Decomposed Adversarial Domain Generalization

Sentao Chen[a]

[a]*Department of Computer Science, Shantou University, China*

ARTICLE INFO

Keywords:
Adversarial training
Domain generalization
f-divergence
Representation learning

ABSTRACT

We tackle the problem of generalizing a predictor trained on a set of source domains to an unseen target domain, where the source and target domains are different but related to one another, *i.e.*, the domain generalization problem. Prior adversarial methods rely on solving the minimax problems to align in the neural network embedding space the components of the domains (*i.e.*, a set of marginal distributions, a set of marginal distributions and multiple sets of class-conditional distributions). However, these methods introduce additional parameters (for each set of distributions) to the network predictor and are difficult to train. In this work, we propose to directly align the domains themselves via solving a minimax problem that can be decomposed and converted into a min one. Particularly, we analytically solve the max problem with respect to (w.r.t.) the domain discriminators, and convert the minimax problem into a min one w.r.t. the embedding function. This is more advantageous since in the end our approach introduces no additional network parameters and simplifies the training procedure. We evaluate our approach on several multi-domain datasets and testify its superiority over the relevant methods.

1. Introduction

Most supervised learning methods are constructed under the assumption that the training (source) and test (target) data are governed by the same domain $P(\boldsymbol{x}, y)$ of the feature variable $\boldsymbol{x}$ and the class label y [1]. In practice, however, control over the data generation process is often less perfect: the source and target data could be drawn from different domains, which may weaken the generalization ability of the resulting predictors, *e.g.*, neural networks. As an important problem in machine learning and computer vision, domain generalization [2–4] is exactly concerned with such non-identically-distributed supervised learning scenario. In this problem, the training data consist of n ($n \geq 2$) datasets drawn from n source domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$, respectively, while the test data are sampled from an unseen target domain $P^t(\boldsymbol{x}, y)$. The source and target domains are different but related to one another, and the goal of domain generalization is to train a predictor on the collection of the n source datasets and generalize it to the target domain.

To tackle the domain difference, one can align the source domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ in the neural network embedding space, whose samples are available during the training procedure. Based on the premise that the source and target domains are related, the unseen target domain $P^t(\boldsymbol{x}, y)$ is expected to become similar to the source ones in the embedding space. Consequently, the classifier trained in the embedding space will generalize well to the target domain [4, 5]. Since adversarial training [6, 7] has been proven effective in aligning probability distributions, it is widely applied to domain generalization. Prior adversarial domain generalization methods [8–14] propose to factorize a domain into the product of marginal distribution and posterior distribution, $P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(y|\boldsymbol{x})$, or the product of class-conditional distribution and prior distribution, $P(\boldsymbol{x}, y) = P(\boldsymbol{x}|y)P(y)$, and attempt to align the domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ via aligning their factorized components. These domain components are (1) a set of marginal distributions (marginals) $P^1(\boldsymbol{x}), \cdots, P^n(\boldsymbol{x})$, or (2) a set of marginals and multiple sets of class-conditional distributions (class-conditionals), *e.g.*, $P^1(\boldsymbol{x}|y), \cdots, P^n(\boldsymbol{x}|y)$ for $y \in \{1, \cdots, c\}$ where c is the number of classes. Specifically, they exploit adversarial training to align the multiple sets of distributions. For each set of distributions, this involves adding a discriminator subnetwork, which usually has several fully-connected layers with parameters, to the base network predictor, and solving a minimax problem of a log loss functional[1] (the adversarial loss). As proved in [9, 11], maximizing the functional with respect to (w.r.t.) the discriminator subnetworks approximates the Jensen-Shannon (JS) divergence (an instance of the f-divergences [17]) among distributions, while minimizing the divergence w.r.t. the embedding function aligns the distributions in the embedding space. However, these adversarial methods [8–14] may not align the domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$, since they only align the domain components, *i.e.*, the marginals $P^1(\boldsymbol{x}), \cdots, P^n(\boldsymbol{x})$, or the marginals and the class-conditionals (Note that generally we do not have $P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(\boldsymbol{x}|y)$). Additionally, as a consequence of domain factorization and adversarial training, these methods introduce multiple sets of additional parameters to the network predictor during training, making the training procedure difficult and converge slowly.

In this work, we propose to directly align the n domains themselves, *i.e.*, $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$, in the network embedding space without domain factorization, and conduct the alignment via solving a minimax problem that can be decomposed and converted into a min one. For clarity, we il-

*Corresponding author
sentaochenmail@gmail.com (S. Chen)
ORCID(s):

[1]According to [15, 16], a functional takes a function or multiple functions as its input(s) and outputs a scalar value. For instance, in adversarial domain generalization [9], the functional there takes an embedding function and multiple discriminator subnetworks as its inputs and outputs a loss value.

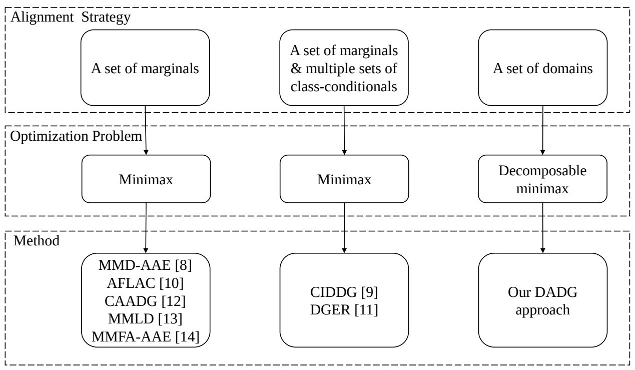


Figure 1: Comparison between our approach and the relevant domain generalization methods. From the perspective of the optimization problem, the minimax problems in previous methods [8–14] consist of maximizing the log loss functionals w.r.t. the discriminator subnetworks which have several fully-connected layers to approximate the JS divergence, and minimizing the JS divergence w.r.t. the embedding function to align the marginals or the marginals and the class-conditionals. By contrast, the decomposable minimax problem in our approach consists of maximizing the quadratic loss functional w.r.t. the linear-in-parameter domain discriminators to approximate the Pearson χ^2 divergence, and minimizing the Pearson χ^2 divergence w.r.t. the embedding function to align the domains. Thanks to the quadratic loss and the linear-in-parameter discriminators, in our minimax problem the max one can be analytically solved (like the problem in linear regression) and hence the minimax problem is decomposed.

lustrate in Figure 1 a brief comparison between our approach and the relevant domain generalization methods [8–14]. To be precise, our minimax problem aims at (1) maximizing a quadratic loss functional (quadratic adversarial loss) w.r.t. n domain discriminators to approximate the Pearson χ^2 divergence (another instance of the f-divergences) among $P^1(\boldsymbol{x}, y)$, $\cdots$, $P^n(\boldsymbol{x}, y)$, and (2) minimizing the divergence w.r.t. the embedding function to align these domains in the network embedding space. We design each domain discriminator as a linear-in-parameter function, which allows us to solve the max problem in an analytic manner and decompose the minimax problem. With other choices (*e.g.*, multi-layer neural network) for the domain discriminator, the decomposition may not be feasible. In addition, we train a classifier in the embedding space, and expect the network predictor containing the embedding function and the classifier to generalize well to the target domain. We name our approach Decomposed Adversarial Domain Generalization (DADG). Compared with the aforementioned adversarial methods, our approach contains no additional network parameters since in the end the max problem w.r.t the domain discriminators is analytically solved. Furthermore, our approach is easy to train and its training can be conveniently carried out by the minibatch Stochastic Gradient Descent (SGD) algorithm. To summarize, our major contributions are listed as follows:

- We propose to directly align domains in the network embedding space, rather than aligning their components (*i.e.*, the marginals, the class-conditionals). Such alignment could better address the domain difference in domain generalization.
- We solve a minimax problem to conduct domain alignment, and convert the problem into a minimization problem. Such conversion avoids going through the challenging adversarial training and simplifies the optimization/training procedure.
- We implement our DADG approach with both shallow and deep neural networks in the experiments. Our approach produces better experimental results than the relevant comparison methods on several multi-domain image classification datasets.

2. Related Work

Domain generalization was first studied by Blanchard *et al.* [2] as a variant of the multi-task learning problem. Since then, a lot of works (*e.g.*, [18–22]) have been conducted to address the problem, most of which are summarized in a recent survey by Zhou *et al.* [4]. Among them, a concentrate line of works [3, 8–11, 23–26] attempt to align domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ via separately aligning their components under certain metrics for domain generalization. Our research is inspired by these works and therefore we discuss them in the following.

Early works [23–25] generally try to align the n domains via (1) aligning a set of n marginal distributions (marginals) $P^1(\boldsymbol{x}), \cdots, P^n(\boldsymbol{x})$, or even a set of n marginals and c sets of n class-conditional distributions (class-conditionals) $P^1(\boldsymbol{x}|y)$, $\cdots, P^n(\boldsymbol{x}|y)$ for $y \in \{1, \cdots, c\}$ in a low dimensional subspace under the Maximum Mean Discrepancy (MMD) [27]. Muandet *et al.* [3] learned a projection matrix to align n marginals and preserved the functional relationship between input and output variables. Ghifary *et al.* [23] reduced the dimensionality of the data to align the marginals and maximized the separability of the classes and the separability of the unlabeled data. Later works [8, 9, 11] exploit the expressive Convolutional Neural Networks (CNNs) to align the domain components in the network embedding space under the JS divergence. Li *et al.* [8] aligned the distributions of the coded features, and simultaneously matched the aligned n marginals to a prior Laplacian distribution. Conditional Invariant Deep Domain Generalization (CIDDG) [9] learns an embedding function to align a set of n marginals and c sets of n class-conditionals. Similarly, Zhao *et al.* [11] learned the embedding function to align a set of n marginals and n sets of c class-conditionals. Essentially, these works approximate the JS divergence as the maximal value of a log loss functional w.r.t. the newly added discriminator subnetwork, which usually has several fully-connected layers with parameters. As a result, when minimizing the divergence (*i.e.*, the maximal value) w.r.t. the embedding function, they therefore lead to solving minimax games between the embedding function and the discriminator subnetworks.

Our DADG work is different from the above adversarial works in the following two aspects. (1) We directly align domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ themselves, rather than factorizing them and aligning their components (*e.g.*, the marginals, the class-conditionals), which may not truly align the domains. (2) Our minimax problem for domain alignment can be decomposed and converted into a min problem as the max one has an analytic solution that explicitly approximates the Pearson χ^2 divergence. Hence, in the end we are merely optimizing the parameters of the base network predictor via minibatch SGD, making the training procedure simple and straightforward. Beyond the f-divergence-based (adversarial) works for domain generalization, we note that the MMD-based works (*e.g.*, [28, 29]) for the related domain adaptation problem [30], also enjoy the same advantages as our DADG work in the sense that their methods introduce no additional network parameters and are easy to train via the minibatch SGD. In the experiments, we also include the comparisons with a representative MMD-based work Joint Adaptation Networks (JAN) [28] to reinforce the advantage of our DADG.

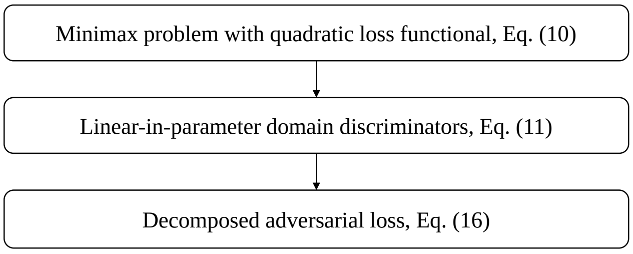
Of course, apart from domain alignment, domain generalization is also addressed via other strategies, including meta-learning [18, 31], data augmentation [20, 32], parameter decomposition [33, 34], and gradient surgery [21]. For example, Li *et al.* [18] provided a meta-learning framework for domain generalization, in which the parameters of the model are updated to minimize the loss over the meta-train and meta-test sets in a coordinated manner. Xu *et al.* [32] developed a Fourier-based data augmentation strategy to force the deep models to capture the Fourier phase information, which is assumed to contain high-level semantics not easily affected by the domain shift. Piratla *et al.* [34] decomposed the parameters of a neural network into a common component which is expected to generalize to the unseen target domain, and a low rank domain-specific component that overfits the source domains. Moreover, Mansilla *et al.* [21] characterized the conflicting gradients emerging in the domain shift scenarios, and devised novel gradient agreement strategies based on gradient surgery to enhance the generalization capability of deep learning models.

The above works exploit strategies complementary to ours to tackle the domain generalization problem. In the experiments, we compare our work with some of them for completeness.

3. Approach

Let $\mathcal{X}$ be an input feature space and $\mathcal{Y} = \{1, \cdots, c\}$ be an output class label space. In domain generalization, we have n $(n \geq 2)$ source domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$, which are reflected by the associated datasets $\mathcal{D}^1 = \{(\boldsymbol{x}_i^1, y_i^1)\}_{i=1}^{m_1}$, $\cdots$, $\mathcal{D}^n = \{(\boldsymbol{x}_i^n, y_i^n)\}_{i=1}^{m_n}$, and a target domain $P^t(\boldsymbol{x}, y)$, whose samples are not available during training. The source and target domains are different but related to each other. Given the n source datasets, the goal is to learn a predictor $h : \mathcal{X} \rightarrow \mathcal{Y}$ that well predicts the target data labels.

We model the predictor h as a neural network and write $h = g \circ \phi$, where $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ is the embedding function generating the embedding space $\mathcal{Z}$ and $g : \mathcal{Z} \rightarrow \mathcal{Y}$ is the classifier. We propose to directly align domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ in the embedding space, which are respectively reflected by the n training datasets $\mathcal{D}^1, \cdots, \mathcal{D}^n$, and train a classifier in this space via minimizing the classification loss. We expect the resulting network predictor containing the embedding function and the classifier to perform well in the target domain.



Minimax problem with quadratic loss functional, Eq. (10)
Linear-in-parameter domain discriminators, Eq. (11)
Decomposed adversarial loss, Eq. (16)

Figure 2: Crucial steps in our adversarial loss decomposition. (1) We align n domains under the Pearson χ^2 divergence, leading to a minimax problem with quadratic loss functional. (2) We design the domain discriminators in the quadratic loss functional as linear-in-parameter functions. (3) We solve the max problem of the minimax in an analytic manner and obtain the decomposed adversarial loss.

In the following two subsections, we first elaborate on our decomposed adversarial loss for domain alignment, and then present the optimization problem for learning the network predictor.

3.1. Decomposed Adversarial Loss

Before diving into the details of this section, we first depict in Figure 2 the crucial steps in our adversarial loss decomposition.

We measure the discrepancy among multiple distributions via computing the average of the discrepancies between each distribution and the average distribution following [3, 9], and employ the Pearson χ^2 divergence as the discrepancy metric between two distributions, which is an instance of the f-divergences [17] with the quadratic generator function $f(u) = u^2 - 1$. Note that, we exploit the Pearson χ^2 divergence due to the following two considerations. (1) Like other divergences (*e.g.*, JS divergence), the Pearson χ^2 divergence can also characterize the discrepancy between distributions, and has been demonstrated to be advantageous in several machine learning problems [35–37]. (2) More importantly, the Pearson χ^2 divergence is generated by the quadratic function. This quadratic function, as we will show shortly, will lead to a minimax problem that can be decomposed via analytically solving the max problem. With other divergences, the resulting minimax problems may not be decomposable. These considerations motivate and encourage us to use the Pearson χ^2 divergence, rather than other divergences. Using the Pearson χ^2 divergence, we define the discrepancy among the n domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ in the embedding space as

$$
\begin{aligned}
&D_{\chi^2}(P^1,\cdots,P^n;\phi)\\
&=\frac{1}{n}\sum_{s=1}^{n}\chi^2\big(P^s(\phi(\boldsymbol{x}),y)\|\bar{P}(\phi(\boldsymbol{x}),y)\big) &(1)\\
&=\frac{1}{n}\sum_{s=1}^{n}\int f\Big(\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}\Big)\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy &(2)\\
&=\frac{1}{n}\sum_{s=1}^{n}\int\Big[\Big(\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}\Big)^2-1\Big]\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy, &(3)
\end{aligned}
$$

where $\bar{P}(\phi(\boldsymbol{x}),y)=\frac{1}{n}\sum_{s=1}^{n}P^s(\phi(\boldsymbol{x}),y)$ is the average distribution. The domain discrepancy $D_{\chi^2}(P^1,\cdots,P^n;\phi)$ is non-negative and equal to 0 if and only if $P^1(\phi(\boldsymbol{x}),y)=\cdots=P^n(\phi(\boldsymbol{x}),y)$. Since the ratio function $\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}$ in the above equations is unknown, it is difficult to minimize the domain discrepancy w.r.t to the embedding function ϕ to align the domains. To solve this problem, we therefore propose to equivalently express the domain discrepancy in another form. In particular, based on the equation $f(u)=u^2-1=\max_v(2uv-v^2-1)$, we first write

$$
\begin{aligned}
&\chi^2\big(P^s(\phi(\boldsymbol{x}),y)\|\bar{P}(\phi(\boldsymbol{x}),y)\big)\\
&=\int f\Big(\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}\Big)\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy &(4)\\
&=\max_{r^s}\int\Big(2\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}r^s(\phi(\boldsymbol{x}),y)-r^s(\phi(\boldsymbol{x}),y)^2-1\Big)\\
&\qquad\times\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy &(5)\\
&=\max_{r^s}\Big(2\int r^s(\phi(\boldsymbol{x}),y)P^s(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy\\
&\qquad-\int r^s(\phi(\boldsymbol{x}),y)^2\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy\Big)-1. &(6)
\end{aligned}
$$

In Eq. (5), $\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}$ is regarded as u and $r^s(\phi(\boldsymbol{x}),y)$ is regarded as v. In Eq. (6), the maximal value is attained at $r^s(\phi(\boldsymbol{x}),y)=\frac{P^s(\phi(\boldsymbol{x}),y)}{\bar{P}(\phi(\boldsymbol{x}),y)}$. Inspired by the terminology in adversarial works [9, 11], below we call $r^s(\phi(\boldsymbol{x}),y)$ the domain discriminator. Then, plugging Eq. (6) into Eq. (1), the domain discrepancy is equivalently expressed in the form of the following maximal value

$$
\begin{aligned}
&D_{\chi^2}(P^1,\cdots,P^n;\phi)\\
&=\frac{1}{n}\sum_{s=1}^{n}\max_{r^s}\Big(2\int r^s(\phi(\boldsymbol{x}),y)P^s(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy\\
&\qquad-\int r^s(\phi(\boldsymbol{x}),y)^2\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy\Big)-1 &(7)\\
&=\max_{r^1,\cdots,r^n}\frac{1}{n}\sum_{s=1}^{n}\Big(2\int r^s(\phi(\boldsymbol{x}),y)P^s(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy\\
&\qquad-\int r^s(\phi(\boldsymbol{x}),y)^2\bar{P}(\phi(\boldsymbol{x}),y)d\phi(\boldsymbol{x})dy\Big)-1. &(8)
\end{aligned}
$$

We use $V(\phi,r^1,\cdots,r^n)$ to represent the quadratic loss functional in Eq. (8). Now, we can align the n domains $P^1(\boldsymbol{x},y),\cdots,P^n(\boldsymbol{x},y)$ via minimizing their discrepancy $D_{\chi^2}(P^1,\cdots,P^n;\phi)$ in the embedding space, leading to the following minimax game between the embedding function ϕ and the domain discriminators $r^1,\cdots,r^n$:

$$\min_{\phi}\max_{r^1,\cdots,r^n}V(\phi,r^1,\cdots,r^n). \tag{9}$$

Since the quadratic (adversarial) loss functional $V(\phi,r^1,\cdots,r^n)$ contains unknown expectations w.r.t. $P^s(\phi(\boldsymbol{x}),y)$ $(1\le s\le n)$ and $\bar{P}(\phi(\boldsymbol{x}),y)$, we approximate these expectations by their empirical averages and solve a minimax problem of the empirical functional $\widehat{V}(\phi,r^1,\cdots,r^n)$:

$$
\begin{aligned}
&\min_{\phi}\max_{r^1,\cdots,r^n}\widehat{V}(\phi,r^1,\cdots,r^n)\\
&\quad=\frac{1}{n}\sum_{s=1}^{n}\Big(\frac{2}{m_s}\sum_{i=1}^{m_s}r^s(\phi(\boldsymbol{x}_i^s),y_i^s)\\
&\qquad-\frac{1}{n}\sum_{s'=1}^{n}\frac{1}{m_{s'}}\sum_{i=1}^{m_{s'}}r^s(\phi(\boldsymbol{x}_i^{s'}),y_i^{s'})^2\Big)-1. &(10)
\end{aligned}
$$

In the following, we convert minimax problem (10) into a min one. To this end, we design each domain discriminator as a linear-in-parameter function. That is,

$$r^s(\phi(\boldsymbol{x}),y;\boldsymbol{\alpha}^s)=\sum_{i=1}^{m}\alpha_i^s p\big((\phi(\boldsymbol{x}),y),(\phi(\boldsymbol{x}_i),y_i)\big), \tag{11}$$

where $\boldsymbol{\alpha}^s=(\alpha_1^s,\cdots,\alpha_m^s)^\top$ are the function parameters, and $p\big((\phi(\boldsymbol{x}),y),(\phi(\boldsymbol{x}_i),y_i)\big)=k(\phi(\boldsymbol{x}),\phi(\boldsymbol{x}_i))\delta(y,y_i)$ is the product kernel function. $k(\phi(\boldsymbol{x}),\phi(\boldsymbol{x}_i))=\exp\big(-\frac{\|\phi(x)-\phi(x_i)\|^2}{\sigma}\big)$ is the Gaussian kernel with kernel width $\sigma>0$, and $\delta(y,y_i)$ is the delta kernel that evaluates 1 if $y=y_i$ and 0 otherwise. Besides, $\{(\phi(\boldsymbol{x}_1),y_1),\cdots,(\phi(\boldsymbol{x}_m),y_m)\}=\{(\phi(\boldsymbol{x}_1^1),y_1^1),\cdots,(\phi(\boldsymbol{x}_{m_n}^n),y_{m_n}^n)\}$ are the $m=m_1+\cdots+m_n$ function centers. As aforementioned in Section 1, we use the linear-in-parameter functions for modeling the domain discriminators, since the functions enable us to decompose minimax problem (10). With the linear-in-parameter domain discriminators, we decompose and convert minimax problem (10) as follows.

$$
\begin{aligned}
&\min_{\phi}\max_{\boldsymbol{\alpha}^1,\cdots,\boldsymbol{\alpha}^n}\widehat{V}(\phi,\boldsymbol{\alpha}^1,\cdots,\boldsymbol{\alpha}^n)\\
&=\min_{\phi}\max_{\boldsymbol{\alpha}^1,\cdots,\boldsymbol{\alpha}^n}\frac{1}{n}\sum_{s=1}^{n}\Big(\frac{2}{m_s}\sum_{i=1}^{m_s}r^s(\phi(\boldsymbol{x}_i^s),y_i^s;\boldsymbol{\alpha}^s)\\
&\qquad-\frac{1}{n}\sum_{s'=1}^{n}\frac{1}{m_{s'}}\sum_{i=1}^{m_{s'}}r^s(\phi(\boldsymbol{x}_i^{s'}),y_i^{s'};\boldsymbol{\alpha}^s)^2\Big)-1 &(12)\\
&=\min_{\phi}\max_{\boldsymbol{\alpha}^1,\cdots,\boldsymbol{\alpha}^n}\frac{1}{n}\sum_{s=1}^{n}\Big(\frac{2}{m_s}\boldsymbol{1}^\top\boldsymbol{P}^s\boldsymbol{\alpha}^s\\
&\qquad-\frac{1}{n}\sum_{s'=1}^{n}\frac{1}{m_{s'}}(\boldsymbol{\alpha}^s)^\top(\boldsymbol{P}^{s'})^\top\boldsymbol{P}^{s'}\boldsymbol{\alpha}^s\Big)-1 &(13)\\
&=\min_{\phi}\max_{\boldsymbol{\alpha}^1,\cdots,\boldsymbol{\alpha}^n}\frac{1}{n}\sum_{s=1}^{n}\Big(2(\boldsymbol{b}^s)^\top\boldsymbol{\alpha}^s-(\boldsymbol{\alpha}^s)^\top\boldsymbol{H}\boldsymbol{\alpha}^s\Big)-1 &(14)
\end{aligned}
$$

$$= \min_{\phi} \frac{1}{n} \sum_{s=1}^{n} \left(2(\boldsymbol{b}^s)^\top \widetilde{\boldsymbol{\alpha}}^s - (\widetilde{\boldsymbol{\alpha}}^s)^\top \boldsymbol{H} \widetilde{\boldsymbol{\alpha}}^s \right) - 1 \tag{15}$$

$$= \min_{\phi} \widehat{D}_{\chi^2}(P^1, \cdots, P^n; \phi). \tag{16}$$

Eq. (12) is obtained by plugging the domain discriminators $r^s(\phi(\boldsymbol{x}), y; \boldsymbol{\alpha}^s)$ $(1 \leq s \leq n)$ into Eq. (10). Eq. (13) equivalently introduces matrix and vector notations, where $\mathbf{1}$ is an m_s-dimensional column vector of ones and $\boldsymbol{P}^s, \boldsymbol{P}^{s'} \in \mathbb{R}^{m_s \times m}$. The (i,j)-th elements of $\boldsymbol{P}^s$ and $\boldsymbol{P}^{s'}$ are respectively defined as $p_{ij}^s = p\big((\phi(\boldsymbol{x}_i^s), y_i^s), (\phi(\boldsymbol{x}_j), y_j)\big)$ and $p_{ij}^{s'} = p\big((\phi(\boldsymbol{x}_i^{s'}), y_i^{s'}), (\phi(\boldsymbol{x}_j), y_j)\big)$. Eq. (14) introduces notations $\boldsymbol{b}^s = \frac{1}{m_s}(\boldsymbol{P}^s)^\top \mathbf{1}$ and $\boldsymbol{H} = \frac{1}{n}\sum_{s'=1}^{n} \frac{1}{m_{s'}}(\boldsymbol{P}^{s'})^\top \boldsymbol{P}^{s'}$ to make the unconstrained quadratic optimization problem become explicit. Eq. (15) analytically solves the max problem with $\widetilde{\boldsymbol{\alpha}}^s = (\boldsymbol{H} + \epsilon \mathbf{I})^{-1} \boldsymbol{b}^s$, where a diagonal matrix $\epsilon \mathbf{I}$ is added to $\boldsymbol{H}$ to ensure that the matrix inversion is always numerically feasible in practice. Here, ϵ is a small positive value and $\mathbf{I}$ is the identity matrix. Finally, Eq. (16) denotes the resulting maximal value as $\widehat{D}_{\chi^2}(P^1, \cdots, P^n; \phi)$, which is an empirical estimate of the Pearson χ^2 divergence among domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ in the embedding space. We call $\widehat{D}_{\chi^2}(P^1, \cdots, P^n; \phi)$ the decomposed adversarial loss.

Remark 1. *We again note that the Pearson χ^2 divergence and the linear-in-parameter domain discriminators are of paramount importance to our decomposition. In particular, these two ingredients enable us to decompose and convert the multi-player minimax problem since the max one is an unconstrained quadratic optimization problem that can be analytically solved. With other f-divergences or other discriminators, this may not be feasible.*

Remark 2. *It is well-known that in every iteration of training a neural network, the loss of the network is usually calculated using a minibatch of samples, rather than all the samples. Here, our decomposed adversarial loss,* i.e., *Eq. (16), is also calculated in this way in every iteration of the SGD algorithm.*

3.2. Optimization Problem

We jointly minimize the classification loss and the decomposed adversarial loss, and present the optimization problem of our Decomposed Adversarial Domain Generalization (DADG) approach as follows

$$\min_{\phi, g} \frac{1}{n} \sum_{s=1}^{n} \frac{1}{m_s} \sum_{i=1}^{m_s} \ell(g(\phi(\boldsymbol{x}_i^s)), y_i^s) + \lambda \widehat{D}_{\chi^2}(P^1, \cdots, P^n; \phi). \tag{17}$$

Here, ℓ is the cross-entropy loss and λ (> 0) is a tradeoff parameter for balancing the two losses. Clearly, in the end we are optimizing the base network predictor containing the embedding function ϕ and the classifier g. For clarity, we illustrate our approach in Figure 3.

We employ minibatch SGD to solve problem (17). In every iteration of the algorithm, a minibatch consists of n mini-batches respectively sampled from the n source datasets, and the objective in (17) is calculated using these minibatches.

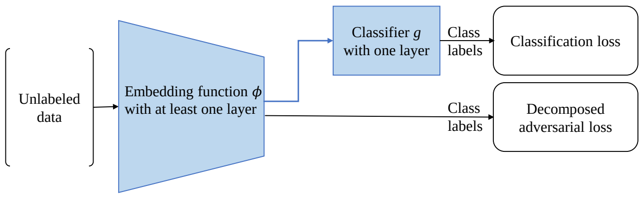

Unlabeled data
Embedding function ϕ with at least one layer
Classifier g with one layer
Class labels
Classification loss
Class labels
Decomposed adversarial loss

Figure 3: Illustration of our DADG approach with a network predictor $h = g \circ \phi$. We optimize the parameters of the predictor to jointly minimize the classification loss and the decomposed adversarial loss (*i.e.*, domain discrepancy).

Remark 3. *From the perspective of computational complexity, we compare our approach with the adversarial methods MMD-AAE [8] and DGER [11], with a focus on the distribution alignment part. In particular, the computational complexity of DADG is $\mathcal{O}(T(n^3b^3 + n^2b^2))$, where T is the number of iterations, n is the number of source domains, and b is the batch size of each domain. The complexities of MMD-AAE and DGER are $\mathcal{O}(T'(2n^2b^2 + nb\sum_{i=1}^{L} L_{i-1}L_i))$ and $\mathcal{O}(T'(nb(c+n+1)\sum_{i=1}^{L} L_{i-1}L_i))$, where T' is the number of iterations, L is the number of fully connected layers in the discriminator subnetwork, and L_i is the number of neurons in the i-th fully connected layer. For the base network predictor being a deep network (*e.g.*, ResNet50 [38]), the adversarial methods usually contain a large number of neurons in the subnetworks and take more iterations to converge [39]. Therefore, our approach is more advantageous in computational complexity than the adversarial methods MMD-AAE and DGER.*

4. Experiments

4.1. Datasets

We evaluate our DADG approach on four multi-domain image classification datasets. (1) **PIE** [40] contains 11,554 gray-scale face images from 5 domains: C05 (left pose), C07 (upward pose), C09 (downward pose), C27 (frontal pose), and C29 (right pose), and each domain has 68 classes. See Figure 4(a) for example images. (2) **VLCS** [41] includes 10,729 images from 4 different domains: VOC2007 (V), LabelMe (L), Caltech-101 (C), and SUN09 (S), and each domain has 5 classes. See Figure 4(b) for example images. (3) **PACS** [33] contains 9,991 images from 4 domains: Art-Painting (A), Cartoon (C), Photo (P), and Sketch (S), and each domain has 7 classes. See Figure 4(c) for example images. (4) **Office-Home** [42] has around 15,500 images of everyday objects organized into 4 domains: Art (A), Clipart (C), Product (P), and RealWorld (R), and each domain has 65 classes. See Figure 4(d) for example images.

4.2. Comparison Methods

We compare our approach against a baseline method and the domain generalization methods. The baseline (Baseline)

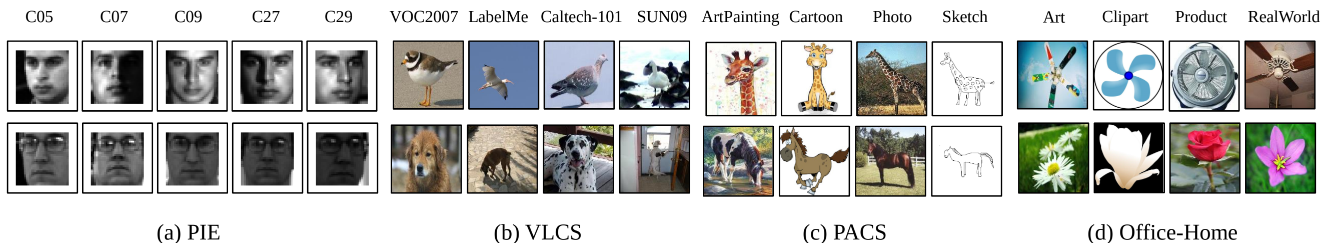


Figure 4: Example images from four multi-domain datasets: PIE [40], VLCS [41], PACS [33], and Office-Home [42].

method trains vanilla neural network classifiers via minimizing the cross-entropy loss on the source data, which is also referred to as DeepAll in some prior works [11, 31, 32, 43]. The domain generalization methods include (1) the ones that perform distribution alignment (Alignment) for domain generalization, and (2) the ones that exploit other strategies (Others) for domain generalization, *e.g.*, meta-learning, data augmentation, parameter decomposition, and gradient surgery. In particular, the former domain generalization methods contain the relevant adversarial methods: Domain-Adversarial Neural Network (DANN) [7], MMD-based Adversarial AutoEncoder (MMD-AAE) [8], CIDDG [9], Domain Generalization via Entropy Regularization (DGER) [11], and Domain Invariant Representation learning with Transformations via Generative Adversarial Networks (DIRT-GAN) [26]. As discussed in the related work, we also include the MMD-based non-adversarial JAN [28] as a comparison method to reinforce the advantage of our DADG approach. Since JAN is originally designed for domain adaptation, we adjust it such that it applies to domain generalization. The latter domain generalization methods are briefly described below. Denoising Multi-Task AutoEncoders (D-MTAE) [41] learns domain invariant features by domain reconstruction; Classification and Contrastive Semantic Alignment (CCSA) [44] uses semantic alignment to regularize the learned feature space; Cross-Gradient (CrossGrad) [45] uses a gradient-based domain perturbation strategy to perturb the input data; Deeper, Broader and Artier Domain Generalization (DBADG) [33] learns a domain-agnostic model and uses tensor factorization to compress model parameters; Meta-Learning for Domain Generalization (MLDG) [18] learns the parameters of the classification model in a meta-learning framework; Meta-Regularization (MetaReg) [19] learns the classifier regularizer in a meta-learning framework; Jigsaw puzzle based Generalization (JiGen) [43] learns the semantic labels by learning from self-supervised signals; Episodic with Feature regularization, Classifier regularization, and the Random classifier regularization (Epi-FCR) [46] develops an episodic training strategy that mimics the train-test domain shift during the training procedure; Model Agnostic learning of Semantic Features (MASF) [31] introduces a model-agnostic episodic learning procedure to regularize the semantic structure of the feature space; Learning to Augment by Optimal Transport (L2A-OT) [20] augments the source datasets with an optimal transport based formulation and trains the classification model using the augmented data; EISNet [47] learns to generalize across domains from extrinsic relationship supervision and intrinsic self-supervision for images from multiple domains; Mixture of Multiple Latent Domains (MMLD) [13] trains the domain-invariant feature extractor shared among the divided latent domains via adversarial learning; Deep Domain-Adversarial Image Generation (DDAIG) [48] makes the classifier robust to unknown domain changes by augmenting the training data. Fourier Augmented Co-Teacher (FACT) [32] develops a Fourier-based data augmentation strategy to capture the high-level semantics. Domain Generalization via Gradient Surgery (DGGS) [21] conducts gradient surgery to improve the performance of deep learning models.

Table 1
Configuration of the backbones for the datasets. 1HLNN represents one-Hidden-Layer Neural Network.

Dataset	PIE	VLCS	PACS	Office-Home
Backbone	1HLNN	2HLNN & AlexNet	ResNet18 & ResNet50	ResNet18 & ResNet50

4.3. Evaluation Protocol

Following the leave-one-domain-out evaluation protocol in [11, 20, 26, 32], we employ network classifiers trained on the source datasets to predict the labels of samples from the remaining target set. The performance of a trained classifier is measured by its target classification accuracy (%). On every domain generalization task, we follow [11, 20, 26, 32] and repeat the experiments 5 times with different random seeds, and report the average classification accuracy.

4.4. Implementation Details

We implement our approach using both shallow and deep neural network predictors (backbones), and present in Table 1 an overview of the configuration of the backbones for the datasets. The Pytorch implementation of our approach will be publicly available upon publication of the paper.

On the PIE dataset with the 1024-dimensional gray-scale image pixel features [40], the backbone is a one-Hidden-Layer Neural Network (1HLNN), which has 512 hidden neurons with the ReLU activation, and 68 output neurons with the softmax transformation. By such design, the number of hidden neurons for the network is half of the number of its input neurons. **On the VLCS dataset**, following previous works [11, 41, 43, 46], we randomly divide each domain dataset into a training set (70%) and a test set (30%), and evaluate on the test set of the held-out target domain. We follow the work of Zhao *et al*. [11] and use two backbones for this dataset: a two-Hidden-Layer Neural Network (2HLNN) and the AlexNet [49]. The 2HLNN takes as its input the 4096-dimensional DeCAF6 features, and has 1024 and 128 neurons for its two hidden layers, both with the ReLU activation, and 5 output neurons. The end-to-end AlexNet is reconstructed to ensure that its final classification layer has 5 output neurons, *i.e.*, the number of classes in VLCS. **On the PACS dataset**, following [11, 26, 32, 33], we split the data from the source domains to 9 (train) : 1 (val) and test on the whole held-out target domain. We use the ResNet18 and ResNet50 backbones [38] for this dataset, where their final classification layers are both reconstructed to have 7 output neurons. **On the Office-Home dataset**, we follow the 9:1 training and validation data splitting protocol in [32, 43] and use the ResNet18 and ResNet50 backbones for this dataset. Finally, for datasets with the deep backbone configuration, we follow the standard practice in [26, 32, 43] and process the images via random resized cropping, horizontal flipping, and color jittering.

For training the networks, our approach with its shallow implementations on PIE and VLCS is trained from scratch by the minibatch SGD with a momentum of 0.9 and a learning rate of 10^{-3}. The tradeoff parameter λ is selected from the range $\{10^{-2}, 10^{-1}, \cdots, 10^{3}\}$ via validation on the training data. Following prior practice [29, 50], we set the Gaussian kernel width σ in the domain discriminators to the median pairwise squared distances on the training data. Moreover, our approach with its deep implementations on VLCS, PACS, and Office-Home is trained from the ImageNet pretrained models. The optimizer is still the minibatch SGD and the learning rate is initially set to 10^{-3} and shrunk to 10^{-4} after 30 iterations. This time, the tradeoff parameter λ is not selected through a grid search as in the experiments with shallow backbones, since the corresponding procedure would be computationally costly. Instead, following [7], we gradually change λ from 0 to 1 by a progressive schedule: $\lambda_t = \frac{2}{1+\exp(-10t)} - 1$, where t is the training progress linearly changing from 0 to 1. As aforementioned in Section 3.2, for both the experiments with shallow and deep backbones, a minibatch in every iteration of the SGD algorithm consists of n minibatches of the same size, which are respectively sampled from the n source datasets.

4.5. Results

We report in Table 2 the classification results on PIE, in Table 3 and Table 4 the results on VLCS, in Table 5 and Table 6 the results on PACS, and in Table 7 and Table 8 the results on Office-Home. Note that, since our experimental settings coincide with prior works [11, 20, 26, 32], we follow the common practice in them and quote the available results of the comparison methods in Table 3 and Table 4 from [11, 21, 26], the results in Table 5 and Table 6 from [11, 20, 26, 32], and the results in Table 7 from [32]. Since some of the comparison methods only have results on some of the datasets, on each dataset the comparison methods could be different. For more extensive comparison with the relevant adversarial methods, we further use the source codes of MMD-AAE [8], CIDDG [9], and DGER [11] to produce their results on datasets PIE and Office-Home. In each table, the names of the source domains are omitted under the leave-one-domain-out evaluation protocol. For every column in the table, the best result is highlighted in bold.

On dataset PIE, we observe that our DADG approach with its shallow implementation consistently outperforms its competitors on all the 5 tasks. The average performance improvement over the second best method CIDDG is more than 2.00%, *i.e.*, 87.25% versus 85.02%. We attribute the outperformance over the relevant adversarial methods (*i.e.*, MMD-AAE, CIDDG, and DGER) to the direct alignment of domains with the decomposed adversarial loss, which bet-

Table 2
Classification accuracy (%) of the distribution-alignment-based (Alignment) methods with the 1HLNN backbone on dataset PIE. The Baseline is also referred to as DeepAll, and the best result is highlighted in bold for each column.

Alignment	Method	C05	C07	C09	C27	C29	Avg
-	Baseline	79.93	78.49	86.61	97.55	69.46	82.41
MMD	JAN [28]	82.37	80.75	87.20	97.90	72.67	84.18
Adversarial	MMD-AAE [8]	81.94	79.19	86.93	97.81	71.51	83.48
	CIDDG [9]	83.67	82.35	86.90	98.17	74.00	85.02
	DGER [11]	82.92	80.91	86.16	97.68	73.47	84.23
Our	DADG	**85.25**	**87.56**	**88.06**	**98.35**	**77.05**	**87.25**

Table 3
Classification accuracy (%) of the Alignment and Other methods with the 2HLNN backbone on dataset VLCS.

Strategy	Method	V	L	C	S	Avg
-	Baseline	70.07	60.54	93.83	65.95	72.60
Others	D-MTAE [41]	63.90	60.13	89.05	61.33	68.60
	DBADG [33]	65.58	58.74	92.43	61.85	69.65
	CCSA [44]	67.10	62.10	92.30	59.10	70.15
	MetaReg [19]	65.00	60.20	92.30	64.20	70.43
	CrossGrad [45]	65.50	60.00	92.00	64.70	70.55
	MLDG [18]	67.70	61.30	94.40	65.90	72.33
	Epi-FCR [46]	67.10	**64.30**	94.10	65.90	72.85
Alignment	DANN [7]	66.40	64.00	92.60	63.60	71.65
	MMD-AAE [8]	67.70	62.60	94.40	64.40	72.28
	DGER [11]	70.54	60.81	94.44	66.11	72.98
	DADG (ours)	**71.15**	63.55	**94.75**	**67.85**	**74.33**

Table 4
Classification accuracy (%) of the Alignment and Other methods with the AlexNet backbone on dataset VLCS.

Strategy	Method	V	L	C	S	Avg
-	Baseline	73.11	58.07	97.15	68.79	74.28
Others	DBADG [33]	69.99	63.49	93.64	61.32	72.11
	JiGen [43]	70.62	60.90	96.93	64.30	73.19
	MMLD [13]	71.96	58.77	96.66	68.13	73.88
	DGGS [21]	68.14	58.56	93.23	63.89	70.96
Alignment	CIDDG [9]	73.00	58.30	97.02	68.89	74.30
	DGER [11]	73.24	58.26	96.92	69.10	74.38
	DIRT-GAN [26]	72.10	**64.00**	97.30	**72.20**	76.40
	DADG (ours)	**74.15**	63.41	**98.55**	69.79	**76.48**

Table 5
Classification accuracy (%) of the Alignment and Other methods with the ResNet18 backbone on dataset PACS.

Strategy	Method	A	C	P	S	Avg
-	Baseline	77.63	76.77	95.85	69.50	79.94
Others	MetaReg [19]	83.70	77.20	95.50	70.30	81.68
	JiGen [43]	79.42	75.25	96.03	71.35	80.51
	Epi-FCR [46]	82.10	77.00	93.90	73.00	81.50
	MASF [31]	80.29	77.17	94.99	71.69	81.04
	MMLD [13]	81.28	77.16	96.09	72.29	81.71
	DDAIG [48]	84.20	78.10	95.30	74.70	83.08
	EISNet [47]	81.89	76.44	95.93	74.33	82.15
	L2A-OT [20]	83.30	78.20	96.20	73.60	82.83
	FACT [32]	85.37	78.38	95.15	79.15	84.51
Alignment	MMD-AAE [8]	75.20	72.70	96.00	64.20	77.03
	DGER [11]	80.70	76.40	96.65	71.77	81.38
	DIRT-GAN [26]	82.56	76.37	95.65	**79.89**	83.62
	DADG (ours)	**85.60**	**79.15**	**97.53**	78.45	**85.18**

Table 6
Classification accuracy (%) of the Alignment and Other methods with the ResNet50 backbone on dataset PACS.

Strategy	Method	A	C	P	S	Avg
-	Baseline	84.94	76.98	97.64	76.75	84.08
Others	MetaReg [19]	87.20	79.20	97.60	70.30	83.58
	MASF [31]	82.89	80.49	95.01	72.29	82.67
	EISNet [47]	86.64	81.53	97.11	78.07	85.84
	FACT [32]	89.63	81.77	96.75	**84.46**	**88.15**
Alignment	DGER [11]	87.51	79.31	98.25	76.30	85.34
	DADG (ours)	**90.15**	**82.57**	**98.95**	80.50	88.04

Table 7
Classification accuracy (%) of the Alignment and Other methods with the ResNet18 backbone on dataset Office-Home.

Strategy	Method	A	C	P	R	Avg
-	Baseline	57.88	52.72	73.50	74.80	64.73
Others	CCSA [44]	59.90	49.90	74.10	75.70	64.90
	CrossGrad [45]	58.40	49.40	73.90	75.80	64.38
	JiGen [43]	53.04	47.51	71.47	72.79	61.20
	DDAIG [48]	59.20	52.30	74.60	76.00	65.53
	L2A-OT [20]	60.60	50.10	74.80	77.00	65.63
	FACT [32]	60.34	**54.85**	74.48	76.55	66.56
Alignment	MMD-AAE [8]	59.27	51.05	74.02	75.32	64.92
	CIDDG [9]	59.94	52.27	74.31	76.83	65.84
	DGER [11]	60.21	51.29	73.90	76.42	65.46
	DADG (ours)	**62.03**	54.43	**75.23**	**77.23**	**67.23**

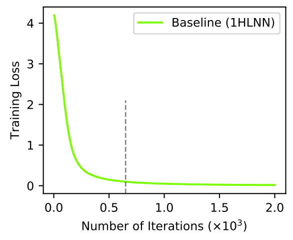
ter reduces the domain difference than the alignment of domain components (*i.e.*, the marginal distributions, the class-conditional distributions) with adversarial training. Consequently, with smaller domain gap our approach produces better classification results than its adversarial competitors. Furthermore, the outperformance of our DADG approach over the non-adversarial JAN [28] further reinforces its advantage and strength in aligning domains for better generalization ability. On dataset VLCS, our approach again yields superior results and achieves the best average performance (*i.e.*, 74.33% and 76.48%) with both its shallow and deep implementations, compared with not only the methods that are based on distribution alignment (Alignment), but also the

Table 8
Classification accuracy (%) of the Alignment methods with the ResNet50 backbone on dataset Office-Home.

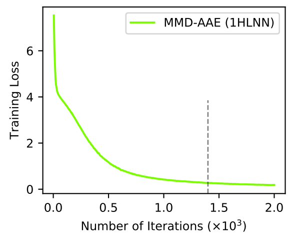
Strategy	Method	A	C	P	R	Avg
-	Baseline	65.75	58.32	78.44	80.12	70.66
Alignment	MMD-AAE [8]	68.71	56.78	79.32	80.53	71.34
	CIDDG [9]	69.37	58.24	80.14	81.30	72.26
	DGER [11]	68.72	56.87	79.51	81.33	71.61
	DADG (ours)	**70.63**	**60.43**	**81.08**	**81.65**	**73.44**

ones that are based on other strategies (Others), *e.g.*, meta-learning, parameter decomposition, and gradient surgery. In particular, our approach outperforms the meta-learning methods (MetaReg, MLDG) in Table 3, since these methods do not align the domains to reduce the domain difference, and since on dataset VLCS, the simulated domain shifts from these methods probably do not reflect the real target domains. On datasets PACS and Office-Home, our DADG continues to produce better results than its competitors on most of the tasks with both the ResNet18 and ResNet50 backbones. The outperformance over MMD-AAE, CIDDG, and DGER again demonstrates the advantage of our approach over these methods. Besides, our approach outperforms the recent popular method DDAIG on both datasets in Table 5 and Table 7. We conjecture that on these two datasets, aligning domains is more effective than adding perturbation to the input images to represent the domain variations, which is performed in DDAIG. Of course, we also notice that in Table 6, while our approach outperforms the method FACT on the first three tasks, it does not outperform FACT on the remaining task where the target domain is Sketch (S). This is because the target domain Sketch is quite different from the three source domains (see Figure 4(c)). As a result, for our approach it may be difficult to significantly reduce the gap between the source and target domains in the embedding space by aligning the source domains. For the method FACT, it learns the spectral phase information from the image data to help the model capture domain-invariant semantic concepts, which tends to be a better strategy in this case. Therefore, the method FACT leads to better result on the target domain Sketch. In summary, the above results suggest that our approach has reduced the domain discrepancy and consequently leads to superior domain generalization performance on these image classification datasets.

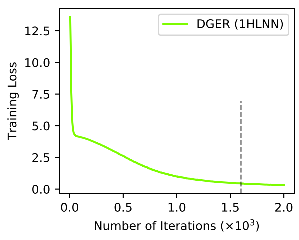
We further make the following conclusions and discussions based on the above results. **(1)** For our approach versus the Baseline, we note that involving our decomposed adversarial loss during training, is beneficial for improving the target performance of the source trained shallow or deep network. **(2)** For our approach versus the relevant adversarial methods MME-AAE, CIDDG, and DGER, we remark that our approach contains fewer network parameters, enjoys a simple procedure of minibatch SGD training, and produces better domain generalization results. **(3)** For our approach versus the other methods, *e.g.*, MetaReg [19], L2A-OT [20], and DGGS [21], which address domain generalization via



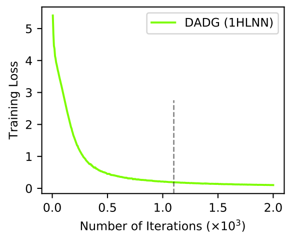
(a)



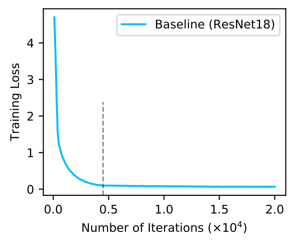
(b)



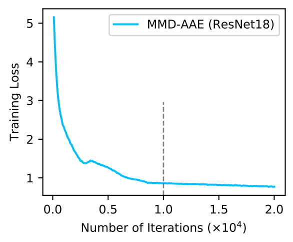
(c)



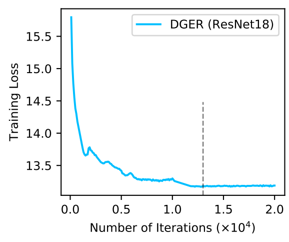
(d)



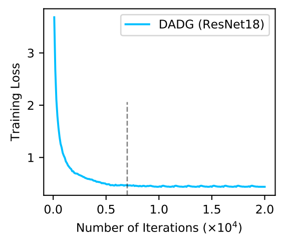
(e)



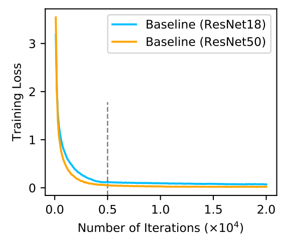
(f)



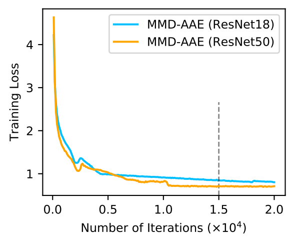
(g)



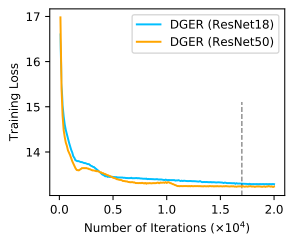
(h)



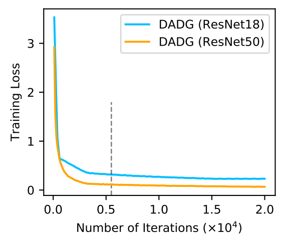
(i)



(j)



(k)



(l)

Figure 5: Convergence performance of Baseline, MMD-AAE, DGER, and DADG. (a)-(d) Convergence results with backbone 1HLNN on the task where the target domain is C07 (PIE). (e)-(h) Convergence results with backbone ResNet18 on the task where the target domain is ArtPainting (PACS). (i)-(l) Convergence results with backbones ResNet18 and ResNet50 on the task where the target domain is Clipart (Office-Home).

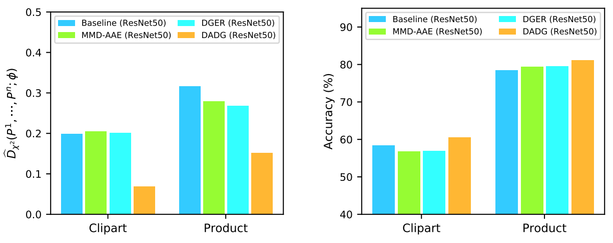
learning a regularization function, augmenting the training data, and refining the optimization procedure, we emphasize that our approach with the decomposed adversarial loss, is complementary to these methods. As a future work, it would be interesting to explore combining our approach with these methods to get the best of both worlds.

4.6. Convergence Performance

We visualize in Figure 5 the convergence performance of Baseline, MMD-AAE, DGER, and DADG using their training losses on the tasks from different multi-domain datasets. From Figure 5(a)-Figure 5(d) for the task where the target domain is C07 (PIE), we observe that while DADG (1HLNN) does not converge as fast as Baseline (1HLNN), it converges faster than the two adversarial methods MMD-AAE (1HLNN) and DGER (1HLNN), as indicated by the dash lines in the figures. From Figure 5(e)-Figure 5(h) for the task where the target domain is ArtPainting (PACS), we again observe that DADG (ResNet18) outperforms MMD-AAE (ResNet18) and DGER (ResNet18) in convergence performance. Moreover, from Figure 5(i)-Figure 5(l) for the task where the target domain is Art (Office-Home), DADG with both the ResNet18 and ResNet50 backbones, yields better convergence results than MMD-AAE and DGER. From these figures, we also notice that during the iterations, the training losses of the two adversarial methods tend to be bigger than DADG. We conjecture that this is because they contain more losses (and also more tradeoff parameters) than our approach with one classification loss and one decomposed adversarial loss. To summarize, the above results confirm that DADG is more advantageous in the training procedure and converges faster than the relevant adversarial methods.

4.7. Domain Discrepancy

We check that reducing the discrepancy among source domains $P^1(\boldsymbol{x}, y), \cdots, P^n(\boldsymbol{x}, y)$ improves the model performance in the unseen but related target domain $P^t(\boldsymbol{x}, y)$. Figure 6(a) plots the domain discrepancy $\widehat{D}_{\chi^2}(P^1, \cdots, P^n; \phi)$ (defined in Eq. (16)) in the embedding spaces of Baseline (ResNet50), MMD-AAE (ResNet50), DGER (ResNet50), and DADG (ResNet50) on two tasks where the target domains are Clipart and Product (Office-Home), respectively. Fig-



(a) (b)

Figure 6: Domain discrepancy and classification accuracy of Baseline (ResNet50), MMD-AAE (ResNet50), DGER (ResNet50), and DADG (ResNet50) on two tasks from Office-Home. (a) Domain discrepancy of the four methods. (b) Classification accuracy of the four methods.

Table 9
Ablation study results (%) of our approach on PIE. DADG-cls means DADG with only the classification loss and DADG-dal means DADG with only the decomposed adversarial loss.

Method	C05	C07	C09	C27	C29	Avg
DADG-cls	79.93	78.49	86.61	97.55	69.46	82.41
DADG-dal	0.99	2.95	1.59	1.59	1.59	1.74
DADG	85.25	87.56	88.06	98.35	77.05	87.25

ure 6(b) shows the corresponding classification accuracy of the four methods. We observe from Figure 6(a) that compared with Baseline, our DADG significantly reduces the domain discrepancy in the ResNet50 embedding space on both tasks. This leads to the improvements in target classification accuracy in Figure 6(b). We also observe from Figure 6(a) that the adversarial methods MMD-AAE and DGER, which align the domain components, do not succeed in reducing the domain discrepancy on the first task, and fail to improve the corresponding target classification accuracy over Baseline in Figure 6(b). To summarize, these results confirm that reducing the source domain discrepancy indeed helps the network predictor to gain better target performance.

4.8. Ablation Study

We conduct ablation study to investigate the contributions of the classification loss and the decomposed adversarial loss in our approach. To this end, we design two DADG variants: DADG-cls with only the classification loss and DADG-dal with only the decomposed adversarial loss. We run these variants on dataset PIE and report the classification results (%) in Table 9. We can observe from Table 9 that without the classification loss, DADG-dal performs poorly on all the tasks. This is natural since the parameters of the classifier are not optimized to minimize the classification loss. We also observe that without the decomposed adversarial loss, DADG-cls yields less superior results than DADG on all the tasks. Clearly, this ablation study suggests that both the classification loss and the decomposed adversarial loss are indispensable to our approach.

5. Conclusion

In this work, we propose the DADG approach to better generalize a source trained network predictor to an unseen target domain. Our approach minimizes the decomposed adversarial loss to align the source domains in the network embedding space, and the cross-entropy loss to learn the downstream classifier. The adversarial loss is decomposed by making use of (1) the quadratic loss brought by the Pearson χ^2 divergence and (2) the linear-in-parameter domain discriminators. We implement our approach using shallow and deep networks, and experimentally demonstrate its advantage over other methods on several multi-domain image classification datasets. In the future, we plan to combine our approach with methods that are based on optimization or data augmentation for improved performance. Beyond domain generalization, we also plan to extend the core decomposition technique in this work to tackle other problems (*e.g.*, generative adversarial learning, unsupervised representation learning), so as to reduce the model parameters and stabilize the training procedure.

Acknowledgements

We thank Zijie Hong for conducting part of the experiments. This work was supported in part by the National Natural Science Foundation of China under Grant 62106137, and in part by Shantou University under Grant NTF21035.

References

[1] V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
[2] G. Blanchard, G. Lee, C. Scott, Generalizing from several related classification tasks to a new unlabeled sample, in: Advances in Neural Information Processing Systems, 2011, pp. 2178–2186.
[3] K. Muandet, D. Balduzzi, B. Schölkopf, Domain generalization via invariant feature representation, in: International Conference on Machine Learning, volume 28, 2013, pp. 10–18.
[4] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, C. C. Loy, Domain generalization: A survey, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022) 1–20.
[5] B. Bhushan Damodaran, B. Kellenberger, R. Flamary, D. Tuia, N. Courty, Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation, in: European Conference on Computer Vision, 2018, pp. 447–463.
[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[7] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, V. Lempitsky, Domain-adversarial training of neural networks, Journal of Machine Learning Research 17 (2016) 1–35.
[8] H. Li, S. J. Pan, S. Wang, A. C. Kot, Domain generalization with adversarial feature learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5400–5409.
[9] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, D. Tao, Deep domain generalization via conditional invariant adversarial networks, in: European Conference on Computer Vision, 2018, pp. 624–639.
[10] K. Akuzawa, Y. Iwasawa, Y. Matsuo, Adversarial invariant feature learning with accuracy constraint for domain generalization, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2019, pp. 315–331.
[11] S. Zhao, M. Gong, T. Liu, H. Fu, D. Tao, Domain generalization via

entropy regularization, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 3118–3129.
[12] M. M. Rahman, C. Fookes, M. Baktashmotlagh, S. Sridharan, Correlation-aware adversarial domain adaptation and generalization, Pattern Recognition 100 (2020) 107124.
[13] T. Matsuura, T. Harada, Domain generalization using a mixture of multiple latent domains, in: AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 11749–11756.
[14] S. Lin, C.-T. Li, A. C. Kot, Multi-domain adversarial feature generalization for person re-identification, IEEE Transactions on Image Processing 30 (2021) 1596–1607.
[15] B. Schölkopf, A. J. Smola, Learning with kernels: support vector machines, regularization, optimization, and beyond, MIT press, 2001.
[16] X. Nguyen, M. J. Wainwright, M. I. Jordan, Estimating divergence functionals and the likelihood ratio by convex risk minimization, IEEE Transactions on Information Theory 56 (2010) 5847–5861.
[17] S. M. Ali, S. D. Silvey, A general class of coefficients of divergence of one distribution from another, Journal of the Royal Statistical Society: Series B (Methodological) 28 (1966) 131–142.
[18] D. Li, Y. Yang, Y.-Z. Song, T. Hospedales, Learning to generalize: Meta-learning for domain generalization, in: AAAI Conference on Artificial Intelligence, volume 32, 2018.
[19] Y. Balaji, S. Sankaranarayanan, R. Chellappa, Metareg: Towards domain generalization using meta-regularization, in: Advances in Neural Information Processing Systems 31, 2018, pp. 998–1008.
[20] K. Zhou, Y. Yang, T. Hospedales, T. Xiang, Learning to generate novel domains for domain generalization, in: European Conference on Computer Vision, 2020, pp. 561–578.
[21] L. Mansilla, R. Echeveste, D. H. Milone, E. Ferrante, Domain generalization via gradient surgery, in: IEEE International Conference on Computer Vision, 2021, pp. 6630–6638.
[22] A. Roy, E. Cambria, Soft labeling constraint for generalizing from sentiments in single domain, Knowledge-Based Systems (2022) 108346.
[23] M. Ghifary, D. Balduzzi, W. B. Kleijn, M. Zhang, Scatter component analysis: A unified framework for domain adaptation and domain generalization, IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2017) 1414–1430.
[24] Y. Li, M. Gong, X. Tian, T. Liu, D. Tao, Domain generalization via conditional invariant representations, in: AAAI Conference on Artificial Intelligence, 2018, pp. 3579–3587.
[25] S. Hu, K. Zhang, Z. Chen, L. Chan, Domain generalization via multidomain discriminant analysis, in: Conference on Uncertainty in Artificial Intelligence, volume 35, 2019.
[26] A. T. Nguyen, T. Tran, Y. Gal, A. G. Baydin, Domain invariant representation learning with domain density transformations, in: Advances in Neural Information Processing Systems, volume 34, 2021.
[27] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, Journal of Machine Learning Research 13 (2012) 723–773.
[28] M. Long, H. Zhu, J. Wang, M. I. Jordan, Deep transfer learning with joint adaptation networks, in: International Conference on Machine Learning, 2017, pp. 2208–2217.
[29] X. Jin, X. Yang, B. Fu, S. Chen, Joint distribution matching embedding for unsupervised domain adaptation, Neurocomputing 412 (2020) 115–128.
[30] S. Chen, L. Han, X. Liu, Z. He, X. Yang, Subspace distribution adaptation frameworks for domain adaptation, IEEE Transactions on Neural Networks and Learning Systems 31 (2020) 5204–5218.
[31] Q. Dou, D. C. de Castro, K. Kamnitsas, B. Glocker, Domain generalization via model-agnostic learning of semantic features, in: Advances in Neural Information Processing Systems, 2019, pp. 6450–6461.
[32] Q. Xu, R. Zhang, Y. Zhang, Y. Wang, Q. Tian, A fourier-based framework for domain generalization, in: IEEE Conference on Computer Vision and Pattern Recognition, 2021, pp. 14383–14392.
[33] D. Li, Y. Yang, Y.-Z. Song, T. M. Hospedales, Deeper, broader and artier domain generalization, in: IEEE International Conference on Computer Vision, 2017, pp. 5542–5550.
[34] V. Piratla, P. Netrapalli, S. Sarawagi, Efficient domain generalization via common-specific low-rank decomposition, in: International Conference on Machine Learning, 2020, pp. 7728–7738.
[35] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, S. P. Smolley, On the effectiveness of least squares generative adversarial networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (2019) 2947–2960.
[36] D. Acuna, G. Zhang, M. T. Law, S. Fidler, f-domain adversarial learning: Theory and algorithms, in: International Conference on Machine Learning, volume 139, 2021, pp. 66–75.
[37] G. Niu, W. Jitkrittum, B. Dai, H. Hachiya, M. Sugiyama, Squared-loss mutual information regularization: A novel information-theoretic approach to semi-supervised learning, in: International Conference on Machine Learning, 2013, pp. 10–18.
[38] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[39] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, Q. He, Deep subdomain adaptation network for image classification, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 1713–1722.
[40] M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, Transfer feature learning with joint distribution adaptation, in: IEEE International Conference on Computer Vision, 2013, pp. 2200–2207.
[41] M. Ghifary, W. Bastiaan Kleijn, M. Zhang, D. Balduzzi, Domain generalization for object recognition with multi-task autoencoders, in: IEEE International Conference on Computer Vision, 2015, pp. 2551–2559.
[42] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.
[43] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, T. Tommasi, Domain generalization by solving jigsaw puzzles, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2224–2233.
[44] S. Motiian, M. Piccirilli, D. A. Adjeroh, G. Doretto, Unified deep supervised domain adaptation and generalization, in: IEEE International Conference on Computer Vision, 2017, pp. 5716–5726.
[45] S. Shankar, V. Piratla, S. Chakrabarti, S. Chaudhuri, P. Jyothi, S. Sarawagi, Generalizing across domains via cross-gradient training, in: International Conference on Learning Representations, 2018.
[46] D. Li, J. Zhang, Y. Yang, C. Liu, Y.-Z. Song, T. M. Hospedales, Episodic training for domain generalization, in: IEEE International Conference on Computer Vision, 2019, pp. 1446–1455.
[47] S. Wang, L. Yu, C. Li, C.-W. Fu, P.-A. Heng, Learning from extrinsic and intrinsic supervisions for domain generalization, in: European Conference on Computer Vision, 2020, pp. 159–176.
[48] K. Zhou, Y. Yang, T. Hospedales, T. Xiang, Deep domain-adversarial image generation for domain generalisation, in: AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 13025–13032.
[49] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[50] S. Chen, H. Wu, C. Liu, Domain invariant and agnostic adaptation, Knowledge-Based Systems 227 (2021) 107192.