can be parameterized with matrix multiplication (d) or channel-wise scaling (e). We found that (c) is the best configuration with matrix parameterization which is further improved by attaching a linear transformation A_β to the end of the network. We adapt the network for a given task by optimizing α and A_β on a few labeled images from the support set, then map query images to the task-specific space and assign them to the nearest class center.

extractor f_ϕ (*i.e.* a convolutional layer) with the weights ϕ_l. Given a support set $\mathcal{S}$, the task-specific adapters r_α parameterized by α, can be incorporated to the output of the layer f_{ϕ_l} as

$$f_{\{\phi_l,\alpha\}}(\boldsymbol{h}) = r_\alpha(f_{\phi_l}(\boldsymbol{h}), \boldsymbol{h}) \tag{3}$$

where $\boldsymbol{h} \in \mathrm{R}^{W\times H\times C}$ is the input tensor, f_{ϕ_l} is a convolutional layer in f_ϕ. Importantly, the number of the task-specific adaptation parameters α are significantly smaller than the task-agnostic ones. The adapters can be designed in different ways.

Next we propose two connection types for incorporating r_α to f_{ϕ_l}: i) serial connection by subsequently applying it to the output of layer $f_{\phi_l}(\boldsymbol{h})$ as

$$f_{\{\phi_l,\alpha\}}(\boldsymbol{h}) = r_\alpha \circ f_{\phi_l}(\boldsymbol{h})$$

which is illustrated in Fig. 2(b), and ii) parallel connection by a residual addition as in [29]

$$f_{\{\phi_l,\alpha\}}(\boldsymbol{h}) = r_\alpha(\boldsymbol{h}) + f_{\phi_l}(\boldsymbol{h})$$

which is illustrated in Fig. 2(c). In our experiments, we found the parallel setting performing the best when α is learned on a support set during meta-test (illustrated in Fig. 2(c)) which we discuss in Sec. 3.

For the parameterization of r_α, we consider two options. Matrix multiplication (illustrated in Fig. 2(d)) with $\alpha \in \mathrm{R}^{C\times C}$:

$$r_\alpha(\boldsymbol{h}) = \boldsymbol{h} * \alpha,$$

where $*$ denotes a convolution, $\alpha \in \mathbb{R}^{C\times C}$ and the transformation is implemented as a convolutional operation with 1×1 kernels in our code. And channelwise scaling (illustrated in Fig. 2(e)):

$$r_\alpha(\boldsymbol{h}) = \boldsymbol{h} \odot \alpha,$$

where $\odot$ is a Hadamard product and $\alpha \in \mathbb{R}^C$. Note that one can also use an additive bias weight in both settings, however, this has not resulted in any significant gains in our experiments. While the matrix multiplication is more powerful than the scaling operation, it also requires more parameters to be estimated or learned. Note that, in a deep neural network, the number of input C_{in} and output channels C_{out} for a layer can be different. In that case, one can still use a non-square matrix: $\alpha \in \mathrm{R}^{C_{out}\times C_{in}}$, however, it is not possible to use a scaling operator in the parallel setting. In our experiments, we use ResNet architecture [14] where most input and output channels are the same. r_α connected in parallel with matrix multiplication form, when its parameters α are learned on the support set, is known as residual adapter [29] and r_α connected serial in channelwise is known as FiLM [26].

An alternative to reduce the dimensionality of α in case of matrix multiplication is matrix decomposition: $\alpha = V\gamma^\top$, where $V \in \mathrm{R}^{C\times B}$ and $\gamma \in \mathrm{R}^{C\times B}$, $B \ll C$. Using a bottleneck, *i.e.* setting $B < C/2$, reduces the number of parameters in the multiplication. In this work, we set $K = [C/N]$ and evaluate the performance for various N in Sec. 3.

Test Dataset	CNAPS [31]	Simple CNAPS [2]	TransductiveCNAPS [1]	SUR [12]	URT [21]	FLUTE [36]	tri-M [22]	URL [20]	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	$\mathbf{59.5 \pm 1.0}$
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	$\mathbf{94.9 \pm 0.4}$
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	$\mathbf{89.9 \pm 0.4}$
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	$\mathbf{81.1 \pm 0.8}$
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	$\mathbf{77.5 \pm 0.7}$
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
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
VGG Flower	88.9 ± 0.5	90.5 ± 0.5	$\mathbf{92.3 \pm 0.4}$	82.9 ± 0.8	87.9 ± 0.6	90.5 ± 0.5	91.6 ± 0.6	92.0 ± 0.5	92.2 ± 0.5
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	$\mathbf{82.8 \pm 1.0}$
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	$\mathbf{57.6 \pm 1.0}$
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	$\mathbf{96.7 \pm 0.4}$
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	$\mathbf{82.9 \pm 0.7}$
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	$\mathbf{70.4 \pm 0.9}$
Average Seen	71.6	73.7	75.1	75.9	77.4	74.5	76.2	**80.4**	**80.4**
Average Unseen	-	67.4	66.5	64.1	62.9	69.9	69.9	70.6	**78.1**
Average All	-	71.2	71.8	71.4	71.8	72.7	73.8	76.6	**79.5**
Average Rank	-	6.1	5.5	5.6	5.5	4.8	4.4	2.5	**1.6**

Table 1. Comparison state-of-the-art methods on Meta-Dataset (using a multi-domain feature extractor of [20]). Mean accuracy, 95% confidence interval are reported. The first eight datasets are seen during training and the last five datasets are unseen and used for test only.

Classifier learning. Finally, the adapted feature extractor $f_{(\phi,\alpha)}$ can be combined with a task-specific classifier c_β, parameterized by β to obtain the final model, *i.e.* $c \circ f_{(\phi,\alpha)}$. Based on the recent works, we investigate use of various linear classifiers in [6, 10, 19, 31], also nonparameteric ones including nearest centroid classifier (NCC) [23, 34] and their variants based on Mahalanobis distance (MD) [2]. Recently, it was shown in [20] that nonparametric classifiers can be successfully combined with a pre-classifier transformation. Concretely, the transformation in [20] that takes in the features computed from the network $f_{\{\phi,\alpha\}} \in \mathrm{R}^d$ and apply an affine transformation $A_\beta : \mathrm{R}^d \rightarrow \mathrm{R}^d$ parameterized by $\beta \in \mathrm{R}^{d \times d}$ to obtain the network embedding that is fed into the classifier, *i.e.* $p_{\phi,\vartheta} = c \circ A_\beta \circ f_{\{\phi,\alpha\}}$. Note that in the case of non-parametric classifier, c is not parameterized by β and we use β to denote the transformation parameters.

In our experiments, the best performing setting uses parallel adapters, whose parameters are in the matrix form, to adapt the feature extractor and followed by the pre-classifier transformation and NCC.

3. Experiments

Here we start with experimental setup, and then we compare our method to the state-of-the-art methods and rigorously evaluate various design decisions. We finally provide further analysis.

3.1. Experimental setup

Dataset. We use the Meta-Dataset [37] which is the standard benchmark for few-shot classification. It contains images from 13 diverse datasets and we follow the standard protocol in [37] (more details in the supplementary).

Implementation details. As in [2, 12, 20], we build our method on ResNet-18 [14] backbone, which is trained over eight training subdatasets by following [20] with the same hyperparameters in our experiments, unless stated otherwise. Once learned, we freeze its parameters and use them as the task-agnostic weights. For learning task-specific weights (ϑ), including the pre-classifier transformation β and the adapter parameters, we directly attach them to the task-agnostic weights and learn them on the support samples in meta-test by using Adadelta optimizer [40].

In the study of various task adaptation strategies in Section 3.3, we consider to estimate the only adapter parameters and learn the auxiliary network parameters by using Adam optimizer as in [2, 31] in meta-train. Note that estimation of pre-classifier and classifier weights via the auxiliary network leads to noisy and poor results and we do not report them. Similarly, we found that the auxiliary network fails to estimate very high-dimensional weights. Hence we only use it to estimate adapter weights that are parameterized with a vector for channelwise multiplication but not with a matrix.

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

We evaluate our method in two settings, with multi-domain or single-domain feature extractor and compare our method to existing state-of-the-art methods. We also evaluate our method incorporated with different feature extractors, *i.e.* SDL, MDL, and URL in the supplementary.

Multi-domain feature extractor. Here we incorporate the proposed residual adapters in matrix form to the multi-domain feature extractor of [20] and compare its performance with the the state-of-the-art methods (CNAPS [31], SUR [12], URT [21], Simple CNAPS [2], Transductive CNAPS [1], FLUTE [36], tri-M [22], and URL [20]) in Tab. 1. To better analyze the results, we divide table into two blocks that show the few-shot classification accuracy in previously seen domains and unseen domains along with their average accuracy. We also report average accuracy over all domains and the average rank as in [20, 36]. Simple CNAPS improves over CNAPS by adopting a simple Mahalanobis

Test Dataset	ResNet-18							ResNet-34		
	Finetune [37]	ProtoNet [37]	fo-Proto-MAML [37]	ALFA+fo-Proto-MAML [37]	BOHB [33]	FLUTE [36]	Ours	ProtoNet [11]	CTX [11]	Ours
ImageNet	45.8 ± 1.1	50.5 ± 1.1	49.5 ± 1.1	52.8 ± 1.1	51.9 ± 1.1	46.9 ± 1.1	$\mathbf{59.5 \pm 1.1}$	53.7 ± 1.1	62.8 ± 1.0	$\mathbf{63.7 \pm 1.0}$
Omniglot	60.9 ± 1.6	60.0 ± 1.4	63.4 ± 1.3	61.9 ± 1.5	67.6 ± 1.2	61.6 ± 1.4	$\mathbf{78.2 \pm 1.2}$	68.5 ± 1.3	82.2 ± 1.0	$\mathbf{82.6 \pm 1.1}$
Aircraft	68.7 ± 1.3	53.1 ± 1.0	56.0 ± 1.0	63.4 ± 1.1	54.1 ± 0.9	48.5 ± 1.0	$\mathbf{72.2 \pm 1.0}$	58.0 ± 1.0	79.5 ± 0.9	$\mathbf{80.1 \pm 1.0}$
Birds	57.3 ± 1.3	68.8 ± 1.0	68.7 ± 1.0	69.8 ± 1.1	70.7 ± 0.9	47.9 ± 1.0	$\mathbf{74.9 \pm 0.9}$	74.1 ± 0.9	80.6 ± 0.9	$\mathbf{83.4 \pm 0.8}$
Textures	69.0 ± 0.9	66.6 ± 0.8	66.5 ± 0.8	70.8 ± 0.9	68.3 ± 0.8	63.8 ± 0.8	$\mathbf{77.3 \pm 0.7}$	68.8 ± 0.8	75.6 ± 0.6	$\mathbf{79.6 \pm 0.7}$
Quick Draw	42.6 ± 1.2	49.0 ± 1.1	51.5 ± 1.0	59.2 ± 1.2	50.3 ± 1.0	57.5 ± 1.0	$\mathbf{67.6 \pm 0.9}$	53.3 ± 1.1	$\mathbf{72.7 \pm 0.8}$	71.0 ± 0.8
Fungi	38.2 ± 1.0	39.7 ± 1.1	40.0 ± 1.1	41.5 ± 1.2	41.4 ± 1.1	31.8 ± 1.0	$\mathbf{44.7 \pm 1.0}$	40.7 ± 1.1	$\mathbf{51.6 \pm 1.1}$	51.4 ± 1.2
VGG Flower	85.5 ± 0.7	85.3 ± 0.8	87.2 ± 0.7	86.0 ± 0.8	87.3 ± 0.6	80.1 ± 0.9	$\mathbf{90.9 \pm 0.6}$	87.0 ± 0.7	$\mathbf{95.3 \pm 0.4}$	94.0 ± 0.5
Traffic Sign	66.8 ± 1.3	47.1 ± 1.1	48.8 ± 1.1	60.8 ± 1.3	51.8 ± 1.0	46.5 ± 1.1	$\mathbf{82.5 \pm 0.8}$	58.1 ± 1.1	$\mathbf{82.7 \pm 0.8}$	81.7 ± 0.9
MSCOCO	34.9 ± 1.0	41.0 ± 1.1	43.7 ± 1.1	48.1 ± 1.1	48.0 ± 1.0	41.4 ± 1.0	$\mathbf{59.0 \pm 1.0}$	41.7 ± 1.1	59.9 ± 1.0	$\mathbf{61.7 \pm 0.9}$
MNIST	-	-	-	-	-	80.8 ± 0.8	$\mathbf{93.9 \pm 0.6}$	-	-	$\mathbf{94.6 \pm 0.5}$
CIFAR-10	-	-	-	-	-	65.4 ± 0.8	$\mathbf{82.1 \pm 0.7}$	-	-	$\mathbf{86.0 \pm 0.6}$
CIFAR-100	-	-	-	-	-	52.7 ± 1.1	$\mathbf{70.7 \pm 0.9}$	-	-	$\mathbf{78.3 \pm 0.8}$
Average Seen	45.8	50.5	49.5	52.8	51.9	46.9	**59.5**	53.7	62.8	**63.7**
Average Unseen	58.2	56.7	58.4	62.4	60.0	53.2	**71.9**	61.1	75.6	**76.2**
Average All	57.0	56.1	57.5	61.4	59.2	52.6	**70.7**	60.4	74.3	**74.9**
Average Rank	7.9	8.3	7.0	5.3	6.0	8.9	**2.8**	5.5	1.8	**1.5**

Table 2. Comparison to state-of-the-art methods on Meta-Dataset (using a single-domain feature extractor which is trained only on ImageNet). Mean accuracy, 95% confidence interval are reported. Only ImageNet is seen during training and the rest datasets are unseen for test only.

distance in stead of learning adapted linear classifier. Transductive CNAPS further improves by using unlabelled test images. SUR and URT fuse multi-domain features to get better performance. FLUTE improves URT by fusing FiLM parameters as initialization which is further finetuned on the support set in meta-test. tri-M adopts the same strategy of learning modulation parameters as CNAPS, where the parameters are further divided into the domain-specific set and the domain-cooperative set to explore the intra-domain information and inter-domain correlations, respectively. URL surpasses previous methods by learning a universal representation with distillation from multiple domains.

From the results, our method outperforms other methods on most domains (10 out of 13), especially obtaining significant improvement on 5 unseen datasets than the second best method, *i.e*. Average Unseen (+7.5). More specifically, our method obtains significant better results than the second best approach on Traffic Sign (+19.5), CIFAR-10 (+8.7), and CIFAR-100 (+6.8). Achieving improvement on unseen domains is more challenging due to the large gap between seen and unseen domain and the scarcity of labeled samples for the unseen task. We address this problem by attaching light-weight adapters to the feature extractor residually and learn the attached adapters on support set from scratch. This allows the model to learn more accurate and effective task-specific parameters (adapters) from the support set to efficiently steer the task-agnostic features for the unseen task, compared with predicting task-specific parameters by an auxiliary network learned in meta-train, *e.g*. Simple CNAPS, tri-M, or fusing representations from multiple feature extractors *e.g*. SUR, URT. Though FLUTE uses a hybrid approach which uses auxiliary networks learned from meta-train to initialize the FiLM parameters for further fine-tuning, their results are not better than URL, which achieves very competitive results as it learns a good universal representation that generalizes well to seen domains and can be further improved with the adaptation strategy proposed in this work, especially significant improvements on unseen domains.

Single-domain feature extractor. We also evaluate our method with a single-domain feature extractor trained on ImageNet only on ResNet-18 as in [37] or ResNet-34 as in [11]. This setting is more challenging than the multi-domain one, as the model is trained only on one domain and tested on both test split of ImageNet but also of other domains. We report the results of our method and state-of-the-art methods (BOHB [33], FLUTE [36], Finetune [37], ProtoNet [37], fo-Proto-MAML [37], and ALFA+fo-Proto-MAML [37], CTX [11]) in Tab. 2. ALFA+fo-Proto-MAML achieves the prior best performance by combining the complementary strengths of Prototypical Networks and MAML (fo-Proto-MAML), with extra meta-learning of per-step hyperparameters: learning rate and weight decay coefficients. FLUTE fails to surpass it with one training source domain, probably due to the lack of FiLM parameters from multiple domains. Our method, when using ResNet18 backbone, outperforms other methods on all domains, especially obtaining significant improvement, *i.e*. Average Unseen (+9.5), on 12 unseen datasets than the second best method. We compare our method to CTX and ProtoNet, which use ResNet-34 backbone. [2] CTX is very competitive by learning coarse spatial correspondence between the query and the support images with an attention mechanism. Ours is orthogonal to CTX and both CTX and our method can potentially be complementary, but we leave this as future work due to high computational cost of CTX. Specifically, we see that our

[2]Note that CTX also uses augmentation strategies such as AutoAugment [8] and other ones from SimClr [5]. We expect applying the same augmentation strategies to our method would yield further improvements, but we leave this for future work.

Test Dataset	classifier	Aux-Net or Ad	serial or residual	M or CW	β	#params	Image -Net	Omni -glot	Air- craft	Birds	Tex- tures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MS- COCO	MNIST	CIFAR -10	CIFAR -100
NCC	NCC	-	-	-	✗	-	57.0	94.4	88.0	80.3	74.6	81.8	66.2	91.5	49.8	54.1	91.1	70.6	59.1
MD	MD	-	-	-	✗	-	53.9	93.8	87.6	78.3	73.7	80.9	57.7	89.7	62.2	48.5	95.1	68.9	60.0
LR	LR	-	-	-	✗	-	56.0	93.7	88.3	79.7	74.7	80.0	62.1	91.1	59.7	51.2	93.5	73.1	60.1
SVM	SVM	-	-	-	✗	-	54.5	94.3	87.7	78.1	73.8	80.0	58.5	91.4	65.7	50.5	95.4	72.0	60.5
Finetune	NCC	-	-	-	✗	-	55.9	94.0	87.3	77.8	76.8	75.3	57.6	91.5	**86.1**	53.1	**96.8**	80.9	65.9
Aux-S-CW	NCC	Aux-Net	serial	CW	✗	76.98%	54.6	93.5	86.6	78.6	71.5	79.3	66.0	87.6	43.3	49.1	87.9	62.8	51.5
Aux-R-CW	NCC	Aux-Net	residual	CW	✗	76.98%	56.1	94.2	88.4	80.6	74.9	82.0	66.4	91.6	48.5	53.5	90.8	70.2	59.7
Aux-S-CW	MD	Aux-Net	serial	CW	✗	76.98%	55.1	93.8	86.8	77.4	73.2	79.9	57.4	88.1	58.4	50.1	92.7	66.5	55.7
Aux-R-CW	MD	Aux-Net	residual	CW	✗	76.98%	54.8	93.8	87.4	78.2	73.4	81.1	58.8	90.1	63.6	48.5	94.8	69.6	60.6
Ad-S-CW	NCC	Ad	serial	CW	✗	0.06%	56.8	94.8	89.3	80.7	74.5	81.6	65.8	91.3	73.9	53.6	95.7	78.4	64.3
Ad-R-CW	NCC	Ad	residual	CW	✗	1.57%	57.6	94.7	89.0	81.2	75.2	81.5	65.4	91.8	79.2	54.7	96.4	79.5	67.4
Ad-S-M	NCC	Ad	serial	M	✗	12.50%	56.2	94.4	89.1	80.6	75.8	81.6	**67.1**	92.1	67.6	54.8	95.9	78.9	66.6
Ad-R-M	NCC	Ad	residual	M	✗	10.93%	57.3	94.9	88.9	81.0	76.7	80.6	65.4	91.4	82.6	55.0	96.6	82.1	66.4
Ad-R-CW-PA	NCC	Ad	residual	CW	✓	3.91%	58.6	94.5	**90.0**	80.5	**77.6**	**81.9**	67.0	**92.2**	80.2	57.2	96.1	81.5	**71.4**
Ad-R-M-PA	NCC	Ad	residual	M	✓	13.27%	**59.5**	**94.9**	89.9	**81.1**	77.5	81.7	66.3	**92.2**	82.8	**57.6**	96.7	**82.9**	70.4

Table 3. Comparisons to methods that learn classifiers and model adaptation methods during meta-test stage based on URL model. NCC, MD, LR, SVM denote nearest centroid classifier, Mahalanobis distance, logistic regression, support vector machines respectively. 'Aux-Net or Ad' indicates using Auxiliary Network to predict α or attaching adapter α directly. 'M or CW' means using matrix multiplication or channel-wise scaling adapters. 'S' and 'R' denote serial adapter and residual adapter, respectively. 'β' indicates using the pre-classifier adaptation. The standard deviation results can be found in the supplementary. The first eight datasets are seen during training and the last five datasets are unseen and used for test only.

method obtains the best average rank and outperforms CTX on most domains (6 out of 10) while our method being more efficient (We train our model on one single Nvidia GPU for around 33 hours while CTX requires 8 Nvidia V100 GPUs and 7 days for training).

3.3. Analysis of task-specific parameterizations

Classifier learning. First we study the adaptation strategies for learning only a task-specific classifier on the pre-trained feature extractor of [20]. We evaluate non-parametric classifiers including nearest cetroid classifier (*NCC*) and NCC Mahalanobis Distance (*MD*) and parametric classifiers including logistic regression (*LR*), support vector machine *SVM* whose parameters are learned on support samples. We also include another baseline with NCC that finetunes all the feature extractor parameters, and report the results in Tab. 3. We observe that NCC obtains the best results for the seen domains and its performance is further improved by MD, while SVM achieves the best for the unseen domains among other classifiers. Finetuning baseline provides competitive results especially for the unseen domains. However, it performs poor in most seen domains.

Feature extractor adaptation. Next we analyze various design decisions for the feature extractor adaptation including connection types (serial, residual), *i.e.* Fig. 2(b), (c), its parameterization including channelwise modulation (*CW*) when they are estimated by an auxiliary network (*Aux-Net*), which has around 77% capacity of the feature extractor. We use with each combination with two nonparameteric classifier, either NCC or MD. *While the adaptation strategies using residual connections performs better than the serial one in almost all cases, the gains are more substantial when generalizing to unseen domains.* Learning adapter weights from few samples only can be very noisy. With residual addition, it is not necessary to change all connections for passing the information forward, which can improve the robustness of useful features and reduce learning burdens for new task, hence increase the generalization ability. While the serial connections may damage the previous learned structures. We also observe that NCC and MD obtain comparable performances. Note that Aux-S-CW with MD corresponds to our implementation of Simple CNAPS [2] with the more powerful feature extractor. We show that replacing its serial connection with a residual one leads to a strong performance boost.

Next we look at the adaptation strategy that learns the task-specific weights directly on the support set as in Eq. (2). We evaluate serial and residual connection types with channelwise and matrix parameterizations by using NCC. We denote this setting as Ad in Tab. 3. Note that we omit MD here, as it produces similar results to NCC. First we observe that learning the weights on the support set outperforms the strategy of estimating them through an auxiliary network almost in all cases. In addition, the learnable weights requires less number of parameters per task, while the capacity of auxiliary network is fixed. We again observe that the residual connections are more effective, especially when used with the matrix parameterization (Ad-R-M). However,the channelwise ones provide a good performance/computation tradeoff. Finally using the pre-classifier alignment (Ad-R-CW-PA and Ad-R-M-PA) further boosts the performance of the best models and we use our best model Ad-R-M-PA to compare against the state-of-the-art.

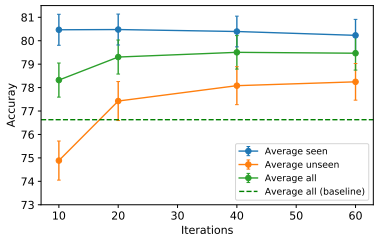

Figure 3. Sensitivity of performance to number of iterations.

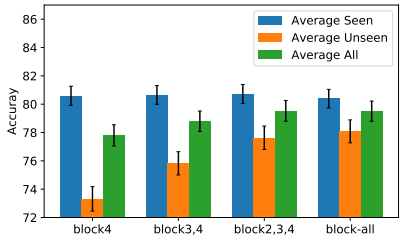

Figure 4. Block (layer) analysis for adapters.

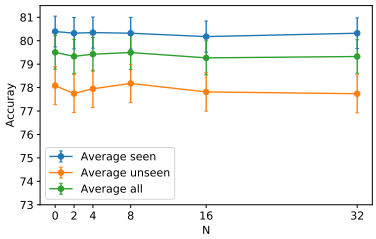

Figure 5. Decomposed residual adapters on block-3,4.

3.4. Further results

Varying-way Five-shot. After evaluating our method over a broad range of varying shots (*e.g.* up to 100 shots), we follow [11, 20] to further analyze our method in 5-shot setting of varying number of categories. In this setting, we sample a varying number of ways with a fixed number of shots to form balanced support and query sets. As shown in Table 4, overall performance for all methods decreases in most datasets compared to results in Table 1 indicating that this is a more challenging setting. It is due to that five-shot setting samples much less support images per class than the standard setting. The top-2 methods remain the same and ours still outperforms the state-of-the-art URL when the number of support images per class is fewer, especially on unseen domains (Average Unseen +6.2).

Five-way One-shot. The similar conclusion can be drawn from this challenging case. Note that there are extremely few samples available for training in this case. As we can see, Ours achieves similar results with URL on seen domains but much better performance on unseen domains due to the learning of attached residual adapters is less over-fitting.

	Varying-Way Five-Shot					Five-Way One-Shot				
Test Dataset	Simple CNAPS [2]	SUR [12]	URT [21]	URL [20]	Ours	Simple CNAPS [2]	SUR [12]	URT [21]	URL [20]	Ours
Average Seen	69.0	71.2	73.8	76.6	**76.7**	65.0	64.0	70.6	73.4	**73.5**
Average Unseen	62.6	56.0	59.6	65.2	**71.4**	57.7	49.6	57.5	62.4	**63.4**
Average All	66.5	65.4	68.3	72.2	**74.6**	62.2	58.5	65.5	69.2	**69.6**
Average Rank	4.1	3.9	3.4	2.1	**1.5**	3.8	4.5	3.3	**1.7**	**1.7**

Table 4. Results of Varying-Way Five-Shot and Five-Way One-Shot scenarios. Mean accuracies are reported and more detailed results can be found in the supplementary.

3.5. Further ablation study

Here, we conduct ablation study for the sensitivity analysis for number of iterations, layer analysis for adapters, and decomposed residual adapters. We summarize results in figures and refer to supplementary for more detailed results.

Sensitivity analysis for number of iterations. In our method, we optimize the attached parameters (α, β) with 40 iterations. Figure 3 reports the results with 10, 20, 40, 60 iterations and indicates that our method (solid green) converges to a stable solution after 20 iterations and achieves better average performance on all domains than the baseline URL (dash green).

Layer analysis for adapters. Here we investigate whether it is sufficient to attach the adapters only to the later layers. We evaluate this on ResNet18 which is composed of four blocks and attach the adapters to only later blocks (block4, block3,4, block2,3,4 and block-all, see Fig. 2). Figure 4 shows that applying our adapters to only the last block (block4) obtains around 78% average accuracy on all domains which outperforms the URL. With attaching residual adapters to more layers, the performance on unseen domains is improved significantly while the one on seen domains remains stable.

Decomposing residual adapters. Here we investigate whether one can reduce the number of parameters in the adapters while retaining its performance by using matrix decomposition (see Sec. 2). As in deep neural network, the adapters in earlier layers are relatively small, we then decompose the adapters in the last two blocks only where the adapter dimensionality goes up to 512×512. Figure 5 shows that our method can achieve good performance with less parameters by decomposing large residual adapters, (*e.g.* when $N = 32$ where the number of additional parameters equal to around 4% vs 13%, the performance is still comparable to the original form of residual adapters, *i.e.* N=0). We refer to supplementary for more details.

4. Conclusion and Limitations

In this work, we investigate various strategies for adapting deep networks to few-shot classification tasks and show that light-weight adapters connected to a deep network with residual connections achieves strong adaptation to new tasks and domains only from few samples and obtains state-of-the-art performance while being efficient in the challenging Meta-Dataset benchmark. We demonstrate that the proposed solution can be incorporated to various feature extractors with a negligible increase in number of parameters.

Our method has limitations too. We build our method on existing backbones such ResNet-18 and ResNet-34, employ fixed adapter parameterizations and connection types which may not be optimal for every layer and task in multi-domain few-shot learning. Thus it would be desirable to have more flexible adaptor structures that can be altered and tuned based on the target task.

References

[1] Peyman Bateni, Jarred Barber, Jan-Willem van de Meent, and Frank Wood. Enhancing few-shot image classification with unlabelled examples. *arXiv preprint arXiv:2006.12245*, 2020. 5

[2] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *CVPR*, pages 14493–14502, 2020. 2, 3, 5, 7, 8

[3] Luca Bertinetto, João F Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531, 2016. 2

[4] Hakan Bilen and Andrea Vedaldi. Universal representations: The missing link between faces, text, planktons, and cat breeds. *arXiv preprint arXiv:1701.07275*, 2017. 3

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 6

[6] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. *arXiv preprint arXiv:2003.04390*, 2020. 2, 3, 5

[7] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *ICML*, pages 794–803. PMLR, 2018. 3

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *CVPR*, pages 113–123, 2019. 6

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 3

[10] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *ICLR*, 2020. 2, 3, 5

[11] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *NeurIPS*, 2020. 2, 6, 8

[12] 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. 2, 3, 5, 8

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICLR*, pages 1126–1135, 2017. 1

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 4, 5

[15] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020. 1

[16] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29:667–675, 2016. 2

[17] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the annual meeting of the cognitive science society*, volume 33, 2011. 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. 2

[19] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665, 2019. 2, 5

[20] Wei-Hong Li, Xialei Liu, and Hakan Bilen. Universal representation learning from multiple domains for few-shot classification. *ICCV*, 2021. 2, 3, 5, 7, 8

[21] Lu Liu, William Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle. A universal representation transformer layer for few-shot image classification. In *ICLR*, 2021. 2, 3, 5, 8

[22] 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. 3, 5

[23] Thomas Mensink, Jakob Verbeek, Florent Perronnin, and Gabriela Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *TPAMI*, 35(11):2624–2637, 2013. 5

[24] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *CVPR*, volume 1, pages 464–471. IEEE, 2000. 1

[25] Boris N Oreshkin, Pau Rodriguez, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018. 1

[26] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings*

of the AAAI Conference on Artificial Intelligence, 2018. 2, 4

[27] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016. 1

[28] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017. 3

[29] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *CVPR*, pages 8119–8127, 2018. 4

[30] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018. 2

[31] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *NeurIPS*, 2019. 2, 3, 5

[32] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2020. 1

[33] Tonmoy Saikia, Thomas Brox, and Cordelia Schmid. Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*, 2020. 2, 6

[34] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017. 1, 5

[35] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *ECCV*, 2020. 3

[36] Eleni Triantafillou, Hugo Larochelle, Richard Zemel, and Vincent Dumoulin. Learning a universal template for few-shot dataset generalization. In *ICML*, 2021. 2, 3, 5, 6

[37] 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. In *ICLR*, 2020. 2, 5, 6

[38] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, 2016. 1, 2

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

[40] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012. 5