Customizing Domain Adapters for Domain Generalization

Yuyang Ji[2⋆], Zeyi Huang[1⋆], Haohan Wang[2†], Yong Jae Lee[1†]
[1]University of Wisconsin-Madison [2]University of Illinois Urbaba-Champaign

Abstract

In this paper, we study domain generalization, where the goal is to develop models that can effectively generalize from multiple source domains to unseen target domains. Different from traditional approaches that aim to create a single, style-invariant model, we propose a new "Customized Domain Adapters" method, named CDA. This method leverages parameter-efficient adapters to construct a model with domain-specific components, each component focusing on learning from its respective domain. We focus on integrating the unique strengths of different adapter architectures, such as ViT and CNN, to create a model adept at handling the distinct statistical properties of each domain. Our experimental results on standard domain generalization datasets demonstrate the superiority of our method over traditional approaches, showcasing its enhanced adaptability and robustness in domain generalization tasks. Code is released at https://github.com/code456-star/CDA.

1. Introduction

Domain generalization is one of the most challenging settings in studying machine learning robustness [38]. This concept studies the ability of algorithms to generalize from multiple source domains to unseen target domains. A core hypothesis behind the setting of domain generalization is that: if the model can perform well on the collection of training domains, it is expected to learn the "semantic" information (i.e., the information that can be generalized across domains) but to ignore the "style" information (i.e., the information that is specific to a particular training domain) from the training data, and thus the model will generalize well to the testing domain.

Given this interesting setting and the promises of domain generalization in studying machine learning robustness, the community has developed a torrent of methods. As summarized in previous works [34]: one branch of the methods builds explicit regularization that pushes a model to learn representations that are invariant to the "style" across these domains [1, 5, 9, 12, 16, 26, 30, 31, 36, 37, 41, 44, 47, 56, 57, 60, 62, 65, 68, 70]; another branch is to perform data augmentation that can introduce more diverse data to enrich the data of certain "semantic" information with the "style" from other domains [13, 20, 49, 58, 64, 69], and also aims to train a model that is invariant to these "styles". More recently, there has been a line of approaches that aims to leverage the rich knowledge from large pre-trained vision language models to improve generalization performance [7, 8, 23, 32, 66]. For example, [8] synthesizes diverse styles via learnable text representations of CLIP to simulate distribution shifts. [23] distills the knowledge from the CLIP model into small models while pushing the models to ignore the "style" information in each domain through text representations.

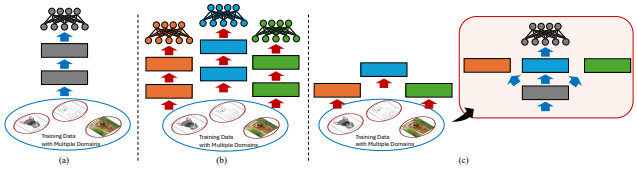

Figure 1. (a) Existing methods share the same avenue where their goal is to learn one model that is invariant to the "style" information from different domains. (b) The alternative way is to learn multiple models, and each model is only specialized in its own domain. (c) We create a Domain Adapter Customization with different components, and each of these components is specially designed to learn the knowledge of one domain, and then merge the learned information together for prediction.

However, we notice that most of the existing methods take a shared avenue in which the aim is to learn one model that is invariant to the "style" information from each domain. In this paper, we take another avenue that values the specification of each domain: whether we can create a model with different components, and each of these components is specially designed to learn the information of one domain, and then merge the learned information together for prediction. We believe our approach is intuitively advantageous because it recognizes the differences between domains, building upon an assumption that distinct domains exhibit unique statistical properties that might not be fully captured by a single model.

⋆,† equal contribution and advising.

In particular, in this paper, we leverage the power of parameter-efficient adapters to construct a customized model with different components. Each of the components specializes in learning from one domain, with different architecture designs, and these adapters are integrated to make a prediction. This method comprises two core steps: the construction of adapter models for each domain and the effective integration of their predictions.

For the first step, building upon the distinct advantages of ViT and CNN in learning shape or texture from image data [40, 42], we tailor architectural choices of adapters automatically by following either tried-and-tested design choices or LLM guidance. For the second step, a domain router is trained to dynamically assign the most suitable adapters for prediction based on the images' domain characteristics. We evaluate on standard domain generalization datasets – the results favor our method.

Unlike standard MoEs [46], whose main goal is to increase model capacity without a proportional increase in inference, our design focuses on learning customized adapters that capture different domain-specific properties (e.g., photos vs. sketches). Our approach also differs from a recent method [27] that applies MoEs to domain generalization. In [27], each expert is indirectly expected to learn a similar visual attribute like parts of an animal, whereas our method explicitly enforces each adapter to learn domain-specific properties. We find that this leads to stronger generalization capabilities to unseen domains.

Contributions. In summary, our main contributions are:

- We propose a new "Domain Adapter Customization" lightweight framework that synergistically combines the inductive biases of different domain adapters. This allows the model to adeptly handle varied distribution shifts, leveraging the strengths of both CNN and ViT architectures.
- We introduce a simple and effective approach for choosing different architectures as different domain adapters. By harnessing the intrinsic biases of domains, the model achieves enhanced domain generalization performances.
- We conduct a rich set of experiments to validate the effectiveness of our method on domain generalization benchmarks and ablate the performance of each of its components.

2. Related Work

2.1. Domain Generalization

The field of Domain Generalization has garnered considerable interest in recent years, as delineated by [38]. This area of study concentrates on the challenge of training models on data from diverse source distributions and testing on a target distribution distinct from those used during training. The underlying premise for method development in this domain is that a model capable of learning a representation invariant across multiple training domains should inherently perform well on an unseen test domain. Numerous methodologies have been developed under this presumption, striving to induce invariance in the training distribution samples. These approaches include explicit regularization-based methods [1, 5, 9, 12, 16, 26, 30, 31, 36, 37, 41, 44, 56, 57, 60, 62, 65, 68, 70] and data augmentation techniques [13, 20, 49, 58, 64, 69]. Recent studies have capitalized on large pre-trained vision-language models to enhance domain generalization. Notably, Domain Prompt Learning (DPL) [66] employs a prompt adaptor to generate prompts that capture domain-specific features from unlabeled data automatically. Other research [7, 32] has utilized CLIP as a foundational model to allocate suitable pre-trained models for each sample and to reframe the domain generalization objective through mutual information. Moreover, [23] leverage knowledge distillation using a large vision language model as a teacher for domain generalization. [8] synthesizes a variety of styles in a joint vision-language space via prompts to effectively tackle source-free domain generalization.

Key differences: A recent work named GMoE [27] directly applies a sparse Mixture-of-Experts framework to DG tasks. Each expert is expected to learn a similar group of visual attributes, such as head, body, leg, etc., of an animal. Instead, our methodology first customizes the architecture of each adapter to align with the intrinsic characteristics of the domain it represents. Then a domain router is trained to guide each adapter to adeptly learn domain-specific features, such as photos, sketches, etc.

2.2. Mixture of Experts

The Mixture-of-Experts (MoEs) models[50], renowned for their innovative gating mechanism, have recently demonstrated significant achievements by incorporating an extensive number of parameters while maintaining a constant computational cost. These models enhance overall performance by employing an input-dependent routing mechanism that amalgamates outputs from multiple sub-models, or 'experts.' This training approach has given rise to a variety of methods applicable across a broad spectrum of fields. However, the integration of MoEs with large-scale models inevitably leads to increased model sizes and prolonged inference times. To address this, Sparse MoEs have been introduced [46], featuring routers that activate only a select few experts, thereby aligning the inference time with that of standalone models. This approach is considered a promising avenue for scaling up vision models. Recent studies have applied the MoE architecture to various generalization tasks [27]. For instance, one study addressed multi-source domain adaptation in NLP by training a classifier for each domain and employing MoEs to ensemble these classifiers [15]. Another study explored the issue of systematic generalization, proposing the dy-

namic composition of experts in MoEs rather than selecting them [27].

Key differences: Contrary to standard MoEs, which aim to expand model capacity without significantly increasing computation, our approach centers on developing customized adapters tailored to capture domain-specific properties. The primary differences in our implementation are: 1) Within each ViT block, the standard FFN is replaced by an MoE layer, where each expert typically uses an FFN. Our approach differs by incorporating lightweight adapters after the MSA in each ViT block. 2) Standard MoE layers utilize a uniform architecture for each expert. Conversely, our method customizes each adapter to address specific domain properties. 3) We employ a domain router loss to dynamically steer incoming images to the adapter best suited to their domain characteristics, a feature not found in traditional MoE frameworks.

2.3. Parameter Efficient Fine-tuning (PEFT)

Existing PEFT methods can be generally divided into two groups. The first fine-tunes a small portion of the internal parameters, such as biases. The second adds tiny learnable modules like Adapters [18] and LoRA [19]. Adapters and LoRA essentially share similar architectures which both look like a bottleneck but are installed at different places. The Adapter is often installed at the output of a block while LoRA is treated as residuals to the projection matrices in a Transformer block. Another popular design in NLP is prompt learning, which turns some text prompt tokens into learnable vectors. Such an idea has recently been applied to vision-language models and is also the source of inspiration for the recently proposed VPT [24] and Visual Prompting. Inspired by these works, we follow some popular adapter architectures to customize our adapters.

3. Method

We first describe our notations and the standard ViT [10] architecture. We then present how we tailor the parameter-efficient adapters to domain generalization, before explaining our design choices for the approach and the implementation of domain adapters.

3.1. Preliminaries

Notation We use $(\mathbf{X}, \mathbf{Y}, \mathbf{D})$ to denote the training dataset with n (data, label, domain ID) paired samples. These data samples can be from multiple domains or distributions. Let $(\mathbf{x}, \mathbf{y}, \mathbf{d})$ denote one sample and $f(\cdot;\theta)$ denote the model we aim to train. Then, the standard empirical risk minimization (ERM) is:

$$\sum_{(\mathbf{x},\mathbf{y})\in(\mathbf{X},\mathbf{Y})} \ell(f(\mathbf{x};\theta),\mathbf{y}), \tag{1}$$

where $\ell(\cdot,\cdot)$ denotes a generic loss function.

Vision Transformer (ViT) For a plain ViT with 12 blocks, each ViT block consists of a Multiheaded Self-Attention (MSA) module and Feed-Forward Networks (FFN) together with LayerNorm and residual connections.

3.2. Customized Domain Adapters (CDA)

Motivated by the study that CNN and ViT networks potentially excel in different styles of images [40, 42], we leverage a compact ViT adapter and CNN adapter and insert them into the pre-trained network at fine-tuning time, hoping they can combine the complementary strengths of CNNs and ViTs to enhance generalization across diverse distributions. In this section, we first introduce the architecture of the ViT adapter and CNN adapter we use. Following this, we elaborate on our strategy for effectively integrating these adapters, harnessing their combined capabilities for enhanced performance.

ViT Adapter We follow the widely used adapter architecture [18] for ViT to design our ViT adapter. This adapter inserts a small number of additional parameter modules between transformer layers. As illustrated in Figure 2, the ViT adapter employs a down-projection with $\mathbf{W}_{\text{down}} \in \mathbb{R}^{q\times r}$ to project the input $\mathbf{h}$ with channel dimension q to a lower-dimensional space specified by bottleneck dimension r, followed by a nonlinear activation function GELU $f(\cdot)$, and an up-projection with $\mathbf{W}_{\text{up}} \in \mathbb{R}^{r\times q}$. The ViT adapter is surrounded by a residual connection, leading to this final form:

$$\mathbf{h}' \leftarrow \mathbf{h} + f(\mathbf{h}\mathbf{W}_{\text{down}})\mathbf{W}_{\text{up}}. \tag{2}$$

For storage efficiency, we set the bottleneck dimension much smaller than q, specifically $r = 48$ in our approach, whereas $q = 384$ serves as the default parameter for ViT-Small.

Conv Adapter As depicted in Figure 2, the Conv adapter comprises three convolutional layers: a 1×1 convolution reducing channels, a 3×3 convolution maintaining the same channel count for both input and output, and a final 1×1 convolution that expands the channels. Given that ViT converts the image into a 1D token sequence, we first revert to a 2D structure before 3×3 convolution. We treat [cls] token as an image and add zero padding to [cls] token to make it a 2D structure before 3x3 convolution. The Conv adapter is integrated within a residual connection, expressed as:

$$\mathbf{h}' \leftarrow \mathbf{h} + \text{Conv}_{1\text{x}1}(f(\text{Conv}_{3\text{x}3}(f(\text{Conv}_{1\text{x}1}(\mathbf{h}))))), \tag{3}$$

where $f(\cdot)$ represents the nonlinear activation function GELU. For storage efficiency, we define the bottleneck channel size for the 3×3 convolution as c. To ensure $c \ll q$, we set $c = 8$ in our method.

It is worth noting that the Conv adapter module is similar to the residual bottleneck blocks of ResNet [17]. By embedding the Conv adapter within the transformer structure, we aim to leverage the inherent visual inductive bias of CNNs

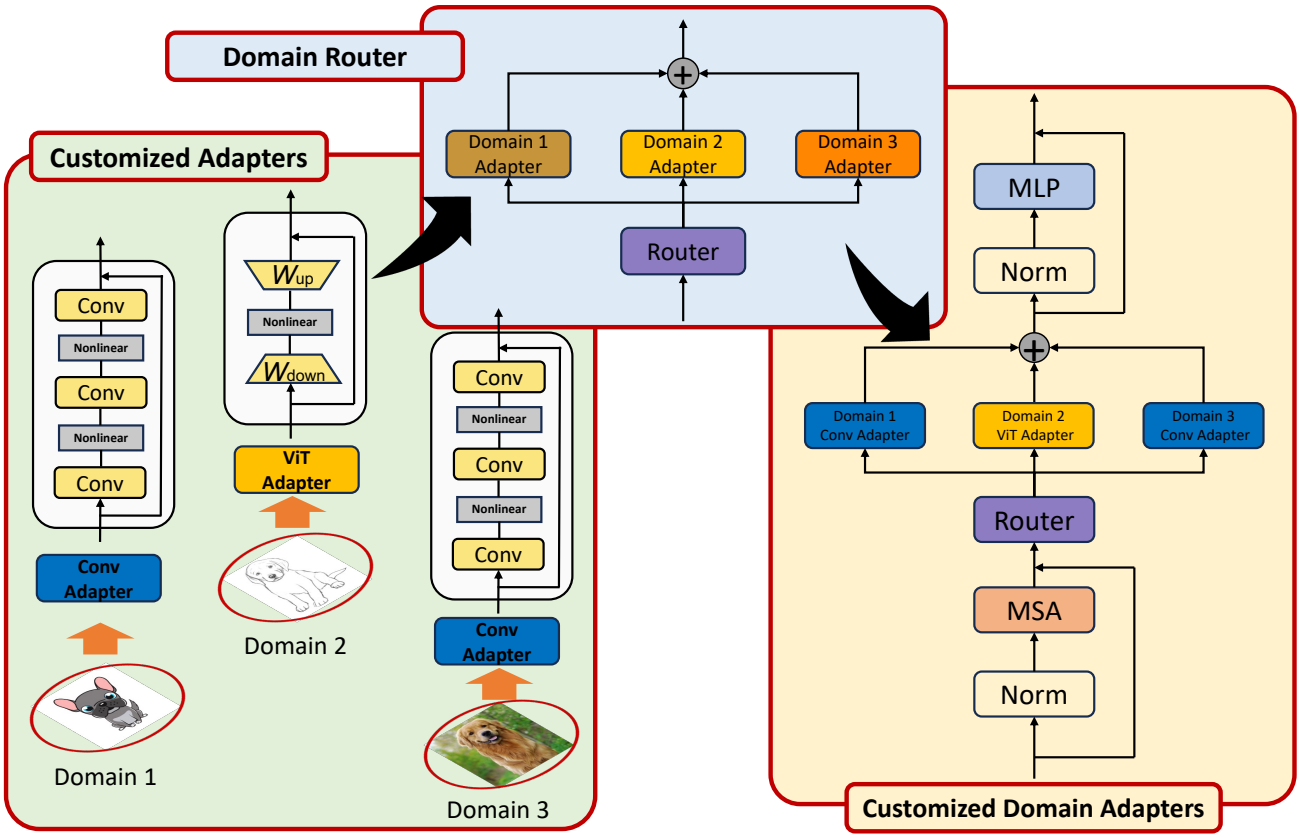

Figure 2. Pipeline of our method. In the first stage "Customized Adapters", we train two models separately, one with a ViT adapter and the other with a CNN adapter, and evaluate them across various domains. The domain is then automatically assigned to the adapter—ViT or CNN—that performs better in that specific domain. The ViT adapter is optimized for domains with less color or texture, while the Conv adapter excels in color and texture-rich domains. In the second stage "Domain Router", the domain router is trained to classify the specific domain to which each training image belongs. In inference, it dynamically routes images to the most fitting adapter based on their domain characteristics.

during the fine-tuning process. Furthermore, the local receptive field of the 3×3 convolution's 2D neighborhood structure offers a complement to the MSA's global receptive field.

Domain Adapter Customization In our network architecture, we integrate both ViT and Conv adapters within the ViT blocks, enabling each adapter to specialize in domain-specific characteristics. This configuration transforms the traditional transformer layers into a synergistic ensemble of transformers and ResNet-like CNNs.

A key consideration is selecting the appropriate ViT and Conv adapters for each domain in the dataset. We customize the adapter type to align with the domain's inherent properties. Typically, ViT-based adapters are preferable for domains lacking color or texture detail, as they excel in global structural analysis. Conversely, CNN-based adapters are ideal for texture-rich domains due to their proficiency in capturing detailed textural features [40, 42]. In practice, we adopt two methods to determine the most effective adapter for each domain. The first method involves randomly sampling one image from each domain, feeding it to GPT-4, and prompting GPT-4 to assign an adapter based on its world knowledge: *"Analyze the domain characteristics of the given image and identify the most suitable neural network adapter: either a ViT-based or CNN-based adapter."* The second method trains two separate models: one with a single ViT adapter and the other with a single CNN adapter. Each model is evaluated on various domains, and the adapter is chosen based on which model performs better in a specific domain. This empirical approach ensures the best adapter match but is more time-consuming compared to the GPT-4-based method, which, while faster, may not guarantee optimal matches. Empirically, both methods achieve comparable results and outperform most state-of-the-art approaches, as demonstrated in the experiment section. Users can choose the appropriate method based on their practical requirements. Our ablation study details the adapter assignments for each dataset, revealing that Conv adapters excel in color-rich domains, while

ViT adapters perform better in less colorful domains.

Domain Router and Loss Function We define the number of adapters N to be equal to the number of domains in our training data. This allows each adapter to specialize in a single domain, potentially improving the model's performance on domain-specific tasks. Therefore, for D domains, we have $N = D$ adapters. Each adapter E_i is trained to learn domain-specific features from domain i. Denoting the output of the MSA as $\mathbf{z}$, the output of the CDA layer with N adapters is given by

$$f_{\text{CDA}}(\mathbf{z}) = \sum_{i=1}^{N} G(\mathbf{z})_i \cdot E_i(\mathbf{z}) = \sum_{i=1}^{N} \text{Softmax}_i(\mathbf{W}\mathbf{z}) \cdot E_i(\mathbf{z}), \tag{4}$$

where N is equal to the number of training domains D, $\mathbf{W}$ denotes the learnable parameters for the gate G. We hope this design allows the model to benefit from the strengths of all the training domains, thus improving its ability to generalize to unseen visual domains.

The router plays a crucial role in dynamically assigning an incoming image to the most suitable adapter based on its domain characteristics. The router, a trainable component of our architecture, analyzes each input image and computes a set of weights that determine the contribution of each adapter for that specific image. To ensure that the router accurately discerns the domain of each image, we employ a router loss function.

$$\sum_{(\mathbf{x},\mathbf{d})\in(\mathbf{X},\mathbf{D})} \ell(\mathbf{W}\mathbf{z}, \mathbf{d}), \tag{5}$$

where $\mathbf{W}$ is the learnable parameter for the gate, $\mathbf{z}$ is the output embedding of the MSA, and $\ell(\cdot,\cdot)$ denotes a generic loss function. This function is designed to penalize the router when it incorrectly predicts the domain of an image. During training, the router loss guides the router in learning to identify the distinguishing features of each domain. We analyze the router's ability to accurately distribute weights to adapters for test images in the ablation study. The overall loss for our method is:

$$\sum_{(\mathbf{x},\mathbf{y},\mathbf{d})\in(\mathbf{X},\mathbf{Y},\mathbf{D})} \ell(f(\mathbf{x};\theta),\mathbf{y}) + \lambda\ell(\mathbf{W}\mathbf{z},\mathbf{d}), \tag{6}$$

where λ is the weight to balance the ERM and domain router loss. In this model, the cross-entropy loss is employed for $\ell(\cdot,\cdot)$. Detailed investigations into the configuration of the domain adapters layer and their placement within the ViT block are presented in supplementary.

4. Experiments

In this section, we evaluate our customized domain adapters on learning domain-specific knowledge for domain generalization. We compare state-of-the-art domain generalization methods and perform ablation studies to analyze the various components of our model.

4.1. Experiment Setup

We follow the setting in [14, 63] to evaluate our domain generalization approach. This includes consistent model selection criteria, dataset splitting, and using the same network backbone for comparability. To ensure a fair comparison with other methods reported on DomainBed, we select the ViT-S/16 model, following [27]. This model aligns closely with ResNet-50 in terms of parameter count and runtime memory usage. The ViT-S/16 features an input patch size of 16×16, 6 heads in its multi-head attention layers, and comprises 12 transformer blocks. Following DomainBed [14] and Ood-bench [63], we choose datasets that cover as much variety as possible from the various OOD research areas for our experiments. We conduct experiments on five OOD datasets: Terra Incognita [3], OfficeHome [55], VLCS [53], PACS [28] and DomainNet [43].

To be consistent with existing line of work, we use the training-validation protocol for model selection: given N domains, it uses 80% the amount of data in $N-1$ domains for training, the other 20% for validation, selects the best model based on the validation result, tests the model on the held-out domain and reports this result. The experiments are averaged over 3 runs.

There is one hyperparameter for our method – the weight of the router loss λ; we set the hyperparameter search space of our method as $\lambda \in [0.01, 0.1]$. We adopt the same hyperparameter search protocol used in [14, 63].

4.2. Main Results

We compare to recent top DG algorithms, using both ViT-Small/16 (21.7M parameters) and ResNet50 (25.6M parameters) pre-trained on ImageNet-1k as the backbone. The results in Table 1 demonstrate that our domain adapter module, trained with ERM, surpasses leading DG algorithms on nearly all datasets. Specifically, even using the smaller backbone ViT-S/16, our method achieves improvements of 0.7%, 0.8%, 1.9%, and 1.6% over the Best SoTA Competitors on the PACS, VLCS, OfficeHome, and DomainNet benchmarks, respectively. The term "Best SoTA Competitor" refers to the highest performance in the literature within the standard DG experimental protocol. In our following ablation study, we explore how our model effectively utilizes the expertise of individual adapters and adapts to images with characteristics spanning multiple domains, thereby enhancing its generalization performance on unseen domains.

4.3. Ablation on Model Design

In this section, we study the impact of each component in our model design. We evaluate our method with a ViT-S/16 backbone on the popular DG benchmark, such as PACS, and DomainNet to conduct the following experimental analyses. We observed that fully finetuning all network parameters outperforms finetuning only custom domain adapters and

Algorithm	Backbone	PACS	VLCS	OfficeHome	TerraInc	DomainNet
ERM [54]	RN50	85.7 ± 0.5	77.4 ± 0.3	67.5 ± 0.5	47.2 ± 0.4	41.2 ± 0.2
IRM [2]	RN50	83.5 ± 0.8	78.5 ± 0.5	64.3 ± 2.2	47.6 ± 0.8	33.9 ± 2.8
Mixup [61]	RN50	84.6 ± 0.6	77.4 ± 0.6	68.1 ± 0.3	47.9 ± 0.8	39.2 ± 0.1
RSC [21]	RN50	85.2 ± 0.9	77.1 ± 0.5	65.5 ± 0.9	46.6 ± 1.0	38.9 ± 0.5
CDANN [30]	RN50	82.6 ± 0.9	77.5 ± 0.1	65.8 ± 1.3	45.8 ± 1.6	38.3 ± 0.3
DANN [11]	RN50	84.6 ± 1.1	78.7 ± 0.3	68.6 ± 0.4	46.4 ± 0.8	41.8 ± 0.2
CORAL [52]	RN50	86.0 ± 0.2	77.7 ± 0.5	68.6 ± 0.4	46.4 ± 0.8	41.8 ± 0.2
MLDG [29]	RN50	84.9 ± 1.0	77.2 ± 0.4	66.8 ± 0.6	47.7 ± 0.9	41.2 ± 0.1
AND-mask [48]	RN50	84.4 ± 0.9	78.1 ± 0.9	65.6 ± 0.4	44.6 ± 0.3	37.2 ± 0.6
MMD [30]	RN50	85.0 ± 0.2	76.7 ± 0.9	67.7 ± 0.1	42.2 ± 1.4	39.4 ± 0.8
Fish [51]	RN50	85.5 ± 0.3	77.8 ± 0.3	68.6 ± 0.4	45.1 ± 1.3	42.7 ± 0.2
SagNet [39]	RN50	86.3 ± 0.2	77.8 ± 0.5	68.1 ± 0.1	48.6 ± 1.0	40.3 ± 0.1
SelfReg [25]	RN50	85.6 ± 0.4	77.8 ± 0.9	67.9 ± 0.7	47.0 ± 0.3	42.8 ± 0.0
mDSDI [4]	RN50	86.2 ± 0.2	79.0 ± 0.3	69.2 ± 0.4	48.1 ± 1.4	42.8 ± 0.1
SWAD [6]	RN50	88.1 ± 0.1	79.1 ± 0.1	70.6 ± 0.2	50.0 ± 0.3	46.5 ± 0.1
Fishr [45]	RN50	85.5 ± 0.2	77.8 ± 0.2	68.6 ± 0.2	47.4 ± 1.6	41.7 ± 0.0
MIRO [7]	RN50	85.4 ± 0.4	79.0 ± 0.0	70.5 ± 0.4	50.4 ± 1.1	44.3 ± 0.2
PCL [59]	RN50	88.7	-	71.6	**52.1**	47.7
ERM [27]	ViT-S	86.2 ± 0.1	79.7 ± 0.0	72.2 ± 0.4	42.0 ± 0.8	47.3 ± 0.2
GMoE [27]	ViT-S	88.1 ± 0.1	80.2 ± 0.2	74.2 ± 0.4	48.5 ± 0.4	48.7 ± 0.2
CDA w GPT (Ours)	ViT-S	89.3 ± 0.2	80.7 ± 0.1	75.5 ± 0.2	50.8 ± 0.6	50.1 ± 0.4
CDA (Ours)	ViT-S	**89.4 ± 0.3**	**81.0 ± 0.2**	**76.1 ± 0.1**	51.3 ± 0.4	**50.3 ± 0.4**

Table 1. Results of recent top DG algorithms, using both ViT-Small/16 (21.7M parameters) and ResNet50 (25.6M parameters) pre-trained on ImageNet-1k as the backbone with train-validation selection criterion. The experiments are averaged over 3 runs. The best result is highlighted in bold. Our method achieves the best performance on PACS, VLCS, OfficeHome, and DomainNet, and second best on TerraInc. Note, "CDA w GPT" indicates using GPT to assist in adapter assignment. This approach achieves results comparable to the tried-and-tested design choice but is significantly more time-efficient. Users can choose the appropriate method based on their practical requirements.

Adapter type	Router Loss	Adapter# / Acc(PACS)	Adapter# / Acc(DomainNet)
ViT adapter	w/o	2 / 87.9 ± 0.2	2 / 48.3± 0.3
ViT adapter	w/o	3 / 88.0 ± 0.1	5 / 48.7± 0.3
ViT adapter	w	3 / 88.5 ± 0.3	5 / 49.4± 0.4
Conv adapter	w/o	2 / 88.1 ± 0.3	2 / 48.8± 0.2
Conv adapter	w/o	3 / 88.3 ± 0.2	5 / 49.0± 0.4
Conv adapter	w	3 / 88.7 ± 0.4	5 / 49.6± 0.4
Customized adapter	w/o	2 / 88.9 ± 0.3	2 / 49.3± 0.3
Customized adapter	w/o	3 / 88.9 ± 0.1	5 / 49.6± 0.5
Customized adapter	w	3 / 89.4 ± 0.3	5 / 50.3± 0.4

Table 2. Analysis of router loss and ViT and Conv adapter customization. Note that since the PACS and DomainNet have three and five training domains respectively, we only apply router loss when the number of adapters is equal to the number of training domains. "Customized adapter" means combining the ViT and Conv adapter according to our method in Section 3.2.

classifier with 1.2% on the PACS dataset in our experiments. This likely stems from the relatively small size of our pre-trained network, where a fine-tune-only adapter may not suffice for downstream tasks. Consequently, we employ full fine-tuning strategies in all our experiments.

Impact of router loss and customization of ViT and Conv adapters We first examine the contribution of domain router loss in Table 2. For example, in PACS, whenever the adapters are trained without router loss, the performance drops by 0.5%, 0.4%, and 0.5% for 3 ViT adapters, 3 Conv adapters, and customized adapters, respectively. We next study the effect of combining the ViT and Conv adapters. The integration of varied adapter types consistently yields a performance increase of at least 0.6%. For example, combining a single ViT adapter and a single Conv adapter results in an 88.9% accuracy-a gain of 1.0% and 0.8% compared to their individual performances without router loss, which are 87.9% and 88.1%, respectively. Lastly, we study the effect of adding more adapters of the same type. It reveals a marginal benefit, with a maximum performance gain of 0.2% across all three adapter types examined in Table 2. The same trend can also be found in the larger DomainNet benchmark. In summary, our findings highlight the significance of the router loss and strategic incorporation of customized adapters in the design of our module. Conversely, increasing the number of identical adapters appears to have a minimal effect on performance.

Analysis of Using CLIP Pretrained Weights To further assess the effectiveness of our CDA, we apply it to CLIP pretrained weights. In Table 3, CDA achieves competitive performance across all benchmark datasets, including PACS, VLCS, and OfficeHome. Notably, CDA outperforms pre-

Method	Backbone	PACS	VLCS	OfficeHome
DPL [67]	CLIP ViT-B16	97.3	84.3	84.2
MIRO [7]	CLIP ViT-B16	95.6	82.2	82.5
CAR-FT [35]	CLIP ViT-B16	96.8	85.5	85.7
SIMPLE+ [33]	Model Pool	99.0±0.1	82.7±0.4	87.7±0.4
CDA(Ours)	CLIP ViT-B16	98.6±0.1	84.0±0.3	91.2±0.1

Table 3. Analysis of Using CLIP Pretrained Weights: To further evaluate the effectiveness of our CDA, we apply it to CLIP pretrained weights. Our results demonstrate that even when using CLIP pretrained weights, CDA continues to deliver SOTA performance, highlighting its adaptability.

Method	Backbone	Param / Acc(PACS)	Param / Acc(DN)	Latency(ms)
ERM [54]	ResNet50	25.6M / 85.7 ± 0.5	25.6M / 41.2 ± 0.2	16.7
ERM [27]	ViT-S/16	21.7M / 86.2 ± 0.1	21.7M / 47.3 ± 0.2	16.9
GMoE [27]	ViT-S/16	33.8M / 88.1 ± 0.1	33.8M / 48.7 ± 0.2	65.6
CDA(Ours)	ViT-S/16	23.9M / 89.4 ± 0.3	24.5M / 50.3± 0.4	20.5 / 28.7
ERM [54]	ViT-B/16	85.8M / 88.8 ± 0.4	85.8M / 50.6± 0.4	17.5
GMoE [27]	ViT-B/16	133.4M / 89.4 ± 0.1	133.4M / 51.3 ± 0.1	160.2
CDA(Ours)	ViT-B/16	90.9M / 90.1 ± 0.8	91.4M / 54.5± 0.3	25.3 / 31.8

Table 4. Analysis of scaling up architecture and inference cost. Our approach achieves notable performance improvements with a small impact on inference speed.

vious methods like DPL [67] and CAR-FT [35] on OfficeHome, with a significant margin of 5.5% over CAR-FT. Moreover, while SIMPLE+ [33] shows slightly better results on PACS, CDA maintains a consistent balance of high performance across all domains, demonstrating its robustness and generalization capabilities. These findings underscore CDA's ability to leverage powerful pretrained weights like CLIP effectively, adapting them to diverse domain generalization tasks.

Scaling up architecture and inference cost We investigate whether similar performance improvements are observed with larger architectures in Table 4. Our Customized Domain Adapter consistently enhances ViT backbones across different scales. Specifically, in the PACS benchmark, CDA adds an extra 2.2M parameters and achieves a 3.2% performance increase for ViT-S, and adds 5.1M parameters for a 1.3% boost in ViT-B. Furthermore, we assessed the additional inference costs introduced by our CDA method. CDA incurs an additional 3.5ms for ViT-S and 8ms for ViT-B, significantly lower than the GMoE's added 45ms for ViT-S and 140ms for ViT-B. Latency measurements are conducted on an 24GB Nvidia RTX 3090 GPU. Similar patterns are observed in the DomainNet benchmark. In conclusion, our approach achieves notable performance improvements with minimal impact on inference speed.

Orthogonal to other DG algorithms Our method lies in an orthogonal direction with most DG algorithms – the design of the backbone architecture. This suggests that integrating DG algorithms could further enhance the performance of our CDA. In this section, we investigate whether our method complements other DG plug-in strategies, such as the data augmentation method W2D [22] and the model ensemble method [6], as shown in Table 5. The results confirm that CDA serves as an effective architecture, synergizing with these methods to yield significant performance gains of 2.2% and 3.3% for SWAD and W2D on PACS, and 2.8% and 3.1% on DomainNet, respectively.

Method	Acc(PACS)	Acc(DomainNet)
CDA	89.4 ± 0.3	50.3± 0.4
SWAD [6]	88.1 ± 0.1	48.2 ± 0.2
SWAD [6] + CDA	90.3 ± 0.2	51.0 ± 0.3
W2D [22]	86.6 ± 0.4	47.8 ± 0.4
W2D [22] + CDA	89.9 ± 0.5	50.7 ± 0.5

Table 5. Analysis of complementarity to other DG algorithms. Our CDA is an effective network architecture for other kinds of DG algorithms.

Method	Backbone	PACS	OfficeHome	VLCS	Memory
ERM [27]	ViT-S/16	2722s	3102s	3635s	5.8GB
GMoE [27]	ViT-S/16	6013s	6996s	6841s	7.6GB
CDA(Ours)	ViT-S/16	6585s	7614s	8235s	6.8GB
ERM [27]	ViT-B/16	8167s	8742s	9233s	11.6GB
GMoE [27]	ViT-B/16	14724s	16404s	15804s	14.7GB
CDA(Ours)	ViT-B/16	13182s	14831s	14283s	13.0GB

Table 6. Comparison of runtime (in seconds) and memory usage (in GB) across PACS, OfficeHome, and VLCS datasets for different methods. Our approach demonstrates competitive performance with manageable memory overhead.

Further analysis of the domain router The domain router is essential for dynamically assigning incoming images to the most suitable adapter based on domain characteristics. In Table 7, we evaluate the router's ability to accurately distribute weights to adapters for each image by averaging all router output weights during inference on the PACS dataset. For instance, in tests within the art domain, routers tend to allocate higher weights to the photo domain due to its color richness and similarity to the art domain, while significantly less weight is given to the sketch domain adapter. These findings confirm the domain router's effectiveness in guiding images to their optimal adapters based on domain traits. Additionally, Figure 3 presents the predicted weights for test samples from PACS across different domains, illustrating this process.

Result of domain adapter customization for each benchmark Our method, detailed in Section 3.2, centers on choosing the right ViT and Conv adapters for each domain

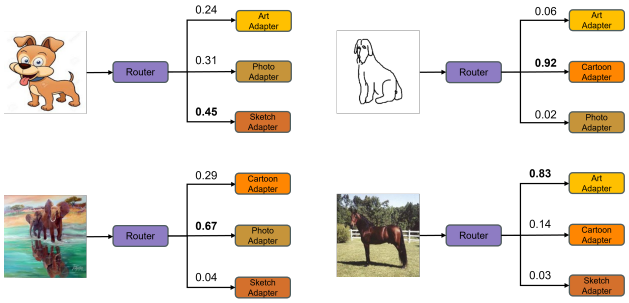

Figure 3. Predicted router weights for four PACS test samples across different domains. For instance, in the test image from the art domain (bottom left), routers tend to assign higher weights to the photo router due to the similarity to the art domain.

Test domain	Art Weight	Cartoon Weight	Photo Weight	Sketch Weight
Art	-	0.29	0.67	0.04
Cartoon	0.24	-	0.31	0.45
Photo	0.83	0.14	-	0.03
Sketch	0.06	0.92	0.02	-

Table 7. Further analysis of the domain router. We assess the router's capability in allocating weights to adapters for each test image by calculating the average of all router output weights during inference across the four test domains of the PACS dataset. For a sanity check, we reported the average router weights for the validation set to assess if the router accurately predicts weights for samples similar to the training distribution. With weights of 0.91, 0.91, 0.92, and 0.92 for the photo, art, cartoon, and sketch domains respectively, this demonstrates the router's effectiveness in allocating weights to adapters for iid and OoD samples.

in a tried-and-tested design manner. We train separate models with a single ViT and a single CNN adapter, evaluate them across training domains, and pair each domain with the adapter that shows better performance. This approach ensures effective domain-adapter matching, enhancing our method's accuracy and efficiency. The adapter paired with each domain for all benchmarks is listed in Table 8. The GPT guided adapter selection is listed in Table 12. Notably, domains with limited color information, like the sketch domain in the PACS dataset, tend to be paired with ViT adapters, while more colorful domains such as photo and art are matched with Conv adapters. In the TerraInc dataset, where all domains comprise photos captured under varying brightness levels, our customization strategy pairs each domain with a Conv adapter. We believe this decision is primarily driven by the Conv adapter's better ability to differentiate brightness variations compared to the ViT adapter.

Further analysis of the ensemble effect of ViT and Conv adapter Our empirical findings highlight the benefits of integrating diverse adapter types. Delving into each domain's performance, Table 9 reveals that ViT and Conv adapters

Dataset	Domain 1	Domain 2	Domain 3	Domain 4	Domain 5	Domain 6
PACS	photo	art	cartoon	sketch	-	-
adapter	Conv	Conv	ViT	ViT	-	-
VLCS	VOC	LABEL	CAL	SUN	-	-
adapter	Conv	Conv	Conv	ViT	-	-
OfficeHome	clip	art	real	product	-	-
adapter	ViT	ViT	ViT	ViT	-	-
Terra	L38	L43	L46	L100	-	-
adapter	Conv	Conv	Conv	Conv	-	-
DomainNet	clip	Info	paint	quick	real	sketch
adapter	ViT	Conv	Conv	ViT	Conv	ViT

Table 8. Domain adapter customization for five benchmarks. To customize the right ViT and Conv adapters for each domain, we train separate models with a single ViT and a single CNN adapter, evaluate them across training domains, and pair each domain with the adapter that shows better performance.

excel in distinct domains-Conv adapters in color-rich ones like photo and art, and ViT adapters in less colorful domains such as sketch. By synergistically combining these adapters, our customized strategy demonstrates better or comparable performance across all domains. This aligns with our hypothesis that a mix of Conv and ViT adapters leverages the strengths of both, validating our method's effectiveness.

Adapter Type	photo	art	cartoon	sketch	Ave
ViT adapter	99.1	91.5	82.6	80.9	88.5
Conv adapter	99.3	92.8	82.8	79.7	88.7
Customized adapter	99.3	93.5	83.2	81.5	89.4

Table 9. Further analysis of the ensemble impact of ViT and Conv adapter on each domain in PACS. For consistency in comparison, all the conducted experiments here employ three adapters alongside router losses.

5. Conclusion

In this paper, we introduced a Customized Domain Adapter network architecture framework for domain generalization. Our approach, leveraging the distinct strengths of ViT and CNN architectures, effectively addresses varied distribution shifts inherent in different domains. By employing a simple method for selecting domain-specific adapter architectures, our model demonstrated enhanced generalization capabilities across standard datasets. Our experimental results underscored the superiority of this approach over existing methods.

Limitations Our method for selecting domain adapters, while effective, may not generalize to all scenarios. However, we show that GPT performs well in domain adapter assignment, reducing the reliance on domain IDs. Moreover, the complexity of the model increases with the number of domains, potentially leading to scalability issues for datasets with a large number of diverse domains. Future work could explore more adaptive and scalable approaches to domain representation even without domain ID information.

Acknowledgments. This work was supported in part by NSF IIS2404180, Microsoft Accelerate Foundation Models Research Program, and Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. 2022-0-00871, Development of AI Autonomy and Knowledge Enhancement for AI Agent Collaboration) and (No. RS-2022-00187238, Development of Large Korean Language Model Technology for Efficient Pre-training).

References

[1] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. Adversarial invariant feature learning with accuracy constraint for domain generalization. *arXiv preprint arXiv:1904.12543*, 2019.

[2] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[3] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *ECCV*, 2018.

[4] Manh-Ha Bui et al. Exploiting domain-specific features to enhance domain generalization. *NeurIPS*, 2021.

[5] Fabio M Carlucci, Paolo Russo, Tatiana Tommasi, and Barbara Caputo. Agnostic domain generalization. *arXiv preprint arXiv:1808.01102*, 2018.

[6] Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *NeurIPS*, 2021.

[7] Junbum Cha, Kyungjae Lee, Sungrae Park, and Sanghyuk Chun. Domain generalization by mutual-information regularization with pre-trained models. In *ECCV*, 2022.

[8] Junhyeong Cho, Gilhyun Nam, Sungyeon Kim, Hunmin Yang, and Suha Kwak. Promptstyler: Prompt-driven style generation for source-free domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15702–15712, 2023.

[9] Yu Ding, Lei Wang, Bin Liang, Shuming Liang, Yang Wang, and Fang Chen. Domain generalization by learning and removing domain-specific features. *arXiv preprint arXiv:2212.07101*, 2022.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

[12] Songwei Ge, Haohan Wang, Amir Alavi, Eric Xing, and Ziv Bar-Joseph. Supervised adversarial alignment of single-cell rna-seq data. *Journal of Computational Biology*, 2021.

[13] Rui Gong, Wen Li, Yuhua Chen, and Luc Van Gool. Dlow: Domain flow for adaptation and generalization. In *CVPR*, 2019.

[14] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[15] Jiang Guo, Darsh J Shah, and Regina Barzilay. Multi-source domain adaptation with mixture of experts. *arXiv preprint arXiv:1809.02256*, 2018.

[16] Beining Han, Chongyi Zheng, Harris Chan, Keiran Paster, Michael R Zhang, and Jimmy Ba. Learning domain invariant representations in goal-conditioned block mdps. *arXiv preprint arXiv:2110.14248*, 2021.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[19] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[20] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. Fsdr: Frequency space domain randomization for domain generalization. In *ICCV*, 2021.

[21] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 124–140. Springer, 2020.

[22] Zeyi Huang, Haohan Wang, Dong Huang, Yong Jae Lee, and Eric P Xing. The two dimensions of worst-case training and their integrated effect for out-of-domain generalization. In *CVPR*, 2022.

[23] Zeyi Huang, Andy Zhou, Zijian Ling, Mu Cai, Haohan Wang, and Yong Jae Lee. A sentence speaks a thousand images: Domain generalization through distilling clip with language guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11685–11695, 2023.

[24] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, pages 709–727. Springer, 2022.

[25] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9619–9628, 2021.

[26] Kyungmoon Lee, Sungyeon Kim, and Suha Kwak. Cross-domain ensemble distillation for domain generalization. In *ECCV*, 2022.

[27] Bo Li, Yifei Shen, Jingkang Yang, Yezhen Wang, Jiawei Ren, Tong Che, Jun Zhang, and Ziwei Liu. Sparse mixture-of-experts are domain generalizable learners. *arXiv preprint arXiv:2206.04046*, 2022.

[28] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *ICCV*, 2017.

[29] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. Learning to generalize: Meta-learning for domain generalization. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

[30] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *CVPR*, 2018.

[31] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *ECCV*, 2018.

[32] Ziyue Li, Kan Ren, Xinyang Jiang, Bo Li, Haipeng Zhang, and Dongsheng Li. Domain generalization using pretrained models without fine-tuning. *arXiv preprint arXiv:2203.04600*, 2022.

[33] Ziyue Li, Kan Ren, Xinyang Jiang, Yifei Shen, Haipeng Zhang, and Dongsheng Li. Simple: Specialized model-sample matching for domain generalization. In *The Eleventh International Conference on Learning Representations*, 2023.

[34] Haoyang Liu, Maheep Chaudhary, and Haohan Wang. Towards trustworthy and aligned machine learning: A data-centric survey with causality perspectives. *arXiv preprint arXiv:2307.16851*, 2023.

[35] Xiaofeng Mao, Yufeng Chen, Xiaojun Jia, Rong Zhang, Hui Xue, and Zhao Li. Context-aware robust fine-tuning. *International Journal of Computer Vision*, 132(5):1685–1700, 2024.

[36] Rang Meng, Xianfeng Li, Weijie Chen, Shicai Yang, Jie Song, Xinchao Wang, Lei Zhang, Mingli Song, Di Xie, and Shiliang Pu. Attention diversification for domain generalization. In *ECCV*, 2022.

[37] Saeid Motiian, Marco Piccirilli, Donald A Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *ICCV*, 2017.

[38] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *ICML*, 2013.

[39] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *CVPR*, 2021.

[40] Muhammad Muzammal Naseer, Kanchana Ranasinghe, Salman H Khan, Munawar Hayat, Fahad Shahbaz Khan, and Ming-Hsuan Yang. Intriguing properties of vision transformers. *Advances in Neural Information Processing Systems*, 34: 23296–23308, 2021.

[41] A Tuan Nguyen, Toan Tran, Yarin Gal, and Atılım Güneş Baydin. Domain invariant representation learning with domain density transformations. *arXiv preprint arXiv:2102.05082*, 2021.

[42] Namuk Park and Songkuk Kim. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022.

[43] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1406–1415, 2019.

[44] Mohammad Mahfujur Rahman, Clinton Fookes, and Sridha Sridharan. Discriminative domain-invariant adversarial network for deep domain generalization. *arXiv preprint arXiv:2108.08995*, 2021.

[45] Alexandre Rame, Corentin Dancette, and Matthieu Cord. Fishr: Invariant gradient variances for out-of-distribution generalization. In *International Conference on Machine Learning*, pages 18347–18377. PMLR, 2022.

[46] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

[47] Mattia Segu, Alessio Tonioni, and Federico Tombari. Batch normalization embeddings for deep domain generalization. *Pattern Recognition*, 135:109115, 2023.

[48] Soroosh Shahtalebi, Jean-Christophe Gagnon-Audet, Touraj Laleh, Mojtaba Faramarzi, Kartik Ahuja, and Irina Rish. Sand-mask: An enhanced gradient masking strategy for the discovery of invariances in domain generalization. *arXiv preprint arXiv:2106.02266*, 2021.

[49] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[50] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[51] Yuge Shi, Jeffrey Seely, Philip HS Torr, N Siddharth, Awni Hannun, Nicolas Usunier, and Gabriel Synnaeve. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

[52] Baochen Sun and Kate Saenko. Deep coral: Correlation alignment for deep domain adaptation. In *ECCV*, 2016.

[53] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[54] Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

[55] Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, 2017.

[56] Haohan Wang, Aaksha Meghawat, Louis-Philippe Morency, and Eric P Xing. Select-additive learning: Improving generalization in multimodal sentiment analysis. 2017.

[57] Ruoyu Wang, Mingyang Yi, Zhitang Chen, and Shengyu Zhu. Out-of-distribution generalization with causal invariant transformations. In *CVPR*, 2022.

[58] Zhuo Wang, Zezheng Wang, Zitong Yu, Weihong Deng, Jiahong Li, Tingting Gao, and Zhongyuan Wang. Domain generalization via shuffled style assembly for face anti-spoofing. In *CVPR*, 2022.

[59] Qiyu Wu, Chongyang Tao, Tao Shen, Can Xu, Xiubo Geng, and Daxin Jiang. Pcl: Peer-contrastive learning with diverse augmentations for unsupervised sentence embeddings. *arXiv preprint arXiv:2201.12093*, 2022.

[60] Zhenyao Wu, Xinyi Wu, Xiaoping Zhang, Lili Ju, and Song Wang. Siamdoge: Domain generalizable semantic segmentation using siamese network. In *ECCV*, 2022.

[61] Minghao Xu, Jian Zhang, Bingbing Ni, Teng Li, Chengjie Wang, Qi Tian, and Wenjun Zhang. Adversarial domain adaptation with domain mixup. In *Proceedings of the AAAI conference on artificial intelligence*, pages 6502–6509, 2020.

[62] Xufeng Yao, Yang Bai, Xinyun Zhang, Yuechen Zhang, Qi Sun, Ran Chen, Ruiyu Li, and Bei Yu. Pcl: Proxy-based contrastive learning for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7097–7107, 2022.

[63] Nanyang Ye, Kaican Li, Lanqing Hong, Haoyue Bai, Yiting Chen, Fengwei Zhou, and Zhenguo Li. Ood-bench: Benchmarking and understanding out-of-distribution generalization datasets and algorithms. *arXiv preprint arXiv:2106.03721*, 2021.

[64] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *ICCV*, 2019.

[65] Jian Zhang, Lei Qi, Yinghuan Shi, and Yang Gao. Mvdg: A unified multi-view framework for domain generalization. In *ECCV*, 2022.

[66] Xin Zhang, Shixiang Shane Gu, Yutaka Matsuo, and Yusuke Iwasawa. Domain prompt learning for efficiently adapting clip to unseen domains. *arXiv e-prints*, pages arXiv–2111, 2021.

[67] Xin Zhang, Shixiang Shane Gu, Yutaka Matsuo, and Yusuke Iwasawa. Domain prompt learning for efficiently adapting clip to unseen domains. *Transactions of the Japanese Society for Artificial Intelligence*, 38(6):B–MC2_1, 2023.

[68] Shanshan Zhao, Mingming Gong, Tongliang Liu, Huan Fu, and Dacheng Tao. Domain generalization via entropy regularization. *NeurIPS*, 2020.

[69] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Deep domain-adversarial image generation for domain generalisation. In *AAAI*, 2020.

[70] Wei Zhu, Le Lu, Jing Xiao, Mei Han, Jiebo Luo, and Adam P Harrison. Localized adversarial domain generalization. In *CVPR*, 2022.