Prompt Vision Transformer for Domain Generalization

Zangwei Zheng[1,] **Xiangyu Yue**[2] **Wang Kai**[1] **Yang You**[1]
[1]National University of Singapore [2]Berkeley
{zangwei, kai.wang, youy}@comp.nus.edu.sg xyyue@berkeley.edu

Abstract

Though vision transformers (ViTs) have exhibited impressive ability for representation learning, we empirically find that they cannot generalize well to unseen domains with previous domain generalization algorithms. In this paper, we propose a novel approach `DoPrompt` based on prompt learning to embed the knowledge of source domains in domain prompts for target domain prediction. Specifically, domain prompts are prepended before ViT input tokens from the corresponding source domain. Each domain prompt learns domain-specific knowledge efficiently since it is optimized only for one domain. Meanwhile, we train a prompt adapter to produce a suitable prompt for each input image based on the learned source domain prompts. At test time, the adapted prompt generated by the prompt adapter can exploit the similarity between the feature of the out-of-domain image and source domains to properly integrate the source domain knowledge. Extensive experiments are conducted on four benchmark datasets. Our approach achieves 1.4% improvements in the averaged accuracy, which is $3.5\times$ the improvement of the state-of-the-art algorithm with a ViT backbone.

1 Introduction

Deep learning has achieved remarkable performances in various computer vision tasks [17, 37, 40]. With the development of network backbones, the learned representation is more generalizable when evaluated in the i.i.d. setting (independent and identical distribution). However, the i.i.d. setting is often violated in the real world due to the remarkable domain shift between training and test data. To tackle this problem, domain generalization (DG) aims to learn a generalizable model from a set of source domains and perform well on the unseen target domain.

Many works in domain generalization propose to learn a domain-invariant feature across source domains [15, 25, 30]. However, when source domains are very diverse, it is difficult to learn a domain-invariant model because each domain contains much domain-specific knowledge. Besides, domain-invariant features among source domains cannot ensure a small domain gap between the source and unseen target domains. Recent works [9, 39, 48] propose to give the prediction for an unseen target domain based on different experts in each source domain. Specifically, they divide network parameters into two sets. The domain-specific one contains components specific for each source domain, and the rest parameters are shared across all domains. In this way, the network can capture the specialized pattern in each domain more efficiently.

Domain-experts-based methods help the model to predict in an unseen test domain. However, two main challenges still remain in the previous methods. First, the specialized components should have the ability to capture different domain knowledge. Previous methods only choose some parameters (*e.g.*, bias, shallow layers, batchnorm) of the network and cannot interact with the whole network. This limits the ability of the network to fully incorporate the domain knowledge. In addition, since there is no access to the target domain, we need a well-designed machanism to extend the learned domain-specific knowledge for the target image at the inference time. [9] uses an ensemble method

Preprint. Under review.

by taking the average of output by the model with different source domain components and [48] adopts the most confident prediction. Both are ignorant of the relationship and similarity between input target images and source domains, and thus cannot fully take advantage of different source knowledge.

To tackle the first problem, we exploit the prompt learning to build powerful domain experts. Vision transformers (ViTs) have achieved striking performance in computer vision tasks [10, 28]. With the self-attention module, ViTs can fully model the relationship between different image patches. Prompt learning [27, 47] is a popular method for downstream task learning in NLP. For different downstream tasks, the knowledge about the task is embedded into the input tokens. Since tokens can interact with the prompt tokens at all layer levels through self-attention layers, the network can well understand the meaning of the task if the prompt carries enough information. Inspired by this, we hope to embed the different domain knowledge into different domain prompts. Compared with previous methods, the domain prompts are more powerful in carrying different knowledge, which means they can more efficiently capture the pattern in different domains.

For the second challenge, we propose to learn how to take better advantage of source information. Matching Network [43] proposes to adaptively use the seen information for the unseen one in the one-shot learning. In the one-shot generalization problem, the source knowledge is semantic information used to assign the class label for the test image. In contrast, for domain generalization, the source knowledge is about the domain information, which should not affect the semantic of each image. Thus, the design of adative knowledge combination should be different from the attention mechanism proposed in [43]. We propose a domain adapter that can analyze the domain distance between the input image and the source domains in the feature space and produces a better prompt for extracting a better representation of images in the unseen domain.

To enable more powerful domain experts and domain adapter, we propose `DoPrompt`, which consists of two objectives during training: Domain Prompt Learning (DPL) and Prompt Adapter Learning (PAL). For Domain Prompt Learning, image patch tokens are prepended with the prompt token of the corresponding domain. The domain prompt learns the domain-specific knowledge in each source domain. To take full advantage of the domain-specific knowledge for target domain prediction, we use a prompt adapter to integrate the learned source domain prompts to generate an well-adapted prompt for each target image. In order to learn a good prompt adapter, we encourage it to distinguish between features from different domains and produce prompts contributive to a correct prediction. During inference time, the adapted prompt generated by the prompt adapter is used for out-of-distribution prediction. Experiments show that the adapted prompt can reflect similarities between input images and different source domains.

We evaluate previous domain generalization methods and our algorithm based on ViT architecture with four standard benchmarks: PACS [23], VLCS [11], OfficeHome [42], and DomainNet [34]. With a controlled computing condition similar to [16], the proposed `DoPrompt` outperforms state-of-the-art methods with ResNet-50 by 4.7% on the averaged accuracy of four datasets. `DoPrompt` outperforms the baseline method with ViT-base as the backbone by 1.4%, which is $3.5\times$ the best improvement that previous performance can achieve.

To summarize, our contributions are as follows:

- We benchmark previous DG algorithms on ViTs and demonstrate that while ViTs outperform ResNet-50, many previous methods can hardly improve the performance of domain generalization with the ViT backbone.

- We propose an effective domain generalization algorithm `DoPrompt` for vision transformers with Domain Prompt Learning (DPL) and Prompt Adapter Learning (PAL).

- To the best of our knowledge, this is the first domain generalization algorithm considering the unique architecture of vision transformers, which outperforms state-of-the-art methods by a large margin across multiple benchmark datasets.

2   Related Work

Domain generalization with different architectures.   Many domain generalization approaches have been proposed during the past decade based on the strong ability of deep neural networks to learn a good representation. Early works [5, 23] on domain generalization evaluated their methods on

AlexNet [22]. In an attempt to show the approach can be applied to different network backbones, many works [6, 24] conducted experiments both on AlexNet and ResNet-18 [17]. However, the two networks are both convolutional neural networks and too shallow from nowadays view. Recently, an increasing number of works [3, 20, 31] applied their methods with ResNet-50 pretrained on ImageNet [8]. Among them, [16] benchmarked a large number of previous approaches on the ResNet-50 backbone with controlled computing budgets. It demonstrated that under the same setting, the basic Empirical Risk Minimization (ERM) loss outperforms many previous methods. This result shows that reconsidering the efficacy of DG methods with different backbones is necessary. In this paper, we investigate whether the past algorithms still work for the transformer-based vision model.

Vision transformer. [10] showed that pure transformer architecture could achieve state-of-the-art performance in image classification. Recently, [45] and [14] propose to use ViTs for domain adaptation. Both of them are based on the aligment method and cannot be extended to the domain generalization setting. [32, 36] found that ViTs learn a different visual representation from the convolutional neural network. They claimed ViTs are less biased to texture, which is assumed to be suspicious signals in the image, and show promising results under image corruption. We find vision transformers can serve as a strong baseline in domain generalization tasks, and we take advantage of the unique properties of ViTs to develop the `DoPrompt` algorithm.

Domain generalization algorithms. Many efforts [1, 13, 24–26, 38, 41, 44] have been devoted to learning a model that can generalize out-of-domains by the research community. We follow the categorization in [46] to overview previous DG algorithm and select the representative ones to be benchmarked with the vision transformer in Appendix A.

3   Method

In this section, we first preliminary knowledge about the domain generalization task and the vision transformer. Then, an overview of `DoPrompt` algorithm is provided. After that, two parts of the algorithm, Domain Prompt Learning (DPL) and Prompt Adapter Learning (PAL) are presented.

3.1   Preliminary

Problem Setting. In domain generalization, the training dataset is composed of K source domains where each domain $\mathcal{D}_i = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_i}$, where $\boldsymbol{x} \in \mathcal{X}$ denotes the image and $y \in \mathcal{Y}$ is the label. The whole source dataset can be represented as, $\mathcal{D}_S = \bigcup_{i=1}^{K} \mathcal{D}_i = \{(\boldsymbol{x}_i, y_i, d_i)\}_{i=1}^{n_s}$ where d_i stands for the domain label of an image and $n_s = \sum_{i=1}^{K} n_i$. DG goal is to learn a model $F : \mathcal{X} \to Y$ that can generalize well to a target domain $\mathcal{D}_T$, which is not accessible during training. Typically, the predictor F can be decomposed into $\phi \circ f$, where the feature extractor $f : \mathcal{X} \to \mathcal{H}$ learns the representation of the input image, and then the classifier $\phi : \mathcal{H} \to \mathcal{Y}$ predicts the class. In our case, the feature extractor f is a vision transformer.

Vision Transformer. The input image to a vision transformer $\boldsymbol{x}$ is first divided into k patches $\{I_i\}_{i=1}^{k}$. The encoding layer E transforms the input patches into patch tokens, and positional embeddings are added to them. The input to the transformer is the encoded patch tokens plus a classification token `[CLS]`. The vision transformer consists of ℓ blocks, and each block contains an attention layer and an MLP layer. The prediction of the vision transformer can be formulated as follows:

$$\texttt{[CLS]} = f([\texttt{[CLS]}, E(I_1), \cdots, E(I_k)) \tag{1}$$
$$y = \phi(\texttt{[CLS]}), \tag{2}$$

where $[\cdot]$ means concatenation of tokens. The interaction of different tokens happens in the attention layers of the vision transformer.

3.2   Overview

In domain generalization, our goal is to learn a robust model that can be applied to an unseen domain with multiple source domains. A trivial way is to learn one feature extractor to embed images from
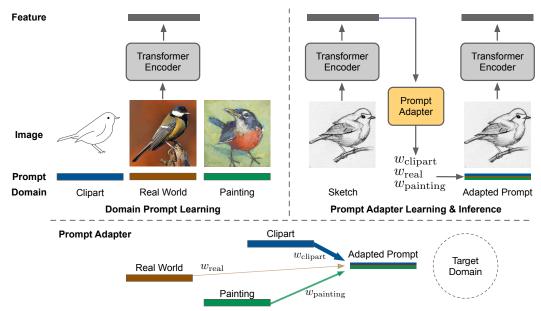
Figure 1: **An overview of the proposed** `DoPrompt` **algorithm**. **Left** shows the Domain Prompt Learning. **Right** shows both the Prompt Adapter Learning and the inference process. **Bottom** shows adapted prompt generation process, where each domain prompt is placed according to the domain position in the feature space. A thicker line denotes a larger weight.

all source domains into a common feature space. With a shared featurizer and classifier across all domains, we can train the model with Empirical Risk Minimization (ERM) loss as the following:

$$\mathcal{L}_{\text{ERM}} = \frac{1}{K} \sum_{i=1}^{K} \frac{1}{n_i} \sum_{(x,y) \in \mathcal{D}_i} \texttt{cross\_entropy}(\phi(f(\boldsymbol{x})), y). \tag{3}$$

However, the network is ignorant of the domain difference if we optimize the above loss directly on all domains. Our motivation is to learn better models for each domain by taking advantage of the domain information and properly exploiting them to perform robust prediction in the target domain. Domain-specific batchnorm [39] uses different batch normalization statistics and affine parameters for each domain. By doing this, the network with one specific batch normalization is optimized for the corresponding domain.

While the domain-specific structure can enhance the generalization performance of the model, there are some limitations of it. First, the domain information is limited to some specific layers. The network can only learn different normalizations for each domain. Second, when predicting in a target domain, the network cannot tell which domain-specific batchnorm is the most suitable one to use. As a result, a simple average of logits predicted by the model with different domain batchnorms is used for prediction. However, the distance between the target domain and source domains are different, as shown in Table 3, and a simple average may lead to bad performance.

In our method, we propose Domain Prompt Learning (DPL) and Prompt Adapter Learning (PAL) to learn a more discriminative feature in each source domain and a more robust combination of knowledge learned from different source domains for inference. Figure 1 shows the framework of our method. The left part shows learning with domain prompts (see 3.3), where the image is input combined with corresponding domain tokens. The right part shows both the Prompt Adapter Learning in training time and the inference method (see 3.4). The domain adapter gives a combination of learned source domain prompts by image features extracted with ViT backbone. The output adapted prompt is optimized for each input image.

3.3 Domain Prompt Learning

To learn network optimized for each source domain, we introduce the domain prompts to carry the domain-specific knowledge. For each domain, we have a set of specific prompts, namely L prompt

Algorithm 1 `DoPrompt` Algorithm

Input: number of steps T, featurizer f, classifier ϕ, domain prompts `prompts`, domain adapter A
1: **for** $s \leftarrow 1$ to T **do**
2:     Sample $(x_1^{i_1}, y_1^{i_1}), \cdots, (x_m^{i_m}, y_m^{i_m})$ from m domains $\mathcal{D}_1, \cdots, \mathcal{D}_m$
3:     $\triangleright$ Learning with domain prompts
4:     Calculate loss $\mathcal{L}_{\text{prompt}}$ according to Equation 6
5:     $\triangleright$ Learning with domain adapter
6:     **for** each domain i **do**
7:         $h_i \leftarrow \text{stop\_gradients}(f(x_i))$
8:         `adapted_prompts` $\leftarrow A(h)$
9:     Calculate loss $\mathcal{L}_{\text{w}}, \mathcal{L}_{\text{adapt}}$ according to Equation 12 and Equation 13
10:    $\mathcal{L} \leftarrow \mathcal{L}_{\text{prompt}} + \mathcal{L}_{\text{adapt}} + \lambda \mathcal{L}_{\text{w}}$
11:    Backward $\mathcal{L}$ and update the parameters

---

tokens. The prompt collections for K source domains are:

$$\text{prompts} = \left\{ \{ [\text{DOM}]^j \}_{i=1}^L \right\}_{j=1}^K. \tag{4}$$

Following the common practice in nature language processing, we append the prompts with the other tokens. With the domain prompt, the domain-aware feature is extracted as:

$$[\text{CLS}] = f_d(\boldsymbol{x}) = f([\ [\text{CLS}], \ \{E(I_i)\}_{i=1}^k, \ \text{prompts}_d \ ]), \tag{5}$$

where prompts_d means the prompts corresponding to domain d, and f_d represents the original model with appended domain prompts. The `prompts` are learnable parameters to capture knowledge contributing to a correct prediction on each domain. The training loss for DAL is:

$$\mathcal{L}_{\text{prompt}} = \frac{1}{|\mathcal{D}_S|} \sum_{(\boldsymbol{x}, y, d) \in \mathcal{D}_S} \text{cross\_entropy}(\phi(f_d(\boldsymbol{x})), y). \tag{6}$$

In vision transformer, domain prompts are treated the same as image tokens and the class token. They pass all attention layers and MLP layers. The domain knowledge is utilized through the attention mechanism, where image and class tokens can get information from the domain tokens. In this way, the whole model can take advantage of the domain information instead of some specific layers to give a more accurate prediction.

3.4 Inference and Prompt Adapter Learning

With Domain Prompt Learning, we can learn a good feature extractor for our source domains. As domain generalization asks for prediction on unseen target domains, it is important to design an approach to extend our knowledge from source domains. One direct method is to average the logits derived by different domain prompts, *i.e.* $\sum_{d=1}^{n_s} \phi(f_d(x))$. However, this method is ignorant of the relationship between different domains. As shown in Table 3, the distances between different domains vary, so we hope to exploit the knowledge in domain prompts while considering this information.

Since each domain prompt is optimized for one specific domain, we believe that for an input image, there also exists an optimal prompt for the domain the image comes from. As we only have domain knowledge of the source domains, we represent the prompt for unknown domains as a linear combination of different source domains:

$$\text{adapted\_prompt} = A(f(\boldsymbol{x})) = \left\{ \sum_{d=1}^K w_d^j \cdot [\text{DOM}]_d^j \right\}_{j=1}^K. \tag{7}$$

We propose a prompt adapter A to predict the linear combination weight w_d based on the extracted feature. Without domain prompts, the extracted features are more domain-discriminative (see Table 2 row 3), and A is able to learn the relationship between different domains and input images. We set A to be two-layer MLP with a appended softmax layer to ensure $\sum_{d=1}^K w_d^j = 1$. The combination is visualized in Figure 1 bottom. The prompt adapter learns an adaptive weight by analyzing the relationship between input target image and the source domains. It can automatically assign a larger weight for the domain prompt whose domain is closer to the target one than the others.

The inference process for any input image from unseen domains can be formulated as follows:

$$[\texttt{CLS}]' = f([\ [\texttt{CLS}],\ \{E(I_i)\}_{i=1}^k]) \tag{8}$$

$$\texttt{adapted\_prompts} = A(\ [\texttt{CLS}]'\ ) \tag{9}$$

$$[\texttt{CLS}]'' = f([\ [\texttt{CLS}]',\ \{E(I_i)\}_{i=1}^k,\ \texttt{adapted\_prompts}\ ]) \tag{10}$$

$$y = \phi(\ [\texttt{CLS}]''\ ). \tag{11}$$

The adapted prompts are conditioned on the input image so that a more appropriate prompt can be used for the prediction.

To train the domain adapter, we model the inference process during training by treating each training image an out-of-distribution image. Two losses are imposed in this process to learn a good prompt adapter. First, given the image from domain t, the most suitable prompts should be its corresponding domain prompts. Thus, we use a cross-entropy loss directly on the predicted weight by $A(f(x))$ as follows:

$$\mathcal{L}_{\text{w}} = \frac{1}{K} \sum_{j=1}^{K} \frac{1}{K} \Big(\texttt{cross\_entropy}(w_j^j, 1) + \sum_{d \neq t} \texttt{cross\_entropy}(w_d^j, 0) \Big). \tag{12}$$

On the other hand, the prompt adapter should learn to generate a prompt that can contribute to the final prediction of the image. No matter which domain the image comes from, prompts of other domains may also help in its prediction. Thus, we optimize the ERM loss when the image is input with the adapted prompts:

$$\mathcal{L}_{\text{adapt}} = \frac{1}{|\mathcal{D}_S|} \sum_{(\boldsymbol{x},y,d) \in \mathcal{D}_S} \texttt{cross\_entropy}(\phi(f_{\texttt{adapted}}(\boldsymbol{x})), y), \tag{13}$$

where $f_{\texttt{adapted}}$ is the original model with appended adapted domain prompts.

Taken together, the training algorithm of $\texttt{DoPrompt}$ is presented in Algorithm 1. Since $\mathcal{L}_{\text{prompt}}$ and $\mathcal{L}_{\text{adapt}}$ are ERM losses, we directly combine them together and use hyper-paramter λ to control loss $\mathcal{L}_{\text{w}}$. The inference process follows equations 8 – 11.

4 Experiment

4.1 Experimental setting

Datasets. We evaluate our approach on four public datasets: PACS, VLCS, OfficeHome, and DomainNet. **PACS** [23] is composed of four domains: **P**hotos, **A**rt, **C**artoon, and **S**ketch. It contains 9,991 images in 7 classes. **VLCS** [11] consists of 10,729 images in 5 classes from 4 real-world datasets, which are **V**OC2007, **L**abelMe, **C**altech, and **S**UN09. **OfficeHome** [42] is a more difficult dataset as it has 30,475 images in 65 classes. There are four different domains in this dataset: **A**rt, **C**lipart, **P**roduct, and **R**eal. **DomainNet** [34] is a large-scale benchmark with 586k images in 345 classes. It contains six domains which are **C**lipart, **I**nfograph, **P**ainting, **Q**uickdraw, **R**eal, and **S**ketch. Each domain is selected once as the target domain for each dataset, while the rest as source domains. Thus, the number of experiments carried out is the same as the number of all domains.

Implementation details. To fairly compare different methods, following [2, 16], we control the same computing condition and budgets. However, [16] conduct a random search of 20 trials on each domain of datasets. With three repeated trials, this results in a total of 1080 experiments to evaluate one method, which is too expensive. Another drawback of the evaluation is that it is hard to choose meaningful hyper-parameters for each algorithm. We adopt a two-stage grid-search to determine hyper-parameters for each method for our implementation. First, we search the general hyper-parameters such as learning rate and weight decay with a grid search for each dataset. Then, we fix these parameters and conduct a grid search for different hyper-parameters for each method. Following [16], we train 5k iterations for each setting.

The hyper-parameter search spaces of stage one and stage two are provided in the Appendix E. Since the hyper-parameter required for the vision transformer may significantly differ from the one for ResNet, we sweep a wide scale of learning rate and weight decay. After the grid search, we use

Table 1: Accuracy (%) on PACS, VLCS, OfficeHome and DomainNet with ViT-Base/16. The last column is the relative improvement of each method compared to ERM ViT baseline.

Algorithm	PACS	VLCS	OfficeHome	DomainNet	Avg.	Δ Avg.
ERM [16] (ResNet-50)	85.7	77.4	67.5	41.2	68.0	-3.8
Best in [16] (ResNet-50)	86.0	77.7	68.6	41.8	68.5	-3.3
ERM [16]	86.4	79.2	74.3	47.4	71.8	0.0
IRM [1]	83.9	79.5	72.3	27.1	65.7	-6.1
CDANN [26]	82.3	79.1	72.3	34.3	67.0	-4.8
DANN [13]	82.6	<u>79.6</u>	71.9	35.0	67.3	-4.5
GDRO [38]	86.6	78.9	74.5	41.3	70.3	-1.5
CORAL [41]	86.2	79.2	74.5	<u>47.7</u>	71.9	+0.1
MLDG [24]	87.0	78.7	<u>74.8</u>	47.6	72.0	+0.2
Mixup [44]	<u>87.4</u>	79.1	74.5	46.7	72.0	+0.2
MMD [25]	86.9	79.8	74.5	47.4	<u>72.2</u>	+0.4
DoPrompt	**88.1**	**80.4**	**76.0**	**48.3**	**73.2**	**+1.4**

Table 2: Averaged domain distances with different models.

Model	cross/in dist.	cross/in avg. class dist.
ResNet-50 (ImageNet Pretrained)	43	114
ViT-B/16 (ImageNet Pretrained)	14	17
DoPrompt w/o. domain prompts	12	10
DoPrompt w/o. fine-tuning backbone	21	41
DoPrompt	6	2

Table 3: Distance between different domains in the OfficeHome dataset on features from pretrained ResNet-50 (left) and ViT-Base/16 (right). In each table, upper right values are the distance between domains while lower left values denote the averaged distance between classes in different domains.

	Art	**C**lipart	**P**ainting	**R**eal		**A**rt	**C**lipart	**P**ainting	**R**eal
A	–	69.7	37.1	14.0	**A**	–	12.4	11.5	7.0
P	*73.3*	–	54.1	69.7	**P**	*10.6*	–	23.5	22.1
R	*178.8*	*159.6*	–	17.6	**R**	*38.4*	*26.3*	–	7.1
C	*124.2*	*99.7*	*51.6*	–	**C**	*13.8*	*10.5*	*5.8*	–

a dropout of 0.1 and weight decay of $1e^{-2}$. The learning rate is $5e^{-6}$ for PACS and VLCS, $1e^{-5}$ for OfficeHome, and $5e^{-5}$ for DomainNet. For DoPrompt, there are two hyper-parameters: the length of prompts L is set to 4 according to experimental performance, and the λ is searched within $\{0.1, 1, 10\}$.

We conduct our experiments based on the DomainBed [16] implementation. The framework is based on PyTorch [33], and we use ViT-Base/16 model pretrained on ImageNet for all experiments. We train the model with AdamW optimizer [21, 29] and the same data augmentation policy as [16]. We adopt the training-domain validation set method [16] for model selection. Specifically, each source domain is split into training and validation subsets. For DoPrompt, we apply the same inference process for validation like the one for the test instead of using the corresponding domain tokens. In this way, the validation result is a more reliable indicator of the test performance.

Domain distance. The difficulty of domain generalization lies in the domain shift across different domains. A well-learned feature space should have images with the same label close and images with different classes distant. However, due to large distribution discrepancy across domains, features of images in the same domain tend to have a smaller distance. We describe this property by cosine distance of domain centroids (detailed in Appendix B). The distance is measured on features from a ImageNet pretrained network. The domain distance measures the distance between two domain centroids, and the average domain class distance is the average class distance between different

Table 4: Accuracy (%) on OfficeHome (left) and DomainNet (right) with ViT-Base/16.

	A	C	P	R	Avg.		clip	info	paint	quick	real	sketch	Avg.
ERM [16]	$71.4_{\pm0.0}$	$59.4_{\pm0.5}$	$81.6_{\pm0.1}$	$84.9_{\pm0.3}$	74.3	ERM [16]	$67.0_{\pm0.1}$	$23.4_{\pm0.6}$	$54.5_{\pm0.3}$	$15.8_{\pm0.6}$	$\underline{69.3}_{\pm0.4}$	$54.6_{\pm0.5}$	47.4
MMD [25]	$71.8_{\pm0.2}$	$60.7_{\pm1.0}$	$81.4_{\pm0.3}$	$84.3_{\pm0.1}$	$\underline{74.5}$	MMD [25]	$\underline{67.0}_{\pm0.2}$	$\underline{23.8}_{\pm0.2}$	$54.0_{\pm0.0}$	$15.9_{\pm0.5}$	$69.0_{\pm0.1}$	$54.6_{\pm0.2}$	47.4
CORAL [41]	$72.3_{\pm0.7}$	$\underline{60.9}_{\pm1.0}$	$\underline{80.4}_{\pm0.4}$	$84.3_{\pm0.0}$	$\underline{74.5}$	CORAL [41]	$66.8_{\pm0.2}$	$24.4_{\pm0.1}$	$54.6_{\pm0.2}$	$16.2_{\pm0.2}$	$\underline{69.3}_{\pm0.1}$	$\underline{54.9}_{\pm0.3}$	$\underline{47.7}$
DANN [13]	$68.7_{\pm0.9}$	$55.8_{\pm0.4}$	$79.2_{\pm0.0}$	$83.9_{\pm0.2}$	71.9	DANN [13]	$45.6_{\pm0.1}$	$14.4_{\pm0.2}$	$44.6_{\pm0.6}$	$8.1_{\pm0.0}$	$52.3_{\pm0.3}$	$44.8_{\pm0.5}$	35.0
CDANN [26]	$68.2_{\pm0.4}$	$56.8_{\pm0.4}$	$79.8_{\pm0.4}$	$84.2_{\pm0.0}$	72.3	CDANN [26]	$45.3_{\pm0.5}$	$13.5_{\pm0.0}$	$45.8_{\pm0.0}$	$7.7_{\pm0.2}$	$49.6_{\pm0.1}$	$43.8_{\pm0.2}$	34.3
MLDG [24]	$72.3_{\pm0.0}$	$60.0_{\pm1.1}$	$82.2_{\pm0.2}$	$84.7_{\pm0.1}$	74.8	MLDG [24]	$65.8_{\pm1.4}$	$\underline{23.8}_{\pm0.1}$	$\underline{54.7}_{\pm0.1}$	$\underline{17.2}_{\pm0.2}$	$69.0_{\pm0.1}$	$\underline{54.9}_{\pm0.3}$	47.6
GDRO [38]	$71.2_{\pm0.7}$	$\underline{60.9}_{\pm0.4}$	$81.1_{\pm0.3}$	$84.7_{\pm0.3}$	$\underline{74.5}$	GDRO [38]	$59.3_{\pm0.2}$	$20.8_{\pm0.2}$	$46.0_{\pm0.6}$	$11.3_{\pm0.5}$	$63.7_{\pm0.1}$	$47.0_{\pm0.3}$	41.3
IRM [1]	$69.5_{\pm0.1}$	$57.1_{\pm0.3}$	$79.1_{\pm0.1}$	$83.3_{\pm0.1}$	72.3	IRM [1]	$35.6_{\pm0.5}$	$13.9_{\pm0.2}$	$33.1_{\pm0.8}$	$6.4_{\pm0.2}$	$40.6_{\pm0.4}$	$32.8_{\pm0.3}$	27.1
Mixup [44]	$\underline{72.7}_{\pm0.4}$	$59.7_{\pm0.5}$	$80.9_{\pm0.3}$	$84.6_{\pm0.2}$	$\underline{74.5}$	Mixup [44]	$65.1_{\pm0.1}$	$23.7_{\pm0.1}$	$54.4_{\pm0.1}$	$15.1_{\pm0.3}$	$67.9_{\pm0.0}$	$54.0_{\pm0.1}$	46.7
DoPrompt	$\mathbf{72.7}_{\pm0.5}$	$\mathbf{62.3}_{\pm0.6}$	$\mathbf{83.2}_{\pm0.1}$	$\mathbf{85.9}_{\pm0.1}$	$\mathbf{76.0}$	DoPrompt	$\mathbf{67.7}_{\pm0.2}$	$\mathbf{24.6}_{\pm0.1}$	$\mathbf{54.9}_{\pm0.1}$	$\mathbf{17.5}_{\pm0.2}$	$\mathbf{69.6}_{\pm0.3}$	$\mathbf{55.2}_{\pm0.5}$	$\mathbf{48.3}$

Table 5: Accuracy (%) on PACS (left) and VLCS (right) with ViT-Base/16.

	A	C	P	S	Avg.		C	L	S	V	Avg.
ERM [16]	$88.6_{\pm0.3}$	$81.3_{\pm0.3}$	$99.1_{\pm0.1}$	$76.1_{\pm0.9}$	86.4	ERM [16]	$96.9_{\pm0.3}$	$65.3_{\pm0.4}$	$75.2_{\pm0.5}$	$79.6_{\pm1.4}$	79.2
MMD [25]	$89.9_{\pm0.5}$	$80.8_{\pm0.5}$	$98.7_{\pm0.1}$	$\underline{77.1}_{\pm0.7}$	86.6	MMD [25]	$96.8_{\pm0.1}$	$65.6_{\pm1.0}$	$\underline{77.3}_{\pm0.6}$	$\underline{79.3}_{\pm1.5}$	$\underline{79.8}$
CORAL [41]	$89.7_{\pm0.3}$	$80.2_{\pm1.0}$	$98.7_{\pm0.2}$	$76.1_{\pm1.6}$	86.2	CORAL [41]	$96.2_{\pm0.7}$	$66.0_{\pm0.3}$	$76.8_{\pm1.6}$	$78.0_{\pm0.9}$	79.2
DANN [13]	$87.5_{\pm0.4}$	$78.6_{\pm0.9}$	$98.0_{\pm0.2}$	$66.3_{\pm1.7}$	82.6	DANN [13]	$\underline{97.7}_{\pm0.3}$	$\underline{66.5}_{\pm0.5}$	$76.0_{\pm1.0}$	$78.0_{\pm0.7}$	79.6
CDANN [26]	$87.3_{\pm0.6}$	$80.1_{\pm0.5}$	$98.5_{\pm0.2}$	$63.1_{\pm2.9}$	82.3	CDANN [26]	$96.3_{\pm0.2}$	$66.3_{\pm0.5}$	$76.5_{\pm0.6}$	$77.3_{\pm0.5}$	79.1
MLDG [24]	$90.4_{\pm0.5}$	$83.0_{\pm0.7}$	$98.7_{\pm0.0}$	$76.0_{\pm0.2}$	87.0	MLDG [24]	$95.7_{\pm0.6}$	$65.9_{\pm0.3}$	$76.3_{\pm0.8}$	$77.1_{\pm0.2}$	78.7
GDRO [38]	$90.1_{\pm0.6}$	$80.5_{\pm1.0}$	$98.8_{\pm0.1}$	$\underline{77.1}_{\pm1.5}$	86.6	GDRO [38]	$97.1_{\pm0.5}$	$65.2_{\pm0.7}$	$75.1_{\pm1.0}$	$78.3_{\pm1.0}$	78.9
IRM [1]	$89.1_{\pm0.7}$	$79.1_{\pm1.1}$	$98.3_{\pm0.1}$	$68.2_{\pm5.1}$	83.9	IRM [1]	$96.2_{\pm0.5}$	$65.7_{\pm0.8}$	$76.8_{\pm0.4}$	$77.7_{\pm0.9}$	79.5
Mixup [44]	$\underline{91.1}_{\pm0.1}$	$\underline{84.0}_{\pm0.3}$	$\underline{99.2}_{\pm0.2}$	$75.4_{\pm0.5}$	$\underline{87.4}$	Mixup [44]	$97.1_{\pm0.9}$	$65.4_{\pm0.2}$	$76.1_{\pm0.4}$	$78.0_{\pm0.3}$	79.1
DoPrompt	$\mathbf{91.1}_{\pm0.3}$	$83.0_{\pm0.2}$	$\mathbf{99.6}_{\pm0.0}$	$\mathbf{78.7}_{\pm0.8}$	$\mathbf{88.1}$	DoPrompt	$97.2_{\pm0.1}$	$\mathbf{67.4}_{\pm0.5}$	$\mathbf{77.4}_{\pm1.0}$	$\mathbf{79.7}_{\pm0.5}$	$\mathbf{80.4}$

Table 6: Ablation study of DoPrompt on OfficeHome.

Algorithm	A	C	P	R	Avg.
DoPrompt	$\mathbf{72.7}_{\pm0.5}$	$\mathbf{62.3}_{\pm0.6}$	$\mathbf{83.2}_{\pm0.1}$	$\mathbf{85.9}_{\pm0.1}$	$\mathbf{76.0}$
w/o. Domain Prompts	$71.4_{\pm0.0}$	$59.4_{\pm0.5}$	$81.6_{\pm0.1}$	$84.9_{\pm0.3}$	74.3
w/o. Domain Adapter	$71.7_{\pm0.3}$	$60.9_{\pm0.5}$	$81.9_{\pm0.3}$	$84.6_{\pm0.2}$	74.8
w/o. $\mathcal{L}_{w}$	$72.1_{\pm0.1}$	$62.1_{\pm0.3}$	$82.8_{\pm0.5}$	$84.9_{\pm0.3}$	75.5
w/o. $\mathcal{L}_{adapt}$	$71.8_{\pm0.1}$	$60.9_{\pm0.5}$	$82.1_{\pm0.2}$	$84.1_{\pm0.1}$	74.7
w/o. Fine-tuning backbone	$71.2_{\pm0.3}$	$58.7_{\pm0.5}$	$81.9_{\pm0.2}$	$84.1_{\pm0.2}$	74.0

domains. Table 3 shows the two distances between different domains, and Table 2 shows the two distances averaged on different domains.

4.2 Domain generalization results

First, we benchmark previous methods with ViT-base/16 backbone on PACS, VLCS, OfficeHome, and DomainNet, with the averaged accuracy of four datasets in Table 1, and the results for each dataset in Table 4 and Table 5. Unless otherwise specified, the bold text denotes the best results, and the underlined one is the best except for our methods. The ERM method on ViT-base/16 outperforms the ERM method, and the best results trained with ResNet-50 as the backbone by 3.8% and 3.3%, respectively. One reason for this phenomenon can be found in Table 3 and Table 2. The domain distance of features from ViT is much smaller than the one from ResNet-50 pretrained on ImageNet. Another observation is that as DG methods only improve the ERM baseline by 0.5% with ResNet-50, they also achieve a slight improvement with the ViT-base backbone. Only 0.4% averaged accuracy is achieved by the best previous methods on the four public datasets.

We compare our proposed DoPrompt with previous methods. DoPrompt improves the ERM baseline with ViT backbone by 1.4% while all other methods have an improvement of no more than 0.4% against the ERM baseline, which is a 3.5× improvement. DoPrompt achieves the best accuracy on each of the four benchmark datasets. If we look at the result of each target domain setting in each dataset, our algorithm outperforms previous methods on 16 out of 18 settings.

4.3 Ablation study

We investigate the effectiveness of different components in DoPrompt on OfficeHome. Table 6 shows the performance of different variants of our design. For the version without a domain adapter, since we cannot generate the adapted prompt, we use the averaged logits from different domain prompts for inference. Our two learning objectives of DoPrompt, Domain Prompt Learning and Prompt Adapter

Table 7: `DoPrompt` test accuracy when predicted with adapted prompt and domain prompts from source domains on OfficeHome dataset.

Target	Adapted	A	C	P	R
A	**72.9**	–	72.5	72.2	72.0
C	**62.1**	62.0	–	61.5	61.9
P	**83.0**	82.8	81.7	–	83.0
R	**85.2**	84.8	84.1	84.0	–

Table 8: Prompt adapter weight analysis on OfficeHome dataset with Clipart as the target domain.

	Percentage (%)			Average Value			Domain Distance		
	A	**P**	**R**	**A**	**P**	**R**	**A**	**P**	**R**
A	67.3	3.5	29.2	0.62	0.08	0.30	–	1.5	1.1
P	24.1	67.7	8.2	0.33	0.56	0.11	1.5	–	0.7
R	3.8	7.9	88.3	0.10	0.11	0.79	1.1	0.7	–
C	70.8	3.7	25.5	0.73	0.09	0.18	4.9	9.5	7.8

Learning, are both vital to the final performance. The great performance drop shows that the two components are both effective. For training the domain adapter, we show that our two losses are both effective. The results in rows 4 and 5 show that each loss is contributive to our final algorithm.

In our experiments, we fine-tune the whole model instead of only the prompt as in the natural language processing. Training only the prompts results in much lower accuracy in Table 6. One reason for this is that the number of parameters of the prompts is not enough to fully learn the downstream tasks. Another reason we find is shown in Table 2. Tuning prompts only cannot reduce the distance between different domains, thus resulting in a bad DG performance.

4.4 Domain adapter learns good prompts

To verify that our prompt adapter can learn a good prompt for the unseen domain prediction, we test the DG performance in the target domain with different domain prompts. Table 7 shows the target accuracy when predicted with the adapted prompt or a domain prompt from one source domain. As we can see, the prompt generated by the prompt adapter is better than the source domain prompts. This shows the prompt adapter does learn how to generate a good combination of domain prompts for generalized prediction.

Table 8 shows the statistics of the weight of adapted prompts generated by the prompt adapter in source and target domains. In the first two tables, each row denotes the classification of one domain, and each column denotes the weight corresponding to each domain prompt. The first table shows the percentage of weights that are the largest in each linear combination weight, and the second one is the averaged weight value for each domain prompt weight. For source domains, the corresponding domain prompts produced by the prompt adapter contain the most information from the corresponding domain as its weight of it is the largest one. This means the prompt adapter learns how to distinguish between different source domains.

The third table shows the domain distance between different domains. For the target domain Clipart, it is most close to the Art domain and far away from the Painting domain. Our prompt adapter learns to give a higher proportion of weights to the Art domain and less to the Painting domain. This reveals the prompt adapter can use the similarity between different domains to produce a better prompt for the target domain during test time.

5 Conclusion

In this paper, we first discovered that previous methods only have slight improvement when applied with vision transformers. We propose a novel domain generalization method `DoPrompt` inspired by prompt learning. `DoPrompt` learns domain-specific knowledge in each domain and a prompt adapter to generate suitably adapted prompt for prediction on out-of-domain images. Extensive experiments on multiple datasets demonstrate the superiority of `DoPrompt` over previous methods.

References

[1] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv*, vol. abs/1907.02893, 2019.

[2] D. Arpit, H. Wang, Y. Zhou, and C. Xiong, "Ensemble of averages: Improving model selection and boosting performance in domain generalization," *arXiv*, vol. abs/2110.10832, 2021.

[3] Y. Balaji, S. Sankaranarayanan, and R. Chellappa, "Metareg: Towards domain generalization using meta-regularization," in *NeurIPS*, 2018.

[4] F. C. Borlino, A. D'Innocente, and T. Tommasi, "Rethinking domain generalization baselines," *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 9227–9233, 2021.

[5] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *NIPS*, 2016.

[6] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *CVPR*, 2019, pp. 2224–2233.

[7] J. Cha, S. Chun, K. Lee, H.-C. Cho, S. Park, Y. Lee, and S. Park, "Swad: Domain generalization by seeking flat minima," in *NeurIPS*, 2021.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

[9] Z. Ding and Y. R. Fu, "Deep domain generalization with structured low-rank constraint," *IEEE Transactions on Image Processing*, vol. 27, pp. 304–313, 2018.

[10] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. De-hghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv*, vol. abs/2010.11929, 2021.

[11] C. Fang, Y. Xu, and D. N. Rockmore, "Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias," in *ICCV*, 2013, pp. 1657–1664.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017.

[13] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, 2016.

[14] C. Ge, R. Huang, M. Xie, Z. Lai, S. Song, S. Li, and G. Huang, "Domain adaptation via prompt learning," *arXiv preprint arXiv:2202.06687*, 2022.

[15] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter component analysis: A unified framework for domain adaptation and domain generalization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 7, pp. 1414–1430, 2016.

[16] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv*, vol. abs/2007.01434, 2021.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, pp. 770–778, 2016.

[18] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *ECCV*, 2016.

[19] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *ICCV*, 2017, pp. 1510–1519.

[20] Z. Huang, H. Wang, E. P. Xing, and D. Huang, "Self-challenging improves cross-domain generalization," in *ECCV*, 2020.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *CoRR*, vol. abs/1412.6980, 2015.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, 2012.

[23] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *ICCV*, 2017, pp. 5543–5551.

[24] ——, "Learning to generalize: Meta-learning for domain generalization," *arXiv*, vol. abs/1710.03463, 2018.

[25] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *CVPR*, 2018, pp. 5400–5409.

[26] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao, "Deep domain generalization via conditional invariant adversarial networks," in *ECCV*, 2018.

[27] X. Liu, K. Ji, Y. Fu, Z. Du, Z. Yang, and J. Tang, "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks," *arXiv*, vol. abs/2110.07602, 2021.

[28] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *ICCV*, pp. 9992–10 002, 2021.

[29] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *ICLR*, 2019.

[30] S. Motiian, M. Piccirilli, D. A. Adjeroh, and G. Doretto, "Unified deep supervised domain adaptation and generalization," in *ICCV*, 2017, pp. 5715–5725.

[31] H. Nam, H. Lee, J. Park, W. Yoon, and D. Yoo, "Reducing domain gap by reducing style bias," in *CVPR*, 2021, pp. 8686–8695.

[32] M. Naseer, K. Ranasinghe, S. H. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Intriguing properties of vision transformers," in *NeurIPS*, 2021.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NeurIPS*, 2019.

[34] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *ICCV*, 2019, pp. 1406–1415.

[35] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv*, vol. abs/1511.06434, 2015.

[36] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" In *NeurIPS*, 2021.

[37] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[38] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, "Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization," *arXiv*, vol. abs/1911.08731, 2019.

[39] S. Seo, Y. Suh, D. Kim, J. Han, and B. Han, "Learning to optimize domain specific normalization for domain generalization," in *ECCV*, 2020.

[40] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, 2017.

[41] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *ECCV Workshops*, 2016.

[42] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *CVPR*, 2017, pp. 5385–5394.

[43] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *NeurIPS*, vol. 29, 2016.

[44] Y. Wang, H. Li, and A. C. Kot, "Heterogeneous domain generalization via domain mixup," *ICASSP*, pp. 3622–3626, 2020.

[45] C. Zhang, M. Zhang, S. Zhang, D. Jin, Q. Zhou, Z. Cai, H. Zhao, X. Liu, and Z. Liu, "Delving deep into the generalization of vision transformers under distribution shifts," in *CVPR*, 2022, pp. 7277–7286.

[46] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization in vision: A survey," *arXiv*, vol. abs/2103.02503, 2021.

[47] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *arXiv*, vol. abs/2109.01134, 2021.

[48] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, "Domain adaptive ensemble learning," *IEEE Transactions on Image Processing*, vol. 30, pp. 8008–8018, 2021.

Appendix

A Additional Domain Generalization Methods

We follow the categorization in [46] to overview previous DG algorithm and select the representative ones to be benchmarked with the vision transformer.

- **Domain Alignment** methods are based on the assumption that models robust to out-of-domain distribution should learn features invariant to different source domains. These algorithms reduce the distance of learned representations between different domains. CORAL [41] minimizes the statistic difference while MMD [25] minimizes the Maximum Mean Discrepancy between different domains. DANN [13] and CDANN [26] are adversarial-based algorithms that use domain discriminator to measure the domain distance, with the latter conditioning on the class label.

- **Meta-Learning** can also help domain generalization. MLDG [24] meta-learns the entire neural network by dividing source domains into nonoverlapping meta-source and meta-target, which simulates the goal of domain generalization to the unseen target domain.

- **Data Augmentation** enlarges the training datasets to cover more unseen distributions. Easy augmentation such as Mixup [44] can help boost generalization ability. The most successful augmentation for domain generalization is related to style transfer [4], where images are transferred into different styles by a style transfer model such as AdaIN [19]. Mixing different styles of images and features is also an effective way [31, 46]. However, these works are based on the assumption that the statistics of the CNN activation map contain the style information, which cannot be directly extended to the vision transformer setting.

- **Ensemble Learning** methods can be generally divided into two groups, one ensembles models trained on different domains [9, 39] while another ensembles models at different steps during training [2, 7]. The prediction of the ensembled model is believed to be more robust than any single model.

- **Regularization Strategies** are also helpful for predicting out-of-distribution data. Among them, IRM [1] forces the linear classifier on each domain should be optimal, and Group-DRO [38] increases the weight of harder domains corresponding to higher classification errors.

B Domain Cosine Distance

To depict the distance between different domains, we measure the distance between different domains by the distance of corresponding centroids. The domain centroid and the class centroid in a domain is defined as follows:

$$\texttt{cent}_d = \frac{1}{|\mathcal{D}_d|} \sum_{i=1}^{\mathcal{D}_d} \boldsymbol{x}_i; \quad \texttt{cent}_c^d = \frac{1}{N_c^d} \sum_{\substack{y=c \\ (\boldsymbol{x},y) \in \mathcal{D}_d}} \boldsymbol{x}, \tag{14}$$

where N_c^d is the number of images with class c in domain d. The domain distance $\texttt{dist}(\cdot, \cdot)$ and averaged class distance between domains $\texttt{class\_dist}(\cdot, \cdot)$ are defined as follows. Since different models have a feature space with a different dimensions and different metrics, we normalize the distance of two centroids by in-domain and in-class distance, respectively:

$$\texttt{cosine\_dist}(\boldsymbol{x}_i, \boldsymbol{x}_j) = 1 - \frac{\boldsymbol{x}_i \cdot \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{x}_j\|} \tag{15}$$

$$\texttt{in\_dist}(D_i) = \frac{1}{|D_i|} \sum_{\boldsymbol{x} \in D_i} \texttt{cosine\_dist}(\boldsymbol{x}, \texttt{cent}_i) \tag{16}$$

$$\texttt{dist}(D_i, D_j) = \frac{\texttt{cosine\_dist}(\texttt{cent}_i, \texttt{cent}_j)}{\frac{1}{2}\big(\texttt{in\_dist}(D_i) + \texttt{in\_dist}(D_j)\big)} \tag{17}$$

$$\texttt{class\_dist}(D_i, D_j) = \frac{1}{n_c} \sum_{i=1}^{n_c} \texttt{dist}(D_i^c, D_j^c), \tag{18}$$

where n_c is the number classes and D_i^c is the set of images with class c in domain i.

C Additional Backbone Details

The detailed backbone information is provided in Table 9.

Table 9: Comparison between different backbones.

Model	# Params	ImageNet Acc@1 (%)
AlexNet	61M	56.5
ResNet-18	11M	69.8
ResNet-50	23M	76.1
ViT-Base/16	86M	81.1

D Ablation Study on Prompt Length

We choose the prompt length to be 4 according to the experimental performance. We experiment with prompt length [2,4,8,16,32] and find that different prompt length does not greatly affect the performance of `DoPrompt` under our experimental setting. However, we do notice that a bigger prompt length requires a higher learning rate to reach the best performance. To make the search space smaller and ensure a fair search space compared with the other methods as shown in Table 1 in the appendix, we set the prompt length to a fixed length. The accuracy of different prompt lengths under our test protocol on OfficeHome is shown in Table 10.

Table 10: Accuracy (%) on OfficeHome with different prompt length.

Length	2	4	8	16	32
Acc (%)	75.5	76.0	75.9	76.0	75.8

E Additional Implementation Details and Hyper-parameter Selection

Table 11: Hyper-parameter search space comparison.

Hyper-parameter	Random Search		Grid Search
	DomainBed [16]	Ensemble [2]	Ours
Learning rate (LR)	$10^{\text{Uniform}(-5,-3.5)}$	$5e^{-5}$	$\{1,5\} \times 10^{\{-6,-5,-4\}}$
Batch size	$2^{\text{Uniform}(3,5.5)}$	32	32
Classifier LR multipler	-	-	$\{1, 10, 100\}$
Last layer dropout [18]	$\{0, 0.1, 0.5\}$	$\{0, 0.1, 0.5\}$	$\{0, 0.1, 0.5\}$
Weight decay	$10^{\text{Uniform}(-6,-2)}$	$10^{\text{Uniform}(-6,-4)}$	$10^{\{-2,-3,-4,-5\}}$

For domain alignment-based methods, we apply the alignment loss on the class token at the last layer. The overall loss is $\mathcal{L} = \mathcal{L}_{\text{ERM}} + \gamma\mathcal{L}_{\text{Align}}$ where the align loss is different for MMD, CORAL, DANN and CDANN. The discriminator-based methods DANN and CDANN implemented in [16] did not follow the original one proposed in [13]. We implement the gradient reverse layer, use global weight decay for the discriminator, and adopt $\beta_1 = 0.5$ for Adam optimizer as suggested in [35]. In this way, we reduce the search space needed for discriminator-based methods. For MLDG, following [16], we adopt the first-order formulation [12] for a fair comparison with other first-order methods with respect to computing budget. For GroupDRO, IRM, and Mixup, we sample several hyper-parameters in the random search space in [16] at a different scale. The hyper-parameters for different methods in the second phase are listed in Table 12.

Table 12: Hyper-parameter search space for different algorithms. Recommended one are in bold text.

Method	Hyper-parameter	Random Search DomainBed [16]	Grid Search Ours
MMD CORAL	γ	$10^{\text{Uniform}(-1,1)}$	$\{\mathbf{0.1}, 1, 10\}$
DANN CDANN	γ	$10^{\text{Uniform}(-2,2)}$	$\{\mathbf{0.1}, 1, 10\}$
	discriminator weight decay	$10^{\text{Uniform}(-6,-2)}$	same to weight decay
	discriminator steps	$2^{\text{Uniform}(0,3)}$	no need
	gradient penalty	$10^{\text{Uniform}(-2,1)}$	no need
	adam β_1	$\{0, 0.5\}$	0.5
MLDG	β	$10^{\text{Uniform}(-1,1)}$	$\{0.1, 1, \mathbf{10}\}$
GroupDRO	η	$10^{\text{Uniform}(-1,1)}$	$\{\mathbf{0.01}, 0.1, 1, 10\}$
IRM	λ	$10^{\text{Uniform}(-1,5)}$	$\{\mathbf{0.1}, 1, 10, 100, 1000\}$
	iterations of penalty annealing	$10^{\text{Uniform}(0,4)}$	500
Mixup	α	0.1	$\{0.1, \mathbf{0.2}, 0.5\}$
DoPrompt	λ	–	$\{0.1, \mathbf{1}, 10\}$

F Additional Dataset Details

The detailed dataset information is provided in Table 13.

Table 13: Detailed dataset statistics

Dataset	Domain	# image	# image	# class
PACS [23]	**A**rt	2,048	9,991	7
	Photo	1,670		
	Clipart	2,344		
	Sketch	3,929		
VLCS [11]	**V**OC2007	3,376	10,729	5
	LabelMe	2,656		
	SUN09	3,282		
	Caltech101	1,415		
OfficeHome [42]	**A**rt	2,427	30,475	65
	Clipart	4,365		
	Product	4,439		
	Real	4,357		
DomainNet [34]	**C**lipart	48,129	586,575	345
	Infograph	51,605		
	Painting	72,266		
	Quickdraw	172,500		
	Real	172,947		
	Sketch	69,128		