Neural Networks 180 (2024) 106739

Contents lists available at ScienceDirect

Neural Networks

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

Joint weight optimization for partial domain adaptation via kernel statistical distance estimation

Sentao Chen

Department of Computer Science, Shantou University, China

ARTICLE INFO

Keywords:
Statistical learning
Partial domain adaptation
Statistical distance estimation
Kernel method

ABSTRACT

The goal of Partial Domain Adaptation (PDA) is to transfer a neural network from a source domain (joint source distribution) to a distinct target domain (joint target distribution), where the source label space subsumes the target label space. To address the PDA problem, existing works have proposed to learn the marginal source weights to match the weighted marginal source distribution to the marginal target distribution. However, this is sub-optimal, since the neural network's target performance is concerned with the joint distribution disparity, not the marginal distribution disparity. In this paper, we propose a Joint Weight Optimization (JWO) approach that optimizes the joint source weights to match the weighted joint source distribution to the joint target distribution in the neural network's feature space. To measure the joint distribution disparity, we exploit two statistical distances: the distribution-difference-based L^2-distance and the distribution-ratio-based χ^2-divergence. Since these two distances are unknown in practice, we propose a Kernel Statistical Distance Estimation (KSDE) method to estimate them from the weighted source data and the target data. Our KSDE method explicitly expresses the two estimated statistical distances as functions of the joint source weights. Therefore, we can optimize the joint weights to minimize the estimated distance functions and reduce the joint distribution disparity. Finally, we achieve the PDA goal by training the neural network on the weighted source data. Experiments on several popular datasets are conducted to demonstrate the effectiveness of our approach. Intro video and Pytorch code are available at https://github.com/sentaochen/Joint-Weight-Optimation. Interested readers can also visit https://github.com/sentaochen for more source codes of the related domain adaptation, multi-source domain adaptation, and domain generalization approaches.

1. Introduction

Statistical learning background. Traditional classification methods generally assume that the joint training distribution $p^{tr}(\boldsymbol{x}, y)$ that independently generates the training data $\{(\boldsymbol{x}_i^{tr}, y_i^{tr})\}_{i=1}^{m_{tr}}$, is identical with the joint testing distribution $p^{te}(\boldsymbol{x}, y)$ that independently generates the testing data $\{(\boldsymbol{x}_i^{te}, y_i^{te})\}_{i=1}^{m_{te}}$, where $\boldsymbol{x}$ are the features and y is the label (Schölkopf & Smola, 2001; Vapnik, 1998). Under such assumption, a classifier g learned by minimizing the training loss $\frac{1}{m_{tr}}\sum_{i=1}^{m_{tr}} \mathcal{L}(g(\boldsymbol{x}_i^{tr}), y_i^{tr})$, will also have a small testing loss $\frac{1}{m_{te}}\sum_{i=1}^{m_{te}} \mathcal{L}(g(\boldsymbol{x}_i^{te}), y_i^{te})$ and a good testing performance in a statistical sense, where $\mathcal{L}$ represents the loss function (Schölkopf & Smola, 2001; Vapnik, 1998). This is because when the two joint distributions $p^{tr}(\boldsymbol{x}, y)$ and $p^{te}(\boldsymbol{x}, y)$ are identical, *i.e.*, $p^{tr}(\boldsymbol{x}, y) = p^{te}(\boldsymbol{x}, y)$, according to the statistical theory (Wasserman, 2004), it holds that $\frac{1}{m_{te}}\sum_{i=1}^{m_{te}} \mathcal{L}(g(\boldsymbol{x}_i^{te}), y_i^{te}) \approx \frac{1}{m_{tr}}\sum_{i=1}^{m_{tr}} \mathcal{L}(g(\boldsymbol{x}_i^{tr}), y_i^{tr})$. In real-world settings, however, the joint training distribution and joint testing distribution may not be identical (Jiang, 2008; Quiñonero-Candela, Sugiyama, Schwaighofer, & Lawrence, 2008), which results in the poor testing performance of the classifier. Therefore, developing methods that can handle the non-identically-distributed learning problems, has become a crucial research focus in the communities of machine learning, pattern recognition, and computer vision (Acuna, Zhang, Law, & Fidler, 2021; Andéol et al., 2023; Nguyen, Tran, Gal, Torr, & Baydin, 2022; Wen, Chen, Xie, Liu and Zheng, 2024).

The PDA problem. Our interest in this work is Partial Domain Adaptation (PDA) (Cao, Long, Wang and Jordan, 2018), which is one of the non-identically-distributed learning problems. In the PDA problem, given a labeled dataset from a source domain (joint source distribution) $p^s(\boldsymbol{x}, y)$, and an unlabeled dataset from a target domain (joint target distribution) $p^t(\boldsymbol{x}, y)$, the goal is to transfer a neural network from the source to the target, such that the network has a small target loss, namely, a good target performance. The challenges here are that the joint source distribution $p^s(\boldsymbol{x}, y)$ is distinct from the joint target distribution $p^t(\boldsymbol{x}, y)$, *i.e.*, $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$, and that the source label space $\mathcal{Y}^s$ is not identical with the target label space $\mathcal{Y}^t$ and contains it as a subset, *i.e.*, $\mathcal{Y}^t \subset \mathcal{Y}^s$ (hence the name "Partial Domain Adaptation").

E-mail address: sentaochenmail@gmail.com.

https://doi.org/10.1016/j.neunet.2024.106739
Received 3 March 2024; Received in revised form 4 September 2024; Accepted 12 September 2024
Available online 16 September 2024

Existing PDA works. To address the PDA problem, existing works have proposed to learn the marginal source weights to match the weighted marginal source distribution $w(\boldsymbol{x})p^s(\boldsymbol{x})$ to the marginal target distribution $p^t(\boldsymbol{x})$ to reduce the marginal distribution disparity, and train the neural network (Cao, Ma, Long and Wang, 2018; Cao, You, Long, Wang, & Yang, 2019; Cao, You, Zhang, Wang, & Long, 2023; Gu, Yu, Yang, Sun, & Xu, 2021; Hu et al., 2020; Zhang, Ding, Li, & Ogunbona, 2018). The marginal source weights are the values of the marginal weight function $w(\boldsymbol{x})$ evaluating at the unlabeled source data. These weights try to decrease the impact of the source data whose labels only exist in the source label space (the source data whose labels belong to the difference set $\mathcal{Y}^s \setminus \mathcal{Y}^t$), since these source data are irrelevant to the target domain and can cause the negative transfer issue (Cao, Long et al., 2018). To be specific, the weighted marginal source distribution and the marginal target distribution are matched by minimizing their disparity, which is measured by statistical distances including the Jensen–Shannon (JS) divergence (Cao, Long et al., 2018; Cao, Ma et al., 2018; Cao et al., 2019, 2023; Hu et al., 2020; Liang, Wang, Hu, He, & Feng, 2020; Zhang et al., 2018), the Maximum Mean Discrepancy (MMD) (Li et al., 2021; Ma et al., 2022; Ren, Ge, Yang, & Yan, 2021), and the Wasserstein distance (Gu et al., 2021). Since the JS divergence and Wasserstein distance are usually expressed as the maximal values of functionals with respect to the input functions (discriminator networks), as noted by Chen (2023) and Nowozin, Cseke, and Tomioka (2016), minimizing these statistical distances therefore leads to the adversarial minimax problem (see detailed discussions in the first paragraph of Section 3.2). Here, a "functional" refers to a function that takes function(s) as its input(s) (Schölkopf & Smola, 2001). Then, based on the outputs of the network's classifier (Cao, Long et al., 2018; Cao, Ma et al., 2018; Hu et al., 2020; Liang et al., 2020; Ma et al., 2022; Ren et al., 2021) or the outputs of the discriminator network (Cao et al., 2019; Zhang et al., 2018), the marginal source weights are learned. While these works have brought the important weighting idea to PDA and achieved promising experimental results, they do not address the disparity between the joint source distribution and the joint target distribution (the joint distribution disparity), which is directly and strongly connected to the neural network's target performance. In the related Domain Adaptation (DA) literature (Bhushan Damodaran, Kellenberger, Flamary, Tuia, & Courty, 2018; Chen, Harandi, Jin, & Yang, 2020; Chen, Hong, Harandi and Yang, 2023; Chen, Zheng and Wu, 2023; Courty, Flamary, Habrard, & Rakotomamonjy, 2017; Turrisi, Flamary, Rakotomamonjy, & Pontil, 2022), it has been clearly and consistently demonstrated that reducing the joint distribution disparity is critical and leads to better target performance than reducing the marginal distribution disparity. Furthermore, most of these works usually learn the marginal source weights in a heuristic manner, and do not have theoretical justifications for the weight learning methods.

Our contributions. In this paper, we address the aforementioned limitations and propose a Joint Weight Optimization (JWO) approach that optimizes the joint source weights to match the weighted joint source distribution $w(\boldsymbol{x},y)p^s(\boldsymbol{x},y)$ to the joint target distribution $p^t(\boldsymbol{x},y)$ in the neural network's feature space. The joint source weights are the values of the joint weight function $w(\boldsymbol{x},y)$ evaluating at the labeled source data. To measure the joint distribution disparity, we utilize two statistical distances: the distribution-difference-based L^2-distance and the distribution-ratio-based χ^2-divergence. These two distances have been successfully applied to various machine learning problems (Chen & Chen, 2023; Chen, Harandi, Jin, & Yang, 2021; Chen, Wang, Hong and Yang, 2023; Suzuki & Sugiyama, 2013; Tangkaratt, Xie, & Sugiyama, 2014; Torkkola, 2003). Importantly, we show that their estimates can be explicitly expressed as functions of the joint source weights, which motives us to utilize them in this work. Since the L^2-distance and χ^2-divergence are unknown in practice, we propose a Kernel Statistical Distance Estimation (KSDE) method to estimate them from the weighted source data and the target data. Our KSDE method first expresses the two distances as two quadratic functionals' maximal values, approximates the expectations in the functionals by the averages of data, then exploits a kernel-based function as the functionals's inputs, and finally solves the resulting quadratic maximization problems with analytic solutions. The analytic solutions are important to our JWO approach, since they allow us to explicitly express the two estimated statistical distances as functions of the joint source weights. Therefore, we can optimize the joint source weights to minimize the estimated distance functions and reduce the disparity between the weighted joint source distribution and the joint target distribution. With other distances, such analytic solutions may not be possible (see Remark 5 for the example with JS divergence). Eventually, to achieve the PDA goal, we train the neural network by minimizing the network's weighted source loss, as well as the target conditional entropy loss (Grandvalet & Bengio, 2004), which is a semi-supervised learning loss commonly employed in the PDA works (Cao, Long et al., 2018; Cao et al., 2019; Gu et al., 2021; Liang et al., 2020; Wen, Chen, Hong and Zheng, 2024; Zhang et al., 2018). Clearly, our JWO approach overcomes the limitations of existing works by reducing the joint distribution disparity and providing theoretically justified weight optimization methods. Also see Remark 3, the second paragraphs of Sections 3.1 and 3.2 for detailed discussions of our work against the DA and other PDA works. We perform experiments on several popular datasets, and demonstrate the benefits of our approach by contrasting its performance with other methods. For clarity, we summarize our contributions as follows.

- **Joint weight optimization.** We propose the JWO approach that optimizes the joint source weights to match the weighted joint source distribution to the joint target distribution under both the L^2-distance and the χ^2-divergence.
- **Kernel statistical distance estimation.** We propose the KSDE method to estimate both the L^2-distance and the χ^2-divergence by solving two quadratic maximization problems with analytic solutions, which allow us to explicitly express the two estimated statistical distances as functions of the joint source weights.

2. Joint weight optimization

2.1. Problem definition

In PDA, one is given a labeled dataset $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{m_s}$ generated by the joint source distribution $p^s(\boldsymbol{x},y)$, and an unlabeled dataset $\mathcal{D}^u = \{\boldsymbol{x}_i^t\}_{i=1}^{m_t}$ generated by the marginal target distribution $p^t(\boldsymbol{x})$, which is obtained by marginalizing the joint target distribution $p^t(\boldsymbol{x},y)$ over y. The goal is to transfer a neural network $f(\boldsymbol{x})$ from the labeled source to the unlabeled target, such that the network has a small target loss $\frac{1}{m_t}\sum_{i=1}^{m_t} \mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t)$, where $\mathcal{L}$ is the loss function and y_i^t is the target label unknown during the training stage. That is, the network $f(\boldsymbol{x})$ should have a good target performance and well predicts the target labels. The challenges in this problem are that the joint source distribution is distinct from the joint target distribution, *i.e.*, $p^s(\boldsymbol{x},y) \neq p^t(\boldsymbol{x},y)$, and that the source label space $\mathcal{Y}^s$ is also distinct from the target label space $\mathcal{Y}^t$ and contains it as a subset, *i.e.*, $\mathcal{Y}^t \subset \mathcal{Y}^s$.

2.2. Problem analysis

Weighted joint distribution matching. We write the neural network as $f(\boldsymbol{x}) = g(h(\boldsymbol{x}))$, where h is the feature extractor generating the feature space and g is the classifier producing the probabilistic prediction. Considering that the joint source distribution is distinct from the joint target distribution and that the source label space contains the target label space, we propose to optimize the joint source weights $w_1, \ldots, w_{m_s}$ to match the weighted joint source distribution $w(\boldsymbol{x},y)p^s(\boldsymbol{x},y)$ to the joint target distribution $p^t(\boldsymbol{x},y)$ in the network's feature space such that $w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y) \approx p^t(h(\boldsymbol{x}),y)$. Here, the joint source weights are the values of the joint weight function $w(h(\boldsymbol{x}),y)$

evaluating at the labeled source data, *i.e.*, $w_i = w(h(\boldsymbol{x}_i^s), y_i^s)$ for $i = 1, \dots, m_s$. For a source data point $(\boldsymbol{x}_i^s, y_i^s)$ irrelevant to the target domain, *i.e.*, $y_i^s \in (\mathcal{Y}^s \setminus \mathcal{Y}^t)$, the target label distribution will be $p^t(y_i^s) = 0$, and as a result the joint source weight will also be $w_i = w(h(\boldsymbol{x}_i^s), y_i^s) \approx 0$, since $w(h(\boldsymbol{x}_i^s), y_i^s)p^s(h(\boldsymbol{x}_i^s), y_i^s) \approx p^t(h(\boldsymbol{x}_i^s), y_i^s) = p^t(h(\boldsymbol{x}_i^s)|y_i^s)p^t(y_i^s) = 0$. Therefore, the joint source weights can decrease or even eliminate the impact of the irrelevant source data (see "**weight visualization**" in Section 4.3 for the empirical evidence).

The PDA goal. Thanks to the weighted joint distribution matching $w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) \approx p^t(h(\boldsymbol{x}), y)$, we have the following equations:

$$\frac{1}{m_t}\sum_{i=1}^{m_t}\mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t)$$

$$\approx \int \mathcal{L}(g(h(\boldsymbol{x})), y)p^t(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy \tag{1}$$

$$\approx \int w(h(\boldsymbol{x}), y)\mathcal{L}(g(h(\boldsymbol{x})), y)p^s(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy \tag{2}$$

$$\approx \frac{1}{m_s}\sum_{i=1}^{m_s} w_i\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s), \tag{3}$$

where the two expectations in Eqs. (1) and (2) are approximated by the averages of data. From these equations, we know that to achieve the PDA goal: training a neural network with a small target loss $\frac{1}{m_t}\sum_{i=1}^{m_t}\mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t)$, one should minimize the neural network's weighted source loss $\frac{1}{m_s}\sum_{i=1}^{m_s} w_i\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s)$. Since this loss involves the joint source weights, in the following two subsections, we conduct weighted joint distribution matching under different statistical distances to learn the joint source weights.

2.3. Joint weight optimization with L^2-distance

The L^2-distance. We first conduct weighted joint distribution matching under the L^2-distance. In particular, we propose to optimize the joint source weights to minimize the L^2-distance between the weighted joint source distribution $w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y)$ and the joint target distribution $p^t(h(\boldsymbol{x}), y)$. The L^2-distance here is defined as

$$L^2\big(w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y), p^t(h(\boldsymbol{x}), y)\big)$$

$$= \int \big(w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) - p^t(h(\boldsymbol{x}), y)\big)^2 dh(\boldsymbol{x})dy. \tag{4}$$

This statistical distance is defined based on the distribution difference $w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) - p^t(h(\boldsymbol{x}), y)$. It is symmetric and non-negative. When two distributions are matched and become identical, the L^2-distance reduces to zero.

Kernel L^2-distance estimation. Since the L^2-distance in Eq. (4) is unknown, we propose to estimate it from the weighted source data and the target data, which reflect the weighted joint source distribution and the joint target distribution. The estimation procedure is conducted as follows.

$$L^2\big(w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y), p^t(h(\boldsymbol{x}), y)\big)$$

$$= \int \max_r \Big[2\big(w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) - p^t(h(\boldsymbol{x}), y)\big)r(h(\boldsymbol{x}), y) - r(h(\boldsymbol{x}), y)^2\Big]dh(\boldsymbol{x})dy \tag{5}$$

$$= \max_r\Big(2\int w(h(\boldsymbol{x}), y)r(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy - 2\int r(h(\boldsymbol{x}), y)p^t(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy - \int r(h(\boldsymbol{x}), y)^2 dh(\boldsymbol{x})dy\Big) \tag{6}$$

$$\approx \max_{\boldsymbol{\alpha}}\Big(\frac{2}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\alpha}) - \frac{2}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\alpha}) - \int r(h(\boldsymbol{x}), y; \boldsymbol{\alpha})^2 dh(\boldsymbol{x})dy\Big) \tag{7}$$

$$= \max_{\boldsymbol{\alpha}}\Big(\frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\boldsymbol{\alpha} - \frac{2}{m_t}\boldsymbol{1}^\top\boldsymbol{K}^t\boldsymbol{\alpha} - \boldsymbol{\alpha}^\top\boldsymbol{H}\boldsymbol{\alpha}\Big) \tag{8}$$

$$= \frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\hat{\boldsymbol{\alpha}} - \frac{2}{m_t}\boldsymbol{1}^\top\boldsymbol{K}^t\hat{\boldsymbol{\alpha}} - \hat{\boldsymbol{\alpha}}^\top\boldsymbol{H}\hat{\boldsymbol{\alpha}}. \tag{9}$$

Eq. (5) is derived from Eq. (4) by leveraging the equation $u^2 = \max_v(2uv - v^2)$ and regarding the distribution difference $w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) - p^t(h(\boldsymbol{x}), y)$ as u and the function $r(h(\boldsymbol{x}), y)$ as v. Eq. (6) expresses the L^2-distance in Eq. (4) as a quadratic functional's maximal value, where the maximal value is reached when the functional's input function $r(h(\boldsymbol{x}), y) = w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) - p^t(h(\boldsymbol{x}), y)$. Eq. (7) approximates the expectations in Eq. (6) by the averages of the labeled source dataset $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{m_s}$ and the labeled target dataset $\mathcal{D}^t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{m_t}$, and evaluates the joint weight function at the labeled source data to obtain the joint source weights $w_1, \dots, w_{m_s}$. The labeled target dataset is assumed to be available here so that we can focus on the estimation. Later in Section 2.5, we discuss how it is obtained. Besides, Eq. (7) exploits a kernel-based function $r(h(\boldsymbol{x}), y; \boldsymbol{\alpha}) = \sum_{i=1}^{m_s}\alpha_i k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i^s))k_y(y, y_i^s)$ as the model for $r(h(\boldsymbol{x}), y)$, where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_{m_s})^\top$ are the function parameters, $k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i^s)) = \exp(-\|h(\boldsymbol{x}) - h(\boldsymbol{x}_i^s)\|_2^2/\sigma)$ is the radial basis function kernel with kernel width σ (> 0), and $k_y(y, y_i^s)$ is the delta kernel that evaluates 1 if $y = y_i^s$ and 0 otherwise. Later in Remark 1, we explain the reasons for choosing the radial basis function kernel and the delta kernel. Eq. (8) introduces matrix and vector notations, where $\boldsymbol{w} = (w_1, \dots, w_{m_s})^\top$ are the joint source weights, $\boldsymbol{1}$ is an m_t-dimensional column vector of ones, and $\boldsymbol{K}^s \in \mathbb{R}^{m_s\times m_s}$, $\boldsymbol{K}^t \in \mathbb{R}^{m_t\times m_s}$, $\boldsymbol{H} \in \mathbb{R}^{m_s\times m_s}$ are three matrices. The (i, j)-th elements of $\boldsymbol{K}^s$, $\boldsymbol{K}^t$ are respectively defined as

$$(\boldsymbol{K}^s)_{ij} = k_x(h(\boldsymbol{x}_i^s), h(\boldsymbol{x}_j^s))k_y(y_i^s, y_j^s), \tag{10}$$

$$(\boldsymbol{K}^t)_{ij} = k_x(h(\boldsymbol{x}_i^t), h(\boldsymbol{x}_j^s))k_y(y_i^t, y_j^s). \tag{11}$$

It is straightforward to verify that $\frac{2}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\alpha}) = \frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\boldsymbol{\alpha}$ and $\frac{2}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\alpha}) = \frac{2}{m_t}\boldsymbol{1}^\top\boldsymbol{K}^t\boldsymbol{\alpha}$. In addition, the (i, j)-th element of $\boldsymbol{H}$ is defined as

$$(\boldsymbol{H})_{ij} = \int k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i^s))k_y(y, y_i^s) \times k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_j^s))k_y(y, y_j^s)dh(\boldsymbol{x})dy \tag{12}$$

$$= \int k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i^s))k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_j^s))dh(\boldsymbol{x}) \times \sum_y k_y(y, y_i^s)k_y(y, y_j^s) \tag{13}$$

$$= \Big(\frac{\pi\sigma}{2}\Big)^{\frac{d}{2}}\exp\Big(\frac{-\|h(\boldsymbol{x}_i^s) - h(\boldsymbol{x}_j^s)\|_2^2}{2\sigma}\Big)k_y(y_i^s, y_j^s), \tag{14}$$

where π is the ratio of a circle's circumference to its diameter and d is the dimensionality of the neural network's features $h(\boldsymbol{x})$. One can also verify that $\int r(h(\boldsymbol{x}), y; \boldsymbol{\alpha})^2 dh(\boldsymbol{x})dy = \boldsymbol{\alpha}^\top\boldsymbol{H}\boldsymbol{\alpha}$. Eq. (9) solves the quadratic maximization problem in Eq. (8) with the analytic solution

$$\hat{\boldsymbol{\alpha}} = (\boldsymbol{H} + \epsilon\mathbf{I})^{-1}\Big(\frac{1}{m_s}(\boldsymbol{K}^s)^\top\boldsymbol{w} - \frac{1}{m_t}(\boldsymbol{K}^t)^\top\boldsymbol{1}\Big). \tag{15}$$

Here, $\epsilon\mathbf{I}$ is a diagonal matrix included to avoid overfitting and ensure invertability, where $\mathbf{I}$ is the identity matrix and ϵ is a small positive value. Thanks to the analytic solution $\hat{\boldsymbol{\alpha}}$, the estimated L^2-distance in Eq. (9) is explicitly expressed as a quadratic function of $\boldsymbol{w}$.

Joint weight optimization with L^2-distance. Given the estimated L^2-distance in Eq. (9), we present the optimization problem of our approach with the L^2-distance as

$$\min_{\boldsymbol{w}\in\Delta} f_{L^2}(\boldsymbol{w}) = \frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\hat{\boldsymbol{\alpha}} - \frac{2}{m_t}\boldsymbol{1}^\top\boldsymbol{K}^t\hat{\boldsymbol{\alpha}} - \hat{\boldsymbol{\alpha}}^\top\boldsymbol{H}\hat{\boldsymbol{\alpha}}. \tag{16}$$

Here, $f_{L^2}(\boldsymbol{w})$ is the objective function to optimize and $\Delta = \{\boldsymbol{w} = (w_1, \dots, w_{m_s})^\top | w_i \geq 0, \sum_{i=1}^{m_s} w_i = m_s\}$ is the constraint set. The constraint set Δ is introduced for the joint source weights $\boldsymbol{w}$ due to the following two considerations. (1) The joint weight function $w(h(\boldsymbol{x}), y)$ should be non-negative and hence $w_i = w(h(\boldsymbol{x}_i^s), y_i^s) \geq 0$. (2) The weighted joint source distribution $w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y)$ should be a probability distribution that sums up to one and hence $\frac{1}{m_s}\sum_{i=1}^{m_s} w_i \approx$

```
Algorithm 1 JWO with $L^2$-distance or $\chi^2$-divergence
Input: Labeled source and labeled target datasets $\mathcal{D}^s$, $\mathcal{D}^t$.
Output: Joint source weights $\boldsymbol{w}$.
  1: Compute $f_{L^2}(\boldsymbol{w})$ in (16) or $f_{\chi^2}(\boldsymbol{w})$ in (24), where $\boldsymbol{K}^s$, $\boldsymbol{K}^t$, $\boldsymbol{H}$, $\widehat{\boldsymbol{\alpha}}$, and
     $\widehat{\boldsymbol{\beta}}$ are defined in (10)-(15), and (23).
  2: Solve minimization problem (16) or (24) by the L-BFGS algorithm (Nocedal & Wright, 1999) to return the optimal solution
     $\boldsymbol{w}$.
```

$\int w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy = 1$. We optimize the joint source weights $\boldsymbol{w}$ to minimize the objective function $f_{L^2}(\boldsymbol{w})$ (the estimated L^2-distance) to reduce the disparity between the weighted joint source distribution and the joint target distribution. To solve minimization problem (16), inspired by Chen (2024) and Li, Murias, Major, Dawson, and Carlson (2019), we first utilize the softmax function to absorb the constraint set Δ, *i.e.*, $w_i \leftarrow m_s \exp(w_i)/\sum_{j=1}^{m_s}\exp(w_j)$, and then solve the resulting unconstrained problem via the L-BFGS algorithm (Nocedal & Wright, 1999), which is implemented in many optimization packages, *e.g.*, pytorch-minimize.[1] To enhance clarity, we summarize our Joint Weight Optimization (JWO) approach with L^2-distance in Algorithm 1.

Remark 1. We explain the reasons for choosing the radial basis function kernel and the delta kernel in the kernel based function $r(h(\boldsymbol{x}),y;\boldsymbol{\alpha})$ from Eq. (7). First, it is common to choose a radial basis function kernel (also known as the Gaussian kernel) for a continuous variable like the neural network's features $h(\boldsymbol{x})$ (Li et al., 2021; Ma et al., 2022; Ren et al., 2021), and a delta kernel for a discrete variable like the class label y (Jin, Yang, Fu, & Chen, 2020; Suzuki & Sugiyama, 2013; Wen, Chen, Xie et al., 2024). Second, the radial basis function kernel and the delta kernel make the third term of Eq. (7) computationally tractable and enable us to compute the matrix $\boldsymbol{H}$ in Eq. (8). Last but not least, in our prior works with the L^2-distance (Chen & Chen, 2023; Chen et al., 2020) and χ^2-divergence (Chen, Hong et al., 2023), such choice has been consistently demonstrated to be advantageous and can yield superior results.

2.4. Joint weight optimization with χ^2-divergence

The χ^2-divergence. We then conduct weighted joint distribution matching under the χ^2-divergence to learn the joint source weights. The χ^2-divergence from the weighted joint source distribution $w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)$ to the joint target distribution $p^t(h(\boldsymbol{x}),y)$ is defined as

$$\chi^2\big(w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)\parallel p^t(h(\boldsymbol{x}),y)\big) = \int\Big[\Big(\frac{w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)}{p^t(h(\boldsymbol{x}),y)}\Big)^2 - 1\Big]p^t(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy, \tag{17}$$

where the symbol "$\parallel$" denotes the divergence between two distributions and is extensively used in prior works (Acuna et al., 2021; Chen, 2023; Nowozin et al., 2016; Zhang et al., 2018). This statistical distance is defined based on the distribution ratio $w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)/p^t(h(\boldsymbol{x}),y)$. It is non-negative and reduces to zero when two distributions are identical. However, unlike the L^2-distance, the χ^2-divergence is an asymmetric statistical distance, which means that the direction in Eq. (17) generally has a different value from the other direction $\chi^2\big(p^t(h(\boldsymbol{x}),y)\parallel w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)\big)$. Later in Remark 2, we explain why we select the current direction, rather than the other one.

[1] https://github.com/rfeinman/pytorch-minimize.

Kernel χ^2-divergence estimation. Since the χ^2-divergence in Eq. (17) is unknown, we propose to estimate it from the weighted source data and the target data. The estimation procedure resembles the procedure of the L^2-distance and is presented as follows.

$$\begin{aligned}
&\chi^2\big(w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)\parallel p^t(h(\boldsymbol{x}),y)\big)\\
&= \int \max_r\Big(2\frac{w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)}{p^t(h(\boldsymbol{x}),y)}r(h(\boldsymbol{x}),y) - r(h(\boldsymbol{x}),y)^2 - 1\Big)p^t(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy && (18)\\
&= \max_r\Big(2\int w(h(\boldsymbol{x}),y)r(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy - \int r(h(\boldsymbol{x}),y)^2p^t(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy\Big) - 1 && (19)\\
&\approx \max_{\boldsymbol{\beta}}\Big(\frac{2}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s),y_i^s;\boldsymbol{\beta}) - \frac{1}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t),y_i^t;\boldsymbol{\beta})^2\Big) - 1 && (20)\\
&= \max_{\boldsymbol{\beta}}\Big(\frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\boldsymbol{\beta} - \frac{1}{m_t}\boldsymbol{\beta}^\top(\boldsymbol{K}^t)^\top\boldsymbol{K}^t\boldsymbol{\beta}\Big) - 1 && (21)\\
&= \frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\widehat{\boldsymbol{\beta}} - \frac{1}{m_t}\widehat{\boldsymbol{\beta}}^\top(\boldsymbol{K}^t)^\top\boldsymbol{K}^t\widehat{\boldsymbol{\beta}} - 1. && (22)
\end{aligned}$$

Eq. (18) is obtained from Eq. (17) by leveraging the equation $u^2 = \max_v(2uv - v^2)$ again and regarding the distribution ratio $w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)/p^t(h(\boldsymbol{x}),y)$ as u and the function $r(h(\boldsymbol{x}),y)$ as v. Eq. (19) expresses the χ^2-divergence in Eq. (17) as a quadratic functional's maximal value, where the maximal value is reached when the functional's input function $r(h(\boldsymbol{x}),y) = w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)/p^t(h(\boldsymbol{x}),y)$. Eq. (20) approximates the expectations in Eq. (19) by the averages of the labeled source and target datasets, and introduces the joint source weights. Moreover, Eq. (20) also uses the kernel-based function $r(h(\boldsymbol{x}),y;\boldsymbol{\beta}) = \sum_{i=1}^{m_s}\beta_i k_x(h(\boldsymbol{x}),h(\boldsymbol{x}_i^s))k_y(y,y_i^s)$ as the model for $r(h(\boldsymbol{x}),y)$, where $\boldsymbol{\beta} = (\beta_1,\ldots,\beta_{m_s})^\top$ are the parameters. Eq. (21) introduces the same notations $\boldsymbol{w}$, $\boldsymbol{K}^s$, and $\boldsymbol{K}^t$ as Eq. (8). Eq. (22) solves the quadratic maximization problem in Eq. (21) with the analytic solution

$$\widehat{\boldsymbol{\beta}} = \big(\frac{1}{m_t}(\boldsymbol{K}^t)^\top\boldsymbol{K}^t + \epsilon\mathbf{I}\big)^{-1}\big(\frac{1}{m_s}(\boldsymbol{K}^s)^\top\boldsymbol{w}\big), \tag{23}$$

where $\epsilon\mathbf{I}$ is a small diagonal matrix included for avoiding overfitting and ensuring invertability. Same as the estimated L^2-distance in Eq. (9), the estimated χ^2-divergence in Eq. (22) is also explicitly expressed as a quadratic function of $\boldsymbol{w}$ due to the analytic solution $\widehat{\boldsymbol{\beta}}$.

Joint weight optimization with χ^2-divergence. Given the estimated χ^2-divergence in Eq. (22), we present the optimization problem of our approach with the χ^2-divergence as

$$\min_{\boldsymbol{w}\in\Delta} f_{\chi^2}(\boldsymbol{w}) = \frac{2}{m_s}\boldsymbol{w}^\top\boldsymbol{K}^s\widehat{\boldsymbol{\beta}} - \frac{1}{m_t}\widehat{\boldsymbol{\beta}}^\top(\boldsymbol{K}^t)^\top\boldsymbol{K}^t\widehat{\boldsymbol{\beta}} - 1, \tag{24}$$

where $f_{\chi^2}(\boldsymbol{w})$ is the objective function and Δ is the constraint set. We optimize $\boldsymbol{w}$ to minimize $f_{\chi^2}(\boldsymbol{w})$ (the estimated χ^2-divergence) to reduce the disparity between the weighted joint source distribution and the joint target distribution. To solve minimization problem (24), we again exploit the softmax function to reflect the constraint set and solve the resulting unconstrained problem via the L-BFGS algorithm. To enhance clarity, we summarize our JWO approach with χ^2-divergence in Algorithm 1.

Remark 2. We explain the reason for choosing the direction in Eq. (17), rather than the other direction. Following a similar procedure in Eqs. (18)–(22), the other direction of the χ^2-divergence is estimated as

$$\chi^2\big(p^t(h(\boldsymbol{x}),y)\parallel w(h(\boldsymbol{x}),y)p^s(h(\boldsymbol{x}),y)\big) \approx \frac{2}{m_t}\mathbf{1}^\top\boldsymbol{K}^t\widehat{\boldsymbol{\beta}}_{\boldsymbol{W}} - \frac{1}{m_s}\widehat{\boldsymbol{\beta}}_{\boldsymbol{W}}^\top(\boldsymbol{K}^s)^\top\boldsymbol{W}\boldsymbol{K}^s\widehat{\boldsymbol{\beta}}_{\boldsymbol{W}} - 1, \tag{25}$$

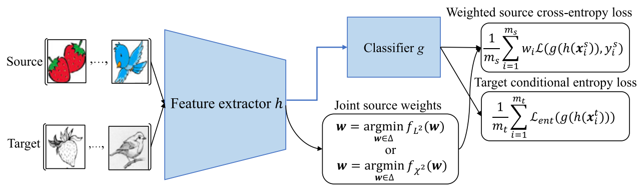

Source
,...,
Target
,...,
Feature extractor h
Classifier g
Weighted source cross-entropy loss
$\frac{1}{m_s}\sum_{i=1}^{m_s} w_i\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s)$
Target conditional entropy loss
$\frac{1}{m_t}\sum_{i=1}^{m_t} \mathcal{L}_{ent}(g(h(\boldsymbol{x}_i^t)))$
Joint source weights
$\boldsymbol{w} = \operatorname{argmin}_{\boldsymbol{w}\in\Delta} f_{L^2}(\boldsymbol{w})$
or
$\boldsymbol{w} = \operatorname{argmin}_{\boldsymbol{w}\in\Delta} f_{\chi^2}(\boldsymbol{w})$

Fig. 1. Graphical illustration of the input data (*e.g.*, images), the neural network $f(\boldsymbol{x}) = g(h(\boldsymbol{x}))$, the joint source weights $\boldsymbol{w}$, and the two loss terms involved in our network training objective.

where $\widehat{\boldsymbol{\beta}}_{\boldsymbol{W}} = \left(\frac{1}{m_s}(\boldsymbol{K}^s)^\top \boldsymbol{W}\boldsymbol{K}^s + \epsilon\mathbf{I}\right)^{-1}\left(\frac{1}{m_t}(\boldsymbol{K}^t)^\top\mathbf{1}\right)$ and $\boldsymbol{W} = \operatorname{diag}(w_1, \ldots, w_{m_s})$ is a diagonal matrix of the joint source weights. Same as the objective function in (24), the estimated value in (25) can also serve as the objective function of the joint source weights $\boldsymbol{W}$. Comparing the two objective functions in (24) and (25), we find that the matrix inverse in $\widehat{\boldsymbol{\beta}}$ is independent of $\boldsymbol{w}$ while the one in $\widehat{\boldsymbol{\beta}}_{\boldsymbol{W}}$ is dependent on $\boldsymbol{W}$. Hence, during the weight optimization procedure, the matrix inverse in (24) only needs to be executed once, while the one in (25) needs to be executed as many times as the iterations of optimization algorithm. This suggests that optimization with (24) has less computational cost than with (25). For this reason, we prefer the direction in Eq. (17) and the corresponding objective function in (24).

Remark 3. Our PDA work in this paper is connected to our previous DA works (Chen et al., 2020; Chen, Hong et al., 2023) in the joint distribution matching idea. However, our PDA work optimizes the weights to match joint distributions, while our DA works optimize the projection matrices (Chen et al., 2020) or the feature extractor (Chen, Hong et al., 2023) to match joint distributions. Additionally, our PDA work estimates the L^2-distance and the χ^2-divergence between the weighted joint source distribution and the joint target distribution, and expresses the estimated statistical distances as functions of the weights. By contrast, our DA works estimate the L^2-distance (Chen et al., 2020) and the χ^2-divergence (Chen, Hong et al., 2023) between the joint source distribution and the joint target distribution, and express the estimated distances as functions of the projection matrices (Chen et al., 2020) or the feature extractor (Chen, Hong et al., 2023).

2.5. Network training

Network training objective. According to the problem analysis, particularly, the second paragraph of Section 2.2, we train the neural network $f(\boldsymbol{x}) = g(h(\boldsymbol{x}))$ by minimizing the network's weighted source loss, and present the optimization problem as follows

$$\min_{g,h} \frac{1}{m_s}\sum_{i=1}^{m_s} w_i\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s) + \frac{\lambda}{m_t}\sum_{i=1}^{m_t}\mathcal{L}_{ent}(g(h(\boldsymbol{x}_i^t))). \tag{26}$$

Here, λ (> 0) is a tradeoff hyperparameter, and $\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s) = -\sum_{k=1}^{|\mathcal{Y}^s|}\mathbb{I}_{\{y_i^s=k\}}\log g_k(h(\boldsymbol{x}_i^s))$ is implemented as the cross-entropy loss, where $g_k(h(\boldsymbol{x}_i^s)) = p(y = k|\boldsymbol{x}_i^s; f = g\circ h)$ is the neural network's kth probabilistic prediction. Besides, following common practice in prior PDA works (Cao, Long et al., 2018; Cao et al., 2019; Gu et al., 2021; Liang et al., 2020; Wen, Chen, Hong et al., 2024; Zhang et al., 2018), the target conditional entropy loss $\mathcal{L}_{ent}(g(h(\boldsymbol{x}_i^t))) = -\sum_{k=1}^{|\mathcal{Y}^s|} g_k(h(\boldsymbol{x}_i^t))\log g_k(h(\boldsymbol{x}_i^t))$ is also included for the purpose of semi-supervised learning (Grandvalet & Bengio, 2004). Minimizing this loss can reduce the uncertainty of target predictions and encourage the target data to be classified with high confidence. For better performance, interested readers can also explore the squared-loss conditional entropy (Tangkaratt et al., 2014), which is a squared-loss variant of the conditional entropy and is more robust against outliers. For clarity, we illustrate in Fig. 1 the elements involved in our network training objective.

Network training algorithm. To train the neural network, we solve minimization problem (26) by the minibatch SGD algorithm, and

Algorithm 2 Network Training Algorithm

```
Input: Labeled source and unlabeled target datasets $\mathcal{D}^s$, $\mathcal{D}^u$
Output: Trained network $f(\boldsymbol{x}) = g(h(\boldsymbol{x}))$.
 1: Initialize joint source weights $\boldsymbol{w}$ to all ones.
 2: while training does not end do
 3:     for $k$ in $1 : K$ do
 4:         Sample minibatches $\mathcal{D}_k^s$, $\mathcal{D}_k^u$, $\boldsymbol{w}_k$ from $\mathcal{D}^s$, $\mathcal{D}^u$, $\boldsymbol{w}$.
 5:         Compute the objective function in (26) on the minibatches $\mathcal{D}_k^s$,
            $\mathcal{D}_k^u$, $\boldsymbol{w}_k$.
 6:         Take a gradient step to update the parameters of feature
            extractor $h$ and classifier $g$.
 7:     end for
 8:     Construct pseudo labeled target dataset $\mathcal{D}^t$ by using the current
        network to label $\mathcal{D}^u$.
 9:     Optimize joint source weights $\boldsymbol{w}$ by Algorithm 1.
10: end while
```

initialize the joint source weights to all ones. In every iteration of the minibatch SGD, three minibatches of data are sampled from the labeled source dataset, the unlabeled target dataset, and the joint source weights to compute the objective function in (26), where each joint source weight corresponds to each labeled source data point. During the network training, the joint source weights are optimized by executing Algorithm 1. Since the inputs of Algorithm 1 include the labeled target dataset $\mathcal{D}^t$, which is not available in PDA, we therefore construct it by applying the widely adopted pseudo labeling technique (Chen, Hong et al., 2023; Jin et al., 2020; Ren, Luo, & Dai, 2023; Wen, Chen, Xie et al., 2024) to the unlabeled target dataset $\mathcal{D}^u$. To be specific, we use the technique to assign pseudo labels to $\mathcal{D}^u$ and construct the labeled target dataset as $\mathcal{D}^t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{m_t}$, where y_i^t is the pseudo label predicted by the neural network. To enhance clarity, we summarize the network training procedure in Algorithm 2. In "**convergence analysis**" of Section 4.3, we study the algorithm's convergence.

Remark 4. For a very large source domain (*e.g.*, source domain from the DomainNet dataset (Peng et al., 2019)), it may not be feasible to optimize the joint source weights all at once by executing Algorithm 1. In this case, we optimize the joint source weights in batches. Each batch of weights are optimized by first sampling two batches of data from the labeled source dataset and the pseudo labeled target dataset and then executing Algorithm 1 on these two batches.

3. Related work and discussion

3.1. Domain adaptation

DA by feature learning. Domain Adaptation (DA) aims to transfer a neural network from a source domain to a distinct target domain, where the two domains share the same label space. Roughly speaking, most existing DA works can be divided into three lines: marginal matching (*e.g.*, Acuna et al. (2021) and Nguyen et al. (2022)), class-conditional matching (*e.g.*, Ge, Ren, Xu, and Yan (2023) and Ren et al. (2023)), and joint matching (*e.g.*, Bhushan Damodaran et al. (2018), Chen, Hong et al. (2023), Sentao Chen et al. (2023) and Wen, Chen, Xie et al. (2024)). In the first line of works, marginal matching, Nguyen et al. (2022) first showed that the KL divergence between the class-posterior source distribution and the class-posterior target distribution in the feature space is less or equal to the KL divergence between these two distributions in the input space, and then optimized the feature extractor to match the marginal source distribution and the marginal target distribution under the KL divergence, which is estimated using Monte Carlo samples. In the second line of works, class-conditional matching, Ge et al. (2023) minimized the proposed Conditional Maximum Mean Discrepancy (CMMD) to match the class-conditional source distribution and the class-conditional target distribution in the feature space, and maximized the mutual information to extract target discriminant information. The CMMD measures the disparity between class-conditional distributions via their embeddings in the RKHS (Schölkopf & Smola, 2001). In the third line of works, joint matching, Chen, Hong et al. (2023) first showed that the expected target loss is bounded by the expected source loss and the χ^2-divergence from the joint source distribution to the joint target distribution in the feature space, and then optimized the feature extractor to minimize the estimated χ^2-divergence, which is derived from the source data and the target data.

Relationship and difference from the DA works. Our PDA work is related to the third line of DA works (Bhushan Damodaran et al., 2018; Chen, Hong et al., 2023; Sentao Chen et al., 2023; Wen, Chen, Xie et al., 2024) in matching the joint distributions, which, as we have explained and emphasized in Section 1, is important to solving the DA and PDA problems. For example, Refs. Chen, Hong et al. (2023) and Chen (2024) show that matching the joint distributions brings strong DA results. However, our PDA work is different from these DA works (Bhushan Damodaran et al., 2018; Chen, Hong et al., 2023; Sentao Chen et al., 2023; Wen, Chen, Xie et al., 2024) in that our work proposes to optimize the joint source weights, rather than the neural network's feature extractor, to match joint distributions. For tackling the PDA problem where the target label space is only a subset of the source label space, matching the (features and label) joint distributions via the joint source weights is better than via the feature extractor. This is supported by the experimental results in Tables 1–3, which show that our PDA approach JWO is superior to the third line of DA approaches including DNA (Chen, Hong et al., 2023), MIEM (Wen, Chen, Xie et al., 2024), and RRL (Sentao Chen et al., 2023) in tackling the PDA problem. In fact, for the PDA problem where the label spaces $\mathcal{Y}^t \subset \mathcal{Y}^s$, it may be problematic to optimize feature extractor h to match joint distributions $p^s(\boldsymbol{x}, y)$ and $p^t(\boldsymbol{x}, y)$ in the feature space such that $p^s(h(\boldsymbol{x}), y) \approx p^t(h(\boldsymbol{x}), y)$. Because by marginalizing both sides of the equation over $h(\boldsymbol{x})$, *i.e.*, $\int p^s(h(\boldsymbol{x}), y) dh(\boldsymbol{x}) \approx \int p^t(h(\boldsymbol{x}), y) dh(\boldsymbol{x})$, we find that such a solution will result in a paradox: $p^s(y) \approx p^t(y)$, which clearly does not hold when $y \in (\mathcal{Y}^s \setminus \mathcal{Y}^t)$.

3.2. Partial domain adaptation

PDA by instance weighting. As aforementioned in the third paragraph of Section 1, most existing PDA works have proposed to learn the marginal source weights to match the weighted marginal source distribution to the marginal target distribution under different statistical distances, such that the marginal distribution disparity can be reduced (Cao, Long et al., 2018; Cao, Ma et al., 2018; Cao et al., 2019, 2023; Gu et al., 2021; Hu et al., 2020; Li et al., 2021; Liang et al., 2020; Ma et al., 2022; Ren et al., 2021; Zhang et al., 2018). Among these works, we are particularly interested in the works of Gu et al. (2021) and Zhang et al. (2018). Below, we discuss these two works in detail. Zhang et al. (2018) estimated the JS divergence between the weighted marginal source distribution $w(h_s(\boldsymbol{x}))p^s(h_s(\boldsymbol{x}))$ and the marginal target distribution $p^t(h_t(\boldsymbol{x}))$ as

$$\begin{aligned}
&\mathrm{JS}\big(w(h_s(\boldsymbol{x}))p^s(h_s(\boldsymbol{x})) \parallel p^t(h_t(\boldsymbol{x}))\big)\\
&= \frac{1}{2}\max_r \Big(\int w(h_s(\boldsymbol{x})) \log r(h_s(\boldsymbol{x})) p^s(h_s(\boldsymbol{x})) dh_s(\boldsymbol{x})\\
&\qquad + \int \log\big(1 - r(h_t(\boldsymbol{x}))\big) p^t(h_t(\boldsymbol{x})) dh_t(\boldsymbol{x}) \Big) + \log 2 \qquad (27)\\
&\approx \frac{1}{2}\max_{\theta_r} \Big(\frac{1}{m_s}\sum_{i=1}^{m_s} w_i \log r(h_s(\boldsymbol{x}_i^s); \boldsymbol{\theta}_r)\\
&\qquad + \frac{1}{m_t}\sum_{i=1}^{m_t} \log\big(1 - r(h_t(\boldsymbol{x}_i^t); \boldsymbol{\theta}_r)\big) \Big) + \log 2. \qquad (28)
\end{aligned}$$

In Eq. (27), the JS divergence is expressed as the maximal value of a functional with respect to its input function r, and h_s, h_t are the source and target feature extractors. In Eq. (28), the input function is implemented as a three-layer discriminator network with parameters $\boldsymbol{\theta}_r$, and the JS divergence is estimated from the weighted source data and the target data. However, in Eq. (28) the marginal source weights $w_1, \ldots, w_{m_s}$ are not optimized to minimize the estimated JS divergence. Instead, they are learned in a heuristic manner according to the outputs of another three-layer discriminator network. Then, the authors optimized the target feature extractor h_t with parameters $\boldsymbol{\theta}_{h_t}$ to minimize the estimated JS divergence in Eq. (28), resulting in the adversarial minimax problem

$$\begin{aligned}
&\min_{\theta_{h_t}} \max_{\theta_r} \Big(\frac{1}{m_s}\sum_{i=1}^{m_s} w_i \log r(h_s(\boldsymbol{x}_i^s); \boldsymbol{\theta}_r)\\
&\qquad + \frac{1}{m_t}\sum_{i=1}^{m_t} \log\big(1 - r(h_t(\boldsymbol{x}_i^t; \boldsymbol{\theta}_{h_t}); \boldsymbol{\theta}_r)\big) \Big). \qquad (29)
\end{aligned}$$

Gu et al. (2021) estimated the Wasserstein distance between the weighted marginal source distribution $w(h(\boldsymbol{x}))p^s(h(\boldsymbol{x}))$ and the marginal target distribution $p^t(h(\boldsymbol{x}))$ as

$$\begin{aligned}
&\mathrm{W}\big(w(h(\boldsymbol{x}))p^s(h(\boldsymbol{x})), p^t(h(\boldsymbol{x}))\big)\\
&= \max_{r:\|r\|_L \le 1} \Big(\int w(h(\boldsymbol{x})) r(h(\boldsymbol{x})) p^s(h(\boldsymbol{x})) dh(\boldsymbol{x})\\
&\qquad - \int r(h(\boldsymbol{x})) p^t(h(\boldsymbol{x})) dh(\boldsymbol{x}) \Big) \qquad (30)\\
&\approx \max_{\theta_r \in \Theta_r} \Big(\frac{1}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s); \boldsymbol{\theta}_r) - \frac{1}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t); \boldsymbol{\theta}_r) \Big). \qquad (31)
\end{aligned}$$

In Eq. (30), the Wasserstein distance is expressed as the maximal value of a functional with respect to its input function r with constraint $\|r\|_L \le 1$, and h is the feature extractor. In Eq. (31), the input function is also implemented as a three-layer discriminator network, which is parameterized by $\boldsymbol{\theta}_r$ with constraint set $\boldsymbol{\Theta}_r = \{\boldsymbol{\theta}_r \mid \|r(\cdot;\boldsymbol{\theta}_r)\|_L \le 1\}$. Then, the authors optimized the marginal source weights $w_1, \ldots, w_{m_s}$ to minimize the estimated Wasserstein distance in Eq. (31), resulting in the adversarial minimax problem

$$\min_{\boldsymbol{w}\in\mathcal{W}} \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_r} \Big(\frac{1}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s); \boldsymbol{\theta}_r) - \frac{1}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t); \boldsymbol{\theta}_r) \Big), \qquad (32)$$

where $\mathcal{W} = \{\boldsymbol{w} = (w_1, \ldots, w_{m_s})^\top \mid w_i \ge 0, \sum_{i=1}^{m_s} w_i = m_s, \sum_{i=1}^{m_s}(w_i - 1)^2 < \rho m_s\}$ is the constraint set for the marginal source weights and ρ is a hyperparameter. To reduce computational cost, the minimax problem (32) is solved by performing alternative optimization between $\boldsymbol{w}$ and $\boldsymbol{\theta}$ for only once, which may not guarantee convergence. Besides these two works, an early work by Huang, Gretton, Borgwardt, Schölkopf, and Smola (2006) is also worth discussing, which tackles the covariate shift problem and optimizes the marginal source weights to match marginal distributions under the MMD.

Relationship and difference from prior PDA works. Our work is related to the above two works (Gu et al., 2021; Zhang et al., 2018) in weighting the source data, which aims to decrease the impact of irrelevant source data and is important to the PDA problem. Interestingly, our work can also be formulated as an adversarial work. For our JWO approach with the L^2-distance, by combining Eqs. (7) and (16), we obtain the adversarial minimax problem

$$\begin{aligned}
&\min_{\boldsymbol{w}\in\Delta} \max_{\boldsymbol{\alpha}} \Big(\frac{2}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\alpha}) - \frac{2}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\alpha})\\
&\qquad - \int r(h(\boldsymbol{x}), y; \boldsymbol{\alpha})^2 dh(\boldsymbol{x}) dy \Big). \qquad (33)
\end{aligned}$$

For our JWO approach with the χ^2-divergence, by combining Eqs. (20) and (24), we obtain the other adversarial minimax problem

$$\begin{aligned}
&\min_{\boldsymbol{w}\in\Delta} \max_{\boldsymbol{\beta}} \Big(\frac{2}{m_s}\sum_{i=1}^{m_s} w_i r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\beta})\\
&\qquad - \frac{1}{m_t}\sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\beta})^2 \Big) - 1. \qquad (34)
\end{aligned}$$

However, based on the detailed discussion above, we note that our work is different from the two works (Gu et al., 2021; Zhang et al., 2018) in several aspects. **(1)** Our work optimizes the joint source weights $w_i = w(h(\boldsymbol{x}_i^s), y_i^s)$ to reduce the joint distribution disparity, rather than the marginal source weights $w_i = w(h(\boldsymbol{x}_i^s))$ to reduce the marginal distribution disparity (Gu et al., 2021; Zhang et al., 2018). **(2)** Our work develops theoretically justified weight optimization methods (16) and (24), rather than relying on the heuristic method (Zhang et al., 2018). **(3)** Our work implements the input function r as a kernel-based function, rather than a three-layer discriminator network (Gu et al., 2021; Zhang et al., 2018). Therefore, our work can solve the maximization parts in (33) and (34) with the analytic solutions in Eqs. (15) and (23), thus converting the two challenging minimax problems (33) and (34) into straightforward minimization problems (16) and (24). In Section 4.2, the experimental results show that our JWO approach is superior to IWAN (Zhang et al., 2018), AR (Gu et al., 2021), and other PDA methods. Importantly, the superiority is confirmed in a statistical sense via the Wilcoxon signed-ranks test (Demšar, 2006).

Remark 5. We provide justification for the choice of L^2-distance and χ^2-divergence for measuring the joint distribution disparity. To this end, we explore our JWO approach with other statistical distances to see what will happen. For convenience, we take the JS divergence discussed in Section 3.2 as an example. Based on the result in Zhang et al. (2018), the JS divergence between the weighted joint source distribution $w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y)$ and the joint target distribution $p^t(h(\boldsymbol{x}), y)$ can be estimated as

$$\begin{aligned} &\mathrm{JS}\big(w(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y) \,\|\, p^t(h(\boldsymbol{x}), y)\big) \\ &\approx \frac{1}{2}\max_{\boldsymbol{\gamma}}\Big(\frac{1}{m_s}\sum_{i=1}^{m_s} w_i \log r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\gamma}) \\ &\qquad + \frac{1}{m_t}\sum_{i=1}^{m_t}\log\big(1 - r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\gamma})\big)\Big) + \log 2, \end{aligned} \tag{35}$$

where $r(h(\boldsymbol{x}), y; \boldsymbol{\gamma}) = \sum_{i=1}^{m_s}\gamma_i k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i^s))k_y(y, y_i^s)$ is the kernel-based function with parameters $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_{m_s})^\top$. Unlike the quadratic maximization problems (7) and (20) from the L^2-distance and χ^2-divergence, the "log" maximization problem (35) from the JS divergence has no analytic solution and hence the estimated JS divergence cannot be explicitly expressed as a function of the joint source weights. In such circumstances, optimizing the joint source weights to minimize the estimated JS divergence in Eq. (35) results in the following minimax problem

$$\begin{aligned} &\min_{\boldsymbol{w}\in\Delta}\max_{\boldsymbol{\gamma}}\Big(\frac{1}{m_s}\sum_{i=1}^{m_s} w_i \log r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\gamma}) \\ &\qquad + \frac{1}{m_t}\sum_{i=1}^{m_t}\log\big(1 - r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\gamma})\big)\Big). \end{aligned} \tag{36}$$

While this challenging minimax problem from the JS divergence can be solved by performing alternative optimization between $\boldsymbol{w}$ and $\boldsymbol{\gamma}$, the computational cost is much higher than solving the straightforward minimization problems (16) and (24) from the L^2-distance and χ^2-divergence. Similar to the JS divergence, other statistical distances may also encounter such a drawback in computational cost, since they may not be analytically estimated and explicitly expressed as functions of the joint source weights. Therefore, we choose the L^2-distance and χ^2-divergence.

4. Experiments

4.1. Experimental setup

Datasets. We conduct experiments on 3 popular image classification datasets, and describe each dataset as follows. **DomainNet** (Peng et al., 2019) is a large and challenging dataset consisting of 345 classes and 6 domains. Since the labels of some classes and domains are noisy, following Gu et al. (2021), 126 classes and 4 domains are selected from this dataset for the experiments. The selected 4 domains are Clipart (Cl), Painting (Pa), Real (Re), and Sketch (Sk), which include 18 703, 31 502, 70 358, and 24 582 images. **Office-Home** (Venkateswara, Eusebio, Chakraborty, & Panchanathan, 2017) contains 65 classes and 4 domains: Art (Ar), Clipart (Cl), Product (Pr), and RealWorld (Rw), which include 2421, 4379, 4428, and 4357 images. **Office** (Saenko, Kulis, Fritz, & Darrell, 2010) contains 31 classes and 3 domains: Amazon (Am), DSLR (Ds), and Webcam (We), which include 2817, 795, and 498 images. See Fig. 2 for the image samples from the 3 datasets.

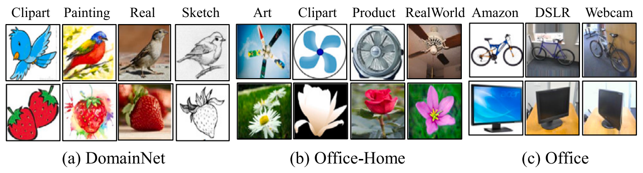


Fig. 2. Images samples from DomainNet (Peng et al., 2019), Office-Home (Venkateswara et al., 2017), and Office (Saenko et al., 2010).

Evaluation protocol. We select 2 distinct domains A and B from a certain dataset as the source and target domains, and construct the PDA task as "A → B". For the DomainNet dataset, following Gu et al. (2021), we use images from the 126 classes to build the source domain and images from the first 40 classes in alphabetical order to build the target domain. For the Office-Home dataset, following Cao et al. (2019), Gu et al. (2021) and Ren et al. (2021), we use images from the 65 classes to build the source domain and images from the first 25 classes in alphabetical order to build the target domain. Finally for the Office dataset, following Cao et al. (2019), Gu et al. (2021) and Ren et al. (2021), we use images from the 31 classes to build the source domain, and images from the 10 classes shared by the Office and Caltech datasets to build the target domain. On each PDA task, a method's performance is measured by its classification accuracy (%) on the target domain.

Comparison methods. We compare our JWO approach against the Baseline, semi-supervised learning, DA, and PDA methods. The Baseline method trains a neural network by minimizing source cross-entropy loss. The semi-supervised learning method EntMin Grandvalet and Bengio (2004) trains a neural network by minimizing both the source cross-entropy loss and target conditional entropy loss. The DA methods include the marginal matching method f-DAL (Acuna et al., 2021) and the joint matching methods DNA[2] (Chen, Hong et al., 2023), MIEM[3] (Wen, Chen, Xie et al., 2024), and RRL[4] (Sentao Chen et al., 2023). The PDA methods contain IWAN (Zhang et al., 2018) and AR (Gu et al., 2021), which are discussed in detail in Section 3.2, and other related methods SAN (Cao, Long et al., 2018), PADA (Cao, Ma et al., 2018), ETN (Cao et al., 2019), BA3US (Liang et al., 2020), and DCAN (Ge et al., 2023).

Implementation details. We exploit the Pytorch framework to implement our approach. The code will be released upon the paper's publication. On the 3 experimental datasets DomainNet, Office-Home, and Office, following prior PDA works (Cao et al., 2019; Gu et al., 2021; Zhang et al., 2018), the neural network is implemented as the ResNet50 model (He, Zhang, Ren, & Sun, 2016), where its feature extractor is pretrained on ImageNet and its classifier is trained from scratch. As presented in the network training algorithm (see Algorithm 2), the network is trained by minibatch SGD. The learning rates for the

[2] https://github.com/sentaochen/Domain-Neural-Adaptation

[3] https://github.com/sentaochen/Mutual-Information-Estimation-and-Minimization

[4] https://github.com/sentaochen/Riemannian-Representation-Learning

network's feature extractor and classifier are respectively set to 0.001 and 0.01. Besides, the joint source weights are optimized by the L-BFGS algorithm (see Algorithm 1) from the pytorch-minimize package. Here, it is worth mentioning that on each task from the DomainNet dataset, since each source domain is very large, we optimize the joint source weights in batches and set the batch size to 3000 (see Remark 4). There are 3 hyperparameters in our approach: the tradeoff parameter λ, the kernel width σ, and the small positive value ϵ. For the tradeoff parameter λ, similar to the practice in Ma et al. (2022), it is gradually changed from 0 to 1 via the formula $\lambda_i = \frac{2}{1+\exp(-10i/I)} - 1$, where $i \in [0, I]$ is the current training iteration and I is the total iterations. According to the explanation in Ma et al. (2022), such setting well balances the cross-entropy loss on the source data and the conditional entropy loss on the target data. For the kernel width σ, it is set to $2/\pi$ following the practice in Chen et al. (2020) for the L^2-distance, and set to the median pairwise squared distances on the unlabeled source and target data following the practice in Chen, Hong et al. (2023) for the χ^2-divergence. Particularly, setting $\sigma = 2/\pi$ for the L^2-distance can avoid the overflow problem in Eq. (14), since it contains the exponential term $(\pi\sigma/2)^{d/2}$. In our previous DA works (Chen et al., 2020; Chen, Hong et al., 2023), these 2 settings for the L^2-distance and the χ^2-divergence had been empirically verified to be effective. Finally for the small positive value ϵ, it is set to 0.01 for both statistical distances. In "**hyperparameter sensitivity**" of Section 4.3, we study the sensitivity of this hyperparameter to justify our setting.

4.2. Experimental results

Results. We report in Tables 1–3 the classification results on DomainNet, Office-Home, and Office, where the best result in each column of the tables is **bolded** and the second best is underlined. Note that the results of some comparison methods are cited from Cao et al. (2019), Ge et al. (2023), Gu et al. (2021) and Zhang et al. (2018), since the experimental setup is the same. To ensure comprehensive comparison, for the comparison methods that lack results on a certain dataset, we use their released source codes and their best hyperparameter settings to produce their results on that dataset. In Table 1 for the DomainNet dataset, our JWO-L^2 and JWO-χ^2 significantly outperform the Baseline, the semi-supervised learning method EntMin, DA, and PDA methods on most of the 12 tasks, achieving the second highest and the highest average classification accuracies of 69.54% and 72.47%. For comparison against the EntMin method, the outperformance shows that our joint source weights play an important role in improving the performance (69.54% versus 63.23%, 72.47% versus 63.23%). For comparison against the DA methods (DNA, MIEM, RRL), the outperformance indicates that in PDA, matching joint distributions via the joint source weights is better than via the feature extractor, as we have emphasized in the second paragraph of Section 3.1. For comparison against the PDA methods including IWAN and AR, the outperformance verifies that reducing joint disparity and providing principled weight optimization methods better solve the PDA problem, as we have discussed in the fourth paragraph of Section 1 and the second paragraph of Section 3.2. In Table 2 for the Office-Home dataset, our JWO-L^2 and JWO-χ^2 again outperform the comparison methods and achieve the best and second best average classification results of 79.00% and 78.94%. In Table 3 for the Office dataset, our JWO-L^2 and JWO-χ^2 are among the top performing methods. Observing from the average results in Tables 1–3, while our JWO-L^2 and JWO-χ^2 alternatively perform better on different datasets, the statistical tests below show that the overall performance of JWO-χ^2 is better than JWO-L^2.

Statistical tests. To be strict in a statistical sense, we conduct the Wilcoxon signed-ranks test (Demšar, 2006) to check if our JWO-χ^2 approach is statistically better than the others. The test exploits a statistic $\mathcal{T}$ to compare two methods' performance on N tasks. Here, we have $N = 30$ PDA tasks and 14 methods for comparison. Hence,

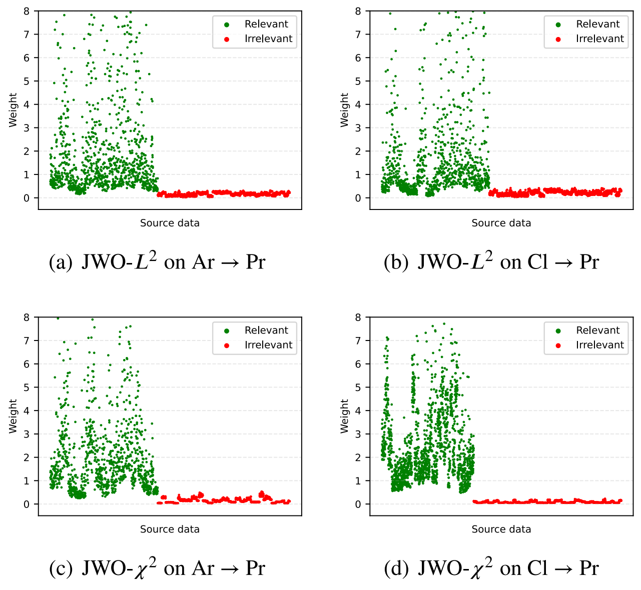


Fig. 3. Weight visualization for the relevant (green) and irrelevant (red) source data on the tasks from Office-Home. The relevant source data are those whose labels fall in the set $\mathcal{Y}^s \cap \mathcal{Y}^t$ (data from the first 25 classes), and the irrelevant ones are those whose labels belong to the set $\mathcal{Y}^s \setminus \mathcal{Y}^t$ (data from the remaining 40 classes). (a)–(b) Visualization of JWO-L^2 weights on "Ar → Pr" and "Cl → Pr". (c)–(d) Visualization of JWO-χ^2 weights on "Ar → Pr" and "Cl → Pr".

we conduct 14 pairs of tests over the tasks: Baseline versus JWO-χ^2, ⋯, DCAN versus JWO-χ^2, JWO-L^2 versus JWO-χ^2, and present the corresponding $\mathcal{T}$ values in Table 4. Table 4 shows that with a confidence level $\alpha = 0.05$ and $N = 30$ tasks, all $\mathcal{T}$ values are below the critical value of 137 for Wilcoxon's test. According to Demšar (2006), this indicates that in a statistical sense, JWO-χ^2 outperforms the Baseline, ⋯, DCAN, and JWO-L^2. From these statistical test results, we conclude that for tackling the PDA problem, our JWO-χ^2 approach is indeed superior to the other methods including the Baseline, ⋯, and DCAN. We also conclude that in our approach, the χ^2-divergence is better than the L^2-distance.

4.3. Experimental analysis

Weight visualization. In Figs. 3(a)–3(d), we visualize the joint source weights of our JWO approach with the L^2-distance and χ^2-divergence on the tasks from Office-Home. For clarity, the relevant source data whose labels fall in the set $\mathcal{Y}^s \cap \mathcal{Y}^t$ (data from the first 25 classes) are gathered and colored in green, while the irrelevant ones whose labels belong to the set $\mathcal{Y}^s \setminus \mathcal{Y}^t$ (data from the remaining 40 classes) are gathered and colored in red. From Figs. 3(a) to 3(d), we observe that the weights for the irrelevant source data (red) are close to 0 or even become 0, while the weights for the relevant source data (green) are larger than 0 and distribute within the interval $(0, 8]$. This suggests that our joint source weights can indeed decrease or even eliminate the impact of irrelevant source data.

Convergence analysis. In Figs. 4(a) and 4(b), we plot the convergence of target classification accuracies of our approach with the L^2-distance and χ^2-divergence on the tasks from DomainNet. We find that on all tasks, the target classification accuracy tends to increase with each iteration, then reaches a plateau and becomes stable after 2×10^4 iterations. This common trend implies that our network training algorithm (Algorithm 2) is convergent. Moreover, we also observe an interesting phenomenon: the target classification accuracy leaps after 1×10^4 iterations. This is because the optimized joint source weights start to take effect in the training procedure and therefore lead to the significant improvement.

Table 1
Classification accuracy (%) for PDA on DomainNet dataset.

Approach	Cl→Pa	Cl→Re	Cl→Sk	Pa→Cl	Pa→Re	Pa→Sk	Re→Cl	Re→Pa	Re→Sk	Sk→Cl	Sk→Pa	Sk→Re	Avg
Baseline (He et al., 2016)	41.21	60.01	42.13	54.52	70.80	48.32	63.10	58.63	50.26	45.43	39.30	49.75	51.96
EntMin (Grandvalet & Bengio, 2004)	50.14	64.05	59.81	65.26	76.12	69.50	75.54	69.74	68.55	50.63	54.95	54.44	63.23
f-DAL (Acuna et al., 2021)	33.34	49.12	41.65	36.07	57.41	46.74	53.61	54.57	47.98	31.79	35.48	41.43	44.10
DNA (Chen, Hong et al., 2023)	54.00	66.90	58.80	63.90	71.70	63.90	68.40	63.40	60.10	61.10	53.80	63.70	62.48
MIEM (Wen, Chen, Xie et al., 2024)	40.49	55.55	46.12	59.48	65.08	46.04	60.52	52.41	46.74	52.24	46.12	58.41	52.43
RRL (Sentao Chen et al., 2023)	53.01	64.29	57.80	64.91	71.25	63.80	68.59	63.09	60.46	62.41	54.09	63.48	62.27
SAN (Cao, Long et al., 2018)	34.35	51.62	46.23	57.13	70.21	58.25	69.61	67.49	67.88	41.69	41.15	48.44	54.50
PADA (Cao, Ma et al., 2018)	22.49	32.85	29.95	25.71	56.47	30.45	65.28	63.35	54.17	17.45	23.89	26.91	37.41
ETN (Cao et al., 2019)	55.01	71.22	56.26	59.33	73.35	64.41	65.09	66.30	59.79	62.62	57.75	68.02	63.26
IWAN (Zhang et al., 2018)	38.42	51.59	48.18	58.80	66.46	53.51	64.63	58.98	56.09	47.96	44.64	55.42	53.72
BA3US (Liang et al., 2020)	42.87	54.72	53.79	64.03	76.39	64.69	79.99	74.31	**74.02**	50.36	42.69	49.65	60.63
AR (Gu et al., 2021)	56.70	70.36	58.56	65.63	74.80	74.85	75.22	71.17	69.08	53.90	55.70	63.09	65.76
DCAN (Ge et al., 2023)	52.85	65.40	55.75	66.83	76.55	70.77	76.65	73.47	70.55	51.13	48.20	55.37	63.63
JWO-L^2 (ours)	56.92	75.54	61.03	72.66	84.79	73.22	78.39	73.46	69.78	**64.76**	55.61	68.35	69.54
JWO-χ^2 (ours)	**60.83**	**81.34**	**65.58**	**73.75**	**85.91**	**78.63**	**81.11**	**75.16**	72.38	64.28	**60.92**	**69.81**	**72.47**

Table 2
Classification accuracy (%) for PDA on Office-Home dataset.

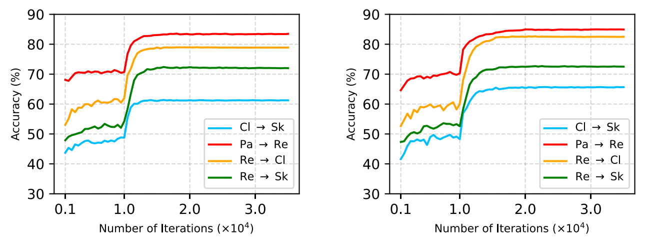
Approach	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg
Baseline (He et al., 2016)	46.33	67.51	75.87	59.14	59.94	62.73	58.22	41.79	74.88	67.40	48.18	74.17	61.35
EntMin (Grandvalet & Bengio, 2004)	54.03	73.61	83.27	69.51	67.56	77.75	69.51	53.73	83.38	74.56	59.34	82.41	70.72
f-DAL (Acuna et al., 2021)	48.72	69.92	78.35	57.94	59.05	64.55	59.96	42.99	76.81	69.33	49.25	75.69	62.71
DNA (Chen, Hong et al., 2023)	53.80	72.60	80.10	60.20	65.20	71.20	62.30	51.50	75.00	69.40	56.80	76.30	66.20
MIEM (Wen, Chen, Xie et al., 2024)	52.96	66.50	75.21	54.73	58.71	67.15	59.23	49.91	74.43	69.05	54.87	73.61	63.03
RRL (Sentao Chen et al., 2023)	53.25	69.69	79.57	59.41	62.24	70.51	64.00	51.40	75.65	69.61	56.66	75.41	65.62
SAN (Cao, Long et al., 2018)	44.42	68.68	74.60	67.49	64.99	77.80	59.78	44.72	80.07	72.18	50.21	78.66	65.30
PADA (Cao, Ma et al., 2018)	51.95	67.00	78.74	52.16	53.78	59.03	52.61	43.22	78.79	73.73	56.60	77.09	62.06
ETN (Cao et al., 2019)	59.24	77.03	79.54	62.92	65.73	75.01	68.29	55.37	84.37	75.72	57.66	84.54	70.45
IWAN (Zhang et al., 2018)	53.94	54.45	78.12	61.31	47.95	63.32	54.17	52.02	81.28	76.46	56.75	82.90	63.56
BA3US (Liang et al., 2020)	60.62	83.16	88.39	71.75	72.79	83.40	75.45	61.59	86.53	79.25	62.80	86.05	75.98
AR (Gu et al., 2021)	62.13	79.22	89.12	73.92	75.57	84.37	78.42	61.91	87.85	**82.19**	65.37	85.27	77.11
DCAN (Ge et al., 2023)	60.20	85.30	89.00	73.80	76.90	83.60	71.70	59.40	88.00	79.60	63.30	85.30	76.34
JWO-L^2 (ours)	62.39	**87.73**	90.28	**76.03**	**80.73**	**86.75**	**79.61**	64.00	88.51	80.62	**66.21**	85.10	**79.00**
JWO-χ^2 (ours)	**66.93**	85.15	**92.05**	74.26	80.62	86.20	77.04	**64.36**	**89.40**	80.90	63.40	**86.95**	78.94

Table 3
Classification accuracy (%) for PDA on Office dataset.

Approach	Am→Ds	Am→We	Ds→Am	Ds→We	We→Am	We→Ds	Avg
Baseline (He et al., 2016)	83.44	75.59	83.92	96.27	84.97	98.09	87.05
EntMin (Grandvalet & Bengio, 2004)	90.45	87.80	94.68	**100.00**	94.36	98.09	94.23
f-DAL (Acuna et al., 2021)	75.54	73.64	88.03	95.79	86.97	98.67	86.44
DNA (Chen, Hong et al., 2023)	85.40	84.40	91.50	97.60	88.30	99.40	91.10
MIEM (Wen, Chen, Xie et al., 2024)	77.87	76.27	90.37	97.12	89.26	99.62	88.42
RRL (Sentao Chen et al., 2023)	78.34	75.25	91.23	98.64	89.35	**100.00**	88.80
SAN (Cao, Long et al., 2018)	94.27	93.90	94.15	99.32	88.73	99.36	94.96
PADA (Cao, Ma et al., 2018)	82.17	86.54	92.69	99.32	95.41	**100.00**	92.69
ETN (Cao et al., 2019)	95.03	94.52	96.21	**100.00**	94.64	**100.00**	96.73
IWAN (Zhang et al., 2018)	90.45	89.15	95.62	99.32	94.26	99.36	94.69
BA3US (Liang et al., 2020)	99.36	**98.98**	94.82	**100.00**	94.99	98.73	97.81
AR (Gu et al., 2021)	91.72	97.63	95.62	**100.00**	95.30	**100.00**	96.71
DCAN (Ge et al., 2023)	93.80	95.40	95.80	**100.00**	95.80	**100.00**	96.80
JWO-L^2 (ours)	98.33	96.20	**96.41**	**100.00**	95.02	**100.00**	97.66
JWO-χ^2 (ours)	**99.65**	96.79	95.45	**100.00**	**95.94**	**100.00**	**97.97**

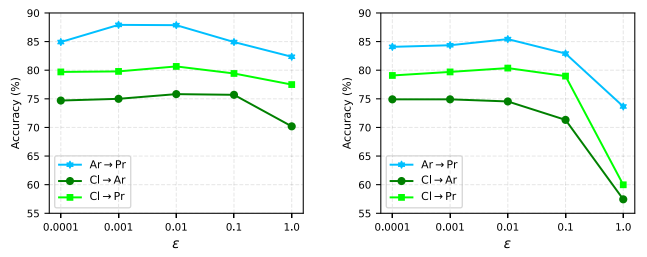
Table 4
Wilcoxon signed-ranks test results of the comparison methods versus our JWO-χ^2 approach.

Approach	Baseline (He et al., 2016)	EntMin (Grandvalet & Bengio, 2004)	f-DAL (Acuna et al., 2021)	DNA (Chen, Hong et al., 2023)	MIEM (Wen, Chen, Xie et al., 2024)	RRL (Sentao Chen et al., 2023)	SAN (Cao, Long et al., 2018)
$\mathcal{T}$ value	0.00	0.50	0.00	0.00	0.00	0.50	0.00
Approach	PADA (Cao, Ma et al., 2018)	ETN (Cao et al., 2019)	IWAN (Zhang et al., 2018)	BA3US (Liang et al., 2020)	AR (Gu et al., 2021)	DCAN (Ge et al., 2023)	JWO-L^2 (ours)
$\mathcal{T}$ value	0.50	4.50	1.00	25.50	37.5	12.50	111.00



(a) JWO-L^2 (b) JWO-χ^2

Fig. 4. Convergence of target classification accuracies of our approach on the tasks from DomainNet. (a) Convergence of JWO-L^2. (b) Convergence of JWO-χ^2.



(a) JWO-L^2 accuracy w.r.t. ϵ (b) JWO-χ^2 accuracy w.r.t. ϵ

Fig. 5. Classification accuracies of our JWO-L^2 and JWO-χ^2 with respect to (w.r.t.) different values of ϵ.

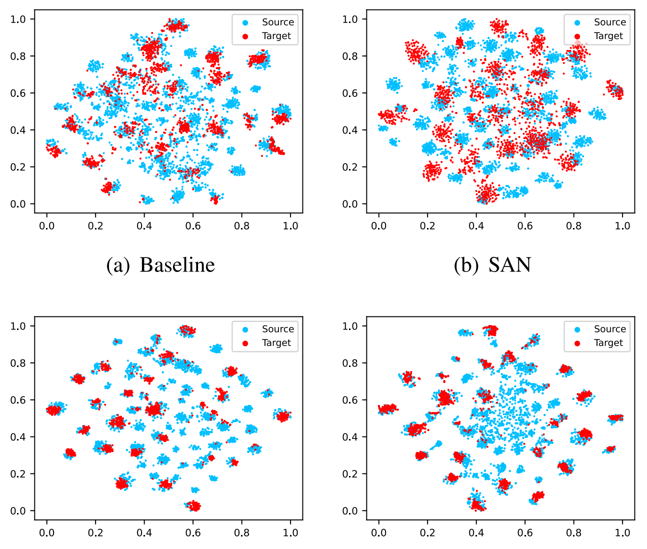
Table 5
Accuracies (%) of pseudo label, JWO-L^2, and JWO-χ^2 on "Pa→Re" and "Re→Cl" from DomainNet.

Pa→Re			Re→Cl		
Pseudo label	JWO-L^2	JWO-χ^2	Pseudo label	JWO-L^2	JWO-χ^2
52.49	70.09	72.82	47.70	70.81	72.55
64.76	77.70	79.34	59.61	74.12	76.85
73.67	81.29	82.49	65.21	75.52	77.09
76.25	84.89	86.03	75.76	78.88	81.50

Hyperparameter sensitivity. In Figs. 5(a) and 5(b), we plot the classification accuracies of JWO-L^2 and JWO-χ^2 by varying the values of ϵ to study the sensitivity of our approach. We find that on the tasks, the best results are attained when ϵ is around 0.01, and the results become less superior once $\epsilon > 0.1$. Hence, we recommend setting $\epsilon = 0.01$ for our approach, as we have done in the main experiments.

Distribution visualization. In Figs. 6(a)–6(d), we plot the t-SNE embeddings (Maaten & Hinton, 2008) of source and target data in the neural network's feature spaces of Baseline, SAN, JWO-L^2, and JWO-χ^2 on the task "Cl→Rw" (Office-Home). Compared with the results of Baseline and SAN, we observe that our JWO approach with the L^2-distance and χ^2-divergence better matches the relevant source data and the target data, and also leaves the irrelevant source data alone. This implies that in the neural network's feature space, our approach can indeed match the weighted joint source distribution and the joint target distribution, both of which are reflected by the source data and target data.

Pseudo label quality. In Table 5, we report the accuracies of pseudo label, JWO-L^2, and JWO-χ^2 on the tasks from DomainNet. The goal is to study how the pseudo label quality, which is measured by accuracy (%), influences the performance of our approach with the L^2-distance and χ^2-divergence. We find that on each task, higher accuracy of the pseudo labels leads to higher accuracy of our approach. This is within our expectation, because better pseudo labels can bring better joint source weights for the relevant and irrelevant source data, which improves the final classification accuracy of our approach.



(a) Baseline (b) SAN (c) JWO-L^2 (d) JWO-χ^2

Fig. 6. T-SNE embeddings of source and target data in the neural network's feature spaces of Baseline, SAN, JWO-L^2, and JWO-χ^2 on the task "Cl→Rw" (Office-Home).

5. Conclusion and outlook

In this paper, we propose the JWO approach that optimizes the joint source weights to match joint distributions for the PDA problem, where the joint distributions are matched under two statistical distances: the L^2-distance and the χ^2-divergence. Furthermore, we also propose the KSDE method that estimates the two statistical distances and expresses them as objective functions of the joint source weights. The experiments demonstrate the superiority of our approach to the other methods, and the experimental analysis reinforces the effectiveness of our approach.

In the future, building on the foundations of our previous DA works (Chen et al., 2020; Chen, Hong et al., 2023; Sentao Chen et al., 2023) and this PDA work, we intend to study the open-set domain adaptation problem. In addition, the KSDE method itself can be of independent interest. We plan to adapt it to solve other machine learning problems (*e.g.*, outlier detection, multi-task learning).

CRediT authorship contribution statement

Sentao Chen: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Funding acquisition, Formal analysis.

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.

Data availability

Data will be made available on request.

Acknowledgments

The author would like to thank Lisheng Wen for conducting part of the experiments. This work was supported in part by the National Natural Science Foundation of China under Grant 62106137 and Grant 62372282, in part by the Guangdong Basic and Applied Basic Research Foundation, China under Grant 2023A1515012954, and in part by Shantou University, China under Grant NTF21035.

References

Acuna, D., Zhang, G., Law, M. T., & Fidler, S. (2021). f-Domain adversarial learning: Theory and algorithms. *Vol. 139*, In *International conference on machine learning* (pp. 66–75).

Andéol, L., Kawakami, Y., Wada, Y., Kanamori, T., Müller, K.-R., & Montavon, G. (2023). Learning domain invariant representations by joint Wasserstein distance minimization. *Neural Networks*, *167*, 233–243.

Bhushan Damodaran, B., Kellenberger, B., Flamary, R., Tuia, D., & Courty, N. (2018). Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation. In *European conference on computer vision* (pp. 447–463).

Cao, Z., Long, M., Wang, J., & Jordan, M. I. (2018). Partial transfer learning with selective adversarial networks. In *IEEE conference on computer vision and pattern recognition* (pp. 2724–2732).

Cao, Z., Ma, L., Long, M., & Wang, J. (2018). Partial adversarial domain adaptation. In *European conference on computer vision* (pp. 135–150).

Cao, Z., You, K., Long, M., Wang, J., & Yang, Q. (2019). Learning to transfer examples for partial domain adaptation. In *IEEE conference on computer vision and pattern recognition* (pp. 2980–2989).

Cao, Z., You, K., Zhang, Z., Wang, J., & Long, M. (2023). From big to small: Adaptive learning to partial-set domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(2), 1766–1780.

Chen, S. (2023). Decomposed adversarial domain generalization. *Knowledge-Based Systems*, *263*, Article 110300.

Chen, S. (2024). Multi-source domain adaptation with mixture of joint distributions. *Pattern Recognition*, *149*, Article 110295.

Chen, S., & Chen, L. (2023). Joint-product representation learning for domain generalization in classification and regression. *Neural Computing and Applications*, *35*, 16509–16526.

Chen, S., Harandi, M., Jin, X., & Yang, X. (2020). Domain adaptation by joint distribution invariant projections. *IEEE Transactions on Image Processing*, *29*, 8264–8277.

Chen, S., Harandi, M., Jin, X., & Yang, X. (2021). Semi-supervised domain adaptation via asymmetric joint distribution matching. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(12), 5708–5722.

Chen, S., Hong, Z., Harandi, M., & Yang, X. (2023). Domain neural adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, *34*(11), 8630–8641.

Chen, S., Wang, L., Hong, Z., & Yang, X. (2023). Domain generalization by joint-product distribution alignment. *Pattern Recognition*, *134*, Article 109086.

Chen, S., Zheng, L., & Wu, H. (2023). Riemannian representation learning for multi-source domain adaptation. *Pattern Recognition*, *137*, Article 109271.

Courty, N., Flamary, R., Habrard, A., & Rakotomamonjy, A. (2017). Joint distribution optimal transportation for domain adaptation. In *Advances in neural information processing systems* (pp. 3730–3739).

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Ge, P., Ren, C.-X., Xu, X.-L., & Yan, H. (2023). Unsupervised domain adaptation via deep conditional adaptation network. *Pattern Recognition*, *134*, Article 109088.

Grandvalet, Y., & Bengio, Y. (2004). Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems* (pp. 529–536).

Gu, X., Yu, X., Yang, Y., Sun, J., & Xu, Z. (2021). Adversarial reweighting for partial domain adaptation. *Vol. 34*, In *Advances in neural information processing systems* (pp. 14860–14872).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hu, J., Tuo, H., Wang, C., Qiao, L., Zhong, H., Yan, J., et al. (2020). Discriminative partial domain adversarial network. In *European conference on computer vision* (pp. 632–648).

Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., & Smola, A. (2006). Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems*, *19*.

Jiang, J. (2008). *Vol. 3*, *A literature survey on domain adaptation of statistical classifiers* (pp. 1–12). URL: http://sifaka.cs.uiuc.edu/jiang4/domainadaptation/survey.

Jin, X., Yang, X., Fu, B., & Chen, S. (2020). Joint distribution matching embedding for unsupervised domain adaptation. *Neurocomputing*, *412*, 115–128.

Li, S., Liu, C. H., Lin, Q., Wen, Q., Su, L., Huang, G., et al. (2021). Deep residual correction network for partial domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(7), 2329–2344.

Li, Y., Murias, M., Major, S., Dawson, G., & Carlson, D. (2019). On target shift in adversarial domain adaptation. In *International conference on artificial intelligence and statistics* (pp. 616–625).

Liang, J., Wang, Y., Hu, D., He, R., & Feng, J. (2020). A balanced and uncertainty-aware approach for partial domain adaptation. In *European conference on computer vision* (pp. 123–140).

Ma, Y., Yao, X., Chen, R., Li, R., Shen, X., & Yu, B. (2022). Small is beautiful: Compressing deep neural networks for partial domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, 1–11.

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(Nov), 2579–2605.

Nguyen, A. T., Tran, T., Gal, Y., Torr, P. H., & Baydin, A. G. (2022). KL guided domain adaptation. In *International conference on learning representations* (pp. 1–12).

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization*. Springer.

Nowozin, S., Cseke, B., & Tomioka, R. (2016). f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems* (pp. 271–279).

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., & Wang, B. (2019). Moment matching for multi-source domain adaptation. In *IEEE international conference on computer vision* (pp. 1406–1415).

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (2008). *Dataset shift in machine learning*. MIT Press.

Ren, C.-X., Ge, P., Yang, P., & Yan, S. (2021). Learning target-domain-specific classifier for partial domain adaptation. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(5), 1989–2001.

Ren, C.-X., Luo, Y.-W., & Dai, D.-Q. (2023). BuresNet: Conditional bures metric for transferable representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(4), 4198–4213.

Saenko, K., Kulis, B., Fritz, M., & Darrell, T. (2010). Adapting visual category models to new domains. In *European conference on computer vision* (pp. 213–226).

Schölkopf, B., & Smola, A. J. (2001). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Cambridge, MA: MIT Press.

Suzuki, T., & Sugiyama, M. (2013). Sufficient dimension reduction via squared-loss mutual information estimation. *Neural Computation*, *25*(3), 725–758.

Tangkaratt, V., Xie, N., & Sugiyama, M. (2014). Conditional density estimation with dimensionality reduction via squared-loss conditional entropy minimization. *Neural Computation*, *27*(1), 228–254.

Torkkola, K. (2003). Feature extraction by non-parametric mutual information maximization. *Journal of Machine Learning Research*, *3*, 1415–1438.

Turrisi, R., Flamary, R., Rakotomamonjy, A., & Pontil, M. (2022). Multi-source domain adaptation via weighted joint distributions optimal transport. In *Uncertainty in artificial intelligence* (pp. 1970–1980).

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Venkateswara, H., Eusebio, J., Chakraborty, S., & Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *IEEE conference on computer vision and pattern recognition* (pp. 5385–5394).

Wasserman, L. (2004). *Vol. 26*, *All of statistics: a concise course in statistical inference*. Springer.

Wen, L., Chen, S., Hong, Z., & Zheng, L. (2024). Maximum likelihood weight estimation for partial domain adaptation. *Information Sciences*, *676*, Article 120800.

Wen, L., Chen, S., Xie, M., Liu, C., & Zheng, L. (2024). Training multi-source domain adaptation network by mutual information estimation and minimization. *Neural Networks*, *171*, 353–361.

Zhang, J., Ding, Z., Li, W., & Ogunbona, P. (2018). Importance weighted adversarial nets for partial domain adaptation. In *IEEE conference on computer vision and pattern recognition* (pp. 8156–8164).

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