Dynamic Domain Generalization

Zhishu Sun[1] , **Zhifeng Shen**[1] , **Luojun Lin**[1,†] , **Yuanlong Yu**[1,†] , **Zhifeng Yang**[1]
Shicai Yang[2] and **Weijie Chen**[2]

[1]College of Computer and Data Science, Fuzhou University, Fuzhou, China
[2]Hikvision Research Institute, Hangzhou, China
linluojun2009@126.com, yu.yuanlong@fzu.edu.cn

Abstract

Domain generalization (DG) is a fundamental yet very challenging research topic in machine learning. The existing arts mainly focus on learning domain-invariant features with limited source domains in a static model. Unfortunately, there is a lack of training-free mechanism to adjust the model when generalized to the agnostic target domains. To tackle this problem, we develop a brand-new DG variant, namely *Dynamic Domain Generalization* (DDG), in which the model learns to twist the network parameters to adapt to the data from different domains. Specifically, we leverage a meta-adjuster to twist the network parameters based on the static model with respect to different data from different domains. In this way, the static model is optimized to learn domain-shared features, while the meta-adjuster is designed to learn domain-specific features. To enable this process, DomainMix is exploited to simulate data from diverse domains during teaching the meta-adjuster to adapt to the agnostic target domains. This learning mechanism urges the model to generalize to different agnostic target domains via adjusting the model without training. Extensive experiments demonstrate the effectiveness of our proposed method. Code is available: https://github.com/MetaVisionLab/DDG

1 Introduction

Deep neural networks have achieved great success on various vision tasks, such as image recognition and object detection [He *et al.*, 2016]. Most of deep models are learnt in a closed perception environment where training and testing data are supposed under *i.i.d* (independent and identical distribution) assumption. However, the reality is open, compound, uncontrolled and incompatible with *i.i.d* assumption. Such complex scenarios can make conventional models suffer tremendous performance degradation, due to the absence of generalization ability to handle domain shift, which undoubtedly will discount the reliability and security of intelligent systems, *e.g.*, autonomous driving system.

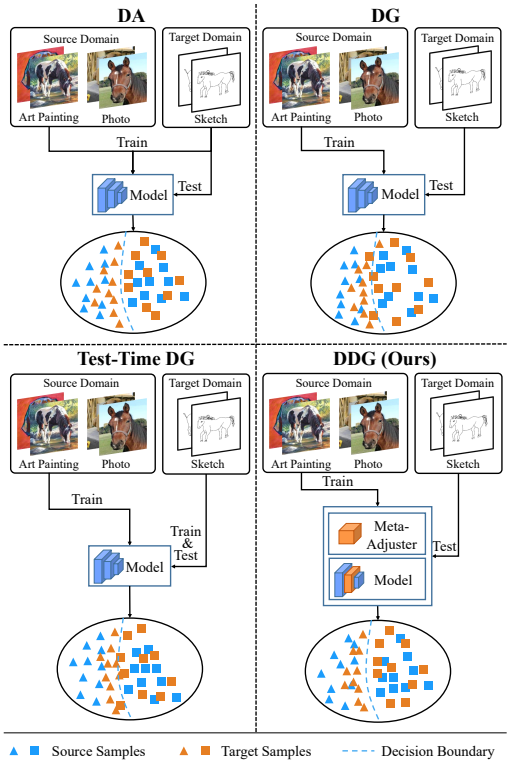


Figure 1: Comparison among DA, DG, Test-time DG and DDG methods. DA and test-time DG are able to adjust the model when encountering novel target domains via fine-tuning, while DG freezes a static model to adapt to any agnostic target domain. Different from the existing methods, DDG is able to adjust the model without any training effort for better domain generalization.

Extensive studies aim to tackle this problem through domain generalization (DG), whose objective is to obtain a robust static model by learning domain-invariant representations via exploiting multiple source domains [Ganin and Lempitsky, 2015; Zhou *et al.*, 2021]. Nevertheless, a severe bottleneck from the static model constrains its freedom to generalize to agnostic target domains with different distribution. In contrast, domain adaptation (DA) [Ganin *et al.*, 2016] and test-time DG [Iwasawa and Matsuo, 2021] are explored to leverage the unlabeled data sampled from the target domain for further unsupervised fine-tuning so as to enhance the target-oriented generalization ability. However, extra train-

†Corresponding authors: Luojun Lin and Yuanlong Yu

ing effort and latency are required during testing, while the manufactured chips and edge devices mostly do not possess training resources. Considering these two perspectives, there is a lack of training-free mechanism to adjust the model when encountering the testing data from novel target domains.

To complement this blank, we propose a new DG variant, *Dynamic Domain Generalization* (DDG). As shown in Figure 1, different from DA, DG, as well as test-time DG methods, the proposed DDG is attached with a meta-adjuster, which learns to adapt to novel domains during pre-training and twists the model during testing without any fine-tuning. To drive the process of learning to adapt, we need to collect training data from numerous novel domains, while the reality is that we are usually hindered by limited source domains. A simple yet friendly solution is that we simulate novel samples from the limited source domains via a random convex interpolation among different source domains, which is termed as DomainMix in this paper. In this way, each simulated sample is restricted to a unique domain. As a consequence, the domain-aware meta-adjuster can be relaxed into an instance-aware meta-adjuster, which modulates the network parameters per instance. Besides, it makes the model easier adapt from source domains to agnostic target domains with undefinable domain distinction, by discarding the utilization of domain label during optimizing meta-adjuster.

Beyond the optimization pipeline of DDG, it is also a critical factor on how to design the meta-adjuster. In order to disentangle the domain-shared features and domain-specific features, the network parameters are decoupled into a static part and a dynamic part where the latter one is generated by the meta-adjuster per instance. To ease the optimization of meta-adjuster, the dynamic parameters are decomposed into a linear combination of several elaborately-designed kernel templates with low-dimensional dynamic coefficients, in which the coefficients are generated dynamically by the meta-adjuster via taking the instance embedding as input. As shown in Figure 3, the kernel templates are designed as four asymmetric kernels for the sake of strengthening the kernel skeletons in spatial as well as channel dimensions, which explicitly regularize the meta-adjuster to generate diverse dynamic parameters to adapt to a large variety of novel samples.

Extensive experiments are conducted on three popular DG benchmarks, including PACS [Li *et al.*, 2017], Office-Home [Venkateswara *et al.*, 2017] and DomainNet [Peng *et al.*, 2019]. These datasets contain various situations varying from small to large scales, from simple to complex scenarios, from small to large domain shifts, which are sufficient to demonstrate the effectiveness of our proposed method. To summarize, DDG opens up a new paradigm to study DG problem.

2 Related Work

Domain Adaptation. Conventional DA aims to transfer the knowledge learnt from labeled source domains to target domains which may contain a few labeled samples or none at all. The discrepancy-based methods are early developed, which mitigate the distribution discrepancy between two domains by minimizing the well-defined distance loss [Haeusser *et al.*, 2017]. Another line of DA methods is based on adversarial learning, where a domain discriminator is frequently-used to train with the feature extractor adversarially, in order to enforce the feature extractor to learn domain-invariant features by confusing the domain discriminator [Ganin *et al.*, 2016]. Compared with DA methods that adjust models to adapt to target domains explicitly, our method can be viewed as a kind of training-free DA via pre-training merely in source domains.

Domain Generalization. Early DG-related research focus on learning domain-invariant representation, which is believed to be robust if it can suppress domain shift among multiple source domains [Motiian *et al.*, 2017]. Another line of DG methods is based on data augmentation that enhances the diversity of image styles for optimizing the generalization capability of models [Zhou *et al.*, 2020a; Zhou *et al.*, 2021]. Self-supervised learning is also applied to DG that learns task-independent representation with less probability of over-fitting [Carlucci *et al.*, 2019; Kim *et al.*, 2021]. Meta-learning is also involved with solving DG [Balaji *et al.*, 2018], where deep model is trained by simulating the meta-train and meta-test tasks in source domains to enhance its generalization ability. Our method can be regarded as a very simple meta-learning-like paradigm, in which meta-adjuster learns to learn parameter generalization for each individual "domain". Besides, a few emerging works show a new insight for DG that deep model can be adapted to target domain at test time [Iwasawa and Matsuo, 2021]. However, the computational cost is inescapable in such methods. In contrast, our method can directly update parameters to adapt to agnostic target domain without any backward propagation.

3 Methods

3.1 Overview

DG aims to train a general model that can be well generalized to any agnostic target domain. In particular, we only consider K-way classification task in this paper. For a vanilla DG task, we are given n_s labeled samples $\{x_s^i, y_s^i, d_s^i\}_{i=1}^{n_s}$ drawn from multi-source domains $\mathcal{X}_S = \{\mathcal{X}_{S_1}, \mathcal{X}_{S_2}, ..., \mathcal{X}_{S_M}\}$, where $x_s^i \in \mathcal{X}_S$, $y_s^i \in \mathcal{Y}_S$ and domain label $d_s^i \in [1, M]$. The objective of DG is to learn a mapping $f_\Theta : \{\mathcal{X}_{S_1}, ..., \mathcal{X}_{S_M}\} \rightarrow \mathcal{Y}_S$ that can also predict precisely on any agnostic target domain.

Vanilla Domain Generalization. For vanilla DG, the learnt mapping f_Θ obtained in training stage will be kept static in inference. It usually causes the performance degradation on agnostic target domains due to the inevitable data distribution distinction between the source and target domains.

Dynamic Domain Generalization. To overcome the weakness of vanilla DG, we propose a new variant of DG, namely *Dynamic Domain Generalization* (DDG). DDG mainly focuses on learning to twist the network parameters to adapt to the instances from diverse novel domains. In this paper, each instance is assumed to be an independent latent domain. Under this assumption, the model parameters Θ can be modeled as a function of input instance x: $\Theta = \Theta(x)$. To this end, a meta-adjuster is designed to generate network parameters to achieve diverse networks so as to adapt to diverse novel samples. In this case, the mapping $f_{\Theta(x)}$ is more likely

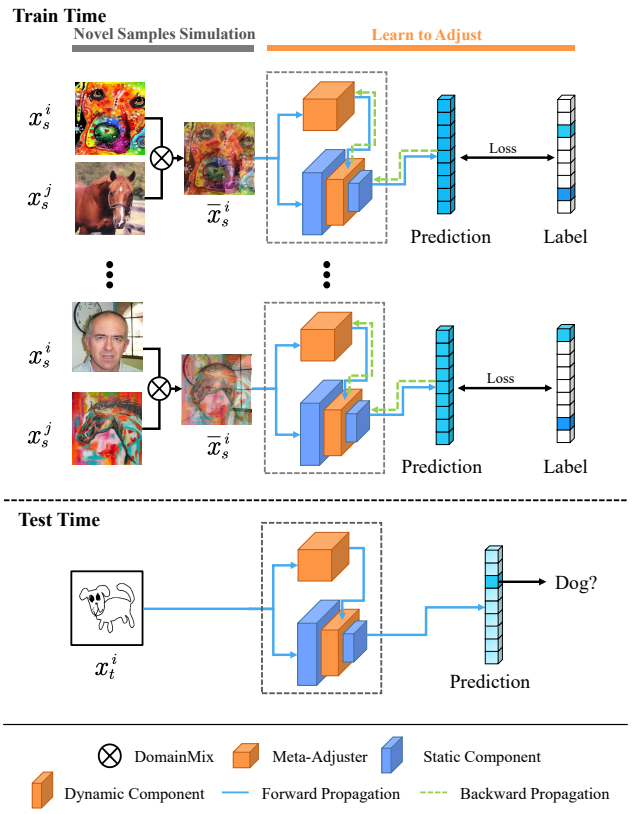


Figure 2: Pipeline of the proposed DDG method. In the training phase, two stages are required, namely novel samples simulation and learning to adjust, which aims to teach the meta-adjuster to adapt to novel samples. In the testing phase, the network can be adjusted without training when encountering agnostic target domain.

to align different instances into the same feature space under the reflection of instance-aware parameters $\Theta(x)$. Compared with the static mapping that is difficult to align instances from different data distributions, DDG is easier to achieve feature alignment among diverse novel domains.

The pipeline of our method is illustrated in Figure 2, where DDG framework is composed of a static component and a dynamic component updated by the meta-adjuster based on each input instance. The former one is responsible to learn domain-shared features while the latter one is designed to learn domain-specific features. In order to teach the meta-adjuster to adapt to diverse novel domains, we simulate numerous novel training samples via a random convex interpolation among the limited source domains.

3.2 Novel Samples Simulation

Numerous novel samples are required to drive the optimization of learning to adjust. By exploiting the limited source domains, we develop a DomainMix method from MixUp [Zhang *et al.*, 2018] to synthesize novel domains with diverse styles. Given two instances (x_s^i, y_s^i) and (x_s^j, y_s^j) randomly sampled from two different source domains, the process of novel samples simulation is formulated as:

$$\bar{x}_s^i = \alpha x_s^i + (1-\alpha)x_s^j, \qquad \bar{y}_s^i = \alpha y_s^i + (1-\alpha)y_s^j, \quad (1)$$

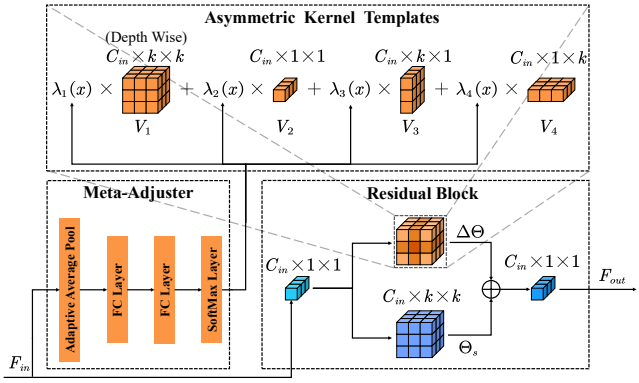


Figure 3: Structure illustration of the basic block for DDG, which contains a static and a dynamic component. The latter one is a learnable linear combination of four asymmetric kernel templates and corresponding dynamic coefficients which are generated by the meta-adjuster. For simplicity, C_{out} is omitted.

where α is randomly sampled from *Beta* distribution in each feed-forward. Each DomainMix sample $(\bar{x}_s^i, \bar{y}_s^i)$ can be viewed as an unique domain and be harnessed to drive the following process of learning to adjust.

3.3 Learn to Adjust

In this section, we decouple the network parameters into a static part and a dynamic part which implicitly disentangles the domain-shared and domain-specific features. Here the dynamic parameters are generated by a shallow generator, termed as meta-adjuster, by taking the instance embedding as input. For a better illustration, we take one of the most popular convolutional neural networks, namely ResNet [He *et al.*, 2016], as the backbone of our dynamic network. As shown in Figure 3, the meta-adjuster is embedded in each middle convolution layer in the residual block (note that each residual block is composed by a sequential of 1×1, 3×3 and 1×1 convolution layers). In the following, we will introduce how to design meta-adjuster effectively.

Meta-Adjuster. Meta-adjuster takes the instance embedding $F_{in}(x)$ (*i.e.*, the output of last block) as input, so that the meta-adjuster practically can be considered as the function of input instance $\mathbf{\Delta\Theta}(x)$. Therefore, the network parameters adjusting process can be formulated as:

$$\mathbf{\Theta}(x) = \mathbf{\Theta}_s + \mathbf{\Delta\Theta}(x), \quad (2)$$

where $\mathbf{\Theta}_s$ and $\mathbf{\Theta}(x)$ represent the static parameters and its corresponding up-to-date parameters, and $\mathbf{\Delta\Theta}(x)$ denotes their dynamic term. Theoretically, the dynamic term $\mathbf{\Delta\Theta}(x)$ should be the same dimension with the static parameters $\mathbf{\Theta}_s$. However, it makes the amount of network parameters grow quadratically and then apparently increases the risk of overfitting. Thus, according to matrix factorization, we propose to decompose the dynamic term into a learnable linear combination of several kernel templates $\{\boldsymbol{V}_n\}_{n=1}^N$ which can be formulated as:

$$\mathbf{\Delta\Theta}(x) = \sum_{n=1}^{N} \lambda_n(x)\boldsymbol{V}_n, \quad (3)$$

where N is a hyper-parameter that represents the number of kernel templates. $\{\lambda_n(x)\}_{n=1}^{N}$ denotes the set of dynamic coefficients, which is a group of instance-aware scalars generated by a learnable shallow meta-adjuster $\lambda(\cdot)$ via taking instance x (practically the instance embedding $F_{in}(x)$) as input. In this way, the dynamic component are restricted into a limited space, which benefits the optimization of whole model.

Asymmetric Kernel Templates. An apparent side effect to decompose meta-adjuster into a linear combination of kernel templates and a few dynamic coefficients is that the modulation brought by dynamic parameters are prone to be tiny and rigid. To avoid this tendency, we design four asymmetric kernel templates to explore the relationship between dynamic parameters from different dimensions, as well as to enforce the meta-adjuster to generate more diverse dynamic parameters for a large variety of novel samples. Another discovery is that the learned knowledge of convolutional kernels is not distributed uniformly, since the skeleton of kernels (*i.e.* the central criss-cross positions) usually has larger weight values [Ding *et al.*, 2019]. For this reason, we propose to enhance the skeleton of kernels by generating skeleton-based dynamic parameters that is derived from a linear combination of these asymmetric skeleton-based kernel templates.

Suppose that the dimension of static kernels $\mathbf{\Theta}_s$ is denoted as $c_{in} \times c_{out} \times k \times k$, where k, c_{in} and c_{out} represent the kernel size, the input channel number as well as the output channel number. As shown in Figure 3, the asymmetric skeleton-based kernel templates are designed as four matrices with different shapes as $c_{in} \times 1 \times k \times k$, $c_{in} \times c_{out} \times 1 \times 1$, $c_{in} \times c_{out} \times k \times 1$ and $c_{in} \times c_{out} \times 1 \times k$, respectively. Note that the kernel template with size of $c_{in} \times 1 \times k \times k$ are implemented as depth-wise convolution, because it merely focuses on spatial relationship. In this way, the dynamic parameters are encouraged to be varied for different input samples. In addition, the skeleton of network parameters can be amplified in spatial as well as channel dimensions.

4 Experiments

4.1 Datasets and Experimental Setup

Datasets. 1) **PACS** [Li *et al.*, 2017] contains more than $9k$ images with seven categories collected from four domains, which includes: *Photo*, *Sketch*, *Cartoon* and *Art Painting*. 2) **Office-Home** [Venkateswara *et al.*, 2017] contains $15,500$ images with 65 categories with a few images per category. All images are sampled from four domains, including: *Art*, *Clipart*, *Product* and *Real-World*. 3) **DomainNet** [Peng *et al.*, 2019] is a large-scale benchmark containing $600k$ images with 345 categories. The images are collected from six different domains which are *Clipart*, *Inforgraph*, *Painting*, *Quickdraw*, *Real* and *Sketch*.

Experimental Setup. For each dataset, we conduct leave-one-out strategy that randomly selects one domain for evaluation and leaves the remaining ones as source domains for training. Each experiment is repeated three times with different random seeds, and the mean and variance of these results are reported in this paper.

4.2 Implementation Details

All the experiments are conducted on RTX 3090 GPU with PyTorch 1.10.0. We use ResNet-50 as the backbone of our dynamic network, by attaching the meta-adjusters to the original residual blocks (as shown in Figure 3), where the spatial size of kernel templates is set as $k = 3$. Note that in each block, the meta-adjuster is implemented by a sequential of a global average pooling layer, a ReLU layer sandwiched between two fully-connected layers, and a SoftMax layer to output instance-aware coefficients.

Before training DG tasks, we pre-train our network on ImageNet so as to compare with the existing works fairly. Then, we train DG tasks by fine-tuning the pre-trained model with SGD. The training objective is a cross-entropy loss function, exactly the same with vanilla DG. The asymmetric kernel templates and meta-adjuster are optimized with the main backbone simultaneously during training. For PACS and Office-Home, the network optimization is set with batch size of 64, training epochs of 50, and the initial learning rate of 1e-3 decayed by cosine scheduler. While training on DomainNet, most of the hyper-parameters keep the same with that of PACS, except that the initial learning rate and max epoch are 2e-3 and 15, and the mini-batches are fetched with random domain sampler strategy [Zhou *et al.*, 2021], in order to ensure that each domain is uniformly sampled.

4.3 Comparisons with the State-of-the-Art

We compare our method with the state-of-the-art methods on PACS, Office-Home and DomainNet datasets, where the results are reported in Table 1, Table 2 and Table 3, respectively. From these tables, we can observe that:

1) Our method achieves an average accuracy of **86.21**, **72.13** and **46.89** percent on PACS, Office-Home and DomainNet benchmark datasets, respectively. Compared to other DG methods, our method achieves the state-of-the-art results on most of DG benchmarks, which shows the superiority of our DDG.

2) Our method is flexible to combine with other methods to further improve the performance. Taking Mixstyle [Zhou *et al.*, 2021] as an example, the performance of our model can be further improved if we insert MixStyle layers into our model. The results can reach **87.87**, **72.31** and **46.93** percent on PACS, Office-Home and DomainNet benchmarks, respectively. It shows that our method is rather robust even on the very strong baselines, i.e. MixStyle.

4.4 Ablation Studies

We conduct ablation studies on PACS dataset to evaluate the effectiveness of asymmetric kernel templates and DomainMix respectively, where the results are shown in Table 4.

Effectiveness of Asymmetric Kernel Templates. We study the effectiveness of asymmetric design of kernel templates, by employing common kernel templates that consists of four identical kernels with spatial dimensions of 1×1 or 3×3. Besides, we also compare with a channel-wise attention method, namely SENet [Hu *et al.*, 2018], which is also a popular and powerful architecture in CNN community. As shown in Table 4, we can see that:

Method	Art Painting	Cartoon	Photo	Sketch	Avg.
D-SAM [D'Innocente and Caputo, 2018]	77.33	72.43	95.30	77.83	80.72
CSD [Piratla *et al.*, 2020]	78.90±1.10	75.80±1.00	94.10±0.20	76.70±1.20	81.40
Epi-FCR [Li *et al.*, 2019]	82.10	77.00	93.90	73.00	81.50
MASF [Dou *et al.*, 2019]	82.89±0.16	80.49±0.21	95.01±0.10	72.29±0.15	82.67
DDAIG [Zhou *et al.*, 2020a]	84.20±0.30	78.10±0.60	95.30±0.40	74.70±0.80	83.10
MetaReg [Balaji *et al.*, 2018]	87.20±0.13	79.20±0.27	97.60±0.31	70.30±0.18	83.60
MixStyle [Zhou *et al.*, 2021]	84.10±0.40	78.80±0.40	96.10±0.30	75.90±0.90	83.70
ERM [Gulrajani and Lopez-Paz, 2021]	88.10±0.10	77.90±1.30	**97.80±0.00**	79.10±0.90	85.70
RSC [Huang *et al.*, 2020]	87.89	82.16	83.35	97.92	87.83
SWAD [Cha *et al.*, 2021]	**89.30±0.20**	**83.40±0.60**	97.30±0.30	82.50±0.50	**88.10**
Ours	85.92±0.17	79.68±0.42	96.65±0.15	82.62±0.27	86.21
Ours w/ MixStyle	87.14±0.21	82.66±0.33	97.77±0.06	**83.89±0.14**	87.87

Table 1: Results on PACS dataset. The best and second-best results are **bold** and underlined, respectively.

Method	Art	Clipart	Product	Real-World	Avg.
D-SAM [D'Innocente and Caputo, 2018]	58.03	44.37	69.22	71.45	60.77
JiGen [Carlucci *et al.*, 2019]	53.04	47.51	71.47	72.79	61.20
L2A-OT [Zhou *et al.*, 2020b]	60.60	50.10	74.80	77.00	62.60
RSC [Huang *et al.*, 2020]	58.42	47.90	71.63	74.54	63.12
DDAIG [Zhou *et al.*, 2020a]	59.20±0.10	52.30±0.30	74.60±0.30	76.00±0.10	65.50
MixStyle [Zhou *et al.*, 2021]	58.70±0.30	53.40±0.20	74.20±0.10	75.90±0.10	65.50
ERM [Gulrajani and Lopez-Paz, 2021]	62.70±1.10	53.40±0.60	76.50±0.40	77.30±0.30	67.50
SWAD [Cha *et al.*, 2021]	66.10±0.40	57.70±0.40	78.40±0.10	80.20±0.20	70.60
Ours	69.39±0.07	57.15±0.17	**80.20±0.19**	**81.79±0.07**	72.13
Ours w/ MixStyle	**69.43±0.19**	**59.40±0.36**	78.56±0.16	81.85±0.11	**72.31**

Table 2: Results on Office-Home dataset. The best and second-best results are **bold** and underlined, respectively.

Method	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	Avg.
C-DANN [Li *et al.*, 2018b]$^+$	54.60±0.40	17.30±0.10	43.70±0.90	12.10±0.70	56.20±0.40	45.90±0.50	38.30
RSC [Huang *et al.*, 2020]$^+$	55.00±1.20	18.30±0.50	44.40±0.60	12.20±0.20	55.70±0.70	47.80±0.90	38.90
Mixup [Zhang *et al.*, 2018]$^+$	55.70±0.30	18.50±0.50	44.30±0.50	12.50±0.40	55.80±0.30	48.20±0.50	39.20
SagNet [Nam *et al.*, 2021]$^+$	57.70±0.30	19.00±0.20	45.30±0.30	12.70±0.50	58.10±0.50	48.80±0.20	40.30
MLDG [Li *et al.*, 2018a]$^+$	59.10±0.20	19.10±0.30	45.80±0.70	13.40±0.30	59.60±0.20	50.20±0.40	41.20
ERM [Gulrajani and Lopez-Paz, 2021]$^+$	58.10±0.30	18.80±0.30	46.70±0.30	12.20±0.40	59.60±0.10	49.8±0.40	40.90
MetaReg [Balaji *et al.*, 2018]	59.77	**25.58**	50.19	11.52	64.56	50.09	43.62
DMG [Chattopadhyay *et al.*, 2020]	65.24	22.15	50.03	15.68	59.63	49.02	43.63
SelfReg [Kim *et al.*, 2021]	62.40±0.10	22.60±0.10	51.80±0.10	14.30±0.10	62.50±0.20	53.80±0.30	44.60
SWAD [Cha *et al.*, 2021]	66.00±0.10	22.40±0.30	53.50±0:10	16.10±0.20	65.80±0.40	**55.50±0.30**	46.50
Ours	**68.07±0.08**	25.02±0.18	53.54±0.37	13.74±0.10	**67.39±0.02**	53.59±0.06	46.89
Ours w/ MixStyle	67.39±0.05	24.21±0.08	**53.59±0.32**	**16.21±0.05**	65.85±0.12	54.34±0.27	**46.93**

Table 3: Results on DomainNet dataset. The best and second-best results are **bold** and underlined, respectively. $^+$: The results are re-implemented and reported by ERM [Gulrajani and Lopez-Paz, 2021]

1) Our method achieves superior performance to the vanilla ResNet and SENet by a large margin, which suggests the effectiveness of meta-adjusters on DG tasks.

2) The result of asymmetric kernel templates is obviously better than that of four identical 1×1 kernel templates and 3×3 kernel templates, which confirms the effectiveness of the asymmetric design of kernel templates. In-depth analysis of asymmetric kernel templates will be introduced in the visualization section.

Effectiveness of Novel Sample Simulation. In order to verify the effectiveness of the novel sample simulation method, we also conduct ablation studies on DomainMix. Instead of using DomainMix, we train the networks with limited source domains. Table 4 shows the comparison results, from which we can observe that our method suffers obvious performance degradation when training without DomainMix strategy. This result demonstrates the importance of novel sample simulation in teaching the meta-adjuster to adapt novel samples from agnostic target domains.

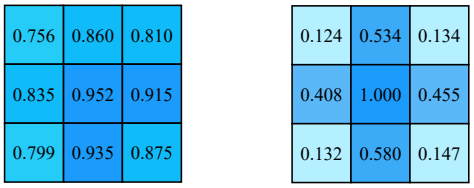
4.5 Visualization

For a better understanding of our method, we visualize the kernel templates and dynamic coefficients in different ways. Note that we use the model trained on the multi-source domains (i.e., Art Painting, Cartoon and Sketch) of PACS.

Visualization of Asymmetric Kernel Templates. In order to further explore the mechanism of asymmetric kernel templates, we have a comparison between static kernels in vanilla ResNet and dynamic kernels in our dynamic network, by the means of visualizing *kernel magnitude matrix* following the literature [Ding *et al.*, 2019]. For ResNet, we sum up the

Method	Art Painting	Cartoon	Photo	Sketch	Avg.	#Params
ResNet [He *et al.*, 2016]	84.99±0.25	77.64±0.42	97.54±0.19	71.63±0.41	82.95	22.43M
SENet [Hu *et al.*, 2018]	82.08±0.36	80.70±0.50	96.89±0.21	75.97±1.84	83.91	24.83M
Ours w/ 1× 1 kernel templates	81.95±0.96	78.20±1.86	97.50±0.31	78.45±0.34	84.03	28.17M
Ours w/ 3× 3 kernel templates	81.10±0.66	78.32±0.71	**97.64±0.27**	77.70±1.90	83.69	66.56M
Ours	**85.92±0.17**	79.68±0.42	96.65±0.15	**82.62±0.27**	**86.21**	31.81M
Ours w/o DominMix	81.18±0.33	**80.18±0.73**	97.35±0.10	78.78±1.11	84.37	31.81M

Table 4: Results of ablation studies in terms of asymmetric kernel templates and DomainMix on PACS dataset.



(a) Static kernel (b) Dynamic kernel

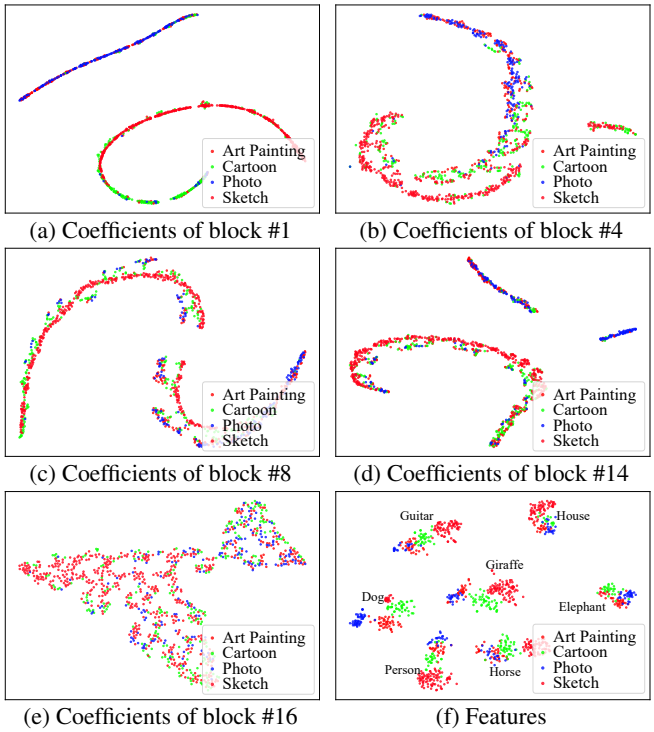
Figure 4: Comparison among static kernel of vanilla ResNet, and dynamic kernel of our network in terms of parameter magnitude. The kernels in sub-figures are acquired by averaging all the kernels, and being normalized in layer dimension.

3×3 kernels along the channel and layer dimensions to obtain a 3×3 kernel magnitude matrix. For dynamic network, we randomly select an image from validation set to feed into the model and aggregate all the kernel templates with corresponding dynamic coefficients, summing up along the channel and layer dimensions. The comparison result is shown in Figure 4, from which we can see that:

1) The central criss-cross positions have larger values than the corners of kernel, no matter whether static network or dynamic network. It confirms the kernel skeletons are more crucial than the corners when learning knowledge.

2) The kernel skeletons are further strengthened in dynamic network (as shown in Figure 4(b)). The reason is that the design of dynamic combination among kernel templates can adjust or enlarge the weights of kernel skeletons in instance-aware manner. Consequently, it can promote the model for learning advanced knowledge, which further improves training-free adaptation ability in inference.

t-SNE Visualization of Dynamic Coefficients and Features. In order to investigate the effect of dynamic network on different distribution data, we select a few blocks to visualize corresponding dynamic coefficients and the features. Figure 5(a)-(e) show the distributions of dynamic coefficients from different blocks of our model, by taking the whole dataset as input, including source (i.e., Art Painting, Cartoon, and Sketch) and target domains (i.e., Photo). We can observe that the dynamic coefficients tend to separate based on domains, and this phenomenon is decelerated from shallow to deep layers. It proves that the instance-aware coefficients are domain-aware practically, even without the explicit supervision of domain labels. Thus, the domain-aware coefficients can realize domain alignment in feature space, as shown in Figure 5(f). It stresses the superiority of dynamic coefficients on learning domain-invariant features for DG tasks.



(a) Coefficients of block #1 (b) Coefficients of block #4
(c) Coefficients of block #8 (d) Coefficients of block #14
(e) Coefficients of block #16 (f) Features

Figure 5: (a)-(e): t-SNE visualization of dynamic coefficients for samples from different domains, where we use colors to distinguish domains. Coefficients of Block #a denotes the dynamic coefficients from the a^{th} block. (f): t-SNE visualization of features (from the last block) for samples from different domains. Best viewed in colors.

5 Conclusion

In this paper, we develop a brand-new DG variant termed as dynamic domain generalization, which aims to explore a training-free mechanism to adjust the model to adapt to agnostic target domains. To this end, we decouple network parameters into static and dynamic parts to disentangle domain-shared and domain-specific features, where the latter ones are dynamically modulated by meta-adjusters with respect to varied novel samples from different domains. To enable this process, DomainMix and Asymmetric Kernel Templates are utilized to simulate novel samples and impose the meta-adjuster to modulate the models diversely for the samples from a large variety of novel domains. Extensive experiments demonstrate the superiority of our proposed DDG to vanilla DG approaches. Our work provides a strong baseline for DDG, and it shows possible to further improve DDG from an optimization perspective in the future.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No.: 83321016, 83418022, U21A20471); the Fujian Provincial Youth Education and Scientific Research Project (No. 650722).

References

[Balaji *et al.*, 2018] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. *NeurIPS*, 2018.

[Carlucci *et al.*, 2019] Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *CVPR*, 2019.

[Cha *et al.*, 2021] 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.

[Chattopadhyay *et al.*, 2020] Prithvijit Chattopadhyay, Yogesh Balaji, and Judy Hoffman. Learning to balance specificity and invariance for in and out of domain generalization. In *ECCV*, 2020.

[Ding *et al.*, 2019] Xiaohan Ding, Yuchen Guo, Guiguang Ding, and Jungong Han. Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. In *ICCV*, 2019.

[Dou *et al.*, 2019] Qi Dou, Daniel Coelho de Castro, Konstantinos Kamnitsas, and Ben Glocker. Domain generalization via model-agnostic learning of semantic features. *NeurIPS*, 2019.

[D'Innocente and Caputo, 2018] Antonio D'Innocente and Barbara Caputo. Domain generalization with domain-specific aggregation modules. In *GCPR*, 2018.

[Ganin and Lempitsky, 2015] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[Ganin *et al.*, 2016] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 2016.

[Gulrajani and Lopez-Paz, 2021] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *ICLR*, 2021.

[Haeusser *et al.*, 2017] Philip Haeusser, Thomas Frerix, Alexander Mordvintsev, and Daniel Cremers. Associative domain adaptation. In *ICCV*, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018.

[Huang *et al.*, 2020] Zeyi Huang, Haohan Wang, Eric P Xing, and Dong Huang. Self-challenging improves cross-domain generalization. In *ECCV*, 2020.

[Iwasawa and Matsuo, 2021] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *NeurIPS*, 2021.

[Kim *et al.*, 2021] Daehee Kim, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. Selfreg: Self-supervised contrastive regularization for domain generalization. In *ICCV*, 2021.

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

[Li *et al.*, 2018a] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Learning to generalize: Meta-learning for domain generalization. In *AAAI*, 2018.

[Li *et al.*, 2018b] 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.

[Li *et al.*, 2019] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *ICCV*, 2019.

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

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

[Peng *et al.*, 2019] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, 2019.

[Piratla *et al.*, 2020] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *ICML*, 2020.

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

[Zhang *et al.*, 2018] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

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

[Zhou *et al.*, 2020b] Kaiyang Zhou, Yongxin Yang, Timothy Hospedales, and Tao Xiang. Learning to generate novel domains for domain generalization. In *ECCV*, 2020.

[Zhou *et al.*, 2021] Kaiyang Zhou, Yongxin Yang, Yu Qiao, and Tao Xiang. Domain generalization with mixstyle. In *ICLR*, 2021.