Task-Adaptive Prompted Transformer for Cross-Domain Few-Shot Learning

Jiamin Wu[1], **Xin Liu**[1], **Xiaotian Yin**[1], **Tianzhu Zhang**[1*], **Yongdong Zhang**[1]

[1]Deep Space Exploration Laboratory/School of Information Science and Technology,
University of Science and Technology of China
{jiaminwu, attcb63442, xiaotianyin}@mail.ustc.edu.cn, {tzzhang, zhyd73}@ustc.edu.cn

Abstract

Cross-Domain Few-Shot Learning (CD-FSL) aims at recognizing samples in novel classes from unseen domains that are vastly different from training classes, with few labeled samples. However, the large domain gap between training and novel classes makes previous FSL methods perform poorly. To address this issue, we propose MetaPrompt, a Task-adaptive Prompted Transformer model for CD-FSL, by jointly exploiting prompt learning and the parameter generation framework. The proposed MetaPrompt enjoys several merits. First, a task-conditioned prompt generator is established upon attention mechanisms. It can flexibly produce a task-adaptive prompt with arbitrary length for unseen tasks, by selectively gathering task characteristics from the contextualized support embeddings. Second, the task-adaptive prompt is attached to Vision Transformer to facilitate fast task adaptation, steering the task-agnostic representation to incorporate task knowledge. To our best knowledge, this is the first work to exploit a prompt-based parameter generation mechanism for CD-FSL. Extensive experimental results on the Meta-Dataset benchmark demonstrate that our method achieves superior results against state-of-the-art methods.

Introduction

Few-Shot Learning (FSL) (Finn, Abbeel, and Levine 2017; Lee et al. 2019; Lifchitz et al. 2019; Wu et al. 2022) aims at building a model that can generalize to unseen novel classes based on only a few labeled examples. The standard few-shot learning paradigm is composed of (i) a meta-training stage where a model is learned from a large training set, and (ii) a meta-test stage where the learned model is adapted to novel classes from a tiny support set. In earlier FSL works (Zhang et al. 2020; Tian et al. 2020), the meta-train and meta-test examples share the same data distribution (*i.e.,* sampling from the same dataset like miniImageNet (Vinyals et al. 2016)). Thus the learned model fails to generalize to a more practical setting, where the meta-test and meta-train classes are sampled from vastly different datasets/domains (Dvornik, Schmid, and Mairal 2020; Liu et al. 2021b). The large domain gap in this cross-domain setting poses a new challenge for few-shot learning (Li, Liu, and Bilen 2022).

*Corresponding Author.

To address this challenge, recent works (Guo et al. 2020; Li, Liu, and Bilen 2022; Requeima et al. 2019) are shifting their interest to *Cross-Domain Few-Shot Learning* (CD-FSL), where the meta-train set and meta-test set are composed of different sets of datasets. To bridge the large domain gap, recent methods (Requeima et al. 2019; Bateni et al. 2020; Triantafillou et al. 2021) learn a set of task-conditioned parameters to adapt the task-agnostic model to the current task. Generally, a set of independent generators are utilized to generate the task-conditioned parameters from the task description gathered from the small support set. For example, as shown in Figure 1 (a), CNAPs (Requeima et al. 2019) and SimpleCNAPs (Bateni et al. 2020) average the support features x^s to obtain a task embedding e_t and send it into convolutional generators $[\theta_{g_1}, \theta_{g_2}, \cdots, \theta_{g_i}]$ to estimate the parameters $[p_1, p_2, \cdots, p_i]$ of FiLM layers (*i.e.*, the scale and shift parameters for feature modulation). Each time we need to generate an additional parameter p_{i+1}, a new generator $\theta_{g_{i+1}}$ would be required, which would increase generator parameters.

By studying the previous FSL methods based on parameter generation (Requeima et al. 2019; Bateni et al. 2020; Oreshkin, Rodríguez López, and Lacoste 2018), we sum up two fundamental points for resolving the substantial domain gap for CD-FSL. **(i) Reliable task description.** To obtain task description as generation conditions, the previous methods directly average the support features x^s into e_t (see Figure 1 (a)). However, the relations between samples from different classes are ignored, which are essential for characterizing the task. Besides, treating the non-representative samples equally as the others may bring noises into the task description, which may further hamper the expressiveness of the generated parameters. **(ii) Flexible parameter generation mechanism.** In the previous methods, the parameters $[p_1, \cdots, p_i]$ are always independently predicted by their corresponding generators $[\theta_{g_1}, \cdots, \theta_{g_i}]$, respectively (see Figure 1 (a)). Thus the scale of generators will increase linearly with the number of generated parameters. This causes inflexibility when requiring abundant task-conditioned parameters, as it may cause overfitting to the few-shot data.

Inspired by the above discussion, we propose ***MetaPrompt***, a ***Task-adaptive Prompted Transformer*** model for CD-FSL, by jointly exploiting the prompt learning and the task-conditioned parameter generation

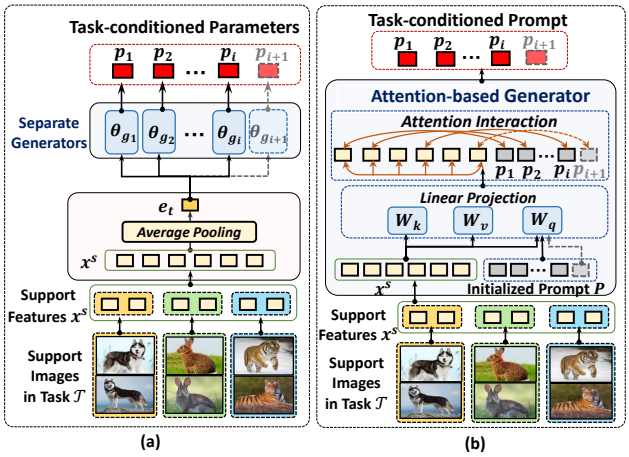

Figure 1: Different ways of generating task-conditioned parameters. (a) Prior works (Requeima et al. 2019; Bateni et al. 2020) average the support features x^s to obtain a task embedding e_t and send it into separate generators $[\theta_{g_1}, \cdots, \theta_{g_i}]$ to independently produce task-conditioned parameters $[p_1, \cdots, p_i]$. (b) Differently, we design an attention-based generator to produce a task-adaptive prompt P with arbitrary number of tokens $[p_1, \cdots, p_i]$ by going through the same linear projection W_k, W_v, W_q in attention layers. The rich relations between P and x^s can be exploited for better task description.

framework. Prompt learning (Lester, Al-Rfou, and Constant 2021; Li and Liang 2021) is a popular method in the NLP field that aims at helping the Transformer model to perform downstream tasks conditionally by attaching the prompt to the input sequence. Motivated by this, we design a **task-conditioned prompt generator** based on attention layers (with parameters W_q, W_k, W_v) to produce a **task-adaptive prompt** P with tokens $[p_1, \cdots, p_i]$, which will not involve additional generators when producing a new additional prompt token p_{i+1} (see Figure 1 (b)). Specifically, given the support features x^s in task $\mathcal{T}$, we send them into a **prompt encoder**, where each support feature is associated with a **class-wise embedding** to discriminate between different classes. A class-aware self-attention layer is utilized to exploit cross-class relations from support features for characterizing the task. Then, several learnable prompt tokens $[p_1, \cdots, p_i]$ along with support features are sent into a **prompt decoder**, which is composed of task-aware cross-attention layers. By fully communicating with support features, the task-adaptive prompt P can absorb crucial task knowledge. When classifying samples in task $\mathcal{T}$, the task-adaptive prompt is attached to the Vision Transformer (Dosovitskiy et al. 2021), empowering the task-agnostic representation to incorporate task characteristics.

Compared to previous generation-based methods (Requeima et al. 2019; Bateni et al. 2020), MetaPrompt enjoys several intriguing properties. **(i)** In contrast to the average operation, our prompt generator can de-emphasize less-representative samples by assigning them smaller attention weights, thereby obtaining **reliable** task description. **(ii)** The attention-based generator is compatible with input sequences with varying length, and thus is **flexible** to generate the prompt with arbitrary length while keeping a fixed model size. **(iii)** Using the task-adaptive prompt imbues our model with necessary **expressiveness**, as they could achieve in-depth feature modification by inserting into the input to affect the attention interaction of subsequent layers.

The contributions of our MetaPrompt could be summarized into three-fold: (1) We propose a Task-adaptive Prompted Transformer by integrating the prompt learning with the parameter generation framework. (2) A task-conditioned prompt generator is proposed to produce a task-adaptive prompt with arbitrary length by gathering task knowledge from contextualized support embeddings. The generated prompt could help with fast task adaptation by attaching to the Transformer model. To our best knowledge, this is the first work to exploit the prompt-based parameter generation for CD-FSL. (3) MetaPrompt achieves superior performance against state-of-the-art CD-FSL methods on Meta-Dataset (Triantafillou et al. 2019), especially for unseen domains.

Related Work

In this section, we introduce several lines of research in parameter generation mechanisms in FSL, cross-domain few-shot learning, and prompt-based learning.

Parameter Generation Mechanisms in FSL. Parameter generation mechanisms have been broadly explored in few-shot learning (Oreshkin, Rodríguez López, and Lacoste 2018; Requeima et al. 2019; Chen et al. 2020; Wu et al. 2021). These methods usually build a meta-learner as a generation network to predict the parameters of task-conditioned modules from a small number of support samples. Some methods (Qi, Brown, and Lowe 2018; Qiao et al. 2018) generate the weights of the classification layer from the extracted features. CNAPs (Requeima et al. 2019) and SimpleCNAPs (Bateni et al. 2020) generate the scaling and shifting parameters of FiLM layers from the averaged support feature (*i.e.*, task embedding). Their generated parameters are used to modulate the feature maps by the affine transformation. However, these methods are limited in flexibility with regard to both task description and multi-parameter generation. By contrast, we develop an attention-based prompt generator to flexibly predict a task-adaptive prompt with arbitrary length by selectively gathering task characteristics.

Cross-Domain Few-Shot Learning. To bridge the large domain gap in CD-FSL, a plethora of methods (Li, Liu, and Bilen 2022, 2021; Liu et al. 2021b) attempt to design task adaptation mechanisms based on the small-sized support set. Some methods obtain task-specific features by dynamically combining the representations from separate domain-specific networks, where the aggregation weights could be produced by linear layers (Liu et al. 2021b), convolution layers (Triantafillou et al. 2021), or Transformer layers (Liu et al. 2021a) for each task. However, training multiple feature extractors inevitably causes additional parameter costs.

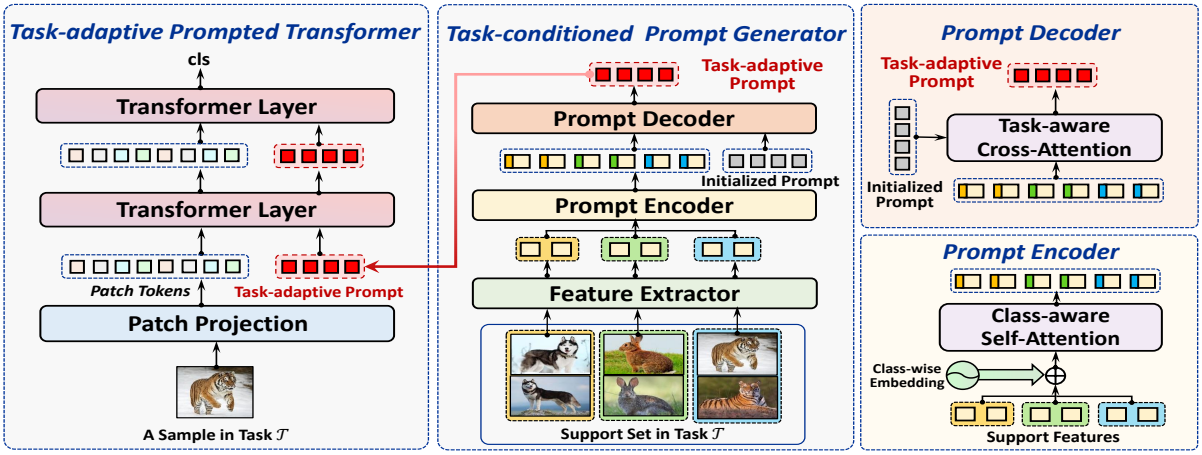

Figure 2: The framework of MetaPrompt: (1) In the Task-conditioned Prompt Generator, the feature extractor produces the feature sequence of the support samples in task $\mathcal{T}$. The prompt encoder exploits relations between support samples from different categories, producing co-adapted support embeddings. The prompt decoder collects task-indicative information from support sequence and transforms it into a task-adaptive prompt. (2) In the Task-adaptive Prompted Transformer, the generated prompt is inserted into ViT-based backbone, producing task-aware representations for samples in $\mathcal{T}$.

Moreover, the adaptability of the fused representations is restricted by backbones learned on seen domains. Other methods introduce additional labeled (Fu et al. 2022) or unlabeled target data (Islam et al. 2021) into meta-training. They adopt self-supervised learning (Islam et al. 2021), self-training (Phoo and Hariharan 2021), or mix-up strategies (Fu et al. 2022) to leverage the target data, hoping to obtain the adaptation ability on the unseen domain. However, it is non-trivial to obtain the target samples in advance in the real world. Another line of methods (Requeima et al. 2019; Triantafillou et al. 2021) proposes to learn task-conditioned parameters from the support set to achieve task adaptation. They meta-learn a parameter generator that is expected to generalize to the target tasks. Our method falls into the parameter generation type. Different from the above methods, we are the first to introduce the prompting strategy in a generative way for building a task-adaptive CD-FSL classifier.

Prompt-based Learning. Prompting methods (Liu et al. 2023; Brown et al. 2020) have made numerous advancements in the field of NLP, whose core idea is to attach prompt tokens to provide instructions for modifying the input in downstream tasks. However, how to design the prompting function is challenging. Recent works (Lester, Al-Rfou, and Constant 2021; Li and Liang 2021) propose to learn a continuous prompt to instruct a frozen pre-trained language model. In this way, prompts are promising to capture task-specific knowledge with small additional parameters. CoOp (Zhou et al. 2022b) and CoCoOp (Zhou et al. 2022a) learn prompts for conditioning the text encoder, thereby adapting vision-language models in various cross-modality tasks. Recently, prompt learning has also been applied to the vision domain. VPT (Jia et al. 2022) prepends prompt tokens to the ViT as instructive task information. L2P (Wang et al. 2022b) and DualPrompt (Wang et al. 2022a) utilize the prompt as an alternative for rehearsal buffer in continual learning. In this paper, we extend the prompt learning to CD-FSL by designing a task-adaptive prompted Transformer for fast few-shot adaptation.

Our Approach

The proposed MetaPrompt includes two modules (see Figure 2): (1) The **Task-conditioned Prompt Generator** is responsible for collecting task-indicative cues and converting them into a task-adaptive prompt. (2) The **Task-adaptive Prompted Transformer** can produce task-aware representations by inserting the generated task-adaptive prompt into the ViT-based backbone.

Preliminaries

Problem Setting. In CD-FSL, a large-scale meta-training set D_{train} (containing seen domains) is used to learn a generalizable model that can transfer to the meta-test set D_{test} (containing unseen domains) to classify novel classes. Notably, D_{train} and D_{test} are defined over a union of diverse datasets/domains, and contain mutually exclusive classes. That is to say, the new classes in D_{test} originate from different datasets from D_{train}. The classification of meta-test samples is conducted over a series of tasks (or episodes) $\mathcal{T}_{test} \in D_{test}$. D_{train} is also segmented into a set of tasks $\mathcal{T}_{train}$ to mimic the meta-test setting, hoping to acquire the cross-task generalization ability. Specifically, an N-way K-shot task $\mathcal{T}$ is composed of a support set $\mathcal{S}$ containing N classes with K samples per class, and a query set $\mathcal{Q} = \{(X_i^q, y_i^q)\}_{i=1}$ including $|Q|$ query samples. Formally, $\mathcal{S} = \{(X_j^s, y_j^s)\}_{j=1}^{NK}$, where X_j^s and y_j^s denote the images and the labels $y_j^s \in \{1, 2, \cdots, N\}$, respectively. During meta-test, the goal is to classify a query sample $x_i^q \in \mathcal{Q}$ from $\mathcal{D}_{test}$ into one of the N support classes given a few labeled samples from $\mathcal{S}$.

Vision Transformer. Following (Hu et al. 2022), we adopt Vision Transformer (ViT) (Dosovitskiy et al. 2021) as our backbone. In ViT, the input image $x \in R^{H\times W\times C}$ is first reshaped and divided into a sequence of L patches with patch size S, *i.e.,* $x \in R^{L\times(S^2\times C)}$. An embedding layer projects the image patches into patch tokens with dimension D: $x_e \in R^{L\times D}$. Usually, a classification token x_{cls} is added to the token sequence, which is then fed into the Transformer block f_t that consists of multiple attention and MLP layers.

Task-conditioned Prompt Generator

We introduce an attention-based task-conditioned prompt generator to produce a task-adaptive prompt, which helps the ViT model to adapt to unseen tasks from arbitrary domains. Specifically, the task-conditioned prompt generator is composed of a feature extractor, a prompt encoder and a prompt decoder. Given the support images $\{X_i^s\}_{i=1}^{NK}$ in the current task $\mathcal{T}$, the feature extractor is used to extract their features $\{x_i^s\}_{i=1}^{NK}, x_i^s \in R^D$. The support features from different classes contain essential information describing task properties, which can be leveraged by the prompt encoder and prompt decoder for learning the task-adaptive prompt.

Prompt Encoder. We associate each support feature with a **class-wise embedding**: $x_i^s = x_i^s + E^c$, where x_i^s is a support sample from class c, $E^c \in R^D$ is the class-wise embedding randomly assigned for class c in $\mathcal{T}$. Here, the learnable class-wise embeddings $E \in R^{N_c\times D}$ are randomly initialized, where N_c is the predefined maximum number of categories in a task. It should be noted that E contain no class-specific information but are just used to discriminate between different categories. As we adopt a nearest-neighbor classifier and class indexes are independent in different episodes, the randomly assigned E will not cause interference. Then, we design a **class-aware self-attention layer** to refine each support embedding, with consideration of its contextual instances from the same or different classes in $\mathcal{T}$. We use the support sequence to serve as query $\mathbf{Q}$, key $\mathbf{K}$, value $\mathbf{V}$ in the self-attention layer (notably, this query $\mathbf{Q}$ should be discriminated from the query set $\mathcal{Q}$ in FSL). Specifically, $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are generated by independent linear projection layers:

$$\mathbf{Q}_i = x_i^s\mathbf{W}_q, \mathbf{K}_j = x_j^s\mathbf{W}_k, \mathbf{V}_j = x_j^s\mathbf{W}_v, \tag{1}$$

where $i, j = 1, 2, \cdots, NK$, and $\mathbf{W}_q \in R^{D\times d_q}, \mathbf{W}_k \in R^{D\times d_k}, \mathbf{W}_v \in R^{D\times d_v}$ are linear projection layers. The query $\mathbf{Q}_i$ is first matched against a list of keys:

$$\tilde{a}_{ij} = \frac{\exp(a_{ij})}{\sum_{j'=1}^{T}\exp(a_{ij'})}, a_{ij} = \frac{\mathbf{Q}_i\mathbf{K}_j^T}{\sqrt{d_k}}, \tag{2}$$

where $\sqrt{d_k}$ is a scaling factor. Using the intra-task correlations in attention scores $\tilde{a}_{ij}$, the support embeddings can be co-adapted, with the discriminative ones amplified. Concretely, the adapted support embedding x_i^s is derived as the weighted sum over all values with $\tilde{a}_{ij}$:

$$x_i^s = \sum_{j=1}^{NK}\tilde{a}_{ij}\mathbf{V}_j, i = 1, 2, \cdots, NK. \tag{3}$$

By exploiting the comprehensive cross-class relations within the task, we can exploit high-level task-related knowledge for building a better adaptation mechanism.

Prompt Decoder. We design **task-aware cross-attention layers** to generate a **task-adaptive prompt** from support embeddings. Specifically, a series of learnable prompt tokens $P = [p_1, p_2, \cdots, p_M] \in R^{MD}$ are initialized to serve as queries $\mathbf{Q}$ with M as the prompt length. Besides, the support sequence is taken as keys $\mathbf{K}$ and values $\mathbf{V}$. Then we use Eq. 1-3 to establish cross-attention paths between prompt tokens and support features. By fully communicating with the support sequence, the prompt tokens could identify representative samples and selectively accumulate task-relevant information. In this manner, each prompt token captures a specific perspective of task knowledge. One advantage of using the attention-based prompt decoder is its scalability, as it can process prompts with arbitrary lengths and tasks with varying sizes in the attention mechanism.

Notably, the task-conditioned prompt generator is meta-trained over numerous tasks, thereby learning shared meta-knowledge that can be well transferred to novel tasks. In contrast to the ambiguous task description based on the sample average, our generator comprehensively considers all support samples and exploits cross-category relations. Therefore, the generated prompt can reliably characterize task information for effective task adaptation.

Task-adaptive Prompted Transformer

The generated task-adaptive prompt is leveraged to instruct the ViT-based backbone to produce task-aware features for samples in the task $\mathcal{T}$. Specifically, give a sample $X \in \mathcal{T}$, the prompt tokens can be applied to the Transformer model f_t by being appended to the patch token sequence x_e of X: $x_{cls} = f_t([x_{cls}; P^{\mathcal{T}}; x_e])$. By the attention interaction between prompt tokens and visual tokens, the high-level task instruction can be injected into the instance embedding, steering the model to incorporate task-aware characteristics.

The class token x_{cls} outputted from f_t is taken as the final representation to perform the prototype-based classification (Snell, Swersky, and Zemel 2017) . Given the query feature $x^q \in \mathcal{T}$, we first compute N class prototypes by average pooling: $\hat{x}_c = \frac{1}{K}\sum_{j=1}^{K} x_j^s$, where x_j^s denotes the j-th support feature from class c. Then, the class probabilities over $c \in \{1, 2, \cdots, N\}$ for x^q are obtained based on proximities:

$$p(y = c|x^q) = \frac{exp(\Phi(x^q, \hat{x}_c))}{\sum_{c'=1}^{N} exp(\Phi(x^q, \hat{x}_{c'}))}, \tag{4}$$

where Φ denotes the cosine similarity metric. Naturally, the classification loss $\mathcal{L}_{cls}$ is formulated as: $\mathcal{L}_{cls} = -\frac{1}{|Q|}\sum_{(x^q,y^q)\in\mathcal{Q}}\log p(y = y^q|x^q)$.

After meta-training on D_{train}, the learned model is applied to novel tasks $\mathcal{T}_{test} \in D_{test}$. In addition to the task-adaptive prompting, we further tune bias parameters of f_t during meta-test by minimizing a classification loss over the support set, for the purpose of enhancing the performance on out-of-domain datasets. This practice of test-time tuning can be found in many prior works (Hu et al. 2022; Li, Liu,

Method	Arch	In-domain								Out-of-domain					ID	OD	Overall
		ImNet	Omni	Acraft	Bird	DTD	QDraw	Fungi	Flwr	Sign	COCO	MNS	CF10	CF100	Avg.	Avg.	Avg.
ProtoNet	RN18	44.5	79.6	71.1	67.0	65.2	64.9	40.3	86.9	46.5	39.9	NA	NA	NA	64.9	43.2	60.6
BOHB-E	RN18	55.4	77.5	60.9	73.6	72.8	61.2	44.5	90.6	57.5	51.9	NA	NA	NA	67.1	54.7	64.6
CNAPs	RN18	52.3	88.4	80.5	72.2	58.3	72.5	47.4	86.0	56.5	42.6	92.7	61.5	50.1	69.7	60.7	66.2
SimpleCNAPs	RN18	58.6	91.7	82.4	74.9	67.8	77.7	46.9	90.7	73.5	46.2	93.9	74.3	60.5	73.8	69.7	72.2
SUR	RN18	56.3	93.1	85.4	71.4	71.5	81.3	63.1	82.8	70.4	52.4	94.3	66.8	56.6	75.6	64.7	71.4
URT	RN18	55.7	94.4	85.8	76.3	71.8	**82.5**	63.5	88.2	51.1	52.2	94.8	67.3	56.9	77.3	64.5	72.3
FLUTE	RN18	51.8	93.2	87.2	79.2	68.8	79.5	58.1	91.6	58.4	50.0	95.6	78.6	67.1	76.2	69.9	73.8
URL	RN18	58.8	94.5	89.4	80.7	77.2	**82.5**	68.1	92.0	63.3	57.3	94.7	74.2	63.5	80.4	70.6	76.6
TSA	RN18	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	80.4	78.1	79.5
TSA*	ViT-S	72.2	**95.2**	86.4	92.5	**87.7**	74.7	67.9	96.0	89.7	60.2	96.4	89.2	76.9	84.1	82.5	83.5
PMF	ViT-S	74.6	91.8	88.3	91.0	86.6	79.2	74.2	94.1	88.9	62.6	95.6	85.4	77.6	85.0	82.0	83.8
MetaPrompt	ViT-S	**75.8**	90.8	89.6	**93.7**	86.3	79.5	**76.9**	**97.1**	**90.9**	**67.2**	96.6	**90.6**	**82.5**	**86.2**	**85.6**	**86.0**

Table 1: Comparison of MetaPrompt to the previous SOTA methods on Meta-Dataset. The in-domain datasets (i.e., the first eight datasets) are seen during meta-training, while the out-of-domain datasets (i.e., the last five datasets) are unseen and used for meta-test only. We report the In-domain Average Accuracy (ID Avg.), Out-of-domain Average Accuracy (OD Avg.), and the Overall Average Accuracy (Overall Avg.). * means the method is reimplemented with the same ViT backbone as MetaPrompt.

and Bilen 2022). The portion of tuned parameters is negligible compared to the whole network. Finally, the fine-tuned model is used to classify the query images by utilizing Eq. 4.

Experiments

In this section, we first introduce datasets and implementation details. Then, we present extensive experimental results.

Dataset Descriptions

We evaluate our model on Meta-Dataset (Triantafillou et al. 2019), a cross-domain few-shot learning benchmark that collects 10 public image datasets from a diverse range of domains: ImNet, Omni, Acraft, Bird, DTD, QDraw, Fungi, Flwr, Sign and COCO. In keeping with prior methods, we use the first 8 datasets for meta-training, where each dataset is further divided into train/val/test splits with disjoint classes. The test split of these datasets is used to evaluate the performance of seen domains (in-domain performance). The remaining two datasets are reserved as unseen domains to measure the out-of-domain performance. Besides, we follow the practice in (Requeima et al. 2019) to incorporate 3 additional meta-test datasets as the unseen domains, *i.e.*, MNS, CF10, and CF100.

Implementation Details

Following the practice in (Hu et al. 2022), we choose ViT-S (Dosovitskiy et al. 2021) with Dino (Caron et al. 2021) pre-trained weights as the backbone. In the task-conditioned prompt generator, we use the fixed ViT as the feature extractor. The prompt decoder is composed of one class-aware self-attention layer and two task-aware cross-attention layers. The class-wise embedding is meta-learned along with the backbone and prompt generator, and is shared across new meta-test tasks. The length of the prompt is set as 8. We follow the episodic training protocol and use the SGD optimizer with a learning rate of $1e-4$ for the ViT backbone and $5e-4$ for the prompt generator. The task-adaptive prompt is inserted into the second layer of ViT, which we empirically found performs best. During meta-test, we randomly sample 600 N-way K-shot tasks from the meta-test split of each dataset, where N varies from 5 to 50 and K varies from 1 to 100. The bias parameters of the backbone are tuned for 30 iterations for each task, using the Adadelta optimizer with a learning rate of 1. Notably, the bias tuning is performed only on out-of-domain datasets.

Comparison to State-of-the-art Methods

We compare MetaPrompt with recent CD-FSL methods and report the in-domain (ID Avg.), out-of-domain (OD Avg.), and overall average accuracy (overall Avg.) in Table 1.

Compared Methods. We select several CD-FSL methods for comparison, including ProtoNet (Snell, Swersky, and Zemel 2017), BOHB-E (Saikia, Brox, and Schmid 2020), CNAPs (Ren et al. 2016), SimpleCNAPs (Bateni et al. 2020), SUR (Dvornik, Schmid, and Mairal 2020), URT (Liu et al. 2021a), FLUTE (Triantafillou et al. 2021), URL (Li, Liu, and Bilen 2021), TSA (Li, Liu, and Bilen 2022), and PMF (Hu et al. 2022). Among them, URL (Li, Liu, and Bilen 2021), TSA (Li, Liu, and Bilen 2022) and PMF (Hu et al. 2022) tune their model or part of the model by several gradient steps during meta-test. Except for PMF, other methods adopt ResNet-18 as the backbone, which is somewhat out-of-date compared to the leading work in other computer vision fields. Therefore, we follow PMF (Hu et al. 2022) to build our model upon a pre-trained ViT backbone. As PMF does not provide results on three additional datasets, we obtain these results by using their officially released code [1]. For fair comparison, we reimplement the best prior CD-FSL method TSA with the same ViT backbone as MetaPrompt, denoted as TSA* in Table 1. We follow TSA to insert linear adapters after each ViT layer in a residual manner.

Result Analysis. As shown in Table 1, our proposed MetaPrompt achieves the highest results in 8 of 13 datasets, setting a new state-of-the-art record of 86.0 (+**2.2** %) in overall accuracy, demonstrating high generalization abil-

[1] https://github.com/hushell/pmf_cvpr22

M	Prompt Generator			Bias Tune	ID Avg.	OD Avg.	Overall Avg.
	TCA	CSA	CWE				
1					83.21	66.31	76.71
2	✓	✓			84.53	74.16	80.54
3	✓		✓		84.23	73.79	80.21
4	✓	✓	✓		**86.21**	75.21	82.00
5				✓	84.31	79.74	82.55
6	✓	✓	✓	✓	85.45	**85.56**	**85.49**

Table 2: Ablation study results on Meta-Dataset. The bold font numbers denote the best results.

M	Method	Arch.	ID Avg.	OD Avg.	Overall
Comparison with the Prompt Tuning strategy					
2	PT (lr=0.1)	ViT-S (Dino)	85.5	72.6	80.5
3	PT (lr=1)	ViT-S (Dino)	84.2	78.9	82.1
4	TAP + PT	ViT-S (Dino)	85.7	83.8	84.9
The Impact of Different Pre-trained Backbones					
5	MetaPrompt	ViT-S (Sup)	86.2 (81.1)	87.7 (75.4)	86.8 (78.9)
6	MetaPrompt	ViT-S (DeiT)	84.8 (82.3)	85.4 (67.5)	85.0 (76.6)
7	MetaPrompt	ViT-B (Dino)	87.4 (84.0)	86.5 (67.9)	87.1 (77.8)
8	MetaPrompt	ViT-B (Sup)	88.3 (83.9)	90.3 (77.0)	89.1 (81.2)

Table 3: Comparison with the prompt tuning strategy and the impact of different pre-trained backbones. $(\cdot)$ denotes the results of the corresponding baselines.

ity and adaptability to arbitrary domains. On in-domain datasets, MetaPrompt obtains competitive results, surpassing the second-best method by 1.2%. Notably, on out-of-domain datasets, our method largely outperforms the prior best methods (**+ 3.1%**), especially on COCO (**+ 4.6%**) and CF100 (**+ 4.9%**), which demonstrates the superiority of MetaPrompt in generalizing to the unseen domains. Compared to the methods based on test-time tuning (*i.e.,* URL, TSA and PMF), we achieve greater generalization performance on out-of-domain datasets. This is because the task-adaptive prompting mechanism could broadly generalize to unseen tasks by customizing task-aware parameters. Besides, the bias tuning strategy is also efficient and lightweight. Apart from that, our method performs consistently better across datasets compared to generation-based methods CNAPs and SimpleCNAPs. They use independent parameter generators to produce modulation parameters. Differently, our proposed attention-based prompt generator produces task-adaptive prompts of varying length, which are more expressive for delivering task instruction.

Ablation Study

In this section, we perform detailed ablation studies to demonstrate the efficacy of each design of MetaPrompt.

Baseline. In the baseline, the extracted features go through a prototype-based classifier for prediction without any task adaptation mechanism used. The training strategy and backbone is the same as the MetaPrompt.

Analysis of Model Components. We perform analysis of model components, including the Class-aware Self-Attention layer (**CSA**) in the prompt encoder, Task-aware Cross-Attention layer (**TCA**) in the prompt decoder, Class-Wise Embedding (**CWE**), and Bias Tuning strategy (performed on both in-domain and out-of-domain datasets). In general, all components contribute to the final performance. From Table 2, we have the following observations: ① **Using the Prompt Generator** (M4) to generate the task-adaptive prompt brings substantial improvement on both in-domain (**+3%**) and out-of-domain datasets (**+ 8.9%**) over the baseline (M1). The results indicate that the prompt generator can flexibly adapt to unseen tasks even without test-time tuning. The generated prompt injects high-level task knowledge into the task-agnostic representation, thus elevating the out-of-domain performance. ② **The removal of CWE** (M2) causes a considerable drop of 1.5% in overall accuracy compared to the complete generator (M4). CWE helps to discriminate samples from different classes, allowing the prompt generator to discover cross-class correlations for task adaptation. ③ **When removing CSA** (M3), the overall accuracy decrease significantly by **2.3%**, revealing the importance of CSA in expressing task knowledge. The co-adaptation process in CSA could emphasize useful samples while de-emphasizing the less-representative ones, leading to more reliable task instructions. ④ **The utilization of bias tuning** (M5) contributes considerably to the out-of-domain performance by optimizing a small number of parameters. By contrast, the in-domain improvements are relatively small than those achieved by using the prompt generator. ⑤ **When combining the prompt generator with bias tuning**, we achieve a whopping improvement of **8.8%** in the overall accuracy. However, the in-domain accuracy drops compared to solely using prompt generator (M4), due to the fact that bias tuning with a large learning rate is unstable on in-domain datasets. Thus, we choose not to perform bias tuning on in-domain datasets in the final version.

M	Generation Strategy	ID Avg.	OD Avg.	Overall Avg.	TC. Params.	Gen. Params.
Generating the prompt with length 8						
1	CNAPs	69.7	60.7	66.2	9.7k	9.24m
2	Avg. + MLP	84.9	70.9	79.5	3.1k	2.37m
3	Max + MLP	85.3	70.3	79.6	3.1k	2.37m
4	**MetaPrompt**	**86.2**	**75.2**	**82.0**	3.1k	2.39m
Generating the prompt with length 12						
5	Avg. + MLP	85.0	69.5	79.0	4.6k	3.55m
6	Max + MLP	84.8	70.3	79.2	4.6k	3.55m
7	**MetaPrompt**	**85.8**	**73.2**	**80.9**	4.6k	2.39m

Table 4: Comparison of different parameter generators in the accuracy, the amount of generator parameters (Gen. Params) and task-conditioned parameters (TC. Params).

Comparison with the Prompt Tuning Strategy

Prompt Tuning (PT) methods, including VPT (Jia et al. 2022) and CoOp (Zhou et al. 2022b), learn new prompts for downstream tasks by iterative tuning. Notably, CoOp is based on cross-modality models, contrasting with CD-FSL methods exclusively encompassing the image modality. We generally reimplement PT by learning prompts in every Transformer layer for each new task during meta-test,

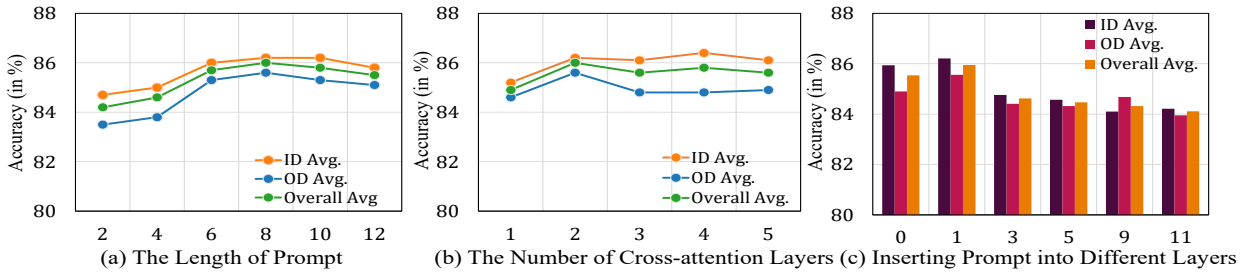

Figure 3: The impact of (a) the length of the prompt, (b) the number of cross-attention layers in the prompt decoder, and (c) the impact of inserting the task-adaptive prompt into different ViT layers.

using the same prompt length as MetaPrompt (see Tab. 3). MetaPrompt outperforms PT (M3) by **3.9%**, showing its superiority over PT. Notably, PT relies on good initialization and is highly sensitive to learning rates (lr) due to limited samples. It requires careful selection of lr, as the optimal lr on different datasets varies a lot (see M2-3). Instead of learning from scratch, MetaPrompt utilizes a meta-learned generator to automatically customize Task-Adaptive prompts (TAP) in a stable and efficient manner, obtaining higher adaptabilities even without test-time tuning (+ 8.9% in OD Avg). Moreover, combining PT with TAP (M4) further improves 2.8%, indicating that the meta-learned prompt and the tuned prompt have **complementary** impact.

Impact of Different Pre-trained Backbones

We compare several pre-trained backbones following PMF in Tab. 3. Compared to baselines, MetaPrompt consistently achieves significant improvements (7%∼9%) on various backbones (ViT-S and ViT-B) and pre-trained weights (M5-8). Notably, the performance of our method scales proportionately with the model size, showcasing substantial gains (7.9% and 9.3%, see M7-8) when using a larger Transformer backbone. It can be observed that the improvements of our MetaPrompt are not dependent on the specific type of pre-trained backbones, validating the effectiveness of the task-conditioned prompt generator.

Comparison of Different Generation Strategies

We compare different parameter generation strategies with regard to the amount of generator parameters (**Gen. Params**) and task-conditioned parameters (**TC. Params**) in Table 4 (w/o bias tuning). Compared to the most representative parameter generation method CNAPs, MetaPrompt claims consistent performance lead and enjoys higher parameter efficiency. The convolutional generators in CNAPs contain more than $\mathbf{3}\times$ TC. Params (*i.e.*, the scale and shift Params) and $\mathbf{3.8}\times$ Gen. Params compared to MetaPrompt (M1 v.s. M4). We also exploit alternative prompt generation strategies by replacing attention layers with M MLPs. The support features are compressed into the task embedding by average pooling and max pooling before sent into 2-layer MLPs. When generating the prompt with length 8, the MLPs have a similar parameter amount as MetaPrompt, but achieve much inferior performance (M2-3). In contrast to MLPs, the attention mechanism enables sufficient interaction between prompts and contextualized task characteristics. Moreover, as the MLP generator is independent to each prompt token, the increase of prompt length causes additional generators (M5-6), while MetaPrompt maintains a fixed capacity for prompts of any length.

Further Experiment Results

Analysis for Length of Prompt. Figure 3 (a) showcases the impact of the length of the task-adaptive prompt. Even when the length is as short as 2, the performance still remains comparable to previous methods, emphasizing the effectiveness of generated prompt. The accuracy increases when the prompt gets longer as it could deliver richer task knowledge. However, the accuracy decreases slightly when the length is beyond 8, as an excessive amount of prompt parameters may bring redundancy and noises. Overall, the accuracy changes relatively smoothly for different prompt lengths.

Analysis for Number of Task-aware Cross-attention Layers. Figure 3 (b) depicts the effect of the number of task-aware cross-attention (CSA) layers used (denoted as N_L). Using just a single CSA layer significantly improves accuracies upon baseline. The accuracy rises as N_L increases from 1 to 2, as one CSA layer may cause insufficient interaction between the prompt and task support samples. When $N_L > 2$, the performance no longer improves obviously, indicating that two CSA layers are flexible enough to enable effective communication for prompt tokens.

Analysis for Layers to Insert the Prompt. The previous prompting method (Lester, Al-Rfou, and Constant 2021) attaches prompt tokens to the first Transformer layer. Here we investigate the impact of attaching the prompt to different layers (see Figure 3 (c)). Inserting the prompt to layer 1 attains highest accuracies. When inserting into the middle and deep layers, the performance deteriorates obviously, as inserting into shallow layers affects more subsequent layers and enables deeper level modification.

Conclusion

We propose a Task-adaptive Prompted Transformer for CD-FSL. We design a task-conditioned prompt generator to generate a task-adaptive prompt from the contextualized task information, which could instruct the model to adapt to unseen domains even under the large domain gap. Experiments show the effectiveness of our proposed method.

Acknowledgments

This work was partially supported by National Defense Basic Scientific Research Program (Grant JCKY2021601B013).

References

Bateni, P.; Goyal, R.; Masrani, V.; Wood, F.; and Sigal, L. 2020. Improved few-shot visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 14493–14502.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33: 1877–1901.

Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE International Conference on Computer Vision*, 9650–9660.

Chen, M.; Fang, Y.; Wang, X.; Luo, H.; Geng, Y.; Zhang, X.; Huang, C.; Liu, W.; and Wang, B. 2020. Diversity Transfer Network for Few-Shot Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 10559–10566.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Dvornik, N.; Schmid, C.; and Mairal, J. 2020. Selecting relevant features from a multi-domain representation for few-shot classification. In *European Conference on Computer Vision*, 769–786.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 1126–1135.

Fu, Y.; Fu, Y.; Chen, J.; and Jiang, Y.-G. 2022. Generalized Meta-FDMixup: Cross-Domain Few-Shot Learning Guided by Labeled Target Data. *IEEE Transactions on Image Processing*, 31: 7078–7090.

Guo, Y.; Codella, N. C.; Karlinsky, L.; Codella, J. V.; Smith, J. R.; Saenko, K.; Rosing, T.; and Feris, R. 2020. A broader study of cross-domain few-shot learning. In *European Conference on Computer Vision*, 124–141.

Hu, S. X.; Li, D.; Stühmer, J.; Kim, M.; and Hospedales, T. M. 2022. Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9068–9077.

Islam, A.; Chen, C.-F.; Panda, R.; Karlinsky, L.; Feris, R.; and Radke, R. 2021. Dynamic Distillation Network for Cross-Domain Few-Shot Recognition with Unlabeled Data. In *Advances in Neural Information Processing Systems*.

Jia, M.; Tang, L.; Chen, B.-C.; Cardie, C.; Belongie, S.; Hariharan, B.; and Lim, S.-N. 2022. Visual prompt tuning. In *European Conference on Computer Vision*, 709–727.

Lee, K.; Maji, S.; Ravichandran, A.; and Soatto, S. 2019. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 10657–10665.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 3045–3059.

Li, W.-H.; Liu, X.; and Bilen, H. 2021. Universal representation learning from multiple domains for few-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 9526–9535.

Li, W.-H.; Liu, X.; and Bilen, H. 2022. Cross-domain few-shot learning with task-specific adapters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7161–7170.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 4582–4597.

Lifchitz, Y.; Avrithis, Y.; Picard, S.; and Bursuc, A. 2019. Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9258–9267.

Liu, L.; Hamilton, W. L.; Long, G.; Jiang, J.; and Larochelle, H. 2021a. A Universal Representation Transformer Layer for Few-Shot Image Classification. In *International Conference on Learning Representations*.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9): 1–35.

Liu, Y.; Lee, J.; Zhu, L.; Chen, L.; Shi, H.; and Yang, Y. 2021b. A multi-mode modulator for multi-domain few-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 8453–8462.

Oreshkin, B.; Rodríguez López, P.; and Lacoste, A. 2018. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, 721–731.

Phoo, C. P.; and Hariharan, B. 2021. Self-training For Few-shot Transfer Across Extreme Task Differences. In *International Conference on Learning Representations*.

Qi, H.; Brown, M.; and Lowe, D. G. 2018. Low-shot learning with imprinted weights. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Qiao, S.; Liu, C.; Shen, W.; and Yuille, A. L. 2018. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2016. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6): 1137–1149.

Requeima, J.; Gordon, J.; Bronskill, J.; Nowozin, S.; and Turner, R. E. 2019. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems*, 7959–7970.

Saikia, T.; Brox, T.; and Schmid, C. 2020. Optimized generic feature learning for few-shot classification across domains. *arXiv preprint arXiv:2001.07926*.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, 4077–4087.

Tian, Y.; Wang, Y.; Krishnan, D.; Tenenbaum, J. B.; and Isola, P. 2020. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, 266–282.

Triantafillou, E.; Larochelle, H.; Zemel, R.; and Dumoulin, V. 2021. Learning a universal template for few-shot dataset generalization. In *International Conference on Machine Learning*.

Triantafillou, E.; Zhu, T.; Dumoulin, V.; Lamblin, P.; Evci, U.; Xu, K.; Goroshin, R.; Gelada, C.; Swersky, K.; Manzagol, P.-A.; et al. 2019. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, 3630–3638.

Wang, Z.; Zhang, Z.; Ebrahimi, S.; Sun, R.; Zhang, H.; Lee, C.-Y.; Ren, X.; Su, G.; Perot, V.; Dy, J.; et al. 2022a. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, 631–648.

Wang, Z.; Zhang, Z.; Lee, C.-Y.; Zhang, H.; Sun, R.; Ren, X.; Su, G.; Perot, V.; Dy, J.; and Pfister, T. 2022b. Learning to prompt for continual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 139–149.

Wu, J.; Zhang, T.; Zhang, Y.; and Wu, F. 2021. Task-aware part mining network for few-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 8433–8442.

Wu, J.; Zhang, T.; Zhang, Z.; Wu, F.; and Zhang, Y. 2022. Motion-modulated temporal fragment alignment network for few-shot action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9151–9160.

Zhang, C.; Cai, Y.; Lin, G.; and Shen, C. 2020. DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover's Distance and Structured Classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12203–12213.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022a. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 16816–16825.

Zhou, K.; Yang, J.; Loy, C. C.; and Liu, Z. 2022b. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9): 2337–2348.