Pattern Recognition 134 (2023) 109086

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Domain Generalization by Joint-Product Distribution Alignment

Sentao Chen [a,*], Lei Wang [b], Zijie Hong [c], Xiaowei Yang [c]

[a] Department of Computer Science, Shantou University, China
[b] School of Computing and Information Technology, University of Wollongong, Australia
[c] School of Software Engineering, South China University of Technology, China

ARTICLE INFO

Article history:
Received 2 April 2022
Revised 15 September 2022
Accepted 28 September 2022
Available online 2 October 2022

Keywords:
Distribution alignment
Distribution divergence
Domain generalization
Feature transformation

ABSTRACT

In this work, we address the problem of domain generalization for classification, where the goal is to learn a classification model on a set of source domains and generalize it to a target domain. The source and target domains are different, which weakens the generalization ability of the learned model. To tackle the domain difference, we propose to align a joint distribution and a product distribution using a neural transformation, and minimize the Relative Chi-Square (RCS) divergence between the two distributions to learn that transformation. In this manner, we conveniently achieve the alignment of multiple domains in the neural transformation space. Specifically, we show that the RCS divergence can be explicitly estimated as the maximal value of a quadratic function, which allows us to perform joint-product distribution alignment by minimizing the divergence estimate. We demonstrate the effectiveness of our solution through comparison with the state-of-the-art methods on several image classification datasets.

1. Introduction

A common assumption underlying most supervised learning algorithms is that the training (source) and test (target) data are drawn from the same domain $P(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ is the feature variable and y is the class label. Under this assumption, a classification model appropriately trained in the source domain is guaranteed to generalize well to the target domain in a probability sense [1]. Unfortunately, in real-world applications, control over the data generation process is less perfect: The source data available for training the classification model can be distributionally different from the target data on which the model will be tested, a problem known as dataset shift [2,3], dataset bias [4], or domain shift [5,6]. Under such circumstances, the source trained model may perform poorly on the target data [7–10].

Domain generalization is concerned with the above non-identically-distributed supervised learning problem, where the training data are respectively drawn from n $(n \geq 2)$ source domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$, while the test data are sampled from a target domain $P^t(\boldsymbol{x}, y)$. The source and target domains are different but related to one another [11–13], and the goal of domain generalization for classification is to learn/train a classification model on the source domains and generalize it to the target domain.

Domain generalization methods aim to exploit the domain relationship (i.e., relationship among distributions) to reduce the domain difference and train a classification model on the source data [11,13–15]. Essentially, these methods aim to learn a feature transformation (i.e., a projection matrix or a neural transformation) to align the n source domains $P^1(\boldsymbol{x}, y)$, ..., $P^n(\boldsymbol{x}, y)$ whose samples are available during training, and expect the learned transformation to generalize to the target domain $P^t(\boldsymbol{x}, y)$ such that its difference from the source ones is reduced. As a result, the source trained model can generalize better to the target domain [11,14,16].

Since a domain $P(\boldsymbol{x}, y)$ can be factorized into $P(\boldsymbol{x}, y) = P(y|\boldsymbol{x})P(\boldsymbol{x})$ or $P(\boldsymbol{x}, y) = P(\boldsymbol{x}|y)P(y)$, generally there are two solutions to align the n domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$. The first one learns a feature transformation to align a set of n marginal distributions (marginals) $P^1(\boldsymbol{x}), \ldots, P^n(\boldsymbol{x})$, and assumes that the posterior distribution $P(y|\boldsymbol{x})$ is stable [7,11,12]. However, as discussed in Zhao et al. [14], Nguyen et al. [15], Li et al. [16], the stability of $P(y|\boldsymbol{x})$ is often violated in practice (e.g., speaker recognition, object recognition), which could result in the under-alignment of domains. Therefore, the second solution proposes to align domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$ by seeking a feature transformation (e.g., a neural transformation) to align a set of n marginals and multiple sets of class-conditional distributions (class-conditionals) [13,14,16]. These sets of class-conditionals could either be c sets of n distributions $P^1(\boldsymbol{x}|y = i), \ldots, P^n(\boldsymbol{x}|y = i)$ for $i \in \{1, \ldots, c\}$, or n sets of c distributions $P^l(\boldsymbol{x}|y = 1), \ldots, P^l(\boldsymbol{x}|y = c)$ for $l \in \{1, \ldots, n\}$, where c is the number of classes. However, since this solution has to align multiple sets of class-conditionals, with each set containing multiple distributions, it may not scale well with the number of classes [15]. Besides, to align distributions in the neural network

* Corresponding author.
E-mail address: sentaochenmail@gmail.com (S. Chen).

https://doi.org/10.1016/j.patcog.2022.109086
0031-3203/

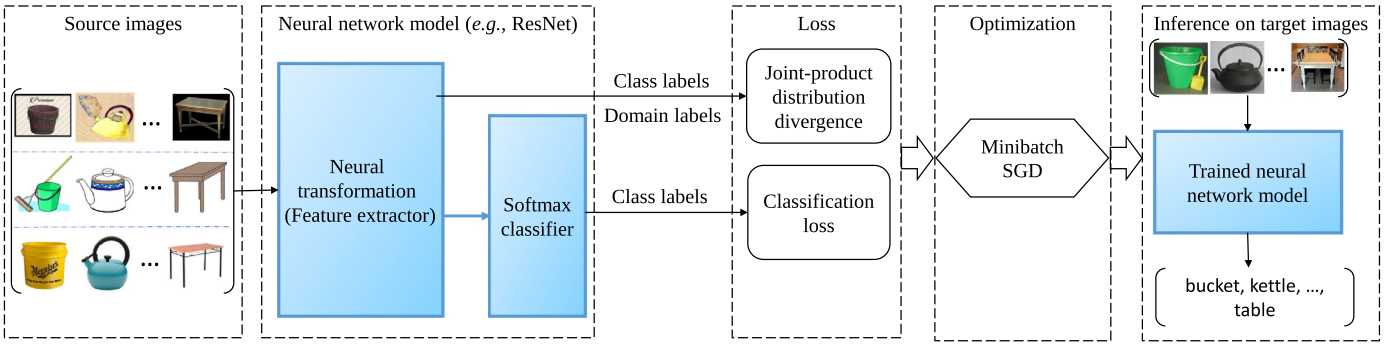


Fig. 1. Illustration of our JPDA solution to domain generalization for image classification, using the Office-Home dataset [17] as an example. In this problem, the source images are collected from 3 different domains (Art, Clipart, and Product). Our solution optimizes the network parameters via the minibatch SGD to perform joint-product distribution alignment in the neural transformation space, and learn a downstream probabilistic classifier with the softmax transformation. The resulting network model is expected to well predict the labels of images drawn from another different target domain RealWorld. In the experiments, we implement our solution with both the shallow and deep neural networks.

context, it usually needs to introduce additional discriminator subnetworks, and solves the challenging minimax problem between the neural transformation and the added subnetworks.

In this work, we address the above issues and propose a Joint-Product Distribution Alignment (JPDA) solution to align the n domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$. To be specific, we first introduce domain label $l \in \{1, \ldots, n\}$ for the n domains and rewrite them as $P(\boldsymbol{x}, y|l = 1), \ldots, P(\boldsymbol{x}, y|l = n)$, respectively. We then learn a neural transformation (feature extractor) T to align the joint distribution $P(\boldsymbol{x}, y, l)$ and the product distribution $P(\boldsymbol{x}, y)P(l)$ such that $P(T(\boldsymbol{x}), y, l) = P(T(\boldsymbol{x}), y)P(l)$, which implies that the distribution of $(T(\boldsymbol{x}), y)$ is independent of the domain label l. This independence conveniently leads to the alignment of the n domains, i.e., $P(T(\boldsymbol{x}), y|l = 1) = \ldots = P(T(\boldsymbol{x}), y|l = n)$. Compared to the aforementioned two solutions from prior works [7,11,13,14], our JPDA solution (1) avoids the factorization of domains and the alignment of the many factorized components, i.e., the marginal distributions and the class-conditional distributions, and (2) only needs to align two distributions, i.e., the joint distribution and the product distribution. Such alignment is algorithmically straightforward and scales well with the number of classes. Namely, unlike previous works [13,14], in our solution the number of distributions aligned is fixed and does not grow with the number of classes. Apart from aligning distributions in the neural transformation space, we learn a downstream classifier for the target classification task.

To be more specific, we align joint distribution $P(\boldsymbol{x}, y, l)$ and product distribution $P(\boldsymbol{x}, y)P(l)$ under the distribution-ratio-based Relative Chi-Square (RCS) divergence [18]. Importantly, we show that the RCS divergence between these two distributions can be analytically estimated as the maximal value of a quadratic function, and consequently obtain an explicit estimate of the RCS divergence. This allows us to directly minimize the divergence estimate with respect to the neural transformation to achieve joint-product distribution alignment. Compared to the existing adversarial methods [7,13,14] that make use of another distribution-ratio-based divergence, the Jensen–Shannon (JS) divergence, our JPDA solution (1) does not need to introduce additional discriminator subnetworks, which could result in learning more network parameters, and (2) avoids solving the challenging minimax problem between the neural transformation and the discriminator subnetworks. Our cost function is a combination of the joint-product distribution divergence and the classification loss. We minimize it via the minibatch Stochastic Gradient Descent (SGD) algorithm, and obtain a network model (containing the neural transformation and the classifier) with better generalization capability. See Fig. 1 that illustrates our solution to domain generalization for image classification. To summarize, our major contributions in this work can be listed as follows:

- We propose to align domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$ via the alignment of joint distribution $P(\boldsymbol{x}, y, l)$ and product distribution $P(\boldsymbol{x}, y)P(l)$, where the domain label $l \in \{1, \ldots, n\}$.
- We analytically derive an explicit estimate of the RCS divergence between $P(\boldsymbol{x}, y, l)$ and $P(\boldsymbol{x}, y)P(l)$ to serve as the alignment loss.
- We demonstrate the effectiveness of our solution by conducting comprehensive experiments on several multi-domain image classification datasets.

2. Related work

We first discuss the domain alignment works in domain generalization, which align domains by factorizing them and aligning the factorized components, i.e., the marginal distributions (marginals), the class-conditional distributions (class-conditionals). Subsequently, we briefly review works that tackle the problem via other strategies.

The study of learning and generalizing a classification model from multiple source domains to a target domain can be traced back to the early works of Blanchard et al. [19] and Khosla et al. [20]. Muandet et al. [11] formally introduced the terminology "domain generalization" with the domain being defined as $P(\boldsymbol{x}, y)$, proposed the Domain Invariant Component Analysis (DICA) that performs dimension reduction to align the marginals under the distributional variance, and trained prediction models on the transformed data. In addition, Akuzawa et al. [21] proposed Adversarial Feature Learning with Accuracy Constraint (AFLAC), which learns domain invariant representations following [22], and prevents the domain invariance from hurting the classification accuracy. By contrast, motivated by Gong et al. [23], Li et al. [13] introduced Conditional Invariant Deep Domain Generalization (CIDDG) that plays two kinds of multi-player minimax games in the deep neural networks, and proved that playing those games is equivalent to aligning the class-conditionals and the marginals, respectively under the JS divergence. Besides, Zhao et al. [14] aligned the marginals and the posterior distributions (posteriors), transformed the alignment of the posteriors into the alignment of the class-conditionals, and aligned the distributions through adversarial training.

Different from prior works that factorize the n domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$ and align their marginals or their class-conditionals (the number of distributions aligned grows with the number of classes), our work introduces the domain label l and aligns the n domains via simply aligning two distributions: the

joint distribution $P(\boldsymbol{x}, y, l)$ and the product distribution $P(\boldsymbol{x}, y)P(l)$. This is algorithmically straightforward and scales well with the number of classes. Furthermore, different from prior works (e.g., Li et al. [13], Zhao et al. [14]) that introduce additional discriminator subnetworks with parameters and solve the challenging minimax problems to align distributions under the JS divergence, our work is free from augmenting the network parameters, analytically solves a quadratic maximization problem to derive an explicit estimate of the RCS divergence, and minimizes that estimate to align the joint distribution and the product distribution.

Apart from domain alignment, domain generalization can also be addressed via meta-learning [6,24,25], data augmentation [26–28], or other strategies [29–31]. (1) The meta-learning based works split the source datasets into meta-train and meta-test sets to simulate domain shift during training, and expect the trained classification model to be robust against the unseen target domain. For example, Li et al. [24] 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. (2) The data augmentation based works generate more source data to increase the probability of covering the data in the target domain, such that classification model trained using the augmented source data can generalize better to the unseen target domain. For instance, Zhou et al. [26] synthesized data from pseudo-novel domains with an optimal transport based formulation to increase the diversity of the source datasets, and then combined both the source and pseudo-novel domain data for training the network classification model. Moreover, Xu et al. [27] 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. (3) Other works explore interesting strategies to improve the generalization ability of the classification model. For example, Mansilla et al. [32] 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 non-domain-alignment works exploit strategies different from ours to tackle the domain generalization problem. In the experiments, we compare our work with some of them for completeness.

3. Methodology

We define the domain generalization problem, give an overview of our solution, and elaborate on the technical details.

3.1. Problem formulation

Let $\mathcal{X}$ be an input feature space and $\mathcal{Y} = \{1, \ldots, c\}$ be an output class label space. According to Chen et al. [9], Muandet et al. [11], Li et al. [13], a domain is represented by $P(\boldsymbol{x}, y)$, where $\boldsymbol{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$. In domain generalization, we have n ($n \geq 2$) source domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$, which are reflected by the associated datasets $\mathcal{D}^1 = \{(\boldsymbol{x}_i^1, y_i^1)\}_{i=1}^{m_1}$, ..., $\mathcal{D}^n = \{(\boldsymbol{x}_i^n, y_i^n)\}_{i=1}^{m_n}$, respectively, 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 one another. Given the n source datasets as training data, the goal is to learn/train a classification model $f : \mathcal{X} \to \mathcal{Y}$ that well predicts the target data labels at test time.

3.2. Overview

We implement the model f with a neural network containing a neural transformation and a probabilistic classifier. As a key solution to the problem, we learn the neural transformation T to align domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$ whose samples are observed during training. For clarity and easy readability, we first present in Table 1 the important notations used in our solution.

Table 1
Notations and their descriptions.

Notation	Description
$\boldsymbol{x}$, $\boldsymbol{z}$	feature/transformed feature variable
y, l	class/domain label
c, n	number of classes/source domains
m_l, m	number of samples in domain l/all domains
λ	tradeoff parameter
α	mixture parameter in the RCS divergence
$\boldsymbol{G}$, $\boldsymbol{K}$, $\boldsymbol{S}$	kernel matrices
$\boldsymbol{H}$	symmetric matrix
$\boldsymbol{b}$	column vector
$\widehat{\boldsymbol{\theta}}$	learned parameter vector

We align domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$ via the following three steps. (1) We first introduce a domain label l, $l \in \{1, \ldots, n\}$, and define a joint distribution $P(\boldsymbol{x}, y, l)$ and a product distribution $P(\boldsymbol{x}, y)P(l)$. (2) We view domains $P^1(\boldsymbol{x}, y), \ldots, P^n(\boldsymbol{x}, y)$ as $P(\boldsymbol{x}, y|l = 1), \ldots, P(\boldsymbol{x}, y|l = n)$, respectively, which is inspired by the probabilistic formulation of the multi-task learning problem [33]. From this viewpoint, the joint distribution $P(\boldsymbol{x}, y, l)$ is reflected by dataset $\mathcal{D}_{xyl} = \{(\boldsymbol{x}_i^1, y_i^1, 1)\}_{i=1}^{m_1} \cup \ldots \cup \{(\boldsymbol{x}_i^n, y_i^n, n)\}_{i=1}^{m_n} = \{(\boldsymbol{x}_i, y_i, l_i)\}_{i=1}^{m}$, where the number of samples $m = m_1 + \ldots + m_n$. The distribution $P(\boldsymbol{x}, y) = \int P(\boldsymbol{x}, y, l)dl$ is reflected by dataset $\mathcal{D}_{xy} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{m}$, and the distribution $P(l) = \int P(\boldsymbol{x}, y, l)d\boldsymbol{x}dy$ is reflected by dataset $\mathcal{D}_l = \{l_i\}_{i=1}^{m}$. (3) We finally learn the neural transformation T to align joint distribution $P(\boldsymbol{x}, y, l)$ and product distribution $P(\boldsymbol{x}, y)P(l)$, i.e., $P(T(\boldsymbol{x}), y, l) = P(T(\boldsymbol{x}), y)P(l)$, which implies that the distribution of $(T(\boldsymbol{x}), y)$ is independent of the domain label l. This independence conveniently leads to the alignment of the n domains: $P(T(\boldsymbol{x}), y|l = 1) = \ldots = P(T(\boldsymbol{x}), y|l = n)$. We expect the learned transformation to generalize to the target domain such that its difference from the source ones is reduced.

We align joint distribution $P(\boldsymbol{x}, y, l)$ and product distribution $P(\boldsymbol{x}, y)P(l)$ under the distribution-ratio-based Relative Chi-Square (RCS) divergence [18], and derive its explicit estimate to serve as the alignment loss. Together with the classification loss of the downstream classifier, we propose to learn the neural network model f via solving a minimization problem in the form

$$\min_{\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T} \mathcal{L}(\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T) = \mathcal{L}_c(\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T) + \lambda \mathcal{L}_t(\boldsymbol{\Theta}_T). \tag{1}$$

Here, $\mathcal{L}_t(\boldsymbol{\Theta}_T)$ is the alignment loss of the transformation T with parameter $\boldsymbol{\Theta}_T$, $\mathcal{L}_c(\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T)$ is the classification loss of the downstream classifier with parameter $\boldsymbol{\Theta}_C$, and λ (> 0) is a tradeoff parameter for balancing these two losses.

In the following subsections, we elaborate on the explicit RCS divergence estimation, the classification loss, and the optimization algorithm.

Remark 1. In prior adversarial works for domain generalization (e.g., Li et al. [7], Li et al. [13], Zhao et al. [14]), another distribution-ratio-based divergence, the JS divergence, is employed for distribution comparison. There, the divergence is iteratively estimated as the maximal value of a value function with respect to the discriminator subnetworks, which results in the challenging minimax problem when minimizing the estimated divergence with respect to the neural transformation. By contrast, the RCS divergence can be analytically estimated as the maximal value of a quadratic function, as we will show shortly, thus converting the minimax problem into a common minimization problem. This strength motivates and encourages us to employ it in this work.

3.3. Explicit divergence estimation

We denote $\boldsymbol{z} = T(\boldsymbol{x}; \boldsymbol{\Theta}_T)$ as the transformed feature variable, and estimate the RCS divergence between joint distribution $P(\boldsymbol{z}, y, l)$ and product distribution $P(\boldsymbol{z}, y)P(l)$.

To be specific, the RCS divergence [18] between $P(\boldsymbol{z}, y, l)$ and $P(\boldsymbol{z}, y)P(l)$ is defined as

$$\begin{aligned}&\mathrm{RCS}_\alpha\big(P(\boldsymbol{z}, y, l)\|P(\boldsymbol{z}, y)P(l)\big)\\&= \int\left[\left(\frac{P(\boldsymbol{z}, y, l)}{P^\alpha(\boldsymbol{z}, y, l)}\right)^2 - 1\right]P^\alpha(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl,\end{aligned} \tag{2}$$

where $P^\alpha(\boldsymbol{z}, y, l) = \alpha P(\boldsymbol{z}, y, l) + (1-\alpha)P(\boldsymbol{z}, y)P(l)$ is the α-mixture distribution with the mixture parameter $\alpha \in [0, 1)$. The RCS divergence compares distributions $P(\boldsymbol{z}, y, l)$ and $P(\boldsymbol{z}, y)P(l)$ based on the ratio $\frac{P(\boldsymbol{z},y,l)}{P^\alpha(\boldsymbol{z},y,l)}$. It is non-negative, upper bounded by $\frac{1}{\alpha} - 1$ for $\alpha \in (0, 1)$, and equals to zero if and only if $P(\boldsymbol{z}, y, l) = P(\boldsymbol{z}, y)P(l)$.

To estimate the RCS divergence, we make use of the equation $a^2 = \max_x(2ax - x^2)$, regard the ratio function $\frac{P(\boldsymbol{z},y,l)}{P^\alpha(\boldsymbol{z},y,l)}$ as a, the function $r(\boldsymbol{z}, y, l)$ as x, and express the original divergence in Eq. (2) as

$$\begin{aligned}&\mathrm{RCS}_\alpha\big(P(\boldsymbol{z}, y, l)\|P(\boldsymbol{z}, y)P(l)\big)\\&= \int \max_r\left[2\frac{P(\boldsymbol{z}, y, l)}{P^\alpha(\boldsymbol{z}, y, l)}r(\boldsymbol{z}, y, l) - r(\boldsymbol{z}, y, l)^2 - 1\right]P^\alpha(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl\end{aligned} \tag{3}$$

$$\geq \max_r \int\left[2\frac{P(\boldsymbol{z}, y, l)}{P^\alpha(\boldsymbol{z}, y, l)}r(\boldsymbol{z}, y, l) - r(\boldsymbol{z}, y, l)^2 - 1\right]P^\alpha(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl \tag{4}$$

$$\begin{aligned}&= \max_r\Big(2\int r(\boldsymbol{z}, y, l)P(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl\\&\quad - \int r(\boldsymbol{z}, y, l)^2P^\alpha(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl\Big) - 1\end{aligned} \tag{5}$$

$$\begin{aligned}&= \max_r\Big(2\int r(\boldsymbol{z}, y, l)P(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl\\&\quad -\alpha\int r(\boldsymbol{z}, y, l)^2P(\boldsymbol{z}, y, l)d\boldsymbol{z}dydl\\&\quad -(1-\alpha)\int r(\boldsymbol{z}, y, l)^2P(\boldsymbol{z}, y)P(l)d\boldsymbol{z}dydl\Big) - 1.\end{aligned} \tag{6}$$

The inequality in Eq. (4) is not difficult to obtain with the knowledge of functional analysis. Let $s^*(x)$ be the maximal function such that $s^*(x) \geq s(x), \forall s(x)$, and $P(x)$ be a probability distribution. Clearly, $\int s^*(x)P(x)dx \geq \int s(x)P(x)dx$, $\forall s(x)$, and therefore $\int s^*(x)P(x)dx \geq \max_s \int s(x)P(x)dx$. In particular, when the function $r(\boldsymbol{x}, y, l) = \frac{P(\boldsymbol{z},y,l)}{P^\alpha(\boldsymbol{z},y,l)}$, Eq. (4) becomes an equality and Eq. (6) attains its maximal value, i.e., Eq. (2). Given datasets $\mathcal{D}_{zyl} = \{(\boldsymbol{z}_i, y_i, l_i)\}_{i=1}^m$, $\mathcal{D}_{zy} = \{(\boldsymbol{z}_i, y_i)\}_{i=1}^m$, and $\mathcal{D}_l = \{l_i\}_{i=1}^m$ from distributions $P(\boldsymbol{z}, y, l)$, $P(\boldsymbol{z}, y)$, and $P(l)$, we can replace the expectations in Eq. (6) by their empirical averages and approximate the RCS divergence as

$$\begin{aligned}&\mathrm{RCS}_\alpha\big(P(\boldsymbol{z}, y, l)\|P(\boldsymbol{z}, y)P(l)\big)\\&\approx \max_r\Big(\frac{2}{m}\sum_{i=1}^m r(\boldsymbol{z}_i, y_i, l_i) - \frac{\alpha}{m}\sum_{i=1}^m r(\boldsymbol{z}_i, y_i, l_i)^2\\&\quad -\frac{1-\alpha}{m^2}\sum_{i,j=1}^m r(\boldsymbol{z}_i, y_i, l_j)^2\Big) - 1.\end{aligned} \tag{7}$$

Let $r(\boldsymbol{z}, y, l)$ be a linear-in-parameter function with the form $r(\boldsymbol{z}, y, l; \boldsymbol{\theta}) = \sum_{i=1}^m \theta_i g\big((\boldsymbol{z}, y, l), (\boldsymbol{z}_i, y_i, l_i)\big)$, where the function parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)^\top$ and the product kernel $g\big((\boldsymbol{z}, y, l), (\boldsymbol{z}_i, y_i, l_i)\big) = k(\boldsymbol{z}, \boldsymbol{z}_i)\delta(y, y_i)\delta(l, l_i)$. The feature kernel $k(\boldsymbol{z}, \boldsymbol{z}_i)$ is a Gaussian kernel $k(\boldsymbol{z}, \boldsymbol{z}_i) = \exp\big(\frac{-\|\boldsymbol{z}-\boldsymbol{z}_i\|^2}{\sigma}\big)$ with kernel width $\sigma > 0$[1], the class label kernel $\delta(y, y_i)$ is a delta kernel that evaluates 1 if $y = y_i$ and 0 otherwise, and the domain label kernel $\delta(l, l_i)$ is also a delta kernel. Such linear-in-parameter function model is a popular choice in statistical machine learning [33,35], and more importantly, can lead to an analytic solution to the maximization problem. Plugging this function model into Eq. (7), we express the estimate of the RCS divergence as

$$\begin{aligned}&\widehat{\mathrm{RCS}}_\alpha\big(P(\boldsymbol{z}, y, l)\|P(\boldsymbol{z}, y)P(l)\big)\\&= \max_{\boldsymbol{\theta}}\Big(\frac{2}{m}\sum_{i=1}^m r(\boldsymbol{z}_i, y_i, l_i; \boldsymbol{\theta}) - \frac{\alpha}{m}\sum_{i=1}^m r(\boldsymbol{z}_i, y_i, l_i; \boldsymbol{\theta})^2\\&\quad -\frac{1-\alpha}{m^2}\sum_{i,j=1}^m r(\boldsymbol{z}_i, y_i, l_j; \boldsymbol{\theta})^2\Big) - 1\end{aligned} \tag{8}$$

$$\begin{aligned}&= \max_{\boldsymbol{\theta}}\Big(\frac{2}{m}\mathbf{1}^\top\boldsymbol{G}\boldsymbol{\theta} - \frac{\alpha}{m}\boldsymbol{\theta}^\top\boldsymbol{G}^\top\boldsymbol{G}\boldsymbol{\theta}\\&\quad -\frac{1-\alpha}{m^2}\boldsymbol{\theta}^\top(\boldsymbol{K}^\top\boldsymbol{K})\circ(\boldsymbol{S}^\top\boldsymbol{S})\boldsymbol{\theta}\Big) - 1\end{aligned} \tag{9}$$

$$= \max_{\boldsymbol{\theta}}(2\boldsymbol{b}^\top\boldsymbol{\theta} - \boldsymbol{\theta}^\top\boldsymbol{H}\boldsymbol{\theta}) - 1 \tag{10}$$

$$= 2\boldsymbol{b}^\top\widehat{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}^\top\boldsymbol{H}\widehat{\boldsymbol{\theta}} - 1. \tag{11}$$

In Eq. (9), $\mathbf{1}$ is an m-dimensional column vector of ones, $\boldsymbol{G}$ is an $m \times m$ matrix with its (i, j)th element defined as $g_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)\delta(y_i, y_j)\delta(l_i, l_j)$. Additionally, the symbol $\circ$ is the Hadamard product, $\boldsymbol{K}$ and $\boldsymbol{S}$ are both $m \times m$ matrices, whose (i, j)th elements are defined as $k_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)\delta(y_i, y_j)$ and $s_{ij} = \delta(l_i, l_j)$, respectively. Eq. (10) introduces two notations

$$\boldsymbol{b} = \frac{1}{m}\boldsymbol{G}^\top\mathbf{1}, \tag{12}$$

$$\boldsymbol{H} = \frac{\alpha}{m}\boldsymbol{G}^\top\boldsymbol{G} + \frac{1-\alpha}{m^2}(\boldsymbol{K}^\top\boldsymbol{K})\circ(\boldsymbol{S}^\top\boldsymbol{S}), \tag{13}$$

which make the unconstrained quadratic optimization problem explicit. Eventually, Eq. (11) solves the problem and presents the analytic solution

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{H} + \epsilon\mathbf{I})^{-1}\boldsymbol{b}. \tag{14}$$

Note that, to ensure that the matrix is always numerically invertible in practice, a diagonal matrix $\epsilon\mathbf{I}$ is added to the solution, where ϵ is a small positive value[2] and $\mathbf{I}$ is the identity matrix.

According to the explicit RCS divergence estimate in Eq. (11), the alignment loss $\mathcal{L}_\mathrm{t}(\boldsymbol{\Theta}_T)$ of the neural transformation T with parameter $\boldsymbol{\Theta}_T$ is defined as

$$\mathcal{L}_\mathrm{t}(\boldsymbol{\Theta}_T) = 2\boldsymbol{b}^\top\widehat{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}^\top\boldsymbol{H}\widehat{\boldsymbol{\theta}} - 1. \tag{15}$$

For clarity, we summarize the above derivation and present the procedure for calculating the alignment loss $\mathcal{L}_\mathrm{t}(\boldsymbol{\Theta}_T)$ in Algorithm 1. In the algorithm, $\boldsymbol{z}$ is the transformed feature variable produced by the neural transformation T, and dataset $\mathcal{D}_{zyl} = \{(\boldsymbol{z}_i, y_i, l_i)\}_{i=1}^m = \{(\boldsymbol{z}_i^1, y_i^1, 1)\}_{i=1}^{m_1} \cup \ldots \cup \{(\boldsymbol{z}_i^n, y_i^n, n)\}_{i=1}^{m_n}$ is the union of the n source datasets with domain labels.

[1] Inspired by Baktashmotlagh et al. [34], in practice we set the kernel width σ to the median squared distance between the source examples.

[2] In the experiments, we set $\epsilon = 10^{-3}$.

Algorithm 1 Procedure for calculating the JPDA loss.

Input: Dataset: $\mathcal{D}_{zyl} = \{(\boldsymbol{z}_i, y_i, l_i)\}_{i=1}^{m}$; Parameter: α.
Output: $\mathcal{L}_t(\boldsymbol{\Theta}_T)$.

1: Construct kernel matrices $\boldsymbol{G}$, $\boldsymbol{K}$, and $\boldsymbol{S}$ from $\mathcal{D}_{zyl}$, whose (i, j)th elements are $g_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)\delta(y_i, y_j)\delta(l_i, l_j)$, $k_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)\delta(y_i, y_j)$, and $s_{ij} = \delta(l_i, l_j)$, respectively;
2: Compute vector $\boldsymbol{b}$, matrix $\boldsymbol{H}$, and vector $\widehat{\boldsymbol{\theta}}$ via Eqs. (12)-(14), respectively;
3: Calculate $\mathcal{L}_t(\boldsymbol{\Theta}_T)$ via Eq. (15).

3.4. Classification loss

The classifier following the neural transformation T is a probabilistic model $P(y|T(\boldsymbol{x}; \boldsymbol{\Theta}_T); \boldsymbol{\Theta}_C)$ parameterized by $\boldsymbol{\Theta}_C$. In line with previous works [7,14,36], we use the cross-entropy loss and define $\mathcal{L}_c(\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T)$ as

$$\mathcal{L}_c(\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T) = \frac{-1}{n}\sum_{l=1}^{n}\frac{1}{m_l}\sum_{i=1}^{m_l}\Big(\sum_{j=1}^{c}\delta(y_i^l, j)\log P(y=j|T(\boldsymbol{x}_i^l; \boldsymbol{\Theta}_T); \boldsymbol{\Theta}_C)\Big), \tag{16}$$

where $\delta(y_i^l, j)$ is the delta kernel function previously defined in Section 3.3 and evaluates 1 if $y_i^l = j$ and 0 otherwise.

3.5. Optimization algorithm

We employ minibatch SGD to solve optimization problem (1) of our JPDA solution. In every iteration of the algorithm, a minibatch consists of n minibatches respectively sampled from the n source datasets with the corresponding domain labels, and the objective $\mathcal{L}(\boldsymbol{\Theta}_C, \boldsymbol{\Theta}_T)$ in (1) is calculated using these minibatches.

After training, we expect the resulting network classification model to perform well on the inference task in the target domain. Again see Fig. 1 that illustrates the procedure of our solution (Data → Network → Loss → Optimization → Inference).

4. Experiments

For conducting the domain generalization experiments, we note that there exist two different experimental settings in the field: one commonly practiced in Nguyen et al. [15], Xu et al. [27], Yang et al. [28], Carlucci et al. [37], and the other one proposed by Gulrajani and Lopez-Paz [38]. We conduct our experiments under the former, which involves following the settings in prior works and citing the available results reported by the authors themselves.

4.1. Datasets

We evaluate our solution on 4 multi-domain image classification datasets: PIE [39], PACS [5], Office-Home [17], and DomainNet [40], which are popular datasets in domain generalization and adaptation [8,27,28,41,42]. In particular, note that datasets PIE, Office-Home, and DomainNet are with many classes in each domain, which can well assess the domain alignment ability and the classification ability of the algorithms. We use them to verify whether our solution can scale well and perform well in such domain generalization tasks with many classes. In the following, we briefly describe the 4 datasets.

PIE [39] contains gray-scale face images from 5 domains: **C05** (3332 left pose images), **C07** (1629 upward pose images), **C09** (1632 downward pose images), **C27** (3329 frontal pose images), and **C29** (1632 right pose images). Each domain has 68 classes. See Fig. 2(a) for the example images.

PACS [5] includes images from 4 domains: ArtPainting (**A**) with 2048 images, Cartoon (**C**) with 2344 images, Photo (**P**) with 1670 images, and Sketch (**S**) with 3929 images. Each domain has 7 classes. See Fig. 2(b) for the example images.

Office-Home [17] contains images of everyday objects organized into 4 domains: Art (**A**) with 2427 images, Clipart (**C**) with 4365 images, Product (**P**) with 4439 images, and RealWorld (**R**) with 4357 images. Each domain has 65 classes. See Fig. 2(c) for the example images.

DomainNet [40] is a visual dataset that has 6 domains and 345 classes in total. Since the labels of some domains and classes are very noisy [43], we follow [43] and select 4 domains and 126 classes from this dataset. These 4 domains are Clipart (**C**) with 18,703 images, Painting (**P**) with 31,502 images, Real (**R**) with 70,358 images, and Sketch (**S**) with 24,582 images. See Fig. 2(d) for the example images.

4.2. Comparison methods

We compare our solution against a baseline (Baseline) method and the state-of-the-art domain generalization methods. The Baseline trains vanilla neural network classifiers via minimizing the cross-entropy loss on the source data, which is also referred to as DeepAll in prior works [14,25,27,37]. The domain generalization methods contain 1) the ones that perform domain alignment (DAlignment) for domain generalization, and 2) the ones that address the problem via the other strategies (Others). Specifically, the former methods include: MMD-AAE [7], CIDDG [13], DGER [14], and DIRT-GAN [15]. The latter ones include: CCSA [44], CrossGrad [45], MLDG [24], MetaReg [46], JiGen [37], Epi-FCR [36], MASF [25], L2A-OT [26], DDAIG [47], EISNet [48], MMLD [49], RSC [50], CSD [51], MixStyle [10], FACT [27], DILU [52], DGGS [32], ATSRL [28], and Tl-SNR [41].

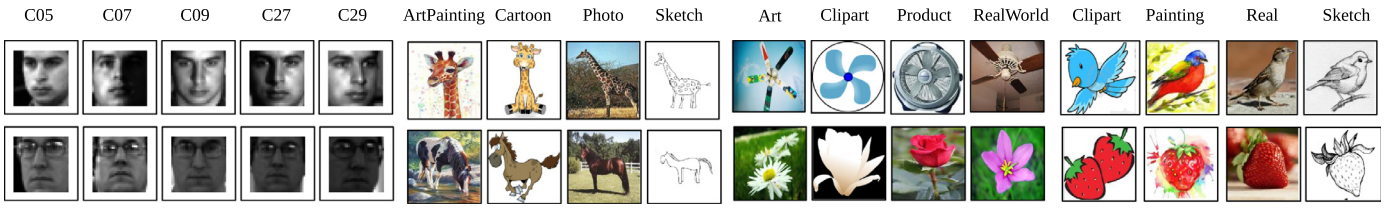
4.3. Evaluation protocol

In line with the leave-one-domain-out evaluation protocol [14,15,26,27], we employ neural 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, following [14,15,26,27], we repeat the experiments 5 times with different random seeds and then report the average classification accuracy.

4.4. Implementation details

We set the mixture parameter $\alpha = 0$ and $\alpha = 0.5$ in the RCS divergence, and denote our solution as JPDA0 and JPDA under these two settings, respectively. Later in Section 4.7, we conduct sensitivity analysis with respect to this hyperparameter. We implement our solution with both the shallow and deep neural networks. The Pytorch implementation (source code) of our solution is available at the link https://github.com/sentaochen/Joint-Product-Distribution-Alignment.

We configure the network backbones for the datasets as below. On the gray-scale image dataset PIE with the 1024-dimensional image pixel features [39], the backbone is a one-Hidden-Layer Neural Network (1HLNN), which has 512 hidden neurons with the ReLU activation, and 68 output neurons (i.e., the number of classes in PIE) 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 each of the 3 RGB image datasets PACS, Office-Home, and DomainNet, we follow the practice in previous works [27,28,42] and use the ResNet18 and ResNet50 backbones [53], where their final classification layers are both recon-



(a) PIE (b) PACS (c) Office-Home (d) DomainNet

Fig. 2. Example images from 4 multi-domain datasets: PIE [39], PACS [5], Office-Home [17], and DomainNet [40].

Table 2
Classification accuracy (%) of the DAlignment and Other methods with the 1HLNN backbone on dataset PIE, where the Baseline is also referred to as DeepAll. In each column, the best result is highlighted in **bold** and the second best is underlined.

Strategy	Method	C05	C07	C09	C27	C29	Avg
–	Baseline	79.93	78.49	86.61	97.55	69.46	82.41
Others	CCSA [44]	80.54	79.44	87.44	97.78	69.88	83.01
	DGGS [32]	82.92	79.91	86.16	97.68	73.47	84.03
DAlignment	MMD-AAE [7]	81.94	79.19	86.93	97.81	71.51	83.48
	CIDDG [13]	83.67	82.35	86.90	98.17	74.00	85.02
	DGER [14]	82.92	80.91	86.16	97.68	73.47	84.23
	JPDA0 (ours)	84.77	**88.34**	88.13	98.31	77.72	87.45
	JPDA (ours)	**85.67**	87.94	**88.50**	**98.43**	**77.86**	**87.68**

structed to have the same number of outputs as the number of classes in that dataset (7 for PACS, 65 for Office-Home, and 126 for DomainNet).

We preprocess and split the training data as follows. On the PIE dataset with the shallow backbone configuration, we utilize the common z-score standardization to preprocess the gray-scale image pixel features. On the PACS, Office-Home, and DomainNet datasets with the deep backbone configurations, we follow the standard practice in Nguyen et al. [15], Xu et al. [27], Carlucci et al. [37] and process the RGB images via random resized cropping, horizontal flipping, and color jittering. Regarding the data splitting protocol, we follow the general practice in prior works [15,27,37] and use 90% of available data as training data and 10% as validation data.

For training the networks, our solution with its shallow implementation on PIE is trained from scratch by the minibatch SGD with a momentum of 0.9 and a learning rate of 10^{-2}. The tradeoff parameter λ is selected from the range $\{10^{-2}, 10^{-1}, \ldots, 10^{3}\}$ by using the validation data. Moreover, our solution with its deep implementations on PACS, Office-Home, and DomainNet 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 backbone, since the corresponding procedure would be computationally costly. Instead, following [22,54], we gradually change λ from 0 to 1 by a progressive schedule: $\lambda_p = \frac{2}{1+\exp(-10p)} - 1$, where p is the training progress linearly changing from 0 to 1.

4.5. Results

We report in Table 2 the classification results on PIE, in Tables 3 and 4 the results on PACS, in Tables 5 and 6 the results on Office-Home, and in Tables 7 and 8 the results on DomainNet. Since our experimental settings coincide with prior works, we therefore directly quote the available results of the comparison methods in Tables 3 and 4 from Zhou et al. [10], Zhao et al. [14], Nguyen et al. [15], Zhou et al. [26], Xu et al. [27], Yang et al. [28], Wang and Bi [41], Piratla et al. [51], Xiao et al. [52], and the results in Tables 5 and 6 from Xu et al. [27], Yang et al. [28], Xiao et al. [52]. For more extensive comparison with other methods, especially the relevant DAlignment ones MMD-AAE [7], CIDDG [13], and DGER [14], we use their source codes, follow their hyperparameter tuning protocols, and produce their results on datasets PIE, Office-Home, and DomainNet. In every 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**, and the second best result is underlined.

In Table 2 for PIE, our solution with the 1HLNN backbone consistently outperforms the Baseline and the domain generalization methods that are based on domain alignment (DAlignment) or other strategies (Others) on all the 5 tasks. Notably, when the target domains are C05, C07, and C29, the improvements of JPDA over the Baseline can reach 5.74% (85.67% versus 79.93%), 9.45% (87.94% versus 78.49%), and 8.40% (77.86% versus 69.46%), respectively. On these 3 tasks, MMD-AAE [7] also obtains better results than the Baseline, but its improvements are not as significant as JPDA0 or JPDA. We conjecture that the method may suffer from the under-alignment of domains since it only aligns the marginal distributions. In summary, in this face recognition problem with 68 classes, the results imply that our solution has significantly reduced the difference among domains and leads to remarkable domain generalization performance. Later in Section 4.6, we visualize the data distribution via t-SNE [55] to further verify this point.

In Tables 3 and 4 for PACS, our solution with the ResNet18 and ResNet50 backbones is among the top performing ones. Specifically, JPDA0 and JPDA perform much better than the relevant DAlignment methods MMD-AAE [7], DGER [14], and DIRT-GAN [15] on the first 3 tasks in Table 3, and they also achieve better average accuracy than the state-of-the-art methods (e.g., DILU [52], FACT [27], ATSRL [28], and Tl-SNR [41]) in both tables. In Tables 5 and 6 for Office-Home, JPDA0 and JPDA again outperform the DAlignment methods on all the 4 tasks with both backbones, and deliver better results than the other methods, including the recent ones MixStyle [10], FACT [27], and ATSRL [28]. In Tables 7 and 8 for DomainNet, since each domain in this dataset contains a large number of classes (126), which makes domain alignment very challenging, the previous DAlignment methods, MMD-AAE [7], CIDDG [13], and DGER [14], do not improve much over the Base-

Table 3
Classification accuracy (%) of the DAlignment 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 [46]	83.70	77.20	95.50	70.30	81.68
	JiGen [37]	79.42	75.25	96.03	71.35	80.51
	Epi-FCR [36]	82.10	77.00	93.90	73.00	81.50
	MASF [25]	80.29	77.17	94.99	71.69	81.04
	MMLD [49]	81.28	77.16	96.09	72.29	81.71
	DDAIG [47]	84.20	78.10	95.30	74.70	83.08
	EISNet [48]	81.89	76.44	95.93	74.33	82.15
	L2A-OT [26]	83.30	78.20	96.20	73.60	82.83
	RSC [50]	83.43	80.31	95.99	**80.85**	85.15
	CSD [51]	78.90	75.80	94.10	76.70	81.38
	DILU [52]	83.92	**81.61**	95.97	80.31	85.45
	MixStyle [10]	84.10	78.80	96.10	75.90	83.73
	ATSRL [28]	85.80	80.70	97.30	77.30	85.28
	FACT [27]	85.37	78.38	95.15	79.15	84.51
	TI-SNR [41]	83.35	79.74	96.20	75.18	83.62
DAlignment	MMD-AAE [7]	75.20	72.70	96.00	64.20	77.03
	DGER [14]	80.70	76.40	96.65	71.77	81.38
	DIRT-GAN [15]	82.56	76.37	95.65	79.89	83.62
	JPDA0 (ours)	86.45	79.90	97.91	78.95	85.80
	JPDA (ours)	**86.63**	79.86	**97.96**	79.12	**85.89**

Table 4
Classification accuracy (%) of the DAlignment 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 [46]	87.20	79.20	97.60	70.30	83.58
	MASF [25]	82.89	80.49	95.01	72.29	82.67
	EISNet [48]	86.64	81.53	97.11	78.07	85.84
	DDAIG [47]	85.40	78.50	95.70	80.00	84.90
	RSC [50]	87.89	82.16	97.92	83.35	87.83
	MixStyle [10]	87.40	83.30	98.00	78.50	86.80
	ATSRL [28]	90.00	83.50	98.90	80.00	88.10
	FACT [27]	89.63	81.77	96.75	**84.46**	88.15
DAlignment	DGER [14]	87.51	79.31	98.25	76.30	85.34
	JPDA0 (ours)	90.56	83.57	98.66	82.77	88.89
	JPDA (ours)	**90.61**	**83.63**	**98.98**	82.84	**89.02**

Table 5
Classification accuracy (%) of the DAlignment 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 [37]	53.04	47.51	71.47	72.79	61.20
	DDAIG [47]	59.20	52.30	74.60	76.00	65.53
	L2A-OT [26]	60.60	50.10	74.80	77.00	65.63
	RSC [50]	58.42	47.90	71.63	74.54	63.12
	DILU [52]	61.81	53.27	74.27	76.31	66.42
	MixStyle [10]	58.70	53.40	74.20	75.90	65.55
	ATSRL [28]	60.70	52.90	75.80	77.20	66.65
	FACT [27]	60.34	**54.85**	74.48	76.55	66.56
DAlignment	MMD-AAE [7]	59.27	51.05	74.02	75.32	64.92
	CIDDG [13]	59.94	52.27	74.31	76.83	65.84
	DGER [14]	60.21	51.29	73.90	76.42	65.46
	JPDA0 (ours)	62.01	54.12	75.87	77.57	67.39
	JPDA (ours)	**62.05**	54.08	**75.92**	**77.65**	**67.43**

line. Nevertheless, our solution, which scales well with the number of classes, achieves consistent improvements over the Baseline on all the tasks and yields better results than its competitors. In summary, in these object recognition problems with a small or large number of classes (65, 7, and 126), the results also demonstrate the effectiveness of our joint-product distribution alignment solution.

According to the above results, we further make the following discussions and conclusions. (1) For our solution versus the Baseline, we note that involving the joint-product distribution divergence in Eq. (15) during training, is beneficial for improving the target performance of the source trained shallow or deep networks (again see Fig. 1 that summarizes the procedure of our solution). (2) For our solution versus the domain generalization meth-

Table 6
Classification accuracy (%) of the DAlignment and Other methods with the ResNet50 backbone on dataset Office-Home.

Strategy	Method	A	C	P	R	Avg
–	Baseline	64.70	58.80	77.90	79.00	70.10
Others	CrossGrad [45]	67.70	57.70	79.10	80.40	71.23
	DDAIG [47]	65.20	59.20	77.70	76.70	69.70
	MixStyle [10]	64.90	58.80	78.30	78.70	70.18
	ATSRL [28]	69.30	60.10	**81.50**	82.10	73.25
	FACT [27]	70.05	58.82	79.61	81.24	72.43
	DGGS [32]	70.21	58.70	79.57	81.98	72.62
DAlignment	MMD-AAE [7]	68.71	56.78	79.32	80.53	71.34
	CIDDG [13]	69.37	58.24	80.14	81.30	72.26
	DGER [14]	68.72	56.87	79.51	81.33	71.61
	JPDA0 (ours)	70.61	60.76	80.32	82.55	73.56
	JPDA (ours)	**70.65**	**60.81**	80.43	**82.59**	**73.62**

Table 7
Classification accuracy (%) of the DAlignment and Other methods with the ResNet18 backbone on dataset DomainNet.

Strategy	Method	C	P	R	S	Avg
-	Baseline	67.30	61.36	70.04	59.06	64.44
Others	FACT [27]	67.58	62.12	70.83	59.86	65.10
	DGGS [32]	68.51	62.53	70.35	60.05	65.36
DAlignment	MMD-AAE [7]	66.28	62.01	69.53	58.97	64.20
	CIDDG [13]	67.60	62.07	70.27	59.19	64.78
	DGER [14]	66.55	61.58	70.13	59.20	64.36
	JPDA0 (ours)	68.42	62.89	71.40	**60.65**	65.84
	JPDA (ours)	**68.45**	**62.92**	**71.43**	60.60	**65.85**

Table 8
Classification accuracy (%) of the DAlignment and Other methods with the ResNet50 backbone on dataset DomainNet.

Strategy	Method	C	P	R	S	Avg
–	Baseline	74.19	68.48	76.08	66.40	71.29
Others	FACT [27]	75.23	68.87	76.55	67.51	72.04
	DGGS [32]	75.51	69.23	76.28	67.95	72.24
DAlignment	MMD-AAE [7]	74.88	68.54	75.02	66.73	71.29
	CIDDG [13]	75.41	69.03	76.37	67.45	72.07
	DGER [14]	74.53	68.27	75.56	67.07	71.36
	JPDA0 (ours)	75.63	**70.21**	76.70	68.75	72.82
	JPDA (ours)	**75.72**	70.12	**76.74**	**68.80**	**72.85**

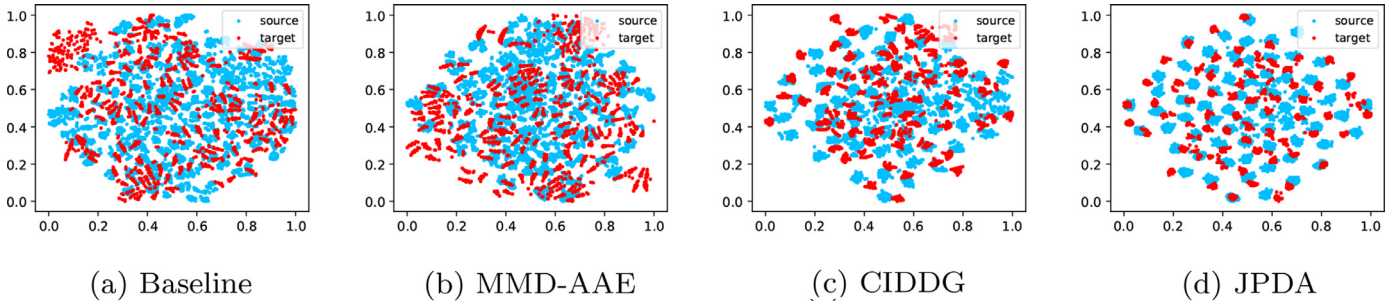
Table 9
Comparison of the DAlignment methods in the case where there are n ($n \geq 2$) source domains and each domain has c classes. To present the text clearly, we abbreviate "Distributions and "adversarial training to "Dist. and "adv. train., respectively.

Method	# Dist. to align	# Added networks	# Losses	Free from adv. train.
MMD-AAE [7]	n	1	4	×
CIDDG [13]	$n(c+1)$	$c+1$	3	×
DGER [14]	$n(c+1)$	$2n$+1	4	×
JPDA (ours)	2	0	2	✓

ods, we note that from the perspective of algorithm design, ours is more straightforward than the others, particularly the most relevant DAlignment methods MMD-AAE, CIDDG, and DGER. To support and reinforce this statement, we compare our solution with these relevant DAlignment methods and present the comparison results in Table 9. The results show that our solution aligns fewer distributions, avoids adding additional subnetworks with parameters, contains fewer losses, and is free from the challenging adversarial training. Therefore, it is more straightforward than the comparison methods from an algorithmic viewpoint. Besides, from the perspective of empirical performance, our solution outperforms its more involved competitors in both face and object recognition problems. (3) For JPDA0 versus JPDA, we note that our solution does not seem to have a special preference for the value of α in the RCS divergence on the tested evaluations. Later in Section 4.7, we conduct experiments to verify such robustness.

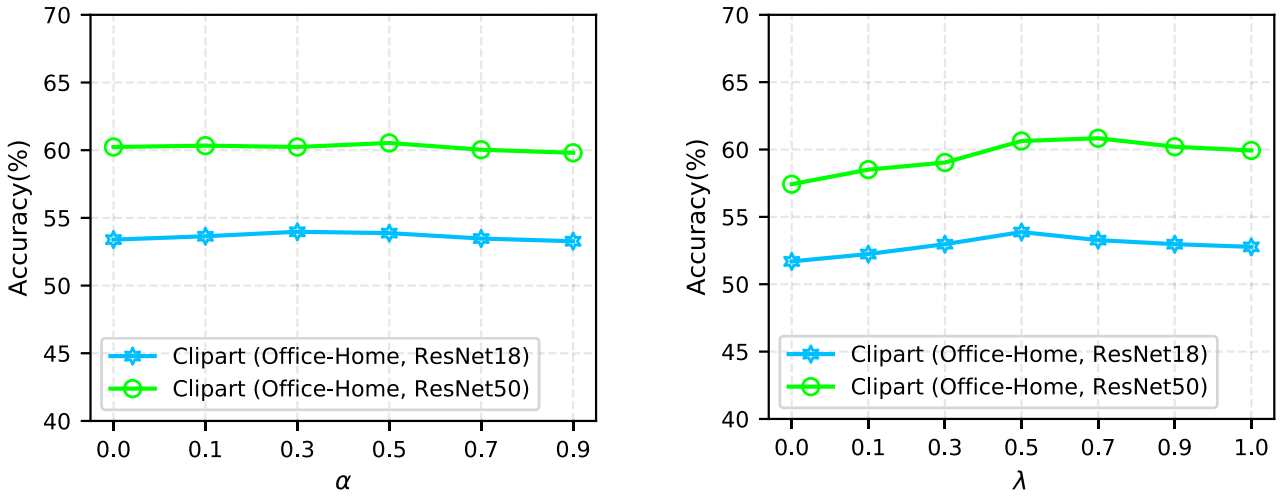
4.6. Feature visualization

We visualize the source and target data represented by the neural transformation features extracted from the Baseline, MMD-AAE, CIDDG, and our JPDA networks. We use PIE as the experimental dataset to better assess the domain alignment ability of our solution, since this dataset has many classes (68) in each domain, which could be challenging for domain alignment. Specifically, we take C07 as the target and the remaining ones as the source domains, and plot in Fig. 3 the t-SNE [55] embeddings of the source and target data. Clearly, compared to the neural transformation

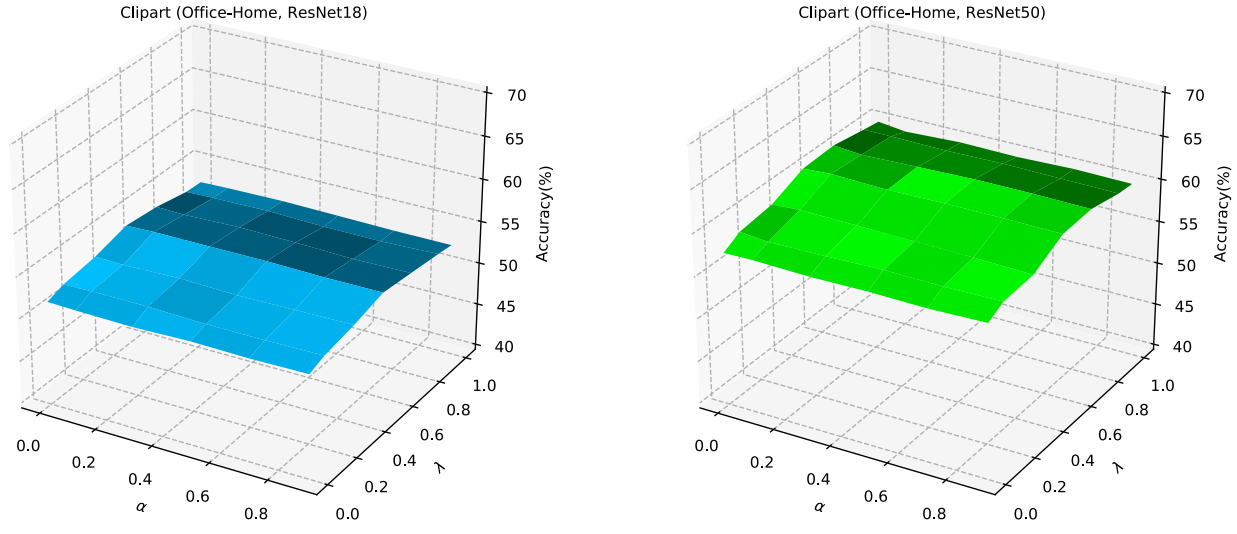


(a) Baseline (b) MMD-AAE (c) CIDDG (d) JPDA

Fig. 3. T-SNE visualization of the source samples (C05, C09, C27, C29, in blue) and the target ones (C07, in red) in the neural transformation spaces of the Baseline, MMD-AAE, CIDDG and our JPDA networks. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Accuracy w.r.t. the value of α (b) Accuracy w.r.t. the value of λ



(c) Accuracy (ResNet18) w.r.t. α and λ (d) Accuracy (ResNet50) w.r.t. α and λ

Fig. 4. Classification accuracy of JPDA with respect to (w.r.t.) different values of mixture parameter α in the RCS divergence and tradeoff parameter λ in optimization problem (1).

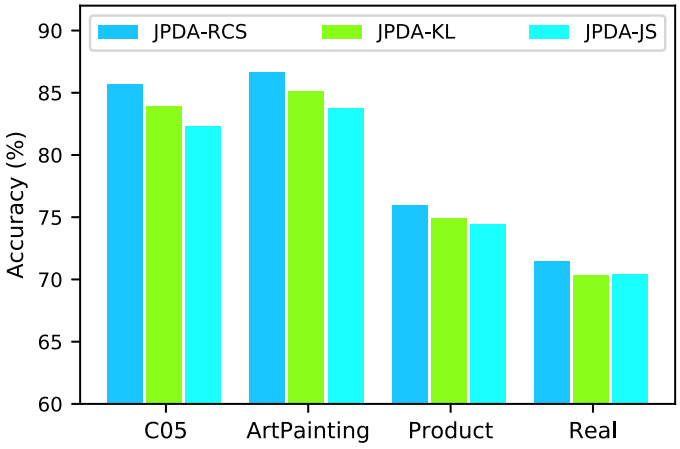


Fig. 5. Classification accuracy of JPDA with the RCS divergence, the KL divergence, and the JS divergence on 4 domain generalization tasks where the target domains are C05 (PIE), ArtPainting (PACS), Product (Office-Home), and Real (DomainNet).

features from the networks of other methods in Fig. 3(a)–(c), the features from our JPDA network in Fig. 3(d) lead to a smaller distribution gap between the source and target data. This shows that while the target data do not participate in training the network, through joint-product distribution alignment our solution manages to better reduce the discrepancy between the source and target domains and therefore leads to better results in the experiments.

4.7. Hyperparameter sensitivity

We investigate the sensitivity of our solution with respect to different choices of the mixture parameter α in the RCS divergence and the tradeoff parameter λ in optimization problem (1). To this end, we plot in Fig. 4(a) and (b) the classification accuracy of our solution with the ResNet18 and ResNet50 backbones, by respectively varying α and λ on the task where the target domain is Clipart (Office-Home). From Fig. 4(a), we observe that on the tested evaluations, our solution is not sensitive to parameter α. By contrast, Fig. 4(b) shows that as the value of λ grows, the accuracy of our solution first monotonically increases and then slightly decreases. This implies the importance of a proper trade-off between the classification loss and the joint-product distribution divergence in optimization problem (1). For completeness, we further plot in Fig. 4(c) and (d) the accuracy of our solution with the ResNet18 and ResNet50 backbones, by simultaneously varying α and λ. The results in Fig. 4(c) and (d) again confirm the robustness of our solution to parameter α, and the importance of keeping a balance between the task loss and the distribution divergence.

4.8. Effect of different divergences

We study how the performance of our JPDA solution is affected by the divergence used for comparing probability distributions. In particular, we replace the original RCS divergence in our solution (i.e., JPDA-RCS) by the Kullback-Leibler (KL) divergence and the JS divergence, and refer to the resulting two solutions as JPDA-KL and JPDA-JS. We run JPDA-KL and JPDA-JS on 4 domain generalization tasks where the target domains are C05 (PIE), ArtPainting (PACS), Product (Office-Home), and Real (DomainNet), and illustrate their classification results in Fig. 5. We observe from Fig. 5 that the performance of JPDA varies with the divergence it applies and that JPDA-RCS consistently outperforms JPDA-KL and JPDA-JS. We conjecture that the outperformance may stem from the property that the RCS divergence can be analytically estimated (see Section 3.3), which makes joint-product distribution alignment under RCS straightforward. By contrast, the KL or JS divergence does not have such nice property and therefore may be less friendly to the distribution alignment. To a certain extent, the results testify that in our solution to domain generalization, the RCS divergence is more advantageous than the KL or JS divergence and can lead to better domain generalization results.

5. Conclusion

In this work, we study the domain generalization problem and propose the JPDA solution to better generalize a source trained network classification model to a different but related target domain. Our solution aligns the joint distribution and the product distribution in the neural transformation space, and minimizes the classification loss. Particularly, the two distributions are aligned under the RCS divergence, which is estimated from empirical data via analytically solving an unconstrained quadratic maximization problem. Compared to the related domain alignment works, ours goes beyond domain factorization and conveniently aligns multiple domains without going through the challenging adversarial training. In the experiments, we implement our solution with the shallow 1HLNN, the deep ResNet18 model, and the deeper ResNet50 model, and we find that our solution performs better than the other methods on several multi-domain image classification datasets, which include not only a small number of classes but also a large number of classes.

When multiple domains are quite different from one another and each domain has a large number of classes (e.g., the DomainNet dataset), it may be challenging to find a neural transformation such that the joint distribution and the product distribution are aligned under that transformation. In this case, our solution may not well align the domains and could result in limited improvement of the model performance. Therefore, in the future we intend to draw inspiration from other complementary domain generalization strategies (e.g., data augmentation [26], gradient surgery [32]) to strengthen our solution and better handle such case.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the Technology and Innovation Major Project of the Ministry of Science and Technology of China under Grant 2020AAA0108404, in part by the National Natural Science Foundation of China under Grant 62106137, and in part by the Shantou University under Grant NTF21035.

References

[1] V.N. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.
[2] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, N.D. Lawrence, Dataset Shift in Machine Learning, MIT Press, 2008.
[3] J.G. Moreno-Torres, T. Raeder, R. Alaiz-Rodriguez, N.V. Chawla, F. Herrera, A unifying view on dataset shift in classification, Pattern Recognit. 45 (1) (2012) 521–530.
[4] A. Torralba, A.A. Efros, Unbiased look at dataset bias, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1521–1528.
[5] 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.
[6] J. Zhang, L. Qi, Y. Shi, Y. Gao, Generalizable model-agnostic semantic segmentation via target-specific normalization, Pattern Recognit. 122 (2022) 108292.
[7] 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.
[8] J. Liang, R. He, Z. Sun, T. Tan, Exploring uncertainty in pseudo-label guided unsupervised domain adaptation, Pattern Recognit. 96 (2019) 106996.

[9] S. Chen, M. Harandi, X. Jin, X. Yang, Domain adaptation by joint distribution invariant projections, IEEE Trans. Image Process. 29 (2020) 8264–8277.
[10] K. Zhou, Y. Yang, Y. Qiao, T. Xiang, Domain generalization with mixstyle, in: International Conference on Learning Representations, 2021, pp. 1–12.
[11] K. Muandet, D. Balduzzi, B. Schölkopf, Domain generalization via invariant feature representation, in: International Conference on Machine Learning, vol. 28, 2013, pp. 10–18.
[12] M. Ghifary, D. Balduzzi, W.B. Kleijn, M. Zhang, Scatter component analysis: a unified framework for domain adaptation and domain generalization, IEEE Trans. Pattern Anal. Mach. Intell. 39 (7) (2017) 1414–1430.
[13] 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.
[14] S. Zhao, M. Gong, T. Liu, H. Fu, D. Tao, Domain generalization via entropy regularization, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 3118–3129.
[15] A.T. Nguyen, T. Tran, Y. Gal, A.G. Baydin, Domain invariant representation learning with domain density transformations, Advances in Neural Information Processing Systems, vol. 34, 2021.
[16] 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.
[17] 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.
[18] M. Yamada, T. Suzuki, T. Kanamori, H. Hachiya, M. Sugiyama, Relative density-ratio estimation for robust distribution comparison, Neural Comput. 25 (5) (2013) 1324–1370.
[19] 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.
[20] A. Khosla, T. Zhou, T. Malisiewicz, A.A. Efros, A. Torralba, Undoing the damage of dataset bias, in: European Conference on Computer Vision, 2012, pp. 158–171.
[21] 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.
[22] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. 17 (59) (2016) 1–35.
[23] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, B. Schölkopf, Domain adaptation with conditional transferable components, in: International Conference on Machine Learning, 2016, pp. 2839–2848.
[24] D. Li, Y. Yang, Y.-Z. Song, T. Hospedales, Learning to generalize: meta-learning for domain generalization, in: AAAI Conference on Artificial Intelligence, vol. 32, 2018.
[25] 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.
[26] 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.
[27] 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.
[28] F.-E. Yang, Y.-C. Cheng, Z.-Y. Shiau, Y.-C.F. Wang, Adversarial teacher-student representation learning for domain generalization, Advances in Neural Information Processing Systems, vol. 34, 2021.
[29] M.M. Rahman, C. Fookes, M. Baktashmotlagh, S. Sridharan, Correlation-aware adversarial domain adaptation and generalization, Pattern Recognit. 100 (2020) 107124.
[30] D. Kim, Y. Yoo, S. Park, J. Kim, J. Lee, SelfReg: self-supervised contrastive regularization for domain generalization, in: IEEE International Conference on Computer Vision, 2021, pp. 9619–9628.
[31] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, P. Yu, Generalizing to unseen domains: a survey on domain generalization, IEEE Trans. Knowl. Data Eng. (2022) 1–14, doi:10.1109/TKDE.2022.3178128.
[32] 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.
[33] S. Bickel, J. Bogojeska, T. Lengauer, T. Scheffer, Multi-task learning for HIV therapy screening, in: International Conference on Machine learning, 2008, pp. 56–63.
[34] M. Baktashmotlagh, M. Harandi, M. Salzmann, Distribution-matching embedding for visual domain adaptation, J. Mach. Learn. Res. 17 (108) (2016) 1–30.
[35] Q. Que, M. Belkin, Back to the future: radial basis function network revisited, IEEE Trans. Pattern Anal. Mach. Intell. 42 (8) (2020) 1856–1867.
[36] 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.
[37] 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.
[38] I. Gulrajani, D. Lopez-Paz, In search of lost domain generalization, in: International Conference on Learning Representations, 2021, pp. 1–13.
[39] 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.
[40] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, B. Wang, Moment matching for multi-source domain adaptation, in: International Conference on Computer Vision, 2019, pp. 1406–1415.
[41] H. Wang, X. Bi, Domain generalization and adaptation based on second-order style information, Pattern Recognit. 127 (2022) 108595.
[42] R. Wang, L. Qi, Y. Shi, Y. Gao, Better pseudo-label: joint domain-aware label and dual-classifier for semi-supervised domain generalization, Pattern Recognit. 133 (2022) 108987.
[43] K. Saito, D. Kim, S. Sclaroff, T. Darrell, K. Saenko, Semi-supervised domain adaptation via minimax entropy, in: International Conference on Computer Vision, 2019, pp. 8049–8057.
[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] Y. Balaji, S. Sankaranarayanan, R. Chellappa, Metareg: towards domain generalization using meta-regularization, in: Advances in Neural Information Processing Systems, vol. 31, 2018, pp. 998–1008.
[47] K. Zhou, Y. Yang, T. Hospedales, T. Xiang, Deep domain-adversarial image generation for domain generalisation, in: AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 13025–13032.
[48] 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.
[49] T. Matsuura, T. Harada, Domain generalization using a mixture of multiple latent domains, in: AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 11749–11756.
[50] Z. Huang, H. Wang, E.P. Xing, D. Huang, Self-challenging improves cross–domain generalization, in: European Conference on Computer Vision, 2020, pp. 124–140.
[51] 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.
[52] Z. Xiao, J. Shen, X. Zhen, L. Shao, C. Snoek, A bit more Bayesian: domain-invariant learning with uncertainty, in: International Conference on Machine Learning, vol. 139, 2021, pp. 11351–11361.
[53] 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.
[54] Y.-H. Liu, C.-X. Ren, A two-way alignment approach for unsupervised multi–source domain adaptation, Pattern Recognit. 124 (2022) 108430.
[55] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.

Sentao Chen received the Ph.D. degree in software engineering from South China University of Technology, Guangzhou, China, in 2020. He is currently a Lecturer with the Department of Computer Science, Shantou University, Shantou, China. His research interests include statistical machine learning, domain adaptation, and domain generalization.

Lei Wang received his Ph.D. degree from Nanyang Technological University, Singapore in 2004. Now he works as associate professor at School of Computing and Information Technology of University of Wollongong, Australia. His research interests include pattern recognition, machine/deep learning, computer vision and image retrieval.

Zijie Hong received the B.S. degree in software engineering from South China University of Technology, Guangzhou, China, in 2019. He is currently pursuing the M.Sc. degree in software engineering in the School of Software Engineering, South China University of Technology. His research interests include domain adaptation and computer vision.

Xiaowei Yang received the B.S. degree in theoretical and applied mechanics, the M.Sc. degree in computational mechanics, and the Ph.D. degree in solid mechanics from Jilin University, Changchun, China, in 1991, 1996, and 2000, respectively. He is currently a full-time professor in the School of Software Engineering, South China University of Technology. His current research interests include designs and analyses of algorithms for large-scale pattern recognition, imbalanced learning, semisupervised learning, support vector machines, tensor learning, and evolutionary computation.