Exploring Efficient Few-shot Adaptation for Vision Transformers

Chengming Xu  
School of Data Science, Fudan University  
cmxu18@fudan.edu.cn

Siqian Yang  
Yabiao Wang  
YouTu Lab, Tencent  
seasonsyang@tencent.com  
caseywang@tencent.com

Zhanxiong Wang  
Tencent  
mazzzwang@tencent.com

Yanwei Fu*  
Xiangyang Xue  
School of Data Science, Fudan University  
yanweifu@fudan.edu.cn  
xiangyangxue@fudan.edu.cn

Reviewed on OpenReview: https://openreview.net/forum?id=n3qLz4eL1l

Abstract

The task of Few-shot Learning (FSL) aims to do the inference on novel categories containing only few labeled examples, with the help of knowledge learned from base categories containing abundant labeled training samples. While there are numerous works into FSL task, Vision Transformers (ViTs) have rarely been taken as the backbone to FSL with few trials focusing on naïve finetuning of whole backbone or classification layer. Essentially, despite ViTs have been shown to enjoy comparable or even better performance on other vision tasks, it is still very nontrivial to efficiently finetune the ViTs in real-world FSL scenarios. To this end, we propose a novel efficient Transformer Tuning (eTT) method that facilitates finetuning ViTs in the FSL tasks. The key novelties come from the newly presented Attentive Prefix Tuning (APT) and Domain Residual Adapter (DRA) for the task and backbone tuning, individually. Specifically, in APT, the prefix is projected to new key and value pairs that are attached to each self-attention layer to provide the model with task-specific information. Moreover, we design the DRA in the form of learnable offset vectors to handle the potential domain gaps between base and novel data. To ensure the APT would not deviate from the initial task-specific information much, we further propose a novel prototypical regularization, which minimizes the similarity between the projected distribution of prefix and initial prototypes, regularizing the update procedure. Our method receives outstanding performance on the challenging Meta-Dataset. We conduct extensive experiments to show the efficacy of our model. Our code is available at https://github.com/loadder/eTT_TMLR2022.

1 Introduction

Modern computer vision models such as ResNet (He et al. 2016) and Faster R-CNN (Ren et al. 2015) are trained on large-scale training sets, and not well generalize to handle the long tail categories with few

*This paper is supported by the project NSFC(62076067).
labeled samples. Few-shot Learning (FSL) has thus been studied to make inference on insufficiently-labeled novel categories typically with the transferable knowledge learned from base categories which are provided with abundant labeled training samples. Essentially, the FSL can be taken as representation learning, as its backbones should ideally extract features representative and generalizable to various novel tasks. Currently Convolutional Neural Networks (CNNs), especially ResNet, are the predominant backbone and widely utilized in most existing FSL works (Ravi & Larochelle, 2017; Finn et al., 2017; Nichol et al., 2018; Li et al., 2017; Sun et al., 2019).

Recently, by taking the merits of Multi-headed Self-Attention (MSA) mechanism and Feed Forward Network (FFN), the transformers have been widely used in the recognition (Alexey et al.; Liu et al., 2021b), detection (Beal et al., 2020) and image editing (Cao et al., 2021). The general pipeline of Pretrain-(Meta-train)-Finetune has been explored in few ViTs on FSL (Hu et al., 2022; Evci et al., 2022; Abnar et al.), recently. Particularly, the ViT models are first pretrained/meta-trained on a large-scale dataset. Then a test-time finetune procedure is set up for each target task on novel data. The finetuning strategy can be generally categorized into linear classifier probing and backbone tuning: the former one optimizes the reasonable decision boundaries by the fixed embeddings, while the latter one considers the adaptation of both embedding space and classifier.

In this paper we focus on the backbone tuning method. (Hu et al., 2022) shows that the naïve Pretrain-Meta-train-Finetune (P>M>F) baseline can generally have satisfactory performance in FSL. Unfortunately, it involves heavy computations and potential overfitting in FSL setting. Particularly, (1) It typically demands extraordinary computing power to formulate episodes from a large number of support classes to update the whole network parameters. Thus it is less efficient in many real-case applications. For example, the edge devices such as mobiles donot have enough computational power to adapt all model parameters by personalized/specialized data collected on these devices. (2) It is very subtle and difficult to directly fine-tune trained deep models on one or two labeled instances per class, as such few-shot models will suffer from severe overfitting (Snell et al., 2017; Fei-Fei et al., 2006; Brian et al.). By contrast, humans have the ability of conducting few-shot recognition from even single example of unseen novel category with very high confidence. Such problems may be the culprit of the phenomenon that their proposed finetune strategy only works on part of datasets and has less effect to the others. This suggests their limited usage of ViT backbone for any potential FSL applications. An alternative choice is to finetune specific layers in a ViT model with much smaller tunable parameters (ViT-s block in Fig. 1(a)). Such a strategy nevertheless can only finetune either low-level or high-level features, leading to inferior performance in many cases. Therefore it is desirable to have an efficient and light-weighted ViT tuning method that shall not only avoid overfitting to small training samples, but also achieve high performance of FSL.

In this paper, we present a novel efficient Transformer Tuning (eTT) for few-shot learning task, which adopts a pretrain-finetune pipeline. To pretrain our transformer, we advocate utilizing the recent self-supervised method - DINO (Caron et al., 2021). Our key novelties are in the finetuning stage. As illustrated in Fig. 1(b), we propose Attentive Prefix Tuning (APT) and Domain Residual Adapter (DRA) as the key components to our eTT, to efficiently learn the newly-introduced tunable parameters over novel support sets. Specifically, we formulate the attentive prototypes by aggregating patch embeddings with the corresponding attention weights of the class token for each image, so as to provide the model with abundant task-specific information and guide each self-attention layer to aggregate more class-related features. To encourage the prefix to keep the prior knowledge from initial prototypes, we further propose a novel prototypical regularization which restricts the relationship between the prefix and prototypes by optimizing the similarity of their projected distributions. Moreover, we propose to additionally adopt a light-weighted domain residual adapter in the form of learnable offset to deal with the potential failure of APT on large domain gaps. Extensive experiments are conducted to evaluate our eTT: we use the ViT-tiny and ViT-small backbones on the large-scale Meta-Dataset (Triantafillou et al., 2019) consisting of ten sub-datasets from different domains; and the results show that our model can achieve outstanding performance with comparable or even much fewer model parameters. Thus our eTT is a promising method on efficiently finetuning ViTs on the FSL tasks.

Our paper has the following contributions.

1. In order to solve the problem of inefficiency and make better use of ViT in FSL, we propose a novel
finetuning method named efficient Transformer Tuning (eTT).
2. Inspired by recent advance in language model, a novel attentive prefix tuning is presented utilizing the attentive prototypes to embed the task-specific knowledge into pretrained ViT model. Particularly, we propose a new initialization strategy tailored for FSL by leveraging prototypical information from the self-attention layers. Moreover, a novel domain residual adapter is repurposed to handle the various domain gaps between training and testing data.
3. We introduce a prototypical regularization term which can constrain the update procedure of prefix during finetuning to maintain the initial task-specific knowledge.
4. By utilizing the proposed eTT, our ViT models receive remarkable performance on Meta-Dataset, overpassing the existing ResNet-based methods without using additional training data. More importantly, both of the model scale and efficiency of our method are comparable with the other competitors, indicating the promising application of ViTs in FSL.

2 Related Works

**Few-shot recognition.** FSL learns transferable knowledge from base classes and adapt it to a disjoint set (novel classes) with limited training data. Among those FSL tasks, few-shot image recognition is the one with most focus and researches. Existing works can be grouped into two main categories. One is optimization-based methods [Ravi & Larochelle, 2017; Finn et al., 2017; Nichol et al., 2018; Li et al., 2017; Sun et al., 2019], which learn parameters that can be better finetuned on few-shot support sets. The other is metric-based methods such as ProtoNet [Snell et al., 2017], RelationNet [Sung et al., 2018], CAN [Hou et al., 2019], DMF [Xu et al., 2021], OSOC [Luo et al., 2021] and CTX [Doersch et al., 2020], which solve FSL by applying an existing or learned metric on the extracted features of images. Particularly, CTX [Doersch et al., 2020] builds up a cross attention module which interacts between query and support images to adaptively aggregate better prototypes than simply averaging all support features. While these methods perform well on classical few-shot learning settings, most of them adopt convnet as backbone, especially ResNet [He et al., 2016]. We, on the opposite, try to make full use of another widely-applied structure, i.e. ViT, in FSL, which requires extra design for training and finetuning strategy.

**Transformer in vision tasks.** Transformers widely utilize the self-attention mechanism which originally are employed to process the feature sequence in Vaswani et al. (2017). Then large scale transformers become increasingly popular in NLP tasks to build complex language models, and also extend to vision tasks.
et al., Yuan et al., 2021, Liu et al., 2021b) by formulating the token sequence with image patches processed with position embedding. It has been shown the efficacy in various applications, such as (Liu et al., 2021a) for image caption, (Sun et al., 2020) for multiple object tracking and (Esser et al., 2021; Cao et al., 2021) for image inpainting and editing. Critically, ViTs is typically trained by very large-scale dataset, and few effort has been dedicated in training or finetuning on few-shot supervision. We follow the pretrain-meta-train-finetune pipeline (Hu et al., 2022), while their method finetune the whole ViTs on few-shot examples, and thus has less efficiency and can easily overfit. In contrast, our proposed eTT has the key components of DRA and APT, demanding much less tunable parameters with much better performance.

**Finetuning algorithm for ViT.** The idea of finetuning ViTs on small-scale datasets has been partly investigated in Natural Language Processing (NLP) communities. Houlsby et al. (2019) proposed to attach two learnable bottleneck adapters to each transformer layer. Other works (Xiang & Percy; Brian et al., 2022) make use of the prompt which trains a small task-specific prompt for each task so that the prompt can guide the model with knowledge corresponding to the task. Such a prompting idea from NLP is inherited and repurposed to finetune a learnable prefix for each novel episode in this paper. However, these works (Xiang & Percy; Brian et al.; Houlsby et al., 2019) initialize the prefix or prompt with word embeddings which is not available in our problem. Instead, we propose an attentive prototype with regularization initializing the visual prefix with object-centric embeddings. Additionally, we notice that a very good concurrent technical report (Hua et al., 2022) also studies finetuning visual prompt for pretrained ViTs in downstream tasks. We highlight the two key differences from our eTT. The first is about the initialization. While initialization strategy does not matter in their method and the corresponding tasks, we show in our experiments that randomly initializing prefix does lead to sub-optimal performance in FSL, which leads to the necessity of a well-designed initialization. The second is that we further propose a regularization term to restrict the prefix, which has never been studied in existing works.

**Task-specific Adapter.** The idea of task-specific adapter has been explored in several works like (Li et al., 2022; Rebuffi et al., 2017; Rusu et al., 2018; Sun et al., 2019; Li et al., 2022; Requeima et al., 2019) to adapt CNNs to learn the whole information from support set. Besides, (Bateni et al., 2019; Bateni et al., 2020) adopt Feature-wise Linear Modulation (FiLM) layers (Perez et al., 2018) to adapt task-specific information into networks. In contrast, we repurpose the adapter as the domain residual to update transformer blocks in a more light-weighted way with less learnable parameters. Beyond different structures, our proposed DRA intrinsically serves as the domain adapter rather than meta-learner for the FSL in (Rusu et al., 2018; Sun et al., 2019; Li et al., 2022; Requeima et al., 2019). While these previous works require meta-training to optimize their adaptation modules, our method simply utilizes the novel support data to learn the DRA, thus reducing the training cost. Furthermore, our DRA is mostly tuned to bridge the visual domain gap between base and novel categories, thus improving the learning of APT on each episode task.

### 3 Methodology

#### 3.1 Problem Setup

We formulate few-shot learning in the meta-learning paradigm. In general, we have two sets of data, namely meta-train set $D_s = \{(I_i, y_i), y_i \in C_s\}$ and meta-test set $D_t = \{(I_i, y_i), y_i \in C_t\}$ which contain the base and novel data respectively and are possibly collected from different domains. $C_s$ and $C_t$ ($C_s \cap C_t = \emptyset$) denote base and novel category sets. FSL aims to train a model on $D_s$ which is generalizable enough on $D_t$. In the testing phase, the model can learn from few labelled data from each category of $C_t$.

While most previous FSL works (Snell et al., 2017; Sung et al., 2018) utilize the setting of $N$-way $K$-shot in mini-ImageNet, i.e., $K$ training samples from $N$ class, we follow CTX (Doersch et al., 2020) to adopt the setting on the large-scale Meta-Dataset (Triantafillou et al., 2019). In each episode $T$, $N$ is first uniformly sampled from $[5, N_{max}]$ where $N_{max}$ equals to $\min(50, |C_s|)$ or $\min(50, |C_t|)$ on training or testing stage, accordingly. $N$ is supposed to be accessible knowledge during both training and testing. In the most naive case, one can get $N$ by directly counting the number of support classes. From each of the sampled category, $M$ query samples per category are randomly selected, and thus constructing the query set $Q = \{(I^q_i, y_i^q)\}_{i=1}^{N_Q}$. After that random amount of samples are taken from the rest of samples belonging to these categories to
form the support set $S = \{(I_{i}^{\text{supp}}, y_{i}^{\text{supp}})\}_{i=1}^{N_S}$. Note that compared to the classical $N$-way $K$-shot setting, such a setting generates class-imbalanced support sets, and different episodes contain different numbers of support samples. This is much more challenging to the model and learning algorithms, as they shall handle both extremely large and small support sets.

### 3.2 Overview of Our Method

To handle the optimization of various episodes on large-scale dataset, we present our novel finetuning model – efficient Transformer Tuning (eTT) as shown in Fig. 2. Our eTT follows the pipeline in Hu et al. (2022), and has key stages of the pretraining and finetuning. We employ DINO as pretraining, and conduct the task tuning by attentive prefix tuning (Sec. 3.4), and backbone tuning with domain residual adapter (Sec. 3.5).

#### Pre-training

As previous work (Hu et al., 2022) shows the importance of self-supervised pre-training to learning vision transformer models, we adopt the same principle and introduce the self-supervised learning model to pre-train our ViT backbone on base data. Specifically, we utilize the recent State-of-the-art self-supervised ViT models – DINO (Caron et al., 2021) to pretrain our model. DINO builds up supervision based on a self-distillation framework by using the multi-crop strategy (Caron et al., 2020). As we will show in our experiments, such a pre-trained model shall have good cluster property even among cross domain images, potentially benefiting our following FSL stages. Note that different from (Hu et al., 2022) which takes an off-the-shelf model pretrained with DINO on full ImageNet, we strictly follow the FSL protocols to retrain the DINO models on the meta-train split in the target dataset to avoid the abuse of extra data.

One would ask whether it is necessary to make use of the annotations for base data, since supervised pretrain has been proven to be effective in many previous FSL works (Ye et al., 2020; Hou et al., 2019). As we will show in the experiments, an additional finetuning with image labels on base data cannot bring consistent improvement and even makes it worse on most datasets, which may be caused by the overfitting on the image labels leads to less generalization ability across different domains. Moreover, compared with vanilla supervised training, the attention maps for models trained by DINO contain more semantic information, which we will utilize in the following context.

#### 3.3 Preliminary: Vanilla Test-time Finetuning

Before fully developing our fine-tuning contributions, we review the simple and effective finetuning method named LT+NCC (Li et al., 2021). The novel modules proposed by us in the following context are all adopted together with this simple baseline method. Given a ViT backbone $f_{\theta}$ that is parameterized by $\theta$ and an
episode $\mathcal{T}$, the support features $\{x_i^{\text{supp}}\}_{i=1}^{N_S}$, where $x_i^{\text{supp}} = f_\theta(I_i^{\text{supp}})$, are extracted from the support set $\{I_i^{\text{supp}}\}_{i=1}^{N_S}$. Then, a learnable linear transformation $\phi$ is added to the model to realize the adaptation, which results in the final support features used for classification $\{\hat{x}_i^{\text{supp}}\}_{i=1}^{N_S}$, where $\hat{x}_i^{\text{supp}} = \phi(x_i^{\text{supp}})$. The prediction of these support images can thus be calculated based on the similarity between the transformed features and the aggregated prototypes as,

$$
\bar{x}_c = \frac{1}{\sum_{i=1}^{N_S} \mathbf{1}_{y_i^{\text{supp}} = c}} \sum_{i=1}^{N_S} \hat{x}_i^{\text{supp}} \mathbf{1}_{y_i^{\text{supp}} = c} 
\hat{y}_i^{\text{supp}}(c) = \frac{\exp(d(\hat{x}_i^{\text{supp}}, \bar{x}_c))}{\sum_{c=1}^{N} \exp(d(\hat{x}_i^{\text{supp}}, \bar{x}_c))}
$$

where $d$ denotes cosine similarity, i.e., $d(a, b) = \frac{a^T b}{\|a\| \|b\|}$. We fix all of the parameters in the original backbone, and adopt the cross entropy loss to optimize the transformation $\phi$. Precisely speaking, for each support image $I_i^{\text{supp}}$ together with its annotation $y_i^{\text{supp}}$, the objective function is as following:

$$
\ell_{CE} = -y_i^{\text{supp}} \cdot \log \hat{y}_i^{\text{supp}}
$$

After finetuning, $\phi$ is applied to query features and the same procedure as above is performed between the processed query features $\{\tilde{x}_m^i\}$ and prototypes $\{\bar{x}_c\}_{c=1}^{N}$ for the inference of each episode.

### 3.4 Task Tuning by Attentive Prefix Tuning

We finetune the pre-trained ViT with support set via an attentive prefix tuning strategy. Specifically, a prefix matrix $\theta_P \in \mathbb{R}^{N_P \times d}$ is first initialized, where $N_P$ denotes the number of prefix. Then a bottleneck $g$ is added upon $\theta_P$ to produce $\tilde{\theta}_P \in \mathbb{R}^{N_P \times (2Ld)}$, where $L$ denotes the number of backbone layers. The $g$ plays the same role as the projector in each self-attention layer, except that all layers share the same module. The produced $\tilde{\theta}_P$ can be reshaped and seen as $L$ value and key pairs $\{\theta_v^l, \theta_k^l\}_{l=1}^L$, $\theta_v^l, \theta_k^l \in \mathbb{R}^{N_P \times d}$. The MSA block in the $L$-th layer can then be reformed by attaching these new pairs to the original key and value sequences:

$$
A^l = \text{Attn}(Q_r, [K; \theta_k^l]) \quad \text{output} = A^l [V; \theta_v^l]
$$

where $[; ;]$ denotes concatenation, $\text{Attn}$ denotes the calculation of MSA matrices. In this way, the prefix can affect the attention matrix $A^l$ and result in different output features from the original ones.

**Remark.** Compared with the naive strategy that finetunes specific layers in ViT (ViT-s block in Fig. 1(a)) which can only adjust part of blocks, the prefix can evenly adapt each layer’s image embedding with almost the same parameter size as one transformer layer, as shown in Tab. 1(a). By fixing the model parameters and optimizing the prefix $\theta_P$ and the transformation module $g$, the support knowledge can be smoothly embedded into the prefix, which further helps the task adaptation.

**Attentive Prototype.** The initialization of the prefix is very important to our APT, as it greatly boosts the performance. Critically, quite different from the prefix or prompt tuning in NLP and visual-context tasks that have task-specific instructions explicitly as word embedding sequences, each episode in our FSL only has the few support images and their labels. Thus, rather than steering the model with ‘what should be done’ as in [Xiang & Percy], our APT shall provide the model with ‘what we have globally’ by leveraging the class-specific information. Thus, the attentive prototype is presented to aggregate the image embeddings with object-centric attention, as the initialization of the prefix. Particularly, each support image $I_i^{\text{supp}}$ is first transformed to a patch embedding sequence $\{\tilde{x}_m^{\text{supp}}\}_{m=1}^{P}$ with the starting patch embedding layer,

$$
\tilde{x}_m^{\text{supp}} = f_{\theta_{pe}}(I_m^{\text{supp}}) + E_m^{\text{pos}}
$$

where $m = 1, \cdots, P^2$ is the patch index; $f_{\theta_{pe}}$ denotes the patch embedding layer which is typically a convolutional layer whose kernel size equals to patch size; and $E_m^{\text{pos}}$ indicates the position embedding. Meanwhile, we can get unnormalized attention score $A \in \mathbb{R}^{h \times P}$ between the class token and image patches from the last MSA layer, where $h$ denotes number of heads in each MSA module. **Such an attention vector** can focus on the foreground in the image, especially for models trained with DINO [Caron et al., 2021], with each head indicating a particular part or an object. We can thus get the initial image-level representation

$$
\hat{A} = \sigma(A) \quad \tilde{x}_m^{\text{supp}} = \frac{1}{h} \sum_{n=1}^{h} \sum_{m=1}^{P^2} \hat{A}_{nm} \tilde{x}_m^{\text{supp}}
$$
where \( \sigma \) is softmax function. Compared with simply averaging all patch embeddings, the attentive embeddings can highlight the objects of interest and suppress the background information. Then the prototypes \( \bar{x} \) can be calculated by averaging the attentive image embeddings belonging to each support category. We set the number of prefix as \( N \), which is available during testing for each episode, and initialize the prefix with \( \bar{x} \).

**Remark.** In this way, commonly-used prototypes can provide the model with comprehensive information about the episode. Also such a first-order statistics is comparable with the normal patch features among the layers. This can benefit the training with more stability. When \( N \) is large, more prefix are required to fully learn the information included by each episode. On the other hand, when \( N \) is small so that the episode is relatively easy, fewer prefix can handle the support knowledge without trouble while decreasing the computing debt.

### 3.5 Backbone Tuning by Domain Residual Adapter

Finetuning few-shot tasks by APT will make a good balance between performance and efficiency. To further improve the model generalization ability on different domains, we further propose the backbone tuning by leveraging the Domain Residual Adapters (DRA), as illustrated in Fig. 2. Specifically, for the \( l \)-th transformer layer, we attach two learnable offset vectors \( \delta_l^a, \delta_l^f \in \mathbb{R}^d \) to the MSA and FFN. After features are processed with MSA and FFN, the corresponding offsets are added to them so that the extreme domain gap can be neutralized. These offsets are expected to represent the gap between source and target domains, and transfer the original manifold to a more appropriate one.

### 3.6 Loss Functions

**Prototypical Regularization.** In addition to the cross entropy loss in Eq. 2 we propose a novel prototypical regularization to ensure the class-specific information, which is embedded in the prefix via initialization, can be maintained during update. The knowledge in attentive prototypes is distilled to the prefix during finetuning. Concretely, in each iteration, the prototypes \( \bar{x} \) and prefix \( \theta_P \) are first projected to a latent space via a projector module \( \psi \), which produces \( \bar{x}' \) and \( \theta'_P \) respectively. Then the distillation loss is computed using these two embeddings as,

\[
\ell_{\text{dist}} = \frac{1}{N} \sum_{n=1}^{N} H(\bar{x}'_n, \theta'_P)
\]

(6)

where \( H(a, b) = -a \log b \). The above objective function can ensure the prototype of each category and the corresponding prefix contain consistent information, which is indicated by the similarity between distributions after projection. To make training more stable and avoid collapse, for each episode we maintain an exponential moving average (EMA) of \( \bar{x}' \) as the center variable \( c_{\text{center}} \). Before calculating \( \ell_{\text{dist}} \), we standardize \( \bar{x}' \) as \( \sigma(\bar{x}' - \bar{x}_{\text{center}}) \), where \( \sigma \) denotes softmax function and \( \tau \) is the temperature typically set as 0.04.

Once having both of the above losses calculated, we can optimize the model parameters including the DRA, the prefix together with the transformation \( g \) and the projector \( \psi \), with the following objective function:

\[
\mathcal{L} = \ell_{\text{CE}} + \lambda \ell_{\text{dist}}
\]

(7)

where the scalar weight \( \lambda \) controls the strength of the regularization.

**Remarks.** For a ViT with \( L \) layers, \( n_h \) heads and \( d \) feature dimension, the size of trainable parameters is \((N + d' + d_{\text{proj}} + d) d + 2(d' + 1) L d\), where \( d' \) is the hidden dimension for transformation module \( g \) and \( d_{\text{proj}} \) denotes output dimension for the projector \( \psi \), which is much smaller than that of the whole backbone model. Specifically, the learnable modules during finetuning have only about 9% parameters with regard to the whole transformer model when using ViT-small and ViT-tiny.


4 Experiments

4.1 Experimental Setup

Datasets. We use Meta-Dataset (Triantafillou et al., 2019) – the most comprehensive and challenging large-scale FSL benchmark. It has 10 sub-datasets such as ImageNet (Deng et al., 2009) and Omniglot (Lake et al., 2015), with various domain gaps. Our experiments are conducted under the single training source setting, i.e. only ImageNet is used for training, and the meta-test split of all ten datasets for evaluation. Some of the test datasets such as CUB share similar or highly-related categories with ImageNet, while the others have greater domain gaps. Note that Hu et al. (2022) claims pretraining on all images in the training set of ImageNet is reasonable for introducing extra data and boosting the performance. However, such a strategy utilizes much more training samples (1.28M images, 1000 classes in ImageNet v.s. 0.9M images, 712 classes in meta-train split of ImageNet). Empirically so many additional images can greatly benefit generalization ability of self-supervised learning methods. Therefore to make a more fair comparison, we strictly follow the experiment protocol used in CTX (Doersch et al., 2020) and shall not use any extra data even in the unsupervised pretraining stage. We resize all images to \(224 \times 224\) for ViT-small and \(84 \times 84\) for ViT-tiny.

Implementation details. We set the patch size as 8 for ViT-tiny (as it has small input image size), and keep the other hyper-parameters as default. We adopt standard ViT-small with 12 layers, 6 attention heads, feature dimension as 384 and patch size as 16. We strictly follow the hyper-parameter setting and data augmentation in DINO (Caron et al., 2021) for pretraining. In test-time finetuning, we empirically set the hidden dimension \(d'\) of the transformation module as \(d/2\), and output dimension \(d_{proj}\) of the projector as 64 for all datasets. We utilize AdamW optimizer finetuning, with learning rate set as \(1 \times 10^{-3}\) for TrafficSign and \(5 \times 10^{-4}\) for other datasets. \(\lambda\) is set as 0.1. For simplicity, the selection of hyper-parameters is conducted on the meta-validation set of ImageNet, which is the only within-domain setting in Meta-Dataset.

Evaluation benchmark. We report the accuracy of randomly sampled 600 episodes for each dataset and the average accuracy when comparing with the existing methods. The comprehensive comparison of both accuracy and 95% confidence interval is in Appendix.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Image size</th>
<th>Params(M)</th>
<th>FLOPs(G)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res18</td>
<td>84×84</td>
<td>11.69</td>
<td>1.82</td>
</tr>
<tr>
<td>ViT-tiny</td>
<td>84×84</td>
<td>5.38</td>
<td>0.72</td>
</tr>
<tr>
<td>Res34</td>
<td>224×224</td>
<td>21.80</td>
<td>3.68</td>
</tr>
<tr>
<td>ViT-small</td>
<td>224×224</td>
<td>21.97</td>
<td>4.61</td>
</tr>
</tbody>
</table>

Table 1: Comparison of parameter size and FLOPs between different backbones.

4.2 Comparison with State-of-the-art Methods

Before the comprehensive comparison, it is necessary to show the comparison between different backbone is fair enough since our backbone model is not the same as the existing method. Therefore we present the comparison of size of model parameters and FLOPs in Tab. 1, in which the FLOPs of all models are computed by fvcore. The results show that (1) compared with Res18, ViT-tiny is a much smaller and efficient model, and (2) ViT-small is approximately comparable to Res34. In this way, the comparison of our proposed method with state-of-the-art methods is reasonable and fair.

We compare our model with ProtoNet (Snell et al., 2017), CTX (Doersch et al., 2020), TSA (Li et al., 2022), etc. These methods take the backbones of ResNet18 or ResNet34. Also, the pretrain-meta-train-finetune baseline (P>M>F) (Hu et al., 2022) is not considered in computing average rank since extra data is used. As in Tab. 2 when using ViT-tiny as backbone whose parameter size is comparable to that of ResNet34, our model receives 1.6 average rank on all dataset. Specifically, on Texture and Fungi, our model outperforms the strongest competitors CTX and TSA by about 8% and 10%, while on other datasets the performance of our model is still comparable with or slight better than that of the existing methods. We notice that our model

1https://github.com/facebookresearch/fvcore
Table 2: Test accuracies and average rank on Meta-Dataset. Note that different backbones are adopted by these methods. * denotes using extra data for training. The bolded items are the best ones with highest accuracies.

<table>
<thead>
<tr>
<th>Model</th>
<th>Backbone</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Aircraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td></td>
<td>45.78</td>
<td>60.85</td>
<td>68.69</td>
<td>57.31</td>
<td>69.05</td>
<td>42.60</td>
<td>38.20</td>
<td>85.51</td>
<td>66.79</td>
<td>34.86</td>
<td>56.96</td>
<td>10.2</td>
</tr>
<tr>
<td>Proto</td>
<td>Res18</td>
<td>50.50</td>
<td>59.98</td>
<td>53.10</td>
<td>68.79</td>
<td>66.56</td>
<td>48.96</td>
<td>39.71</td>
<td>85.27</td>
<td>47.12</td>
<td>41.00</td>
<td>56.10</td>
<td>10.5</td>
</tr>
<tr>
<td>Relation</td>
<td></td>
<td>34.69</td>
<td>45.35</td>
<td>40.73</td>
<td>49.51</td>
<td>52.97</td>
<td>43.30</td>
<td>30.55</td>
<td>68.76</td>
<td>33.67</td>
<td>29.15</td>
<td>42.87</td>
<td>14.6</td>
</tr>
<tr>
<td>P-MAML</td>
<td></td>
<td>49.53</td>
<td>63.37</td>
<td>55.95</td>
<td>68.66</td>
<td>66.49</td>
<td>51.52</td>
<td>39.96</td>
<td>87.15</td>
<td>48.83</td>
<td>43.74</td>
<td>57.52</td>
<td>9.2</td>
</tr>
<tr>
<td>BOHB</td>
<td></td>
<td>51.92</td>
<td>67.57</td>
<td>54.12</td>
<td>70.69</td>
<td>68.34</td>
<td>50.33</td>
<td>41.38</td>
<td>87.34</td>
<td>51.80</td>
<td>48.03</td>
<td>59.15</td>
<td>8.2</td>
</tr>
<tr>
<td>TSA</td>
<td></td>
<td>59.50</td>
<td>78.20</td>
<td>72.20</td>
<td>74.90</td>
<td>74.90</td>
<td>67.97</td>
<td>51.23</td>
<td>93.30</td>
<td>84.09</td>
<td>55.68</td>
<td>70.54</td>
<td>4.3</td>
</tr>
<tr>
<td>Ours</td>
<td>ViT-t</td>
<td>56.40</td>
<td>72.52</td>
<td>72.84</td>
<td>73.79</td>
<td>77.57</td>
<td>67.97</td>
<td>51.23</td>
<td>93.30</td>
<td>84.09</td>
<td>55.68</td>
<td>70.54</td>
<td>4.1</td>
</tr>
<tr>
<td>Proto</td>
<td>Res34</td>
<td>53.70</td>
<td>68.50</td>
<td>58.00</td>
<td>74.10</td>
<td>68.80</td>
<td>53.30</td>
<td>40.70</td>
<td>87.00</td>
<td>58.10</td>
<td>41.70</td>
<td>60.39</td>
<td>7.4</td>
</tr>
<tr>
<td>CTX</td>
<td></td>
<td>62.76</td>
<td>82.21</td>
<td>79.49</td>
<td>80.63</td>
<td>75.57</td>
<td>72.68</td>
<td>51.58</td>
<td>95.34</td>
<td>82.65</td>
<td>59.90</td>
<td>74.28</td>
<td>2.8</td>
</tr>
<tr>
<td>TSA</td>
<td></td>
<td>63.73</td>
<td>82.58</td>
<td>80.13</td>
<td>83.39</td>
<td>79.61</td>
<td>71.03</td>
<td>51.38</td>
<td>94.05</td>
<td>81.71</td>
<td>61.67</td>
<td>74.93</td>
<td>2.5</td>
</tr>
<tr>
<td>P&gt;M&gt;F*</td>
<td></td>
<td>74.69</td>
<td>80.68</td>
<td>76.78</td>
<td>85.04</td>
<td>86.63</td>
<td>71.25</td>
<td>54.78</td>
<td>94.57</td>
<td>88.33</td>
<td>62.57</td>
<td>77.53</td>
<td>—</td>
</tr>
<tr>
<td>Ours</td>
<td>ViT-s</td>
<td>67.37</td>
<td>78.11</td>
<td>79.94</td>
<td>85.93</td>
<td>87.62</td>
<td>71.34</td>
<td>61.80</td>
<td>96.57</td>
<td>85.09</td>
<td>62.33</td>
<td>77.61</td>
<td>1.6</td>
</tr>
</tbody>
</table>

is inferior to the best ones in Omniglot, while this is reasonable. Since Omniglot images represent simple characters with monotone color patterns, each image patches contain less information than images in other datasets. Vanilla ViTs have less efficiency in dealing with these image patches due to limited interaction among patch embeddings. This problem can be solved with much sophisticated variants of ViT like Swin (Liu et al., 2021b), and will be taken as future works. Moreover, our proposed method is better than P>M>F, which not only utilizes extra data for training but also finetunes all model parameters during testing, on more than half of the datasets, which strongly indicates the effectiveness of the proposed finetuning strategy in this paper. As for using ViT-tiny which has much less parameter than Res18, our model is still comparable to the state-of-the-art methods and outperforms many popular baselines. Particularly, compared with ProtoNet which is one of the most famous and efficient methods in FSL, our eTT shows significant boost on Aircraft by 19.74% and TrafficSign by 36.97%. The reason of the inferior results on several datasets against TSA can be two folds. Firstly, the ViT-tiny intrinsically has smaller capacity than Res18. On the other hand, while it is common to train ViT with large scale images and patches so that the images are splitted into abundant patches and each patch-level token can receive enough information. In contrast, we adopt $84 \times 84$ images with $8 \times 8$ patch size for ViT-tiny so that the comparison with Res18 is fair, which lead to less patches with smaller size and may have negative influence on the performance. In general, the results indicate that our proposed eTT can make ViT models a desirable choice for large scale FSL problems.

### 4.3 Model Analysis

To further validate the effectiveness of our method, we conduct a series of ablation studies on Meta-Dataset using ViT-small below.

#### 4.3.1 Design of Each Module

**Can finetuning on meta-train set boost the performance?** One would ask whether it is necessary to make use of base annotations, as supervised pretraining is also effective in many FSL works (Ye et al., 2020; Hou et al., 2019). To verify it, we finetune DINO-pre-trained ViT-small on meta-train split of ImageNet, in which the options of all hyper-parameters and data augmentations follow DeiT (Touvron et al., 2021) using either way of class token features or averaged patch features as image representations. After such a supervised finetuning, we test the models both with the basic test-time finetuning method as in Sec. 3.3 which we denote as LT+NCC, and with our proposed eTT. The results are shown in Fig. 3, from which we find out that (1) Supervised finetuning does improve test accuracies on ImageNet, CUB and MSCOCO. Particularly, the token finetune model receives 89.83% accuracy on CUB when testing with our eTT, which is remarkably better than any other models. This is reasonable as similar images between ImageNet and these datasets. By
training on the image annotations of ImageNet, the model learns class-specific knowledge which cannot be obtained during self-supervised learning. Since the categories are highly correlated and overlapping among these datasets, the learned knowledge can also benefit the recognition on these novel datasets even though the specific novel classes do not appear in the meta-train set. (2) Despite the improvement on the three datasets, models with supervised finetuning degrade on the other datasets, especially on Traffic Sign and VGG Flower. This is due to fitting class labels weakens the effect of these features and makes it harder to generalize to novel domains. When taking into account the performance of all datasets, pretraining with DINO is generally the much more desirable choice for better generalization over different domains. (3) The improvement of our proposed method against the basic LT+NCC is not consistent among three different kinds of pretraining strategy. For example, while our method can boost the performance of DINO pre-trained model by 9.47% on Aircraft and 4.83% on CUB, it can only bring much less advantage on models with supervised finetuning.

**Effectiveness of APT and DRA.** We test the DINO pre-trained model with different kinds of testing strategies including (1) Proto: Directly generating predictions based on ProtoNet. The prototypes are computed using averaged class token features from each category. (2) LT+NCC: The basic test-time finetuning method in Sec. 3.3. (3) Last: Finetuning the last transformer layer during testing, together with LT+NCC. which has similar parameter size to our method. (4) First: Finetuning the first transformer layer during testing, together with LT+NCC. which has similar parameter size to our method. (5) LN: We try to finetune the affinity parameter in each layer normalization as an alternative finetune strategy, which is utilized in many cross-domain FSL works (Tseng et al.; Tsutsui et al., 2022). (6) APT: The model is finetuned using APT together with LT+NCC, using cross entropy loss and the proposed prototypical regularization. (7) Adapter: The model is finetuned using DRA together with LT+NCC, using cross entropy loss. (8) eTT: The model is finetuned using our proposed APT, DRA and LT+NCC. The results in Tab. 3 show that while LT+NCC can fundamentally improve the model which indicates the importance of test-time finetuning, adding our proposed modules to the finetuning procedure can consistently bring higher performance. Also, finetuning specific transformer layer can only bring limited improvement on few datasets: finetuning the last
layer leads to good performance on Aircraft, CUB and Texture, while updating the first layer leads to good performance on Quickdraw and MSCOCO. However, this simple finetuning strategy cannot bring consistent improvement on all datasets. This indicates that different data requires different levels of adaptation, and the improvement is much smaller than that of our method. Moreover, we give the tSNE visualization of feature embeddings of a randomly sampled episode from TrafficSign in Fig. 4, which demonstrates that utilizing our proposed method can better regulate the feature embeddings into proper clusters.

Is prototypical initialization necessary? One of the most important parts of our APT is the attentive prototypical initialization in which we use attentively aggregated patch embeddings to initialize the prefix matrix. To verify this strategy, we compare several different choices of initialization, including (1) Random: random initialization from normal distribution. (2) Avg: simply averaging all patch embeddings from each category. (3) Sampling: randomly sampling one image for each category, and then initializing the prefix matrix with the averaged patch embeddings of each image. (4) Full: computing prototypes with our proposed attentive prototype. Results in Tab. 3 show that random initialization performs the worst, which can be resulted from insufficient task-specific information provided by the prefix in this way. Meanwhile, among all other strategies, using the attention map to aggregate patch embeddings as in Eq. 5 is better than simply averaging, leading to about 1% improvement on average.

Do we need a more complex adapter structure? One would argue that our DRA structure is too simple to learn the complex knowledge from support images. In Tab. 3 we compare three different instantiations of adapters including (1) Linear: As in Li et al. (2022), we use a linear layer for each adapter, whose output are than added to the original features in the MSA and FFN. (2) Bottleneck: We expand the linear layer
to a bottleneck structure where two linear layers are used for each adapters. (3) FiLM: We compare DRA with a FiLM-like variant, in which we add a scaling vector for each adapter as in FiLM layer [Perez et al. (2018)]. Note that such a method is similar to MTL [Sun et al. (2019)] in FSL. The difference lies in that we still use the original way to directly tune the parameters on the novel support sets, instead of using another meta-trained module to generate the parameters. (4) Offset: Only an offset vector is adopted for each adapter.

The results reveals that the linear adapter performs the worst, which means such a structure is improper for ViT in finetuning. Moreover, we also find that using the bottleneck adapter will result in a dilemma. If we use small initial value for the adapter, the weights of each layer can only achieve gradient with extremely small values. As the result, these weights, except the bias term of the last layer, can hardly be updated based on the support knowledge, which means such an architecture almost equals to our design where only an offset vector is utilized. On the other hand, if large initial values are adopted to avoid gradient diminishing, then the output features from the adapters can make the predictions severely deviate from those without adapters, thus leading to worse performance. As for the FiLM-like DRA, it is worse than offset DRA by about 0.8% on average, while it doubles the parameter size based on offset DRA, leading to no significant additional efficacy.

Effectiveness of prototypical regularization. We also validate this regularization. In Tab. 3 we present the test accuracy when finetuning with and without this loss function. We can find that by applying this objective function, the model can have higher results on most datasets. Besides, as described in Sec. 3.6, we use a standardization technique when computing the prototypical regularization. To verify its efficacy, we compare the model with and without such a standardization. The results are shown in Tab. 3. When not using standardization, the results are generally worse given comparable confidence intervals (Tab. 11). The results verify that this strategy can help the model with more stable finetuning procedure.

<table>
<thead>
<tr>
<th>$d_{proj}$</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>67.18</td>
<td>75.30</td>
<td>78.88</td>
<td>86.20</td>
<td>87.09</td>
<td>69.82</td>
<td>61.61</td>
<td>96.31</td>
<td>82.24</td>
<td>62.14</td>
<td>76.68</td>
</tr>
<tr>
<td>96</td>
<td>66.23</td>
<td>75.69</td>
<td>78.26</td>
<td>85.67</td>
<td>87.28</td>
<td>70.25</td>
<td>61.97</td>
<td>96.59</td>
<td>84.10</td>
<td>62.17</td>
<td>76.82</td>
</tr>
<tr>
<td>128</td>
<td>67.31</td>
<td>76.83</td>
<td>78.81</td>
<td>85.77</td>
<td>87.36</td>
<td>70.16</td>
<td>60.81</td>
<td>95.36</td>
<td>84.29</td>
<td>62.12</td>
<td>77.00</td>
</tr>
<tr>
<td>256</td>
<td>66.83</td>
<td>78.04</td>
<td>78.38</td>
<td>84.60</td>
<td>86.68</td>
<td>70.43</td>
<td>61.03</td>
<td>96.23</td>
<td>85.33</td>
<td>62.10</td>
<td>76.97</td>
</tr>
<tr>
<td>192</td>
<td>67.37</td>
<td>78.11</td>
<td>79.94</td>
<td>85.93</td>
<td>87.62</td>
<td>71.34</td>
<td>61.80</td>
<td>96.57</td>
<td>85.09</td>
<td>62.33</td>
<td>77.61</td>
</tr>
</tbody>
</table>

Table 4: Test accuracies on Meta-Dataset of different variants of our proposed method. The bolded items are the best ones with highest accuracies.

4.3.2 Comparison among Different Hyper-parameter Settings

In additional to the ablation study about the proposed module, We further verify different choices of hyper-parameters in our model. Especially, $d_{proj}$ for the transformation module in APT and $\lambda$ for the prototypical regularization are tested in Tab. 4 and Tab. 5 in the Appendix. In general, the improvement is not consistent. For $d_{proj}$, we can find that using 192-d hidden dimension can get globally better results, which indicates that such a choice can make a good balance between the model capacity and scale so that the finetuning can be conducted both efficiently and effectively. As for $\lambda$, 0.1 seems to be a desirable choice. Intuitively, smaller $\lambda$ leads to less control of the prefix from the proposed prototypical regularization. Therefore, the prefix may lose the desired information during the optimization on the support set. On the other hand, when $\lambda$ is too large, the regularization overwhelms the label supervision, and thus the model can hardly adapt to the support knowledge, leading to worse performance especially on Omniglot and Aircraft.

5 Conclusion

We propose a novel finetuning method named efficient Transformer Tuning (eTT) for few-shot learning with ViT as our backbone. By fixing the parameters in the backbone and utilizing attentive prefix tuning and domain residual adapter, our method can guide the ViT model with comprehensive task-specific information, which leads to better representations and performance. This is demonstrated by the fact that we establish new state-of-the-arts on the large-scale benchmark Meta-Dataset.
References


Chenjie Cao, Yuxin Hong, Xiang Li, Chengrong Wang, Chengming Xu, Yanwei Fu, and Xiangyang Xue. The image local autoregressive transformer. Advances in Neural Information Processing Systems, 34, 2021.


Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. 2006.


A Appendix

A.1 Limitations and Future Work

Despite the marginal effectiveness and efficiency of our proposed eTT, we mainly notice two points that should be explored in the future: (1) The plain ViT backbone utilized in this paper, may not be the best choice to the simple dataset, e.g., Omniglot, while a well-designed ViT backbone may potentially better improve the efficacy of our method on such dataset. (2) A flexible finetuning algorithm such as (Lee et al.) may have better generalization ability when facing episodes with various shots and ways, than the commonly-used methods that adopt fixed test-time finetuning iterations. However, it is non-trivial to directly merge such methods with our proposed eTT due to different network structures and tuning strategies. It can be taken as the future work to properly utilize these flexible finetuning algorithm to further improve the performance of ViT in FSL.
A.2 Additional Experiment Results

A.2.1 Full Comparison with state-of-the-art methods

We show the accuracies together with confidence interval in Tab. 6. Beyond the accuracies which is analyzed in the main context, the confidence interval of our eTT on both ViT-tiny and ViT-small is comparable with the other competitors, which reflects that our method is stable and robust enough among different testing episodes.

A.2.2 Influence of Training set

As we have stated in the main context, our eTT is trained on the meta-train split of ImageNet to make fair comparison with other methods. To show to what extent training on the full ImageNet training set instead of the meta-train set can impact the performance, we test our eTT using off-the-shelf DINO ViT-s model. The results are shown in Tab. 7. We can find that (1) For those datasets on which DINO meta-train performs better than P>M>F, using full ImageNet to train DINO can bring further improvement. (2) With the help of more data, our eTT overpasses P>M>F on ILSVRC and MSCOCO. (3) While more data does improve the results on Omniglot and TrafficSign, the final results are still worse than those of P>M>F, which we think may be correlated with the limitations of our method as analyzed above. Given all these results, as a lighter model in that no meta-training phase is utilized and only few parameters are engaged in the test-time tuning, our method can still enjoy comparable performance with P>M>F when training on same amount of data.

A.2.3 Influence of DINO

The DINO pretrain procedure is an important part of our method. To verify the effectiveness of DINO pretrain so that the comparison with other methods is fair enough, we present in Tab. 8 the results of TSA using DINO-pretrained ResNet-34 and eTT using supervised pretrained ViT-s. We can find that (1) The effect of DINO is not consistent on two backbones. While DINO benifits our eTT with about 5% accuracy on average, it severely weakens the performance of TSA with a large margin. It means that for FSL, different backbones require different pretrain strategy respectively. (2) While supervised pretrained ViT-s performs worse on most datasets, it is better on CUB and COCO, which indicates learning label information from ImageNet can help the model understanding novel knowledge from these two datasets.

A.2.4 Verification of potential overfitting in finetuning

As we have stated in Sec. 4 finetuning the whole backbone model with few support data will meet potential overfitting problem. To reveal if such a problem exists in Meta-Dataset, we conduct an experiment as follow: during a normal testing phase, we select all episodes whose minimum shot (minimum number of support images for each class) is no larger than 2 (extremely small number of labelled instances), and compare the average accuracies of eTT and simple finetuning based on these episodes. Tab. 9 and Tab. 10 show the accuracies on support sets and query sets respectively. We can find that most of the datasets both methods receive nearly 100% accuracy, which means these two methods can well learn the support data. Given this fact, finetuning is much worse than eTT in terms of query accuracies. The overfitting can be reflected given high training accuracies and worse testing performance, and to some extent our proposed eTT can fix this problem.

A.2.5 More Visualization

We visualize the self-attention map from models with and without DRA on ILSVRC and TrafficSign in Fig. 5. Specifically, we randomly sample an episode from each dataset and use our eTT to tune the model based on the support samples. Then we calculate the self-attention map of the last layer’s class token and highlight the areas with top 20% attention scores. We can find that for the in-domain ILSVRC episode, the model can attend to similar regions no matter whether DRA is used. In contrast, the model without DRA can easily attend to background regions with less valuable information, which reveals a potential reason that these two models has similar accuracies on ILSVRC but large performance gap on TrafficSign.
Furthermore, we visualize more episodes from Aircraft, TrafficSign and MSCOCO in Fig. 6, Fig. 7 and Fig. 8 which shows that our proposed eTT can remarkably improve the embedding space after test-time finetuning.

### A.3 Broader Impact

Our paper presents a more efficient and practical FSL pipeline utilizing ViT. We hope this work can shed light on the broader usage of ViT in FSL tasks. On the other hand, the proposed method can provide researchers with alternative choice for FSL applications in real-case scenarios with large-scale meta-train set and challenging various test episodes.

<table>
<thead>
<tr>
<th>$d_{proj}$</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>67.01</td>
<td>76.56</td>
<td>78.34</td>
<td>85.53</td>
<td>86.96</td>
<td>70.03</td>
<td>61.20</td>
<td>96.17</td>
<td>85.00</td>
<td>62.67</td>
<td>76.95</td>
</tr>
<tr>
<td>0.05</td>
<td>66.49</td>
<td>77.40</td>
<td>78.92</td>
<td>85.80</td>
<td>87.54</td>
<td>70.23</td>
<td>60.78</td>
<td>96.28</td>
<td>84.95</td>
<td>62.38</td>
<td>77.08</td>
</tr>
<tr>
<td>0.5</td>
<td>66.88</td>
<td>77.73</td>
<td>78.65</td>
<td>86.00</td>
<td>87.15</td>
<td>70.48</td>
<td>61.64</td>
<td>96.23</td>
<td>84.39</td>
<td>63.39</td>
<td>77.25</td>
</tr>
<tr>
<td>0.9</td>
<td>67.03</td>
<td>76.55</td>
<td>77.89</td>
<td>85.78</td>
<td>87.04</td>
<td>70.08</td>
<td>62.45</td>
<td>96.20</td>
<td>84.44</td>
<td>62.83</td>
<td>77.03</td>
</tr>
<tr>
<td>1.0</td>
<td><strong>67.37</strong></td>
<td><strong>78.11</strong></td>
<td><strong>79.94</strong></td>
<td><strong>85.93</strong></td>
<td><strong>87.62</strong></td>
<td><strong>71.34</strong></td>
<td><strong>61.80</strong></td>
<td><strong>96.57</strong></td>
<td><strong>85.09</strong></td>
<td><strong>62.33</strong></td>
<td><strong>77.61</strong></td>
</tr>
</tbody>
</table>

**Table 5:** Test accuracies on Meta-Dataset of different variants of our proposed method. The bolded items are the best ones with highest accuracies.

<table>
<thead>
<tr>
<th>Model</th>
<th>Backbone</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>Res18</td>
<td>67.10</td>
<td>60.65</td>
<td>58.69</td>
<td>57.31</td>
<td>60.05</td>
<td>42.63</td>
<td>42.17</td>
<td>38.20</td>
<td>42.02</td>
<td>56.79</td>
<td>34.86</td>
</tr>
<tr>
<td>Proto</td>
<td>Res18</td>
<td>66.49</td>
<td>60.65</td>
<td>58.69</td>
<td>57.31</td>
<td>60.05</td>
<td>42.63</td>
<td>42.17</td>
<td>38.20</td>
<td>42.02</td>
<td>56.79</td>
<td>34.86</td>
</tr>
<tr>
<td>Relation</td>
<td>Res18</td>
<td>66.49</td>
<td>60.65</td>
<td>58.69</td>
<td>57.31</td>
<td>60.05</td>
<td>42.63</td>
<td>42.17</td>
<td>38.20</td>
<td>42.02</td>
<td>56.79</td>
<td>34.86</td>
</tr>
<tr>
<td>P-MAML</td>
<td>Res18</td>
<td>66.49</td>
<td>60.65</td>
<td>58.69</td>
<td>57.31</td>
<td>60.05</td>
<td>42.63</td>
<td>42.17</td>
<td>38.20</td>
<td>42.02</td>
<td>56.79</td>
<td>34.86</td>
</tr>
<tr>
<td>BOHB</td>
<td>Res18</td>
<td>66.49</td>
<td>60.65</td>
<td>58.69</td>
<td>57.31</td>
<td>60.05</td>
<td>42.63</td>
<td>42.17</td>
<td>38.20</td>
<td>42.02</td>
<td>56.79</td>
<td>34.86</td>
</tr>
<tr>
<td>TSA</td>
<td>Res18</td>
<td>66.49</td>
<td>60.65</td>
<td>58.69</td>
<td>57.31</td>
<td>60.05</td>
<td>42.63</td>
<td>42.17</td>
<td>38.20</td>
<td>42.02</td>
<td>56.79</td>
<td>34.86</td>
</tr>
</tbody>
</table>

**Table 6:** Test accuracies, confidence interval and average rank on Meta-Dataset. Note that different backbones are adopted by these methods. The bolded items are the best ones with highest accuracies.

<table>
<thead>
<tr>
<th>Model</th>
<th>Train Set</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTT</td>
<td>meta-train</td>
<td>67.37</td>
<td>78.11</td>
<td>79.94</td>
<td>85.93</td>
<td>87.62</td>
<td><strong>71.34</strong></td>
<td>61.80</td>
<td>96.57</td>
<td>85.09</td>
<td>62.33</td>
<td>77.61</td>
</tr>
<tr>
<td>eTT</td>
<td>full</td>
<td><strong>74.76</strong></td>
<td>78.73</td>
<td><strong>80.10</strong></td>
<td><strong>86.99</strong></td>
<td><strong>87.72</strong></td>
<td>71.20</td>
<td>61.95</td>
<td><strong>96.66</strong></td>
<td>85.83</td>
<td><strong>64.25</strong></td>
<td><strong>78.82</strong></td>
</tr>
<tr>
<td>P&gt;M&gt;F</td>
<td>full</td>
<td>74.69</td>
<td><strong>80.68</strong></td>
<td>76.78</td>
<td>85.04</td>
<td>86.63</td>
<td>71.25</td>
<td>54.78</td>
<td>94.57</td>
<td><strong>88.33</strong></td>
<td>62.57</td>
<td>77.53</td>
</tr>
</tbody>
</table>

**Table 7:** Test accuracies on Meta-Dataset of different variants of our proposed method. The bolded items are the best ones with highest accuracies. The highlighted rows denote the final model in our main paper.
Table 8: Test accuracies on Meta-Dataset of different variants of our proposed method. The bolded items are the best ones with highest accuracies. The highlighted rows denote the final model in our main paper.

<table>
<thead>
<tr>
<th>Model</th>
<th>Pretrain</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSA</td>
<td>Sup.</td>
<td>59.50</td>
<td>78.20</td>
<td>72.20</td>
<td>74.90</td>
<td>77.30</td>
<td>67.60</td>
<td>44.70</td>
<td>90.90</td>
<td>82.50</td>
<td>59.00</td>
<td>70.68</td>
</tr>
<tr>
<td>TSA</td>
<td>DINO</td>
<td>48.18</td>
<td>64.94</td>
<td>56.74</td>
<td>45.49</td>
<td>69.06</td>
<td>59.51</td>
<td>31.13</td>
<td>81.01</td>
<td>48.70</td>
<td>26.18</td>
<td>53.09</td>
</tr>
<tr>
<td>eTT</td>
<td>Sup.</td>
<td>65.17</td>
<td>67.47</td>
<td>73.30</td>
<td>87.71</td>
<td>84.50</td>
<td>67.46</td>
<td>55.51</td>
<td>92.55</td>
<td>64.08</td>
<td>63.68</td>
<td>72.14</td>
</tr>
<tr>
<td>eTT</td>
<td>DINO</td>
<td>67.37</td>
<td>78.11</td>
<td>79.94</td>
<td>85.93</td>
<td>87.62</td>
<td>71.34</td>
<td>61.80</td>
<td>96.57</td>
<td>85.09</td>
<td>62.33</td>
<td>77.61</td>
</tr>
</tbody>
</table>

Table 9: Support set accuracies of eTT and Finetune on testing episodes whose minimum shots is no larger than 2.

<table>
<thead>
<tr>
<th>Model</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>eTT</td>
<td>100.00</td>
<td>99.99</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>99.79</td>
<td>100.00</td>
<td>100.00</td>
<td>99.15</td>
<td>99.89</td>
<td>99.89</td>
</tr>
<tr>
<td>FT</td>
<td>100.00</td>
<td>99.87</td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
<td>96.95</td>
<td>100.00</td>
<td>100.00</td>
<td>95.20</td>
<td>95.20</td>
<td>99.20</td>
</tr>
</tbody>
</table>

Table 10: Query set accuracies of eTT and Finetune on testing episodes whose minimum shots is no larger than 2. The bolded items are the best ones with highest accuracies.

<table>
<thead>
<tr>
<th>Model</th>
<th>ILSVRC</th>
<th>Omni</th>
<th>Acraft</th>
<th>CUB</th>
<th>DTD</th>
<th>QDraw</th>
<th>Fungi</th>
<th>Flower</th>
<th>Sign</th>
<th>COCO</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT</td>
<td>29.19</td>
<td>54.54</td>
<td>35.10</td>
<td>41.54</td>
<td>53.66</td>
<td>43.37</td>
<td>38.53</td>
<td>76.76</td>
<td>72.90</td>
<td>41.21</td>
<td>48.68</td>
</tr>
<tr>
<td>eTT</td>
<td>40.22</td>
<td>64.79</td>
<td>41.33</td>
<td>55.11</td>
<td>66.20</td>
<td>49.14</td>
<td>56.33</td>
<td>85.03</td>
<td>75.29</td>
<td>56.19</td>
<td>58.96</td>
</tr>
</tbody>
</table>

Table 11: The corresponding confidence intervals of models in ablation study on standardization.
Figure 5: Visualization of self-attention from model with and without DRA on ILSVRC (left) and TrafficSign (right). The white regions are those with top 20% attention scores.
Figure 6: More visualization of feature embeddings from a randomly sampled episode of TrafficSign.
Figure 7: More visualization of feature embeddings from a randomly sampled episode of Aircraft.
Figure 8: More visualization of feature embeddings from a randomly sampled episode of MSCOCO.