SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2024, Vol. 67, Iss. 5, 152101:1–152101:15
https://doi.org/10.1007/s11432-022-3857-5

Adversarial data splitting for domain generalization

Xiang GU, Jian SUN* & Zongben XU

School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049*, China*

Received 18 July 2022/Revised 15 March 2023/Accepted 13 April 2023/Published online 18 December 2023

Abstract Domain generalization aims to learn a model that is generalizable to an unseen target domain, which is a fundamental and challenging task in machine learning for out-of-distribution generalization. This paper proposes a novel domain generalization approach that enforces the learned model to be able to generalize well over the train/val subset splitting of the training dataset. This idea is modeled herein as an adversarial data splitting framework, formulated as a min-max optimization problem inspired by the meta-learning approach. The min-max optimization problem is solved by iteratively splitting the training dataset into the training and val subsets to maximize the domain shift measured by the objective function and updating the model parameters to enable the model to generalize well from the training subset to the val subset by minimizing the objective function. This adversarial training approach does not assume the known domain labels of the training data; instead, it automatically investigates the "hard" splitting of the train/val subsets to learn the generalizable model. Extensive experimental results using three benchmark datasets demonstrate the superiority of this approach. In addition, we derive a generalization error bound for the theoretical understanding of our proposed approach.

Keywords domain generalization, adversarial learning, data splitting, meta-learning, out of distribution

1 Introduction

Deep learning (DL) is greatly successful in image recognition [1–3]. However, DL methods mostly succeed when the training and test data are from the same distribution (i.e., the i.i.d. assumption). This assumption is often violated because the equipment/environments that the data captured/generated in the training and test phases are often different from those in real applications. In the presence of a distributional difference (i.e., domain shift [4]) between the training and test sets, the trained model performance will significantly degrade on the test set.

To tackle the domain shift issue, domain adaptation (DA) [5] and domain generalization (DG) [6, 7] methods have been used to learn transferable models from the source to the target domain. DA methods commonly align the distributions of different domains either in a feature space [8–12] or a raw pixel space [13], but require knowledge on the unlabeled dataset of the target domain at the training time, which is unrealistic in real-world applications. DG does not assume the known target domain dataset and learns a generalizable model for the unknown target domain, which is the focus of this work.

DG methods usually use multiple source domains to train a model that can generalize to an unseen target domain. The methods in [7, 14, 15] aligned the distributions of different source domains to learn domain-invariant features. In their approach, Refs. [16–19] sampled meta-train and meta-test data from different source domains to simulate the domain shift and train the model in a meta-learning approach. Another set of methods [20–22] augmented the new domain data by mixing the statistics of different domains to train the model and enhance its generalization ability.

Conventional DG methods assume that domain labels are available; that is, the domain of each training data is given. However, in a more realistic scenario, the domain labels of the training data may be unknown or even hard to identify/define. The DG task is more challenging in this "domain-free" setting because the standard meta-learning or adversarial training methods for DG commonly assume known domain labels. Only a few studies on DG have been conducted in the domain-free setting. These studies

* Corresponding author (email: jiansun@xjtu.edu.cn)

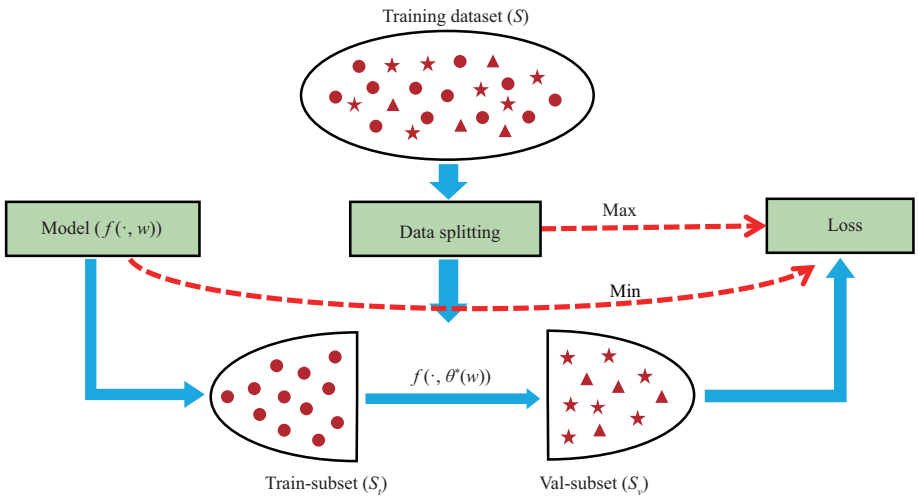

Figure 1 (Color online) Illustration of our ADS framework for DG. We split the training dataset into the train and val subsets. The model trained on the training subset will minimize the loss on the val subset, while the data splitting operation will maximize the generalization loss of the model on the val subset. The proposed ADS approach enforces the model to become generalizable, even in worst-case data splitting.

enhanced the model's generalization ability through self-supervised learning [23], training with clustered multiple latent domains [24], self-challenge of the model [25], or inference time target projections [26]. Although these methods have shown promising results, the DG task in the domain-free setting has not been solved well.

In this study, we propose a novel adversarial training framework for domain-free DG, investigating the domain gap within the training dataset for learning the generalizable model. We specifically aim to learn a model that can generalize well against all possible domain shifts between the train and val subsets from the training dataset splitting. Compared to a baseline model trained by directly minimizing the cross-entropy loss on the training dataset, this model is expected to perform better on unseen, but related test data. However, model training on all possible train/val subsets is infeasible. We alternatively investigate the "hard" train/val subsets with a large domain shift. More concretely, we adversarially generate "hard" train/val subsets with a large domain shift on the training dataset and learn a generalizable model to minimize the domain shift over splitting (Figure 1). We implement this idea by employing our proposed adversarial data splitting (ADS) model formulated as a min-max optimization problem that unifies train/val splitting and meta-learning in the same framework. Accordingly, we propose an iterative optimization algorithm to solve the min-max problem by adversarially splitting the training dataset into the train and val subsets by maximizing their domain shift based on the given model and updating the model by minimizing the prediction error on the val subset through the meta-learning approach with the given splitting. We optimize the min-max problem to enforce the model to generalize well, even in worst-case train/val splitting.

We verify the effectiveness of our method by conducting extensive experiments on the PACS, Office-Home, and CIFAR-10 benchmark datasets under different settings with multiple/single source domain(s). The experimental results show that our proposed method outperforms the baseline and the compared DG methods. The method's effectiveness is confirmed by the ablation studies. We theoretically derive the upper bound of the generalization error for the DG problem and find that our method implicitly minimizes the upper-bound terms. This theoretical analysis partially explains the success of our method.

The contributions of this work are summarized as follows.

• We propose a novel adversarial approach, called ADS, based on meta-learning for DG that adversarially finds the hard data splitting to the recognition model and trains the recognition model to generalize well over this splitting.

• Extensive experiments under different settings with multiple/single source domain(s) show the effectiveness of the proposed approach.

• We present a theoretical analysis for ADS by deriving the upper bound of the generalization error.

The remainder of this paper is structured as follows: Section 2 presents the related studies; Section 3 discusses our proposed method in detail; Section 4 describes the analysis of our method; Section 5 reports

the experimental results; and Section 6 concludes this work.

2 Related work

This section summarizes and compares our method with those in related works, including DG methods with and without domain labels and min-max-related DA and adversarial machine learning approaches.

DG with domain labels. DG methods are mainly classified into three categories when the domain labels are available. The first category includes the methods of [7, 14, 15, 27, 28] that learn the domain-invariant features by aligning the feature distributions or by common/specific feature decomposition. Ref. [29] proposed achieving the domain invariance of the classifier by rearranging the appropriate spatial attention to diverse task-related features. The second category includes methods from [16–19, 30–32]. These methods are based on a meta-learning approach that splits the given source domains into the meta-train and -test domains, trains the model in an episodic training paradigm, and must know the domain labels of the training data. The third and last category involves the approaches of [20–22] that generate fake domain data using the domain label to train the model and enhance its generalization ability. The generated fake domain data enrich the training dataset, making the trained model more generalizable. Our proposed method may mostly be related to the second category. However, unlike previous studies, we considered herein the DG problem in a domain-free setting and an adversarially split training dataset to synthesize the domain shift in a principled min-max optimization method instead of using leave-one-domain-out splitting as in these approaches.

DG without domain labels. When the domain labels are unavailable, the method of [33] learns robust feature representation by projecting out style-related features, such as color and texture. The approach from [23] proposes solving jigsaw puzzles on training images. Meanwhile, the method presented by [24] divides samples into several latent domains via clustering and learns domain-invariant features on these domains through adversarial training. In the work of [25], the method drops features with larger loss gradients to force the trained model to activate more features. The approach of [26] projects target features to the source data manifold at inference time, which may increase the inference computational cost. The abovementioned methods enhance the generalization ability by enforcing the model to extract robust features [23–25, 33] or manifold projection [26]. By contrast, our method is a novel meta-learning approach for DG that adversarially splits the training dataset. Refs. [34, 35] proposed adversarial data augmentation methods to tackle the single-domain generalization problem of the training set comprising a single domain. Refs. [34, 35] adversarially found the "hard" data augmentation that the model performs worse. In contrast, our approach searches for the "hard" dataset splitting that the model cannot generalize well from the train to val subset, which is orthogonal to the methods of [34, 35].

Min-max-related DA. Min-max/adversarial training methods are a typical kind of DA methods [8, 36–38], wherein both labeled source and unlabeled target data are given during training. These methods commonly train a discriminator to distinguish the source and target domains and learn a feature extractor to fool the discriminator to extract the domain-invariant features. Different from the adversarial methods for DA, in methodology, our approach adversarially finds the "hard" dataset splitting for the model and trains the model to generalize well on the given dataset splitting in a meta-learning manner.

Adversarial machine learning. Adversarial machine learning methods [39–41] commonly generate adversarial samples by maximizing the prediction error to improve the model's robustness to data corruption. Our method is different from adversarial machine learning methods in terms of both goal and methodology. Our approach specifically aims to improve the model's robustness to the domain shift for DG, while adversarial machine learning methods [39–41] aim to improve the model's robustness to data corruption. In terms of methodology, adversarial machine learning methods [39–41] often optimize the noise to be added to the input data to worsen the prediction performance, and then train the model to perform well on the corrupted data. By contrast, our approach adversarially splits the training dataset into the train/val subsets to find the hard data splitting with a larger domain shift and train the model to generalize well over this splitting based on meta-learning.

3 Method

Given the training dataset $S = \{(x_i, y_i)\}_{i=1}^{N}$, our goal is to train the model on S to generalize well on an unseen target domain related to the training domain. In our domain-free setting, the domain labels of

the training data in S are unknown. We propose an ADS model for the domain-free DG.

3.1 Adversarial data splitting

In domain-free DG, we did not have the training data of the unseen target domain. We also did not have the domain labels of the samples in the training dataset S. We modeled the task of domain-free DG as a learning problem that enforces the model to be able to generalize well over any train/val (S_t/S_v) subset splitting of the training dataset S. Intuitively, if the model can generalize well against any of the possible domain shifts within the training dataset, we expect it to perform better on the unseen but related target domain data when compared to the baseline model trained by directly minimizing the cross-entropy loss on the training dataset. We denote $f(\cdot,\theta)$ as the model parameterized by θ that outputs the classification probability of its input, w as the initialization of θ, and l as the cross-entropy loss. Our idea is formulated as the following optimization problem:

$$\begin{aligned}&\min_{w}\frac{1}{|\Gamma_\xi|}\sum_{S_v\in\Gamma_\xi}\mathcal{L}(\theta^*(w);S_v)+\mathcal{R}(w)\\&\text{s.t. }\ \theta^*(w)=\arg\min_{\theta}\mathcal{L}(\theta;S_t,w),\ \ S_t=S-S_v.\end{aligned}\tag{1}$$

where Γ_ξ denotes the set of all possible val-subsets of S, and ξ denotes the relative size of val-subset S_v (i.e., $|S_v|\,/\,|S|$). The effect of ξ is evaluated in Subsection 5.2.

The problem of (1) is a meta-learning-based bi-layer optimization problem defined over all the possible splitting of the train and val subsets (S_t/S_v) of the training dataset S. In the problem of (1), we aim to optimize the model parameters w, such that the updated model parameters $\theta^*(w)$ trained on the training subset S_t with w as the initialization can minimize the loss on the val subset S_v coupled with an additional constraint $R(w)$. In the constraint of (1), $\min_\theta \mathcal{L}(\theta;S_t,w)$ is a nonconvex optimization problem, of which multiple minimizers depend on the initialization w. By parameterizing θ^* using w and meta-learning to learn w in (1), we will find a good initialization w, such that the validation loss of $\theta^*(w)$ is minimized. Note that taking w as the initialization to parameterize $\theta^*(w)$ is a commonly utilized strategy in meta-learning-based DG methods [17,18]. We will experimentally justify its effectiveness in Subsection 5.2. The loss functions in the training and val subsets in (1) are defined as follows:

$$\mathcal{L}(\theta;S_t,w)=\frac{1}{|S_t|}\sum_{(x,y)\in S_t}l(f(x,\theta),y),\tag{2}$$

$$\mathcal{L}(\theta^*(w);S_v)=\frac{1}{|S_v|}\sum_{(x,y)\in S_v}l(f(x,\theta^*(w)),y),\tag{3}$$

where the loss in (3) on the val subset is called generalization loss reflecting the validation error. This meta-learning-based formulation enforces that the model with parameters w can generalize well from train subset S_t to the val subset S_v over all possible S_t/S_v pairs. For the train and val subsets with a smaller (resp. larger) domain shift, the model trained on the training subset should perform better (resp. worse) on the val subset, leading to a lower (resp. higher) generalization loss. We then measure the domain shift using the generalization loss.

The number of possible train/val splitting may be prohibitively large; thus, the summation over all possible splitting in (1) is infeasible. We propose the following ADS framework as an alternative:

$$\begin{aligned}&\min_{w}\max_{S_v\in\Gamma_\xi}\mathcal{L}(\theta^*(w);S_v)+\mathcal{R}(w)\\&\text{s.t. }\ \theta^*(w)=\arg\min_{\theta}\mathcal{L}(\theta;S_t,w),\ \ S_t=S-S_v.\end{aligned}\tag{4}$$

Given w, the objective function in (4) is an upper bound of that in (1). In the min-max problem of (4), the train/val (S_t/S_v) splitting is optimized to maximize the generalization loss to increase the domain shift between the train and val subsets by finding the hard splitting to the model. w is optimized by minimizing the generalization loss of the model over the splitting. By the abovementioned interleaved maximization and minimization problems, solving (4) enforces the trained model to be generalizable, even for the worst-case of train/val subset splitting. The model performing efficiently for this worst-case splitting could also work well for the other splittings. The additional constraint $\mathcal{R}(w)$ is set to be the

training loss on S_t (i.e., $\mathcal{R}(w) = \mathcal{L}(w; S_t)$) that constrains the following: the model with parameters w should predict the data labels well on S_t [18]. We stop the backward propagation of $\mathcal{R}(w)$ when updating S_v. The optimization algorithm of (4) is presented in Subsection 3.2.

The model f comprises a feature extractor and a classifier. ResNet [1] can be taken as the feature extractor. The extracted feature is normalized to the unit sphere through L_2-normalization. We followed [42] to utilize the marginal-softmax-based classifier that enforces a larger margin between classes for L_2-normalized features. The output probability for the k-th class of input x is derived as follows:

$$p(k|x) \propto \exp(s(u_k^{\mathrm{T}} z - m\mathbb{I}_{\{k=y\}})), \tag{5}$$

where m and s are the hyperparameters indicating the margin and the radius, respectively; z is the L_2-normalized feature of x; y is the class label; u_k is the parameter with the unit norm; and $\mathbb{I}_{\{\cdot\}}$ is the indicator function. m is set to zero at test time. The effectiveness of L_2-normalization is verified in Subsection 5.2.

3.2 Optimization

We will now discuss the optimization of the bi-layer min-max optimization problem of (4). This optimization problem is challenging mainly because the $\theta^*(w)$ variable in the objective function is a minimizer of another optimization problem. Inspired by the idea of using finite gradient descent to estimate the optimal $\theta^*(w)$ [18, 43–45], we adopt herein one step of gradient descent [18, 44]:

$$\hat{\theta}(w) = w - \alpha g_w^t, \tag{6}$$

where $g_w^t = \nabla_\theta \mathcal{L}(\theta; S_t, w)$, and α is the step size. With (6), our adversarial splitting model in (4) becomes

$$\min_w \max_{S_v \in \Gamma_\xi} \mathcal{L}(w - \alpha g_w^t; S_v) + \mathcal{R}(w). \tag{7}$$

Eq. (7) is a min-max optimization problem; thus, we alternately update S_v and w by fixing the other one as known. The details for this are provided below.

Optimizing w with fixed S_v. Fixing S_v ($S_t = S - S_v$ is then fixed), w can then be updated by gradient descent as follows:

$$w = w - \eta \nabla_w \left(\mathcal{L}(w - \alpha g_w^t; S_v) + \mathcal{R}(w) \right), \tag{8}$$

where η is the learning rate.

Finding the hardest splitting S_v with fixed w. Fixing w, to maximize the objective function in the problem of (7) w.r.t. S_v, we must maximize the generalization loss $\mathcal{L}(w - \alpha g_w^t; S_v)$. In other words, the updated model trained on the training subset fails to generalize well on the val subset (i.e., the domain shift is maximized). For the maximization, we perform a first-order Taylor expansion by $\mathcal{L}(w - \alpha g_w^t; S_v) \approx \mathcal{L}(w; S_v) - \alpha \langle g_w^v, g_w^t \rangle$, where $g_w^v = \nabla_w \mathcal{L}(w; S_v)$, and $\langle \cdot, \cdot \rangle$ denotes the inner product. With the $\mathcal{L}(\cdot; S_v)$ definition in (3), g_w^v definition, and Taylor expansion, the optimization problem of (7) w.r.t. S_v can be further written as $\max_{S_v \in \Gamma_\xi} \{\frac{1}{|S_v|} \sum_{(x,y) \in S_v} l\left(f(x,w), y\right) - \alpha \langle \nabla_w l(f(x,w), y), g_w^t \rangle\}$. This problem is equivalent to the following formulation:

$$\begin{aligned} &\max_{S_v, A} \sum_{(x,y) \in S_v} l\left(f(x,w), y\right) - \alpha \langle \nabla_w l(f(x,w), y), A \rangle \\ &\text{s.t. } A = g_w^t, S_v \in \Gamma_\xi, \end{aligned} \tag{9}$$

where we introduce an auxiliary variable, A. The problem of (9) is solved by alternately updating S_v and A. We initialize A with the gradient of a randomly selected sample. Given A, maximizing the objective function in (9) w.r.t. $S_v \in \Gamma_\xi$ means to find a subset of S with $\xi|S|$ samples, such that the objective function on this subset is maximized, because Γ_ξ is the set consisting of all the S subsets composed of $\xi|S|$ samples. Accordingly, we can select $\xi|S|$ samples from S to constitute S_v, such that the mean of the objective function values on the selected samples is maximized. We then compute and rank the values of $l\left(f(x,w), y\right) - \alpha \langle \nabla_w l(f(x,w), y), A \rangle$ for all $(x,y) \in S$ and select the largest $\xi|S|$ samples to constitute the S_v. Given S_v (S_t is then given), we update A by $A = g_w^t = \frac{1}{|S_t|} \sum_{(x,y) \in S_t} \nabla_w l(f(x,w), y)$ based on the g_w^t

definition. We discuss the details and convergence of this alternate iteration in Appendix A. Computing gradient w.r.t. all parameters is time- and memory-consuming; hence, we only compute gradient w.r.t. parameters of the final layer (i.e., classifier) of model f. We avoid the trivial solution that the samples in the train and val subsets are of different classes by learning the splitting for each class in turn, as shown in Algorithm 1. For the k-th class, we denote S^k as its samples and I as the sample indexes. We then solve (9), in which S is replaced by S^k to find the "hard" val subset S_v^k of S^k using the abovementioned optimization process. In other words, we select the largest $\xi|I|$ samples according to the rank of their objective values in (9) to constitute S_v^k.

In summary, we alternately use (8) to update the model parameters and solve (9) to find the hardest S_v. Algorithm 2 presents the training algorithm.

Algorithm 1 Updating the data splitting

Input: Training dataset S, model $f(\cdot, w)$.
Output: Train/val subsets S_t, S_v.

Set $S_v = \emptyset$, $S_t = \emptyset$;
for $k = 1, 2, \ldots, K$ **do**
  Denote I as the indexes of samples of the k-th class and $S^k = \{(x_i, y_i) : i \in I\}$;
  Extract feature z_i of sample x_i for $i \in I$, and feed z_i to the classifier to calculate the loss l_i and its gradient g_i w.r.t. the classifier parameters;
  Randomly select a gradient to initialize A;
  for $t' = 1, 2, \ldots, M_1$ **do**
    Compute $l_i - \alpha \langle g_i, A \rangle$ and select the largest $\xi|I|$ samples to constitute S_v^k;
    $S_t^k = S^k - S_v^k$;
    Update A by $\frac{1}{|S_t^k|} \sum_{i \in I} \mathbb{I}_{\{(x_i, y_i) \in S_t^k\}} g_i$;
  end for
  $S_v = S_v \cup S_v^k$, $S_t = S_t \cup S_t^k$;
end for

Algorithm 2 Training

Input: Training dataset S.
Output: Trained model $f(\cdot, w)$.

Randomly split S into S_t and S_v;
for $t = 1, 2, \ldots, M_0$ **do**
  Sample mini-batch data from S_t and S_v;
  Update w by (8) using the mini-batch SGD on the sampled data;
  if $t\%T == 0$ **then**
    Update S_t and S_v using Algorithm 1;
  end if
end for

4 Analysis

We first derive a generalization error bound in Theorem 1 for the general meta-learning-based framework for DG. We then analyze our method based on Theorem 1.

Without loss of generality, we consider the binary classification problem. Denoting $\mathcal{H} = \{f(\cdot, w)\}$ and $\mathcal{H}_{S_t} = \{f(\cdot, \theta^*(w)) : \theta^*(w) = \arg\min_\theta \mathcal{L}(\theta; S_t, w)\}$, the meta-learning methods [14, 17] for DG find a model f in $\mathcal{H}_{S_t}$ to minimize the generalization loss on S_v. We then denote $\mathcal{P}$ as the source distribution, $\epsilon_{\mathcal{Q}}^{\Psi}(f) = \mathbb{E}_{(x,y)\sim\mathcal{Q}}[\mathbb{I}_{\{\Psi(f(x)) \neq y\}}]$ as the generalization error on the distribution $\mathcal{Q}$ of the unseen target domain, $\hat{\epsilon}_{S_v}^{\Psi}(f) = \frac{1}{|S_v|} \sum_{(x,y)\in S_v} \mathbb{I}_{\{\Psi(f(x)) \neq y\}}$ as the empirical error on S_v, $VC(\mathcal{H})$ as the VC dimension of $\mathcal{H}$, and $\Psi(\cdot)$ as the prediction rule (e.g., the Bayes optimal predictor). We derive the following theorem based on the DA theory [46, 47] and inspired by the analysis in [48].

Theorem 1. Given a constant $\gamma > 0$, if we assume $\mathbb{E}_{\mathcal{Q}}[\mathbb{I}_{\{l(f(x),y)>\gamma\}}] \geqslant \mathbb{E}_{\mathcal{P}}[\mathbb{I}_{\{l(f(x),y)>\gamma\}}]$, then for any $S_v \in \Gamma_\xi$ and $S_t = S - S_v$ and $\delta \in (0, 1)$, with a probability of at least $1 - 2\delta$, we have $\forall f \in \mathcal{H}_{S_t}$,

$$\epsilon_{\mathcal{Q}}^{\Psi_l}(f) \leqslant \hat{\epsilon}_{S_v}^{\Psi_l}(f) + B(S_v) + 2\sqrt{\frac{8}{\xi\,|S|}\left(C_H + \frac{4}{\delta}\right)} + C^*(\mathcal{Q}, S_t), \tag{10}$$

where $B(S_v) = -\inf_{f' \in \mathcal{H}_{S_t}} \frac{1}{|S_v|} \sum_{(x,y)\in S_v} \mathbb{I}_{\{l(f'(x),y)>\gamma\}}$, $C_H = \sup_{S'_v \in \Gamma_\xi} VC(\mathcal{H}_{S-S'_v}^{\Psi_l}) \log \frac{2e\xi|S|}{VC(\mathcal{H}_{S-S'_v}^{\Psi_l})}$, $C^*(\mathcal{Q}, S_t) = \sup_{f' \in \mathcal{H}_{S_t}} \mathbb{E}_{\mathcal{Q}}[\mathbb{I}_{\{l(f'(x),y)>\gamma\}}] + \inf_{f' \in \mathcal{H}_{S_t}} \{\epsilon_{\mathcal{P}}^{\Psi_l}(f') + \epsilon_{\mathcal{Q}}^{\Psi_l}(f')\}$. $\mathcal{H}_{S_v}^{\Psi_l} = \{\Psi_l \circ f : f \in \mathcal{H}_{S_v}\}$,

Ψ_l is a loss-related indicator:

$$\Psi_l(f(x)) = \begin{cases} 1, & \text{if } l(f(x), y) > \gamma, \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Appendix B provides the proof. The $\mathbb{E}_{\mathcal{Q}}[\mathbb{I}_{\{l(f(x),y)>\gamma\}}] \geqslant \mathbb{E}_{\mathcal{P}}[\mathbb{I}_{\{l(f(x),y)>\gamma\}}]$ assumption in Theorem 1 is realistic because the model trained on the $\mathcal{P}$ data should have a smaller classification loss on $\mathcal{P}$ than $\mathcal{Q}$. In Theorem 1, C_H is determined by the diversity of the hypothesis space and the number of training data. The network structure is predefined in training; hence, we could informally treat C_H as a constant. $C^*(\mathcal{Q}, S_t)$ depends on the distribution gap between Q and S_t. Intuitively, if S_t is close to the unseen distribution Q, the models trained on S_t (i.e., models $f' \in \mathcal{H}_{S_t}$) have a smaller error/loss on the Q data, and $C^*(\mathcal{Q}, S_t)$ could be smaller. However, we can hardly estimate $C^*(\mathcal{Q}, S_t)$ in training due to the lack of the Q data in the DG task.

We will now illustrate that our ADS model in (4) implicitly minimizes the upper bound $\hat{\epsilon}^{\Psi_l}_{S_v}(f) + B(S_v)$ in (10). First, the first term $\hat{\epsilon}^{\Psi_l}_{S_v}(f)$ of the bound as the empirical error on S_v is just the generalization loss $\mathcal{L}(\theta^*(w); S_v)$ in (4), which is minimized in our ADS model of (4). Second, minimizing the second term $B(S_v)$ of the bound w.r.t. S_v is equivalent to

$$\max_{S_v \in \Gamma_\xi} \inf_{f \in \mathcal{H}_{S_t}} \frac{1}{|S_v|} \sum_{(x,y) \in S_v} \mathbb{I}_{\{l(f(x),y)>\gamma\}}, \tag{12}$$

based on the $B(S_v)$ definition. Our ADS model in (4) is closely related to the realization of the goal of (12). Eqs. (12) and (4) are min-max problems w.r.t. the model parameters and the train/val subset splitting. Meanwhile, the objective function in (12) can be taken as a variant of the empirical loss on S_v using the indicator function $\mathbb{I}$ in contrast to our cross-entropy-based loss $\mathcal{L}(\theta^*(w); S_v)$ in (4). In summary, our ADS model is closely related to the minimization of the upper bound in (10).

5 Experiments

We evaluated our method under three types of experimental settings: multi-source with domain shift (MSDS), in which the training data are from multiple source domains, and a domain shift exists between the training and test data; single source with domain shift (SSDS), in which the training data are from a single source domain, and a domain shift exists between the training and test data; and same source and target domain (SSTD), in which the training and test data are from the same domain.

We conducted experiments on three benchmark datasets. PACS [49] contains four domains sharing seven classes, namely art painting (A), cartoon (C), photo (P), and sketch (S). It is utilized to conduct experiments in the MSDS and SSDS settings. Office-Home [50] consists of four domains, namely Art (Ar), Clipart (Cl), Product (Pr), and Real World (Rw), which share 65 classes. Office-Home is used for the MSDS setting. Meanwhile, CIFAR-10 [51] is taken for the SSTD setting.

We compared our method with the following multi-source DG methods, including D-SAM [52], JiGen [23], MASF [17], MMLD [24], RSC [25], DDAIG [20], L2A-OT [53], MixStyle [21], PAR [54], SagNet [55], EFDMix [56], PCL [57], MiRe [58], CB+CORAL [59], and the baseline ERM [1]. ERM denotes the empirical risk minimization that directly trains the model with the ResNet [1] backbones on all the training data using cross-entropy loss without any DG technique and L_2-normalization. We also compared our approach with the typical single-source DA methods, including M-ADA [34] and GUD [35].

5.1 Implementation details and results

MSDS. For the MSDS setting, we trained the model on three domains and evaluated it using the remaining domain for the PACS and Office-Home datasets. We used ResNet18 and ResNet50 [1] pre-trained on ImageNet [60] as the backbones. For each, the last fully connected layer was replaced by a bottleneck layer, and the corresponding network was taken as the feature extractor. The bottleneck layer dimension was set to 512 for the ResNet18 backbone, as in [48], and 256 for the ResNet50 backbone, as in [61]. We employed the lower bound of the radius in [61] to set s to 7.5 and 10.0 for PACS and Office-Home, respectively. m was set to 0.2 and 0.1 for PACS and Office-Home. The stochastic gradient descent (SGD) with a momentum of 0.9 was utilized to update the model parameters. The classifier and bottleneck layer were trained from scratch; hence, their learning rates were 10 times that of convolutional

Table 1 Results in the MSDS setting on PACS[a)]

Task	C, P, S→A	A, P, S→C	A, C, S→P	A, C, P→S	Average
			ResNet18		
ERM	$80.2^{\pm.4}$	$75.5^{\pm.5}$	$95.9^{\pm.1}$	$70.1^{\pm.9}$	80.4
D-SAM	77.3	72.4	95.3	77.8	80.7
JiGen	79.4	75.3	96.0	71.4	80.5
MASF	$80.3^{\pm.2}$	$77.2^{\pm.1}$	$95.0^{\pm.1}$	$71.7^{\pm.2}$	81.0
MMLD	81.3	77.2	96.1	72.3	81.8
RSC	83.4	**80.3**	96.0	80.9	85.1
DDAIG	84.2	78.1	95.3	74.7	83.1
L2A-OT	83.3	78.2	96.2	73.6	82.8
MixStyle	84.1	78.8	96.3	73.8	83.7
PAR	80.6	78.3	94.9	79.3	83.3
SagNet	83.6	77.7	95.5	76.3	83.3
EFDMix	$83.9^{\pm.4}$	$79.4^{\pm.7}$	$\mathbf{96.8}^{\pm.4}$	$75.0^{\pm.7}$	83.9
MiRe	$84.6^{\pm.5}$	$79.5^{\pm.4}$	$\mathbf{96.8}^{\pm.2}$	$78.4^{\pm1.0}$	84.8
ADS (ours)	$84.2^{\pm.1}$	$79.5^{\pm.3}$	$95.8^{\pm.1}$	$82.1^{\pm.4}$	85.4
ADS+RandAug (ours)	$\mathbf{86.2}^{\pm.3}$	$80.1^{\pm.7}$	$96.0^{\pm.1}$	$\mathbf{87.4}^{\pm.9}$	**87.4**
			ResNet50		
ERM	$86.1^{\pm.2}$	$79.2^{\pm.4}$	$97.6^{\pm.1}$	$70.3^{\pm.7}$	83.3
MASF	$82.9^{\pm.2}$	$80.5^{\pm.2}$	$95.0^{\pm.1}$	$72.3^{\pm.2}$	82.7
MMLD	86.3	77.0	97.3	71.7	83.1
RSC	87.9	82.2	97.9	83.4	87.8
DDAIG	87.4	77.2	95.8	78.3	84.7
MixStyle	88.9	81.9	97.2	73.8	85.5
PAR	86.5	81.8	96.8	85.2	87.6
SagNet	86.3	81.1	97.8	77.4	85.7
EFDMix	$\mathbf{90.6}^{\pm.3}$	$82.5^{\pm.7}$	$\mathbf{98.1}^{\pm.2}$	$76.4^{\pm1.2}$	86.9
PCL	90.2	83.9	**98.1**	82.6	88.7
CB+CORAL	$87.8^{\pm.8}$	$81.0^{\pm.1}$	$97.1^{\pm.4}$	$81.1^{\pm.8}$	86.7
ADS (ours)	$89.1^{\pm.1}$	$84.6^{\pm.2}$	$96.8^{\pm.2}$	$85.6^{\pm.3}$	89.0
ADS+RandAug (ours)	$90.4^{\pm.5}$	$\mathbf{84.7}^{\pm1.1}$	$96.8^{\pm.2}$	$\mathbf{88.3}^{\pm.8}$	**90.1**

a) The best results are bolded.

layers. Following [8], the learning rate of the convolutional layers was adjusted by $\eta = \frac{0.001}{(1+10p)^{0.75}}$, where p is the training progress, which linearly changes from 0 to 1. The step size α was set to 10^{-5}. M_0, M_1, and T in Algorithms 1 and 2 were set to 20000, 10, and 500, respectively. The batch size was set to 64. ξ was set to 0.5. The running mean and the variance of the batch normalization (BN) layers were fixed as the pre-trained values on ImageNet discussed in [31]. Considering the memory limit, for ResNet50, we adopted the first-order approximation [44] that stops the g_w^t gradient in (8) to reduce the memory and the computational cost.

In Table 1, our ADS with ResNet18/ResNet50 on PACS achieved results of 85.4%/89.0%, which were competitive to the best results (85.1%/88.7%) of the previous methods. In Table 2, on Office-Home, our ADS achieved competitive accuracies of 64.3% using ResNet18 and 70.3% using ResNet50. Our approach splits the training dataset without data/domain augmentation. We also investigated the performance of our proposed ADS utilizing data/domain augmentation that achieved a promising performance in DG. For simplicity, we applied the RandAugment [62] to our approach and denoted this as ADS+RandAug. Tables 1 and 2 show that ADS+RandAug further improved the ADS performance and achieved the best results compared with the previous state-of-the-art (SOTA) methods. Compared with the baseline ERM, our ADS using ResNet18 and ResNet50 improved its performance by 5.0% and 5.7% on PACS and 2.1% and 2.1% on Office-Home, respectively. In Table 1, on PACS, the ADS outperformed the ERM in almost all tasks, except for task P. Note that, in task S, in which the target domain style was extremely different from the source domains, our ADS boosted the ERM accuracy by 12.0% and 15.3% based on ResNet18 and ResNet50, respectively. On the more challenging dataset of Office-Home (because the number of classes was larger than PACS), our ADS consistently outperformed the ERM in almost all tasks (Table 2).

Table 2 Results in the MSDS setting on Office-Home[a)]

Task	Cl, Pr, Rw→Ar	Ar, Pr, Rw→Cl	Ar, Cl, Rw→Pr	Ar, Cl, Pr→Rw	Average
		ResNet18			
ERM	$56.7^{\pm.3}$	$47.6^{\pm.2}$	$71.4^{\pm.1}$	$72.9^{\pm.3}$	62.2
D-SAM	58.0	44.4	69.2	71.5	60.8
JiGen	53.0	47.5	71.5	72.8	61.2
MMLD	49.9	46.0	65.2	75.2	59.1
RSC	58.4	47.9	71.6	74.5	63.1
DDAIG	59.2	52.3	74.6	76.0	65.5
L2A-OT	60.6	50.1	74.8	**77.0**	65.6
MixStyle	58.7	53.4	74.2	75.9	65.5
PAR	55.5	45.6	66.8	70.6	59.6
SagNet	60.2	45.4	70.4	73.4	62.3
MiRe	$60.2^{\pm.8}$	$53.2^{\pm.9}$	$\mathbf{75.1}^{\pm.6}$	$76.4^{\pm0.6}$	66.2
ADS (ours)	$\mathbf{62.0}^{\pm.2}$	$48.6^{\pm.3}$	$71.4^{\pm.1}$	$75.2^{\pm.1}$	64.3
ADS+RandAug (ours)	$61.2^{\pm.7}$	$\mathbf{54.7}^{\pm.5}$	$73.8^{\pm.3}$	$75.4^{\pm.2}$	**66.3**
		ResNet50			
ERM	$64.9^{\pm.4}$	$51.8^{\pm.2}$	$76.5^{\pm.2}$	$79.4^{\pm.3}$	68.2
MMLD	60.3	52.4	71.3	80.1	66.0
DDAIG	58.1	52.7	76.0	77.8	66.1
MixStyle	66.5	57.2	76.8	79.4	70.0
PAR	63.8	52.9	74.4	77.7	67.2
SagNet	68.9	50.1	76.8	80.5	69.1
PCL	67.3	59.9	78.7	80.7	71.6
CB+CORAL	$65.6^{\pm.6}$	$56.5^{\pm.6}$	$77.6^{\pm.3}$	$78.8^{\pm.5}$	69.6
ADS (ours)	$\mathbf{70.2}^{\pm.2}$	$53.5^{\pm.4}$	$77.4^{\pm.1}$	$80.2^{\pm.2}$	70.3
ADS+RandAug (ours)	$69.0^{\pm.5}$	$\mathbf{60.2}^{\pm.2}$	$\mathbf{79.2}^{\pm.3}$	$\mathbf{80.8}^{\pm.4}$	**72.3**

a) The best results are bolded.

SSDS. We evaluated our method in this setting on PACS using ResNet18. We trained the model on one domain and tested it on each of the remaining three domains. The implementation details are the same as those in MSDS. Table 3 presents the results. The JiGen and SagNet results were quoted from [55]. Our ADS method achieved the best average accuracy (66.6%), outperforming the baseline ERM by 4.2%. It also outperformed the baseline approach of the ERM in 10 tasks among 12. Like our proposed ADS, the RSC, Jigen, and SagNet approaches did not use the domain labels; hence, they can be applied to the SSDS setting. The ADS outperformed these methods in the SSDS setting (Table 3).

SSTD. We also applied our DG method to the common recognition task of the training and test data being from the same domain (i.e., SSTD) on the CIFAR-10 using ResNet18. We investigated the effect of the training size by sampling different training data sizes from the provided training set (i.e., source domain). Note that the same data were used to train models of different methods. We evaluated the trained model on the original test set of CIFAR-10 and its corresponding corrupted dataset CIFAR-10-C [63]. The hyperparameters of s and m were set to 8.0 and 0.2, respectively. The other implementation details were similar to those in MSDS, except that the running mean and variance were updated in the BN layers. Figure 2 presents the results. The JiGen [23] and MMLD [55] results were obtained by running their codes on CIFAR-10. The ADS outperformed the other methods under all different training sizes. The performance improvement over the ERM was generally larger when the training size was smaller, which may be because the model can more possibly overfit when the training size is smaller, and our ADS is designed to learn generalizable features.

5.2 Analysis

Effectiveness of ADS. We conducted ablation studies on PACS using ResNet18 in both the MSDS and SSDS settings to justify the method's effectiveness. In Tables 4 and 5, L_2 denotes the feature L_2-normalization. ADS with rand-split denotes our approach of randomly splitting the train/val subsets at each training step, while ADS (w/o split) denotes the approach that trains our model on all training data without using data splitting. Tables 4 and 5 show that ADS outperformed ADS (without split) by 3.1% and 2.7% in the SSDS and MSDS settings, respectively, indicating that our ADS model is effective.

Table 3 Results in SSDS setting on PACS (ResNet18)[a)]

Task	A→C	A→P	A→S	C→A	C→P	C→S	P→A	P→C	P→S	S→A	S→C	S→P	Average
ERM	$63.7^{\pm.3}$	$95.6^{\pm.1}$	$63.5^{\pm.4}$	$72.0^{\pm.3}$	**86.5**$^{\pm.1}$	$73.3^{\pm.2}$	$68.4^{\pm.4}$	$32.7^{\pm.5}$	$42.2^{\pm.4}$	$41.6^{\pm.5}$	$60.3^{\pm.2}$	$49.3^{\pm.3}$	62.4
JiGen	57.0	**96.1**	50.0	65.3	85.5	65.9	62.4	27.2	35.5	26.6	41.1	42.8	54.6
SagNet	67.1	95.7	56.8	**72.1**	85.7	69.2	69.8	35.1	40.7	41.1	62.9	46.2	61.9
RSC	66.3	94.3	61.2	72.6	86.3	**75.6**	67.9	32.5	49.2	54.7	67.1	52.5	65.0
M-ADA	62.6	95.5	60.7	69.1	86.3	68.3	56.2	36.8	**55.5**	45.9	**72.1**	56.8	63.8
GUD	60.8	80.1	60.7	49.3	59.4	61.6	48.4	34.3	42.6	22.1	52.2	27.1	49.9
ADS (ours)	**67.5**$^{\pm.2}$	$94.5^{\pm.1}$	**67.1**$^{\pm.3}$	$69.0^{\pm.2}$	**86.5**$^{\pm.1}$	$73.8^{\pm.3}$	**70.2**$^{\pm.2}$	**36.1**$^{\pm.4}$	$52.1^{\pm.2}$	**56.7**$^{\pm.2}$	$67.9^{\pm.3}$	**57.4**$^{\pm.1}$	**66.6**

a) The best results are bolded.

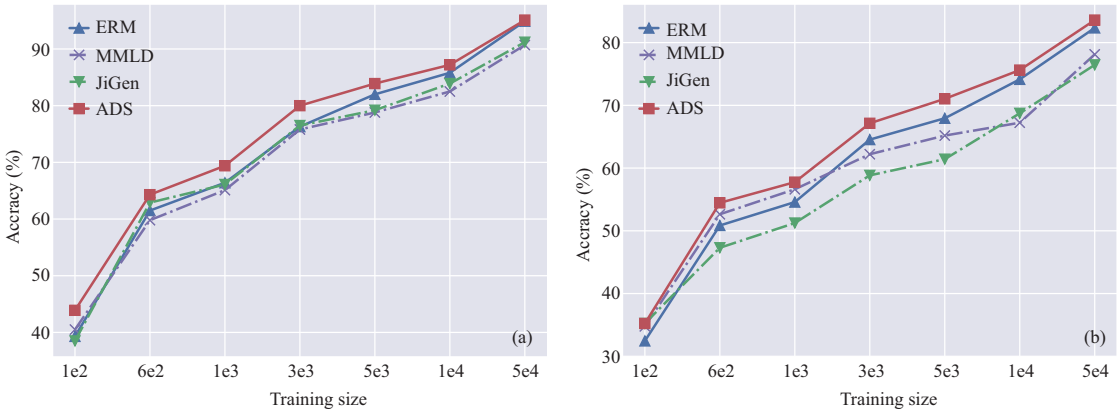

Figure 2 (Color online) Results in the SSTD setting on CIFAR-10 and CIFAR-10-C (ResNet18). (a) CIFAR-10; (b) CIFAR-10-C.

Table 4 Ablation study on PACS in the SSDS setting[a)]

Task	A→C	A→P	A→S	C→A	C→P	C→S	P→A	P→C	P→S	S→A	S→C	S→P	Average
ADS (w/o split)	64.5	**95.4**	**67.2**	69.8	87.2	73.3	67.2	31.7	38.5	44.3	67.9	49.8	63.1
ADS (w/o L_2)	**67.6**	95.2	66.2	**76.9**	**88.4**	73.5	**70.4**	30.5	35.3	52.7	67.7	57.3	65.1
ADS (w/ rand-split)	67.2	95.3	63.7	70.8	85.3	**75.5**	70.0	**37.2**	45.9	53.5	64.5	55.8	65.4
ADS	67.5	94.5	67.1	69.0	86.5	73.8	70.2	36.1	**52.1**	**56.7**	**67.9**	**57.4**	**66.6**

a) The best results are bolded.

Table 5 Ablation study on PACS in the MSDS setting[a)]

Target	A	C	P	S	Average
ADS (w/o split)	82.5	77.5	95.2	75.6	82.7
ADS (w/o L_2)	83.1	77.0	94.8	79.3	83.6
ADS (w/ rand-split)	83.2	78.6	**95.8**	79.5	84.3
ADS	**84.2**	**79.5**	**95.8**	**82.1**	**85.4**

a) The best results are bolded.

Furthermore, ADS improved the results of ADS with rand-split by 1.2% and 1.1% in the two settings, confirming the effectiveness of adversarial splitting compared to random splitting. This improvement may be because the random splitting cannot cover all the possible splitting of the train/val subsets. Our ADS approach learned to generalize on the adversarially learned hardest train/val splitting, enforcing that the model can generalize, even in worst-case splitting. The ADS also outperformed ADS (w/o L_2) by 1.5% and 1.8% in two settings, indicating that L_2-normalization was effective and contributed to the performance improvements.

Comparison with domain label-based data splitting. We compared our adversarial splitting (adv-split) with the domain label-based splitting (label-split) that utilized leave-one-domain-out splitting. In Figure 3(a), when the domain label is available, domain label-based splitting is less effective than adversarial splitting. The reasons for the abovementioned findings are explained here. The internal distributional gaps within the training dataset cannot be sufficiently captured by the given fixed domain labels due to the differences in styles, poses, sub-classes, etc. within the training data. Our adversarial splitting method does not rely on the domain label. Instead, it iteratively finds the hardest train/val

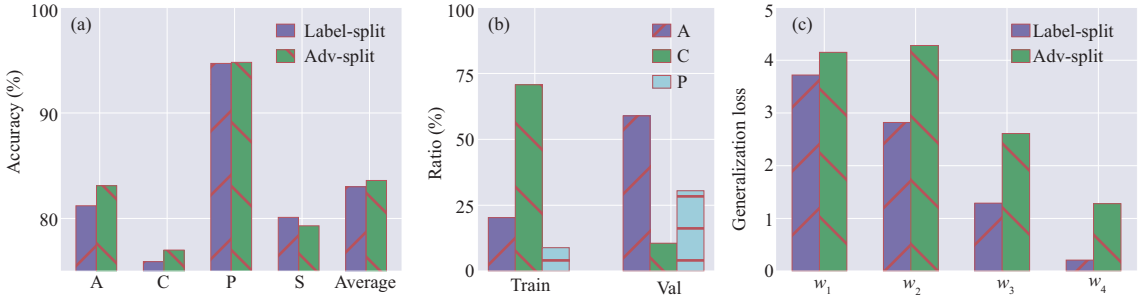

Figure 3 (Color online) (a) Results of adversarial (adv-split) and domain label-based (label-split) splitting on the PACS dataset in the MSDS setting; (b) ratio of samples in the learned train/val subsets from different source domains (A, C, and P); (c) generalization losses (a larger value means harder splitting) of adv- and label-split for the model with parameters w_1, w_2, w_3, w_4 at four different increasing training steps.

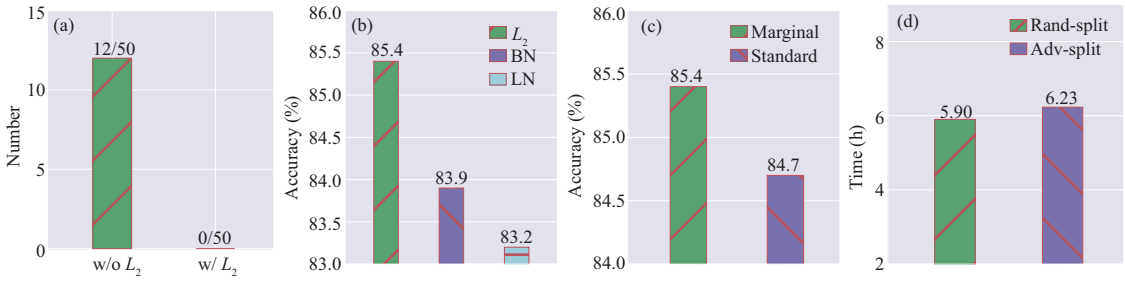

Figure 4 (Color online) (a) Number of occurrences of gradient explosion; (b) results of the L_2-normalization (L_2), BN, and LN; (c) results of the marginal and standard softmax; (d) computational cost of adversarial and random splitting. All these experiments were conducted on PACS in the MSDS setting.

splitting to the current model by maximizing the domain shift and then trains the model to generalize on this splitting. This strategy more flexibly investigates the distributional gaps adaptive to the model on the training dataset and can potentially enhance the generalization ability of the learned model. Figure 3(b) presents the ratio of the samples in the learned train/val subsets (in task S in the MSDS setting) belonging to the underlying source domains A, C, and P. Figure 3(b) indicates that the learned splitting is different from the domain label-based splitting. Figure 3(c) depicts the splitting hardness using the generalization loss as a measure. The domain label-based splitting was not as hard as adversarial splitting to the model. When the domain label was absent, the domain label-based splitting cannot be used anymore, while our adversarial splitting still worked well (Tables 1–3 and Figure 2).

***L_2*-normalization mitigates gradient explosion.** We also experimentally found that the L_2-normalization can mitigate the gradient explosion that frequently occurs in meta-learning-based DG approaches [17, 18]. We independently conducted the same experiments on task A in the MSDS setting 50 times with L_2-normalization (w/ L_2) and without L_2-normalization (w/o L_2). We then counted the number of gradient explosion occurrences. Figure 4(a) illustrates the results. Figure 4(a) shows that with the L_2-normalization, the number of gradient explosion occurrences significantly decreased, indicating that the L_2-normalization mitigates the gradient explosion. We also theoretically analyze this finding in Appendix C.

Comparison of different normalizations. Figure 4(b) compares the L_2-normalization (L_2) with the BN and layer normalizations (LN) in our framework. L_2 outperformed BN and LN by 1.5% and 2.2%, respectively, conforming the effectiveness of the L_2-normalization for DG.

Comparison with the standard softmax. Figure 4(c) compares the marginal softmax-based classifier with the standard softmax-based classifier in our method. The marginal softmax outperformed the standard softmax by 0.7%, confirming the effectiveness of the marginal softmax-based classifier in our framework.

Computational cost. We compared the computational cost by the total training time of the adversarial and random splitting in the same number of steps (20000), as in Figure 4(d). The training time of the adversarial splitting was slightly higher than that of the random splitting by 1.95% (0.33/5.90 h).

Convergence and loss stability. We testified the convergence of ADS with errors and losses in different tasks in Figures 5(a)–(c). Figures 5(a) and (b) illustrate the classification error curves on the

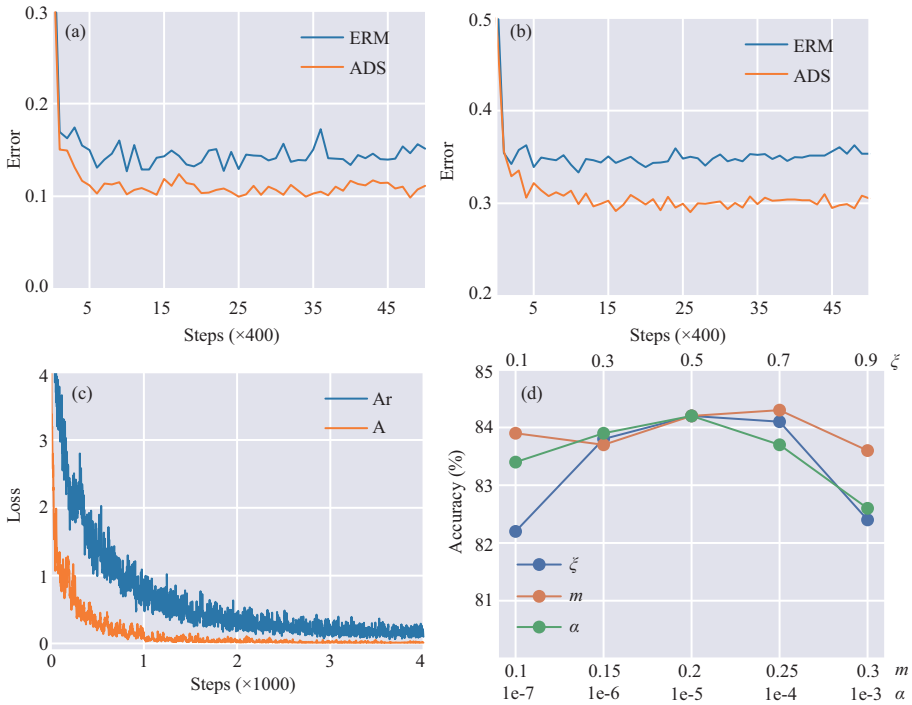

Figure 5 (Color online) Target errors and losses of task A (PACS) and Ar (Office-Home) during training based on ResNet50 in the MSDS setting. (a) Error of task A; (b) error of task Ar; (c) loss of ADS; (d) effect of the hyperparameters of ξ, α, and m. The results are of task A on the PACS dataset based on ResNet18 in the MSDS setting.

target domains of A and Ar, respectively, in the MSDS setting. Figure 5(c) depicts the training loss of ADS in tasks A and Ar in the MSDS setting. The curves indicate that ADS converged in the training process. ADS also showed better stability than the baseline ERM in Figures 5(a) and (b).

Sensitivity to hyperparameters. Figure 5(d) presents the effect of the hyperparameters of ξ, α and m. The results were similar when ξ ranged from 0.3 to 0.7. As for the effect of α, the accuracy was stable to the α values in the large range of 1e−6 to 1e−4. A small α results in a small step size for parameter updating in the meta-learning framework and limits the benefits of meta-learning and adversarial splitting. By contrast, a larger α results in a larger step size for gradient descent-based network updating, which may fail to decrease the training loss from the optimization perspective. Figure 5(d) shows that the results are relatively stable to the m value.

Visualization of the learned feature space. We visualized the feature space learned by our ADS method and the baseline ERM (Figure 6) by t-SNE [64]. ADS seemed to yield a better class separation and a better alignment of the distributions of source and unseen target domains, which can possibly explain the accuracy improvements achieved by our ADS.

Comparison of strategies for $\theta^*(w)$ parameterization. As discussed in Section 3, we parameterized $\theta^*(w)$ by considering w as the initialization of θ. Another strategy of parameterizing $\theta^*(w)$ is to use a neural network, denoted as $\eta(w)$, with w as the input. For this network-based strategy, $\eta(w)$ was trained on the training subset, and the S_v optimization was the same as that in Subsection 3.2 based on the Taylor expansion of the generation loss at w. We conducted experiments to compare these two strategies. $\eta(w)$ comprised two fully connected layers with rectified linear unit (ReLU) activation function after the first layer. To reduce the computational burden due to the large number of parameters, we fixed the feature extractor in this experiment for both strategies. The results on the PACS dataset in the MSDS setting for the initialization- and network-based strategies were 70.8% and 67.9%, respectively. The initialization-based strategy produced a better result and was simpler to implement.

6 Conclusion

In this study, we unified adversarial training and meta-learning in a novel proposed ADS framework to tackle the general DG problem. Our extensive experiments showed the effectiveness of the proposed

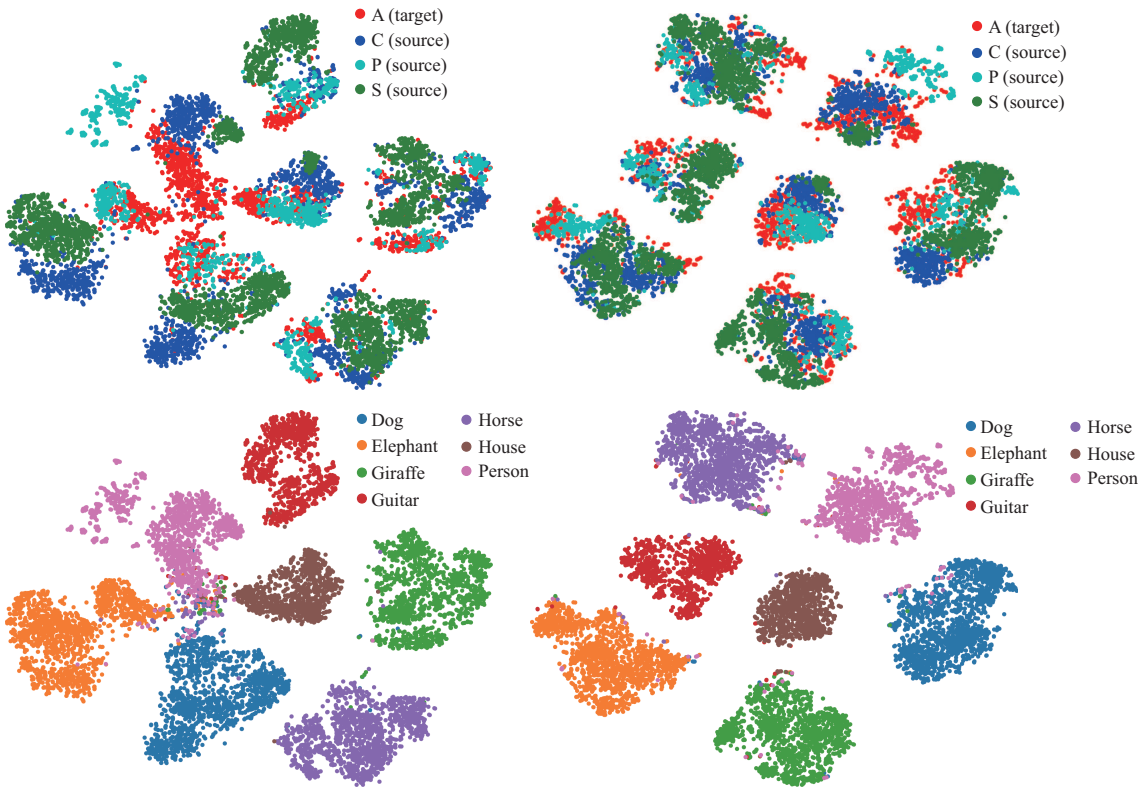

Figure 6 (Color online) t-SNE visualization of the extracted features using our proposed ADS (domain) (b), ADS (class) (d), and ERM (domain) (a), ERM (class) (c) on the PACS dataset in the MSDS setting. The different colors in (a) and (b) indicate the different domains, while those in (c) and (d) depict different classes.

method. Our future work will present a deeper theoretical understanding and more applications of our proposed method.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2021YFA1003002), National Natural Science Foundation of China (Grant Nos. U20B2075, 12125104, 11971373, 61721002), and Fundamental Research Funds for the Central Universities.

Supporting information Appendixes A–D. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 He K M, Zhang X Y, Ren S Q, et al. Deep residual learning for image recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016
2 Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, 2012
3 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014. ArXiv:1409.1556
4 Torralba A, Efros A A. Unbiased look at dataset bias. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011
5 Pan S J, Yang Q. A survey on transfer learning. IEEE Trans Knowl Data Eng, 2010, 22: 1345–1359
6 Blanchard G, Lee G, Scott C. Generalizing from several related classification tasks to a new unlabeled sample. In: Proceedings of Advances in Neural Information Processing Systems, 2011
7 Muandet K, Balduzzi D, Schölkopf B. Domain generalization via invariant feature representation. In: Proceedings of Advances in Neural Information Processing Systems, 2013
8 Ganin Y, Ustinova E, Ajakan H, et al. Domain-adversarial training of neural networks. J Mach Learn Res, 2016, 17: 2096–2030
9 Long M S, Cao Y, Wang J M, et al. Learning transferable features with deep adaptation networks. In: Proceedings of International Conference on International Conference on Machine Learning, 2015
10 Tao J W, Chung F L, Wang S T. A kernel learning framework for domain adaptation learning. Sci China Inf Sci, 2012, 55: 1983–2007
11 Wang Y, Peng J J, Wang H B, et al. Progressive learning with multi-scale attention network for cross-domain vehicle re-identification. Sci China Inf Sci, 2022, 65: 160103
12 Huang L Q, Liu Z G, Pan Q, et al. Evidential combination of augmented multi-source of information based on domain adaptation. Sci China Inf Sci, 2020, 63: 210203
13 Hoffman J, Tzeng E, Park T, et al. CyCADA: cycle-consistent adversarial domain adaptation. In: Proceedings of International Conference on International Conference on Machine Learning, 2018

14 Li H L, Pan S J, Wang S Q, et al. Domain generalization with adversarial feature learning. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018
15 Zhao S S, Gong M M, Liu T L, et al. Domain generalization via entropy regularization. In: Proceedings of Advances in Neural Information Processing Systems, 2020
16 Balaji Y, Sankaranarayanan S, Chellappa R. Metareg: towards domain generalization using meta-regularization. In: Proceedings of Advances in Neural Information Processing Systems, 2018
17 Dou Q, de Castro D C, Kamnitsas K, et al. Domain generalization via model-agnostic learning of semantic features. In: Proceedings of Advances in Neural Information Processing Systems, 2019
18 Li D, Yang Y X, Song Y Z, et al. Learning to generalize: meta-learning for domain generalization. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligenc, 2018
19 Li Y Y, Yang Y X, Zhou W, et al. Feature-critic networks for heterogeneous domain generalization. In: Proceedings of International Conference on Machine Learning, 2019
20 Zhou K Y, Yang Y X, Hospedales T M, et al. Deep domain-adversarial image generation for domain generalisation. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2020
21 Zhou K Y, Yang Y X, Qiao Y, et al. Domain generalization with mixstyle. In: Proceedings of International Conference on Learning Representations, 2021
22 Xu Q W, Zhang R P, Zhang Y, et al. A fourier-based framework for domain generalization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021
23 Carlucci F M, D'Innocente A, Bucci S, et al. Domain generalization by solving jigsaw puzzles. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019
24 Matsuura T, Harada T. Domain generalization using a mixture of multiple latent domains. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2020
25 Huang Z Y, Wang H H, Xing E P, et al. Self-challenging improves cross-domain generalization. In: Proceedings of European Conference on Computer Vision, 2020
26 Pandey P, Raman M, Varambally S, et al. Generalization on unseen domains via inference-time label-preserving target projections. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021
27 Ghifary M, Kleijn W B, Zhang M, et al. Domain generalization for object recognition with multi-task autoencoders. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2015
28 Piratla V, Netrapalli P, Sarawagi S. Efficient domain generalization via common-specific low-rank decomposition. In: Proceedings of International Conference on International Conference on Machine Learning, 2020
29 Meng R, Li X F, Chen W J, et al. Attention diversification for domain generalization. In: Proceedings of European Conference on Computer Vision, 2022
30 Du Y J, Xu J, Xiong H, et al. Learning to learn with variational information bottleneck for domain generalization. In: Proceedings of European Conference on Computer Vision, 2020
31 Du Y, Xu J, Xiong H, et al. Learning to optimize domain specific normalization for domain generalization. In: Proceedings of European Conference on Computer Vision, 2020
32 Li D, Zhang J S, Yang Y X, et al. Episodic training for domain generalization. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2019
33 Wang H, He Z, Lipton Z C, et al. Learning robust representations by projecting superficial statistics out. In: Proceedings of International Conference on Learning Representations, 2019
34 Qiao F C, Zhao L, Peng X. Learning to learn single domain generalization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020
35 Volpi R, Namkoong H, Sener O, et al. Generalizing to unseen domains via adversarial data augmentation. In: Proceedings of Advances in Neural Information Processing Systems, 2018
36 Xiao N, Zhang L. Dynamic weighted learning for unsupervised domain adaptation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021. 15242–15251
37 Shi W X, Zhang L, Chen W J, et al. Universal domain adaptive object detector. In: Proceedings of the 30th ACM International Conference on Multimedia, 2022. 2258–2266
38 He Z W, Zhang L, Yang Y, et al. Partial alignment for object detection in the wild. IEEE Trans Circ Syst Video Technol, 2022, 32: 5238–5251
39 Goodfellow I J, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. In: Proceedings of International Conference on Learning Representations, 2015
40 Nguyen A, Yosinski J, Clune J. Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015
41 Madry A, Makelov A, Schmidt L, et al. Towards deep learning models resistant to adversarial attacks. In: Proceedings of International Conference on Learning Representations, 2018
42 Wang H, Wang Y T, Zhou Z, et al. CosFace: large margin cosine loss for deep face recognition. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018
43 Fan Y, Tian F, Qin T, et al. Learning to teach. In: Proceedings of International Conference on Learning Representations, 2018
44 Finn C, Abbeel P, Levine S. Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of International Conference on International Conference on Machine Learning, 2017
45 Sun J, Tappen M F. Learning non-local range Markov random field for image restoration. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011
46 Ben-David S, Blitzer J, Crammer K, et al. Analysis of representations for domain adaptation. In: Proceedings of Advances in Neural Information Processing Systems, 2007
47 Ben-David S, Blitzer J, Crammer K, et al. A theory of learning from different domains. Mach Learn, 2010, 79: 151–175
48 Saito K, Kim D, Sclaroff S, et al. Semi-supervised domain adaptation via minimax entropy. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2019
49 Li D, Yang Y X, Song Y Z, et al. Deeper, broader and artier domain generalization. In: Proceedings of IEEE International Conference on Computer Vision (ICCV), 2017
50 Venkateswara H, Eusebio J, Chakraborty S, et al. Deep hashing network for unsupervised domain adaptation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017
51 Krizhevsky A, Hinton G. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009

52 D'Innocente A, Caputo B. Domain generalization with domain-specific aggregation modules. In: Proceedings of German Conference on Pattern Recognition, 2018
53 Zhou K, Yang Y, Hospedales T, et al. Learning to generate novel domains for domain generalization. In: Proceedings of European Conference on Computer Vision, 2020
54 Huang J X, Guan D Y, Xiao A R, et al. FSDR: frequency space domain randomization for domain generalization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021
55 Nam H, Lee H, Park J, et al. Reducing domain gap via style-agnostic networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021
56 Zhang Y B, Li M H, Li R H, et al. Exact feature distribution matching for arbitrary style transfer and domain generalization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022
57 Yao X F, Bai Y, Zhang X Y, et al. PCL: proxy-based contrastive learning for domain generalization. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022
58 Chen C Q, Tang L Y, Liu F, et al. Mix and reason: reasoning over semantic topology with data mixing for domain generalization. In: Proceedings of Advances in Neural Information Processing Systems, 2022
59 Wang X, Saxon M, Li J, et al. Causal balancing for domain generalization. In: Proceedings of International Conference on Learning Representations, 2023
60 Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. Int J Comput Vis, 2015, 115: 211–252
61 Gu X, Sun J, Xu Z B. Spherical space domain adaptation with robust pseudo-label loss. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020
62 Cubuk E D, Zoph B, Shlens J, et al. Randaugment: practical automated data augmentation with a reduced search space. In: Proceedings of Advances in Neural Information Processing Systems, 2020
63 Hendrycks D, Dietterich T. Benchmarking neural network robustness to common corruptions and perturbations. In: Proceedings of International Conference on Learning Representations, 2019
64 van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mach Learn Res, 2008, 9: 2579–2605