MVDG: A Unified Multi-view Framework for Domain Generalization

Jian Zhang[1,2], Lei Qi[3(✉)], Yinghuan Shi[1,2(✉)], and Yang Gao[1,2]

[1] State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing, China
`syh@nju.edu.cn`
[2] National Institute of Healthcare Data Science, Nanjing University,
Nanjing, China
[3] School of Computer Science and Engineering, Southeast University,
Nanjing, China
`qilei@seu.edu.cn`

Abstract. To generalize the model trained in source domains to unseen target domains, domain generalization (DG) has recently attracted lots of attention. Since target domains can not be involved in training, overfitting source domains is inevitable. As a popular regularization technique, the meta-learning training scheme has shown its ability to resist overfitting. However, in the training stage, current meta-learning-based methods utilize only one task along a single optimization trajectory, which might produce a biased and noisy optimization direction. Beyond the training stage, overfitting could also cause unstable prediction in the test stage. In this paper, we propose a novel multi-view DG framework to effectively reduce the overfitting in both the training and test stage. Specifically, in the training stage, we develop a multi-view regularized meta-learning algorithm that employs multiple optimization trajectories to produce a suitable optimization direction for model updating. We also theoretically show that the generalization bound could be reduced by increasing the number of tasks in each trajectory. In the test stage, we utilize multiple augmented images to yield a multi-view prediction to alleviate unstable prediction, which significantly promotes model reliability. Extensive experiments on three benchmark datasets validate that our method can find a flat minimum to enhance generalization and outperform several state-of-the-art approaches. The code is available at https://github.com/koncle/MVRML.

1 Introduction

Traditional supervised learning assumes that training and test data are from the same distribution. However, this conventional assumption is not always
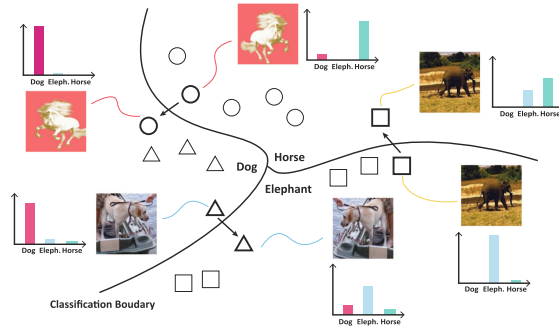

Fig. 1. The examples for the variation of the prediction probability when the test images are slightly perturbed in the unseen domain on the PACS dataset.

satisfied in the real world when domain shifts exist between training and test data. Recently, learning a robust and effective model against domain shift has raised considerable attention [3,20]. As one of the most representative learning paradigms under domain shift, unsupervised domain adaption (UDA) aims to tackle the adaptation from the labeled source domain to the unlabeled target domain under domain shift. Despite the great success of current UDA models [33,36,54], when deploying the previously trained UDA model to other unseen domains, we should re-train the model by incorporating the newly collected data from the unseen domain. This re-training process not only increases extra space/time costs but also violates privacy policy in some cases (e.g., clinical data), rendering these UDA methods not applicable to some real tasks.

The dilemma above motivates us to focus on a more applicable yet challenging setting, namely domain generalization (DG) [35]. In DG, by only learning the related knowledge from existing source domains, the trained model is required to be directly applied to previously unseen domains without any re-training procedure. To guarantee the model's efficacy on the unseen target domain, previous DG methods [25,35,41] intend to reduce the domain-specific influence from source domains via learning domain-invariant representations.

Well fitting source domains is easy, whereas well generalizing to an unseen target domain is hard. Previous methods inevitably suffer from overfitting when concentrating only on fitting source domains. Therefore, the meta-learning-based methods [10,24] have arisen as one of the most popular methods to resist overfitting during training, which simulates the domain shift episodically to perform regularization. However, these methods train their model with a single task at each iteration, which could cause a biased and noisy optimization direction.

Besides, after investigating the predictions of the trained model during the test stage, we notice that overfitting also results in unstable predictions. We conduct an experiment by perturbing (e.g., random crop and flip) the test images. As shown in Fig. 1, their predictions usually changed after being perturbed. It is because the feature representations of unseen images learned by the overfitted model are more likely to lie near the decision boundary. These representations

are easily perturbed across the boundary, thus, producing different predictions. This phenomenon is more challenging in DG due to the domain discrepancy.

As aforementioned, the overfitting problem not only appears in the training stage but also largely influences the following test procedure. To fight against overfitting, we innovatively propose a multi-view framework to deal with the inferior generalization ability and unstable prediction. Specifically, in the training stage, we design a multi-view regularized meta-learning algorithm that can regularize the training process in a simple yet effective way. This algorithm contains two steps. The first step is to guide a model to pursue a suitable optimization direction via exploiting multi-view information. Unlike most previous methods (e.g., MLDG [24]) that train the model using only one task along a single optimization trajectory (i.e., the training path from the current model to another model) with limited single-view information, we propose to train the model using multiple trajectories with different sampled tasks to find a more accurate direction with integrated multi-view information. In the second step, we update the model with the learned direction. We present a theoretical analysis that the number of tasks is also critical for generalization ability. Besides, we empirically verify that integrating multiple trajectory information can help our method find a flat minimum for promising generalization.

In the test stage, we propose to deal with the unstable prediction caused by the overfitted model using multi-view prediction. We argue that current test images with a single view cannot prevent unstable prediction. Nevertheless, different augmentations applied to the test image can bring abundant information from different views. Thus, if using the image pre-processing perturbations in the test procedure (e.g., the cropping operation), we can obtain multi-view information for a single image. Therefore, we augment each test image into multiple views during the test stage and ensemble their predictions as the final output. By exploiting the multi-view predictions of a single image, we can eliminate the unreliability of predictions and obtain a more robust and accurate prediction.

In summary, we propose a unified multi-view framework to enhance the generalization of the model and stabilize the prediction in both training and test stages. Our contributions can be summarized as follows:

– During training, we design an effective multi-view regularized meta-learning scheme to prevent overfitting and find a flat minimum that generalizes better.
– We theoretically prove that increasing the number of tasks in the training stage results in a smaller generalization gap and better generalizability.
– During the test stage, we introduce a multi-view prediction strategy to boost the reliability of predictions by exploiting multi-view information.
– Our method is validated via conducting extensive experiments on multiple DG benchmark datasets and outperforms other state-of-the-art methods.

2   Related Work

Domain Generalization. Domain generalization (DG) has been proposed recently to deal with learning the generalizability to the unseen target domain.

Current DG methods can be roughly classified into three categories: *Domain-invariant feature learning*, *Data augmentation* and *Regularization*.

Domain-invariant feature learning-based methods aim to extract domain-invariant features that are applicable to any other domains. One widely employed technique is adversarial training [28,41,56], which learns the domain-invariant features by reducing the domain gap between multiple source domains. Instead of directly learning the domain-invariant features, several methods [4,8,39] try to disentangle it from domain-specific features.

Data augmentation based methods try to reduce the distance between source and unseen domains via augmenting unseen data. Most of them perform image-level augmentation [26,52,53,57] that generates images with generative adversarial networks [14] or adaptive instance normalization [16]. Since feature statistics contains style information and easy to manipulate, other methods augment features by modifying its statistics [19,38,59] or injecting generated noise [27,43].

As overfitting hurts the generalization ability of a model, regularization-based methods prevent this dilemma by regularizing the model during training. Several works [6,48] add auxiliary self-supervised loss to perform regularization. [10, 24,55] adopt a meta-learning framework to regularize the model by simulating domain shift during training. Ensemble learning [7,58] also has been employed to regularize training models. Our method also belongs to this category which can better prevent overfitting by exploiting multi-view information.

Meta-Learning. Meta-learning [45] is a long-standing topic that learns how to learn. Recently, MAML [11] has been proposed as a simple model-agnostic method to learn the meta-knowledge, attracting lots of attention. Due to the unpredictable large computational cost of second-order derivatives, first-order methods are thus developed [11,37] to reduce the cost. Later, meta-learning is introduced into DG to learn generalizable representation across domains. These methods perform regularization under the meta-learning framework. For example, MLDG [24] utilizes the episodic training paradigm and updates the network with simulated meta-train and meta-test data. MetaReg [2] explicitly learns regularization weights, and Feature-critic [29] designs a feature-critic loss to ensure that the updated network should perform better than the original network. Recent DG methods [30,40] all adopt the episodic training scheme similar to MLDG due to its effectiveness. Although these methods can alleviate overfitting by training on a single task, they may produce biased optimization direction. However, our method can mitigate this problem by learning from multiple tasks and optimization trajectories.

Test Time Augmentation. The test time augmentation (TTA) is originally proposed in [21], which integrates the predictions of several augmented images for improving the robustness of the final prediction. Besides, several methods [34,44] try to learn automatic augmentation strategy for TTA. Other methods apply TTA to estimate uncertainty in the model [1,22]. However, according to our best knowledge, TTA has not been explored in DG, which can alleviate prediction uncertainty via generating multi-view augmented test images.

3 Our Method

3.1 Episodic Training Framework

Given the data space and label space as $\mathcal{X}$ and $\mathcal{Y}$, respectively, we denote N source domains as $\mathcal{D}_1, \ldots, \mathcal{D}_N$, where $\mathcal{D}_i = \{x_k, y_k\}_1^{N_i}$. N_i is the number of samples in the i-th source domain $\mathcal{D}_i$. We denote the model parameterized by θ as f. Given input x, the model outputs $f(x|\theta)$. As previously reviewed, meta-learning-based methods usually train the model with an episodic training paradigm. Similar to the meta-learning based methods [24] that split the source domains into meta-train and meta-test sets at each iteration, we leave one domain out as meta-test domain $\mathcal{D}_{te}$ and the remaining domains as meta-train domains $\mathcal{D}_{tr}$. Hereafter, we sample mini-batches from these domains and obtain meta-train $\mathcal{B}_{tr}$ and meta-test data $\mathcal{B}_{te}$. A *sub-DG task* is defined as a pair of them $t = (\mathcal{B}_{tr}, \mathcal{B}_{te})$. We define the loss on a batch $\mathcal{B} \in \{\mathcal{B}_{tr}, \mathcal{B}_{te}\}$ with parameter θ as:

$$\mathcal{L}(\mathcal{B}|\theta) = \sum_{(x_i, y_i) \in \mathcal{B}} \ell\big(f(x_i|\theta), y_i\big),$$

where ℓ is the traditional cross-entropy loss.

Different from previous meta-learning algorithms (e.g., MLDG [24]) that take second-order gradients to update model parameters, Reptile [37] is proposed as a first-order algorithm to reduce the computational cost. Therefore, we adopt Reptile version of MLDG. To be specific, given model parameter θ_j at the j-th iteration, we sample a task and train the model first with $\mathcal{L}(\mathcal{B}_{tr}|\theta_j)$ and then $\mathcal{L}(\mathcal{B}_{te}|\theta_j)$. Afterwards, we can obtain a temporarily updated parameter θ_{tmp} along this optimization trajectory. Finally, we take $(\theta_{tmp} - \theta_j)$ as an optimization direction, i.e., the direction points from the temporary parameter to the original parameter, to update the original parameter θ_j:

Fig. 2. An illustration of reptile and our multi-view regularized meta-learning algorithm (best viewed in color). (Color figure online)

$$\theta_{j+1} = \theta_j + \beta(\theta_{tmp} - \theta_j). \tag{1}$$

In this way, θ_{tmp} could exploit a part of current weight space and find a better optimization direction for current sampled tasks.
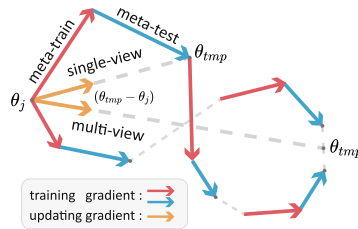
3.2   Multi-view Regularized Meta-Learning

Although Reptile reduces the com-
putational cost, the training scheme
above suffers from several problems.
Firstly, the model is trained along
a single optimization trajectory, pro-
ducing only one temporary parameter
θ_{tmp}. It is only a partial view of the
weight space for the current param-
eter θ_j, which is noisy and insuffi-
cient to produce a robust optimiza-
tion direction. To better understand
this problem, we plot the loss surface
of three temporarily trained parame-

Fig. 3. The loss surface [13] of three tem-
porarily trained parameters (denoted as
dot) starting from θ_j and optimized along
a different trajectory. The star denotes the
averaged parameter.

ters (denoted as a dot) in Fig. 3. Two of the three parameters have higher loss
values on both source and target domains. If we only consider one view (i.e.,
using a single temporary parameter), we may not find a good optimization
direction for updating. Recent researchs [7,12,18] find that the flat minimum
can produce better generalization performance than a sharp minimum, and sim-
ple weight averaging can achieve it if these weights share part of optimization
trajectory [12]. Inspired by this conclusion, we also plot the averaged parame-
ter (denoted as a star) and find that it can produce a more robust parameter
with lower loss than most parameters on both training and test domains. There-
fore, the ensembling of temporary models can produce a better and more stable
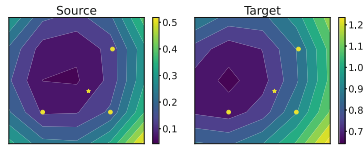optimization direction.

Secondly, since the model is trained with a single task in each trajectory,
it cannot explore the weight space fully and is hard to escape from the local
minimum, resulting in the overfitting problem. We have theoretically proved
that more sampled tasks can help minimize the domain generalization error and
produce better performance in Sect. 3.3.

Aiming to better explore the weight space to obtain a more accurate opti-
mization direction and erase the impact of overfitting, we develop a simple yet
effective multi-view regularized meta-learning (MVRML) algorithm by exploit-
ing multi-view information at each iteration. Specifically, to find a robust opti-
mization direction, we obtain T temporary parameters $\{\theta_{tmp}^1, ..., \theta_{tmp}^T\}$ along
different optimization trajectories. Different from MLDG, which only samples
a single task for the training stage, we train each temporary parameter with s
sampled tasks to help it escape from the local minimum. Besides, we sample
different tasks in different trajectories to encourage the diversity of temporary
models. Learning from these tasks allows to explore different information from
weight space with supplementary views. This sampling strategy plays an impor-
tant role in our method, and we will verify it in Sect. 4.3. Then we average their
weights to obtain a robust parameter: $\theta_{tmp} = \frac{1}{T}\sum_{t=1}^{T}\theta_{tmp}^t$. Since these models
share a part of the optimization trajectory, ensembling their weights can help

find parameters located in the flat minimum that generalizes better. The full algorithm is shown in Algorithm 1 and illustrated in Fig. 2.

Algorithm 1. Multi-view Regularized Meta-Learning

Input: source data $\mathcal{D}_{src}$, network parametrized by θ, hyperparameters: inner loop learning rate α, outer loop learning rate β, the number of optimization trajectories T and the number of sampled tasks s.

Output: the trained parameter

1: $\theta_0 \leftarrow \theta; j \leftarrow 0$ // `initialize parameter`
2: **while** not converged **do**
3:     $\theta_j^0 \leftarrow$ Initialized by θ_j
4:     **for** $t \in \{1, ..., T\}$ **do** // `train temporary models`
5:       **for** $i \in \{1, ..., s\}$ **do**
6:         $\mathcal{D}_{tr}, \mathcal{D}_{te} \leftarrow$ Random split $\mathcal{D}_{src}$ // $\mathcal{D}_{tr} \cup \mathcal{D}_{te} = \mathcal{D}_{src}, \ \mathcal{D}_{tr} \cap \mathcal{D}_{te} = \emptyset$
7:         Sample mini-batch $\mathcal{B}_{tr}, \mathcal{B}_{te}$ from $\mathcal{D}_{tr}, \mathcal{D}_{te}$
8:         $\theta_j^{i-1} \leftarrow \theta_j^{i-1} - \alpha \nabla_\theta (\mathcal{L}(\mathcal{B}_{\text{tr}} | \theta_j^{i-1}))$ // `train with meta-train data`
9:         $\theta_j^i \leftarrow \theta_j^{i-1} - \alpha \nabla_\theta (\mathcal{L}(\mathcal{B}_{\text{te}} | \theta_j^{i-1}))$ // `train with meta-test data`
10:       **end for**
11:       $\theta_{tmp}^t \leftarrow \theta_j^i$ // `assign current temporary parameter`
12:     **end for**
13:     $\theta_{j+1} \leftarrow \theta_j + \beta(\frac{1}{T} \sum_{t=1}^{T} \theta_{tmp}^t - \theta_j)$ // `update the original parameter`
14:     $j \leftarrow j + 1$
15: **end while**

Relation to Reptile. Our algorithm is similar to the batch version of Reptile [37], but there are three key differences. Firstly, the goal is different. Reptile aims to find a good weight initialization that can be fast adapted to new tasks, while since our classification task is fixed, we are more interested in the generalization problem without adaptation. Secondly, the task sampling strategy is different. Reptile only samples a single task for each trajectory, while we sample multiple different tasks for better generalization. Finally, the training scheme is different. In Reptile, since its goal is to adapt to the current classification task, it trains the model on this task iteratively. However, it is easy to overfit this single task in DG since the performance of the training model is already good in the source domains. Differently, we train different tasks in each trajectory to find a more generalizable parameter and prevent the overfitting problem.

Relation to Ensemble Learning. There are two ensemble steps in our method. First, at each iteration, we ensemble several temporary model parameters to find a robust optimization direction. Second, if we change the formulation of Eq. (1) as $\theta_{i+1} = (1 - \beta)\theta_i + \beta\theta_{\text{tmp}}$, we can obtain an ensemble learning algorithm that combines the weights of current model and temporary model. Therefore, this training paradigm implicitly ensembles models in the weight space and can lead to a more robust model [18].

3.3   Theoretical Insight

Traditional meta-learning methods train the model with only one task, which could suffer from the overfitting problem. We theoretically prove that increasing the number of tasks can improve the generalizability in DG. We denote the source domain as $\mathcal{S} = \{\mathcal{D}_1, ...\mathcal{D}_N\}$ and target domain as $\mathcal{T} = \mathcal{D}_{N+1}$. A task is defined as $t = (\mathcal{B}_{tr}, \mathcal{B}_{te})$, which is obtained by sampling from $\mathcal{S}$. At each iteration, a sequence of sampled tasks along a single trajectory is defined as $\mathbf{T} = \{t_0, \ldots, t_m\}$ with a size of m. The training set of task sequences is defined as $\mathbb{T} = \{\mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_n\}$ with a size of n. A training algorithm $\mathbb{A}$ trained with $\mathbb{T}$ or $\mathbf{T}$ is denoted as $\theta = \mathbb{A}(\mathbb{T})$ or $\theta = \mathbb{A}(\mathbf{T})$. We define the expected risk as $\mathcal{E}_\mathcal{P}(\theta) = \mathbb{E}_{(x_i,y_i)\sim\mathcal{P}}\ell(f(x_i|\theta), y_i)$. With a little abuse of notation, we define the loss with respect to $\mathbf{T}$ as:

$$\mathcal{L}(\mathbf{T}; \theta) = \frac{1}{m} \sum_{(\mathcal{B}_{tr}, \mathcal{B}_{te})\in\mathbf{T}} \frac{1}{2}(\mathcal{L}(\mathcal{B}_{tr}; \theta) + \mathcal{L}(\mathcal{B}_{te}; \theta)), \qquad (2)$$

and the loss with respect to $\mathbb{T}$ as $\mathcal{L}(\mathbb{T}; \theta) = \frac{1}{n}\sum_{\mathbf{T}\in\mathbb{T}}\mathcal{L}(\mathbf{T}; \theta)$.

Theorem 1. Assume that algorithm $\mathbb{A}$ satisfies β_1-uniform stability [5] with respect to $\mathcal{L}(\mathbb{T}; \mathbf{A}(\mathbb{T}))$ and β_2-uniform stability with respect to $\mathcal{L}(\mathbf{T}; \mathbf{A}(\mathbf{T}))$. The following bound holds with probability at least $1 - \delta$:

$$\mathcal{E}_\mathcal{T}(\theta) \le \hat{\mathcal{E}}_\mathcal{S}(\theta) + \frac{1}{2}\sup_{\mathcal{D}_i\in\mathcal{S}} \mathbf{Div}(\mathcal{D}_i, \mathcal{T}) + 2\beta_1 + (4n\beta_1 + M)\sqrt{\frac{\ln\frac{1}{\delta}}{2n}} + 2\beta_2, \quad (3)$$

where M is a bound of loss function ℓ and $\mathbf{Div}$ is KL divergence. β_1 and β_2 are functions of the number of task sequences n and the number of tasks m in each task samples. When $\beta_1 = o(1/n^a), a \ge 1/2$ and $\beta_2 = o(1/m^b), b \ge 0$, this bound becomes non-trivial. Proof is in **Supplementary Material**.

This bound contains three terms: (1) the empirical error estimated in the source domain; (2) the distance between the source and target domain; (3) the confidence bound related to the number of task sequences n and the number of tasks m in each sequence. Traditional meta-learning methods in DG train with a large number of task sequences (i.e., n). However, the number of tasks in the sequence is only one (i.e., m). *By increasing the number of sampled tasks in each sequence, we can obtain a lower error bound. In this case, we could expect a better generalization ability.*

3.4   Multi-view Prediction

As our model is trained in source domains, the feature representations of learned data are well clustered. However, when unseen images come, they are more likely to be near the decision boundary because of the overfitting and domain

discrepancy, leading to unstable feature representations. When we apply small perturbations to the test images, their feature representations will be pushed across the boundary, as shown in Fig. 1.

However, current test images only present a single view (i.e., the original image) with limited information. As a result, it cannot completely prevent the unstable prediction caused by overfitting. Besides, we argue that different views of a single image could bring in more information than a single view. Therefore, instead of only using a single view to conduct the test, we propose to perform multi-view prediction (MVP). By performing multi-view predictions, we can integrate complementary information from these views and obtain a more robust and reliable prediction. Assuming we have an image x to be tested, we can generate different views of this image with some weak stochastic transformations $T(\cdot)$. Then the image prediction p is obtained by:

$$p = \text{softmax}\big(\frac{1}{m} \sum_{i=1}^{m} f(T(x)|\theta)\big),$$

where m is the number of views for a test image. We only apply the weak transformations (e.g., random flip) for MVP because we find that the strong augmentations (e.g., the color jittering) make the augmented images drift off the manifold of the original images, resulting in unsatisfactory prediction accuracy, which will be shown in **Supplementary Material**.

Note that the improvement brought by MVP does not mean the learned model has poor generalization capability on the simple transformations since our method without MVP has superior performance compared to other methods, and MVP can also improve other SOTA models. Besides, most predictions can not be changed if a model is robust. We will verify these claims in Sect. 4.3.

4   Experiments

We describe the details of datasets and implementation details as follows:

Datasets. To evaluate the performance of our method, we consider three popularly used domain generalization datasets: PACS, VLCS and OfficeHome. PACS [23] contains 9,991 images with 7 classes and 4 domains, and there is a large distribution discrepancy across domains. VLCS [46] consists of 10,729 images, including 5 classes and 4 domains with a small domain gap. Office-Home [47] contains 15,500 images, covering 4 domains and 65 categories which are significantly larger than PACS and VLCS.

Implementation Details. We choose ResNet-18 and Resnet-50 [15] pretrained on ImageNet [9] as our backbone, the same as previous methods [10]. All images are resized to 224×224, and the batch size is set to 64. The data augmentation consists of random resize and crop with an interval of [0.8, 1], random horizontal flip, and random color jittering with a ratio of 0.4. The model is trained for 30 epochs. We use SGD as our outer loop optimizer and Adam as the inner loop

optimizer, both with a weight decay of 5×10^{-4}. Their initial learning rates are 0.05 and 0.001 for the first 24 epochs, respectively, and they are reduced to 5×10^{-3} and 1×10^{-4} for the last 6 epochs. The β_1 and β_2 are 0.9 and 0.999 for Adam optimizer, respectively. The number of optimization trajectories and sampled tasks are both 3. For multi-view prediction, we only apply weak augmentations, i.e., the random resized crop with an interval of $[0.8, 1]$ and random horizontal flip. The augmentation number t is set to 32. If not specially mentioned, we adopt this implementation as default.

We adopt the leave-one-out [23] experimental protocol that leaves one domain as an unseen domain and other domains as source domains. We conduct all experiments three times and average the results. Following the way in [23], we select the best model on the validation set of source domains. DeepAll indicates that the model is trained without any other domain generalization modules.

Table 1. Domain generalization accuracy (%) on PACS dataset with ResNet-18 (left) and ResNet-50 (right) backbone. The best performance is marked as **bold**.

Method	A	C	P	S	Avg.	Method	A	C	P	S	Avg.
DeepAll	78.40	75.76	**96.49**	66.21	79.21	DeepAll	86.72	76.25	**98.22**	76.31	84.37
MLDG [24]	79.50	77.30	94.30	71.50	80.70	MASF [10]	82.89	80.49	95.01	72.29	82.67
MASF [10]	80.29	77.17	94.99	71.69	81.04	MetaReg [2]	82.57	79.20	97.60	70.30	83.58
MetaReg [2]	83.70	77.20	95.50	70.30	81.70	MatchDG [31]	85.61	82.12	78.76	97.94	86.11
VDN [50]	82.60	78.50	94.00	82.70	84.50	DSON [42]	87.04	80.62	95.99	82.90	86.64
FACT [51]	85.37	78.38	95.15	79.15	84.51	RSC [17]	87.89	82.16	97.92	83.35	87.83
RSC [17]	83.43	80.31	95.99	80.85	85.15	FSDCL [19]	88.48	83.83	96.59	82.92	87.96
FSDCL [19]	85.30	**81.31**	95.63	81.19	85.86	SWAD [7]	89.30	83.40	97.30	82.50	88.10
MVDG (Ours)	**85.62**	79.98	95.54	**85.08**	**86.56**	MVDG (Ours)	**89.31**	**84.22**	97.43	**86.36**	**89.33**

4.1   Comparison with State-of-the-Art Methods

We evaluate our method (namely **MVDG**) to different kinds of recent state-of-the-art DG methods on several benchmarks to demonstrate its effectiveness.

Table 2. Domain generalization accuracy (%) on VLCS and OfficeHome datasets. The best performance is marked as **bold**.

Method	C	L	P	S	Avg.	Method	A	C	P	R	Avg.
DeepAll	96.98	62.00	73.83	68.66	75.37	DeepAll	58.65	50.35	73.74	75.67	64.60
JiGen [6]	96.17	62.06	70.93	71.40	75.14	RSC [17]	58.42	47.90	71.63	74.54	63.12
MMLD [32]	97.01	62.20	73.01	**72.49**	76.18	DAEL [58]	59.40	55.10	74.00	75.70	66.10
RSC [17]	96.21	62.51	73.81	72.10	76.16	FSDCL [19]	60.24	53.54	74.36	76.66	66.20
						FACT [51]	**60.34**	**54.85**	74.48	76.55	66.56
MVDG (Ours)	**98.40**	**63.79**	**75.26**	71.05	**77.13**	MVDG (Ours)	60.25	54.32	**75.11**	**77.52**	**66.80**

PACS. We perform evaluation on PACS with ResNet-18 and ResNet-50 as our backbone. We compare with several meta-learning based methods (i.e., MLDG [24], MASF [10], MetaReg [2]), augmentation based methods (i.e., FACT [51], FSDCL [19]), ensemble learning based methods (i.e., SWAD [7]), domain-invariant feature learning (i.e., VDN [50]) and causal reasoning (i.e., MatchDG [31]). As shown in Table 1, our method can surpass traditional meta-learning methods in a large margin by 4.86% (86.56% vs. 81.70%) on ResNet-18 and 6.66% (89.33% vs. 82.67%) on ResNet-50. Besides, our method also achieves SOTA performance compared to recent other methods. Note that the improvement of our method on the hardest "sketch" domain is significant compared to the DeepAll (i.e., 85.08% vs. 66.21%), owing to its better regularization and robustness.

VLCS. To verify the trained model can also generalize to unseen domains with a small domain gap, we conduct an experiment on VLCS. As seen in Table 2, our method outperforms several SOTA methods and achieves the best performance on three domains (CALTECH, LABELME, and PASCAL), demonstrating that our method can also perform well in this case.

OfficeHome. We compare our method with SOTA methods on OfficeHome to prove the adaptation of our method to the dataset with a large number of classes. The result is reported in Table 2. Our method is able to achieve comparable performance to current SOTA methods.

4.2 Ablation Study

To further investigate our method, we conduct an ablation study on MVDG: i.e., 1) Reptile version of MLDG, 2) multi-view regularized meta-learning (MVRML), 3) multi-view prediction (MVP).

Table 3. The accuracy (%) of the ablation study on PACS dataset on each component: DeepAll, Reptile, and multi-view regularized meta-learning (MVRML), multi-view prediction (MVP).

Method	A	C	P	S	Avg.	Method	A	C	P	S	Avg.
Resnet-18	78.40	75.76	96.49	66.21	79.21±0.49	Resnet-50	86.72	76.25	98.22	76.31	84.37±0.42
+MVRML	85.20	79.97	95.29	83.11	85.89±0.27	+MVRML	89.02	84.47	97.09	85.24	88.95±0.06
+MVRML+MVP	85.62	79.98	95.54	85.08	86.56±0.18	+MVRML+MVP	89.31	84.22	97.43	86.36	89.33±0.18
+Reptile	80.49	76.23	94.91	77.65	82.34±0.71	+Reptile	85.48	78.58	97.07	77.31	84.61±0.15

Reptile. As seen in Table 3, the performance of Reptile can achieve satisfactory performance compared to DeepAll. Note that, although its performance in the "sketch" domain improves a lot (i.e., 77.65% vs. 66.21%), the performance in the "photo" domain decreases. We hypothesize that the feature spaces learned by meta-learning and DeepAll are different. Since ResNet-18 is pretrained on

ImageNet (photo-like dataset), it shows high performance in the "photo" domain at the beginning. When the training procedure continues, the model is hard to move far away from its initial weight space. Thus its performance is promising in the "photo" domain. However, when trained with meta-learning, it can obtain a good performance by the episodic training scheme but with a little sacrifice of its original performance in the "photo" domain.

Multi-view Regularized Meta-Learning. When we apply multi-view regularized meta-learning (MVRML), the performance is improved a lot on the baseline in Table 3, which shows its efficacy in dealing with the overfitting issue. We observe that the "photo" domain also decreases a little. It may be caused by the better weight space produced by the meta-learning algorithm, which is far away from the initial weight space (i.e., the initial model trained by ImageNet).

Multi-view Prediction. We employ multi-view prediction (MVP) to enhance model reliability. As shown in Table 3, the performance improves on both baselines. We notice that there is a large improvement in the "sketch" domain because the "sketch" domain has only the outline of the object, and thus it is more sensitive to small perturbations. With MVP, the model's prediction can be more reliable and accurate.

4.3   Further Analysis

We further analyze the properties of our method with ResNet-18 as backbone. More experiments could be found in **Supplementary Material**.

Local Sharpness Comparison. As mentioned in Sect. 3.2, our method can achieve better performance because it can find a flat minimum. To verify it, we plot local flatness via the loss gap between the original parameter and perturbed parameter, i.e., $\mathbb{E}_{\theta'=\theta+\epsilon}[\mathcal{L}(\theta, \mathcal{D}) - \mathcal{L}(\theta', \mathcal{D})]$ [7], where ϵ is perturbation sampled from gaussian distribution with a radius of γ. We conduct an experiment on the target domain of PACS, and the results are shown in Fig. 4. With a larger radius, the sharpness of the three methods all increases. However, MVRML can find a better flat minimum where the loss increases the most slowly compared to the other methods, achieving better generalization performance.
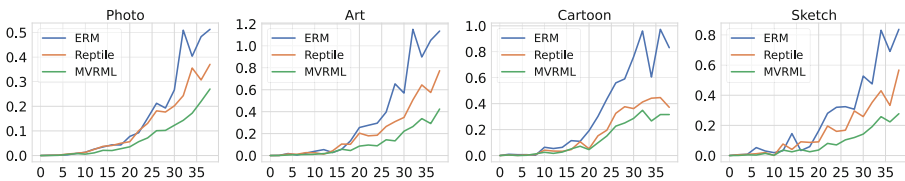

Fig. 4. Local sharpness comparison between ERM, Reptile, and MVRML on target domain of PACS. X-axis indicates the distance γ to the original parameter, and Y-axis indicates the sharpness of the loss surface (the lower, the flatter).

Table 4. The comparison of different task sampling strategies for MVRML on PACS dataset.

Strategies	A	C	P	S	Avg.
S1	82.81	76.85	94.65	82.78	84.27
S2	84.11	78.48	93.75	82.95	84.82
S3 (Ours)	85.20	79.97	95.29	83.11	85.89

Influence of Task Sampling Strategies. As mentioned in Sect. 3.2, task sampling strategy is crucial for performance improvement in MVRML. We compare different sampling strategies at each iteration: each trajectory samples 1) from the same domain, denoted as S1; 2) from all domains, denoted as S2; 3) from random split meta-train and meta-test domains, denoted as S3. As shown in Table 4, with a better sampling strategy, the generalizability of the trained model increases. We hypothesize that it is owing to the batch normalization layer that normalizes data with statistics calculated on a batch. When only sampling from a single domain, the diversity of BN statistics is limited to three domains. If we sample batches from different domains, although the diversity of BN statistics increases a little, the statistics in a batch tend to be the same on average. Finally, if we sample tasks from meta-train and meta-test splits, we can ensure that the diversity of statistics changes drastically, encouraging temporary models to explore diverse trajectories and produce a more robust optimization direction.

Impact of the Number of Tasks and Trajectories. For MVRML, we sample a sequence of tasks on multiple trajectories to train the model. Both the number of tasks and trajectories can affect the generalization performance. Therefore, we train our model with a different number of tasks and trajectories. When we experiment with one factor, we set the other to 3. The result can be shown in Fig. 5a and Fig. 5b. With the increasing number of tasks, the performance first improves, as our theory suggests that more tasks can benefit the generalizability. However, the performance plateaus afterward. We suspect that the first few tasks are critical for the optimization procedure of temporary models since it decides the direction to optimize. With more tasks to be learned, the optimization direction does not change too much. Thus, more tasks could not largely improve performance. There is a similar trend in the number of trajectories. Because three trajectories are good enough to provide a robust optimization direction, more trajectories also could not help too much. To reduce the computational cost, we choose three trajectories in our experiments.

Influence of the Number of Augmented Images in MVP. When applying multi-view prediction (MVP), we need to augment the test images into different views and ensemble their results. Therefore, the number of augmented images has an influence on the result. We apply MVP to both DeepAll and our model with a different number of augmented images. As shown in Fig. 5c, the performance improves when more augmented images are generated, which results from the increasing diversity of the test images. When the number is large enough (e.g.,

64), the diversity created by the weak augmentation cannot increase anymore, and the performance plateaus.
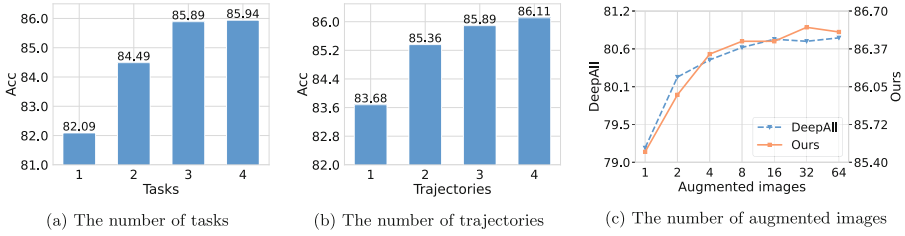

(a) The number of tasks   (b) The number of trajectories   (c) The number of augmented images

Fig. 5. The impact of the number of tasks (a) and trajectories (b) in MVRML. The number of augmented images (c) in MVP.

Table 5. The left table shows accuracy and prediction change rate (PCR) of different methods on PACS dataset with ResNet-18. The weak augmentation is denoted as WA. The right table shows the accuracy (%) of applying MVP to other SOTA methods on PACS dataset.

	Method	A	C	P	S	Avg.
Acc(%)	DeepAll (w/o WA)	77.90	70.60	96.41	66.03	77.73
	DeepAll	78.40	75.76	96.49	66.21	79.21
	MVRML	85.20	79.97	95.29	83.11	85.89
PCR(%)	DeepAll (w/o WA)	43.0	42.7	9.9	63.6	39.8
	DeepAll	36.5	37.6	8.1	49.3	32.9
	MVRML	27.7	28.4	8.7	25.4	22.6

Method	A	C	P	S	Avg.
MixStyle [59]	80.43	77.55	97.11	76.15	82.81
MixStyle+MVP	83.05	78.45	97.50	78.99	84.49
RSC [17]	84.67	76.75	95.81	82.59	84.96
RSC+MVP	84.91	76.83	96.41	83.81	85.49
FSR [49]	84.57	80.76	96.11	82.16	85.90
FSR+MVP	85.60	81.40	96.59	83.48	86.76

Unstable Prediction. In the previous sections, we argue that if the model overfits the source domains, it is easy to produce unstable predictions by perturbing the test images slightly (random resized crop and flip). By contrast, a robust model can reduce this effect and perform well. To verify it, we test several models in the unseen domain: DeepAll without weak augmentations (i.e., color jittering, random crop, and flip), DeepAll, and the model trained with MVRML. We introduce prediction change rate (PCR), calculated by the ratio of the number of predictions changed after applying the augmentations and the number of total predictions. We compare the test accuracy and PCR in Table 5. The larger this measure, the more unstable the model in the unseen domains. As seen, DeepAll without augmentation produces the highest PCR and lowest Acc because this model overfits source domains. Meanwhile, with data augmentation and a better training strategy, the performance of the model largely improves, and PCR decreases drastically.

MVP on SOTA Methods. MVP is a plug-and-play method that can be easily adapted to other methods. To validate its adaptation ability, we integrate it into three SOTA methods, i.e., Mixstyle [59], RSC [17] and FSR [49]. For RSC and FSR, we directly use their pre-trained model. For MixStyle, we implement it by ourselves. The result is shown in Table 5. MVP can improve all of these trained models, which suggests its effectiveness in DG.

5 Conclusion

In this paper, to resist overfitting, with the observation that the performance in DG models can be benefited by task-based augmentation in training and sample-based augmentation in testing, we propose a novel multi-view framework to boost generalization ability and reduce unstable prediction caused by overfitting. Specifically, during training, we designed a multi-view regularized meta-learning algorithm. During testing, we introduced multi-view prediction to generate different views of a single image for the ensemble to stabilize its prediction. We provide theoretical proof that increasing the number of tasks can boost generalizability, and we also empirically verified that our method can help find a flat minimum that generalizes better. By conducting extensive experiments on three DG benchmark datasets, we validated the effectiveness of our method.

References

1. Ayhan, M.S., Berens, P.: Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. In: MIDL (2018)
2. Balaji, Y., Sankaranarayanan, S., Chellappa, R.: MetaReg: towards domain generalization using meta-regularization. In: NeurIPS (2018)
3. Ben-David, S., Blitzer, J., Crammer, K., Pereira, F., et al.: Analysis of representations for domain adaptation. In: NeurIPS (2007)
4. Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., Erhan, D.: Domain separation networks. In: NeurIPS (2016)
5. Bousquet, O., Elisseeff, A.: Stability and generalization. In: JMLR (2002)
6. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: CVPR (2019)
7. Cha, J., et al.: SWAD: domain generalization by seeking flat minima. arXiv (2021)
8. Chattopadhyay, P., Balaji, Y., Hoffman, J.: Learning to balance specificity and invariance for in and out of domain generalization. In: ECCV (2020)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
10. Dou, Q., de Castro, D.C., Kamnitsas, K., Glocker, B.: Domain generalization via model-agnostic learning of semantic features. In: NeurIPS (2019)

11. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: ICML (2017)
12. Frankle, J., Dziugaite, G.K., Roy, D., Carbin, M.: Linear mode connectivity and the lottery ticket hypothesis. In: ICML (2020)
13. Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D.P., Wilson, A.G.: Loss surfaces, mode connectivity, and fast ensembling of dnns. In: NeurIPS (2018)
14. Goodfellow, I., et al.: Generative adversarial nets. In: NeurIPS (2014)
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
16. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. In: ICCV (2017)
17. Huang, Z., Wang, H., Xing, E.P., Huang, D.: Self-challenging improves cross-domain generalization. In: ECCV (2020)
18. Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., Wilson, A.G.: Averaging weights leads to wider optima and better generalization. arXiv (2018)
19. Jeon, S., Hong, K., Lee, P., Lee, J., Byun, H.: Feature stylization and domain-aware contrastive learning for domain generalization. In: ACMMM (2021)
20. Kouw, W.M., Loog, M.: A review of domain adaptation without target labels. In: TPAMI (2019)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NeurIPS (2012)
22. Lee, H., Lee, H., Hong, H., Kim, J.: Test-time mixup augmentation for uncertainty estimation in skin lesion diagnosis. In: MIDL (2021)
23. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: ICCV (2017)
24. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Learning to generalize: meta-learning for domain generalization. In: AAAI (2018)
25. Li, H., Jialin Pan, S., Wang, S., Kot, A.C.: Domain generalization with adversarial feature learning. In: CVPR (2018)
26. Li, L., et al.: Progressive domain expansion network for single domain generalization. In: CVPR (2021)
27. Li, P., Li, D., Li, W., Gong, S., Fu, Y., Hospedales, T.M.: A simple feature augmentation for domain generalization. In: ICCV (2021)
28. Li, Y., et al.: Deep domain generalization via conditional invariant adversarial networks. In: Ferrari, Vittorio, Hebert, Martial, Sminchisescu, Cristian, Weiss, Yair (eds.) ECCV 2018. LNCS, vol. 11219, pp. 647–663. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01267-0_38
29. Li, Y., Yang, Y., Zhou, W., Hospedales, T.M.: Feature-critic networks for heterogeneous domain generalizationx. arXiv (2019)
30. Liu, Q., Dou, Q., Heng, P.A.: Shape-aware meta-learning for generalizing prostate MRI segmentation to unseen domains. In: MICCAI (2020)
31. Mahajan, D., Tople, S., Sharma, A.: Domain generalization using causal matching. In: ICML (2021)
32. Matsuura, T., Harada, T.: Domain generalization using a mixture of multiple latent domains. In: AAAI (2020)
33. Melas-Kyriazi, L., Manrai, A.K.: PixMatch: unsupervised domain adaptation via pixelwise consistency training. In: CVPR (2021)
34. Molchanov, D., Lyzhov, A., Molchanova, Y., Ashukha, A., Vetrov, D.: Greedy policy search: a simple baseline for learnable test-time augmentation. arXiv (2020)
35. Muandet, K., Balduzzi, D., Schölkopf, B.: Domain generalization via invariant feature representation. In: ICML (2013)

36. Na, J., Jung, H., Chang, H.J., Hwang, W.: FixBi: bridging domain spaces for unsupervised domain adaptation. In: CVPR (2021)
37. Nichol, A., Schulman, J.: Reptile: a scalable metalearning algorithm. arXiv (2018)
38. Nuriel, O., Benaim, S., Wolf, L.: Permuted adain: Enhancing the representation of local cues in image classifiers. arXiv (2020)
39. Piratla, V., Netrapalli, P., Sarawagi, S.: Efficient domain generalization via common-specific low-rank decomposition. In: ICML (2020)
40. Qiao, F., Zhao, L., Peng, X.: Learning to learn single domain generalization. In: CVPR (2020)
41. Rahman, M.M., Fookes, C., Baktashmotlagh, M., Sridharan, S.: Correlation-aware adversarial domain adaptation and generalization. In: PR (2019)
42. Seo, S., Suh, Y., Kim, D., Han, J., Han, B.: Learning to optimize domain specific normalization for domain generalization. In: ECCV (2020)
43. Shu, M., Wu, Z., Goldblum, M., Goldstein, T.: Prepare for the worst: generalizing across domain shifts with adversarial batch normalization. arXiv (2020)
44. Sypetkowski, M., Jasiulewicz, J., Wojna, Z.: Augmentation inside the network. arXiv (2020)
45. Thrun, S., Pratt, L. (eds.): Larning to Learn(1998)
46. Torralba, A., Efros, A.A.: Unbiased look at dataset bias. In: CVPR (2011)
47. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: CVPR (2017)
48. Wang, S., Yu, L., Li, C., Fu, C.W., Heng, P.A.: Learning from extrinsic and intrinsic supervisions for domain generalization. In: ECCV (2020)
49. Wang, Y., Qi, L., Shi, Y., Gao, Y.: Feature-based style randomization for domain generalization. arXiv (2021)
50. Wang, Y., Li, H., Chau, L.P., Kot, A.C.: Variational disentanglement for domain generalization. arXiv (2021)
51. Xu, Q., Zhang, R., Zhang, Y., Wang, Y., Tian, Q.: A fourier-based framework for domain generalization. In: CVPR (2021)
52. Xu, Z., Liu, D., Yang, J., Raffel, C., Niethammer, M.: Robust and generalizable visual representation learning via random convolutions. arXiv (2020)
53. Yue, X., Zhang, Y., Zhao, S., Sangiovanni-Vincentelli, A., Keutzer, K., Gong, B.: Domain randomization and pyramid consistency: simulation-to-real generalization without accessing target domain data. In: ICCV (2019)
54. Yue, Z., Sun, Q., Hua, X.S., Zhang, H.: Transporting causal mechanisms for unsupervised domain adaptation. In: ICCV (2021)
55. Zhang, J., Qi, L., Shi, Y., Gao, Y.: Generalizable model-agnostic semantic segmentation via target-specific normalization. In: PR (2022)
56. Zhao, S., Gong, M., Liu, T., Fu, H., Tao, D.: Domain generalization via entropy regularization. In: NeurIPS (2020)
57. Zhou, K., Yang, Y., Hospedales, T.M., Xiang, T.: Deep domain-ad image generation for domain generalisation. In: AAAI (2020)
58. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Domain adaptive ensemble learning. In: TIP (2021)
59. Zhou, K., Yang, Y., Qiao, Y., Xiang, T.: Mixstyle neural networks for domain generalization and adaptation. arXiv (2021)