Joint Distribution Weighted Alignment for Multi-Source Domain Adaptation via Kernel Relative Entropy Estimation

Sentao Chen, Ping Xuan, and Zhifeng Hao, *Senior Member, IEEE*

***Abstract*—The objective of Multi-Source Domain Adaptation (MSDA) is to train a neural network on labeled data from multiple joint source distributions (source domains) and unlabeled data from a joint target distribution (target domain), and use the trained network to estimate the target data labels. The challenge in this MSDA problem is that the multiple joint source distributions are relevant but distinct from the joint target distribution. To address this challenge, we propose a Joint Distribution Weighted Alignment (JDWA) approach to align a weighted joint source distribution to the joint target distribution under the relative entropy. Specifically, the weighted joint source distribution is defined as the weighted sum of the multiple joint source distributions, and is parameterized by the relevance weights. Since the relative entropy is unknown in practice, we propose a Kernel Relative Entropy Estimation (KREE) method to estimate it from data. Our KREE method first reformulates relative entropy as the negative of the minimal value of a functional, then exploits a function from the Reproducing Kernel Hilbert Space (RKHS) as the functional's input, and finally solves the resultant convex problem with a global optimal solution. We also incorporate entropy regularization to enhance the network's performance. Together, we minimize cross entropy, relative entropy, and entropy to learn both the relevance weights and the neural network. Experimental results on benchmark image classification datasets demonstrate that our JDWA approach performs better than the comparison methods. Intro video and Pytorch code are available at https://github.com/sentaochen/Joint-Distribution-Weighted-Alignment. Interested readers are also welcome to visit https://github.com/sentaochen for more source codes of the domain adaptation, partial domain adaptation, multi-source domain adaptation, and domain generalization approaches.**

***Index Terms*—Statistical learning, image classification, multi-source domain adaptation, statistical distance.**

I. Introduction

STATISTICAL learning background. Traditional classification algorithms 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 to 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 input features and y is the class label [1], [2]. Under this independent and identically distributed (i.i.d.) assumption, a classifier g trained 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 can well estimate the testing data labels in a statistical sense, where $\mathcal{L}$ represents the loss function [1], [2]. 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 [3], 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 scenarios, however, the joint training distribution and joint testing distribution may not be identical [4], which degrades the performance of the classifier in estimating the testing data labels. Therefore, developing algorithms to relax the i.i.d. assumption has become a crucial research topic in the communities of machine learning and computer vision [5]–[10].

The MSDA problem. Our focus in this work is Multi-Source Domain Adaptation (MSDA) [11], a problem that relaxes the i.i.d. assumption and includes data from multiple non-identical joint distributions. In this problem, given n ($n \geq 2$) labeled datasets from the source domains (joint source distributions) $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y)$, and an unlabeled dataset from the target domain (joint target distribution) $p^t(\boldsymbol{x}, y)$, the objective is to train a neural network such that it achieves a small target loss and accurately estimates the target data labels. According to the previous paragraph "**Statistical learning background**", the challenge in MSDA is that the multiple joint source distributions $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y)$ are relevant but distinct from the joint target distribution $p^t(\boldsymbol{x}, y)$, *i.e.*, $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$ for $s = 1, \cdots, n$.

Existing MSDA works. Existing MSDA works can be roughly categorized into three lines: marginal alignment, class-conditional alignment, and joint alignment. The first line of works, marginal alignment, proposes to align the marginal source distributions $p^1(\boldsymbol{x}), \cdots, p^n(\boldsymbol{x})$ and marginal target distribution $p^t(\boldsymbol{x})$ in the neural network's feature space [11]–[16]. The marginals are aligned in a pairwise manner by minimizing the moment distance (*e.g.*, [14]), the $\mathcal{H}$-divergence (*e.g.*, [11], [12]), or in a concurrent manner by minimizing the mutual information (*e.g.*, [15]). However, considering that $p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y|\boldsymbol{x})$, these works do not align the remaining class-posterior distributions $p^1(y|\boldsymbol{x}), \cdots, p^n(y|\boldsymbol{x}), p^t(y|\boldsymbol{x})$, which, as noted by [9], [17], [18], are distinct among domains. As a result, they may fail to address the challenge in MSDA, *i.e.*, $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$ for $s = 1, \cdots, n$. The

This work was supported in part by the National Natural Science Foundation of China under Grants 62372282, 62106137, and 62476163, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2023A1515012954, and in part by Shantou University under Grant NTF21035. *(Corresponding author: Sentao Chen.)*

Sentao Chen and Ping Xuan are with the Department of Computer Science and Technology, Shantou University, Shantou 515063, China (e-mail: sentaochenmail@gmail.com).

Zhifeng Hao is with the Department of Mathematics, Shantou University, Shantou 515063, China.

second line of works, class-conditional alignment, proposes to align not only the aforementioned marginals, but also the class-conditional source distributions $p^1(\boldsymbol{x}|y), \cdots, p^n(\boldsymbol{x}|y)$ and class-conditional target distribution $p^t(\boldsymbol{x}|y)$ [19]–[21]. The class-conditionals are aligned in a pairwise manner by minimizing the Maximum Mean Discrepancy (MMD) or its variants. While these works have improved the performance over the marginal alignment ones, they may also fail to align the joint source distributions and joint target distribution. This is because in general, a joint distribution cannot be factorized into the product of a marginal distribution and a class-conditional distribution, *i.e.*, $p(\boldsymbol{x}, y) \neq p(\boldsymbol{x})p(\boldsymbol{x}|y)$. As a consequence, aligning the marginals and class-conditionals may not align the joints. Finally, the third line of works, joint alignment, proposes to optimize the neural network embeddings to align the joint source distributions $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y)$ and joint target distribution $p^t(\boldsymbol{x}, y)$ [22], [23]. The joint distributions are aligned by minimizing the Hellinger distance (*e.g.*, [22]) or mutual information (*e.g.*, [23]). However, these works naively treat all the sources to be equally relevant to the target, and assign the same weight to them during training the network. Since some of the sources may be less relevant to the target in practical applications, treating them equally to other more relevant sources is sub-optimal and can hurt the target performance. Besides, these works [22], [23] do not consider semi-supervised learning with the unlabeled data, which can be exploited to boost the network's performance.

Our contributions. In this paper, our goal is to tackle the challenge in MSDA, *i.e.*, $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$ for $s = 1, \cdots, n$, and overcome the limitations in the aforementioned three lines of MSDA works. Detailed discussions of the limitations are provided in Section IV "**Distribution alignment**" to strengthen the motivation of this work. To be specific, we leverage a neural network $f = g \circ h$ containing a feature extractor h and a classifier g, and align a weighted joint source distribution $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y)$ to the joint target distribution $p^t(\boldsymbol{x}, y)$ in the network's feature space generated by h, such that $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \approx p^t(h(\boldsymbol{x}), y)$. The weighted joint source distribution $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y)$ is the weighted sum of the n joint source distributions $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y)$, *i.e.*, $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y) = \sum_{s=1}^{n} \alpha^s p^s(\boldsymbol{x}, y)$, and is parameterized by the relevance weights $\boldsymbol{\alpha} = (\alpha^1, \cdots, \alpha^n)^\top$. These weights can reflect the various degrees of relevance of the sources to the target, where large weights indicate high relevance and small weights indicate low relevance. When the joint distributions are aligned, we further utilize the semi-supervised learning technique to enhance the network's performance. In the reminder, we name this MSDA approach Joint Distribution Weighted Alignment (JDWA). A detailed discussion of the differences between our JDWA approach and other methods is presented in Subsection III-B and Section IV. To be more specific, we utilize relative entropy (also known as Kullback-Leibler divergence) as the alignment loss, which reduces to zero when the two distributions in question are identical [24]. Compared with other statistical distances (*e.g.*, $\mathcal{H}$-divergence, MMD), the relative entropy is better than them for the following reasons. (i) It is well defined for the distance between joint distributions, while some statistical distances (*e.g.*, $\mathcal{H}$-divergence, MMD) are only defined for the distance between marginal distributions or class-conditional distributions; (ii) It has demonstrated its superiority in the relevant problems of Domain Adaptation (DA) [7], [25], [26] and Domain Generalization (DG) [17]. (iii) It is strongly connected to the geodesic distance on statistical manifold and hence can better characterize the statistical distance between probability distributions [25], [27]. We minimize the relative entropy with respect to the relevance weights and the feature extractor to align the weighted joint source distribution and the joint target distribution. Considering that the relative entropy is unknown in real situations, we estimate it from samples to obtain the practical alignment loss. To estimate the relative entropy, we first reformulate it as the negative of the minimal value of a functional[1], then substitute the expectations in the functional by sample averages, exploit a function from the Reproducing Kernel Hilbert Space (RKHS) [2] as the functional's input, and finally solve a convex optimization problem. Thanks to the convexity, the global optimal solution to the problem is easy to obtain. In the reminder, we name this estimation method Kernel Relative Entropy Estimation (KREE). Furthermore, we perform entropy regularization [28] for the predictions on the target unlabeled data. This semi-supervised learning technique reduces the uncertainty of the target predictions, such that the target unlabeled data can be classified with high confidence. Our optimization for JDWA consists of optimizing both the relevance weights and the neural network to jointly minimize the relative entropy for joint distribution weighted alignment, the cross entropy for classification, and the entropy for semi-supervised learning. Eventually, we conduct experiments on benchmark datasets with varied characteristics (*i.e.*, few-class/many-class datasets, small/large-scale datasets). We find that our JDWA approach outperforms the comparison methods in a statistical sense. For clarity, we summarize our contributions below.

- We propose the JDWA approach, which aligns the weighted joint source distribution to the joint target distribution to address the challenge in MSDA. Our approach assigns large weights to more relevant sources and small weights to less relevant sources, and performs entropy regularization to better estimate the target data labels.
- We introduce the KREE method, which estimates the relative entropy between the weighted joint source distribution and joint target distribution to obtain the practical alignment loss. Our method is kernel-based and transforms the estimation into solving a convex problem.
- We conduct extensive experiments with statistical tests to demonstrate the advantages of our approach. Our experiments are conducted on few-class/many-class datasets, and small/large-scale datasets.

II. Joint Distribution Weighted Alignment

A. Problem Definition

In Multi-Source Domain Adaptation (MSDA), one is given n $(n \geq 2)$ labeled source datasets $\mathcal{D}^1 = \{(\boldsymbol{x}_i^1, y_i^1)\}_{i=1}^{m_1}, \cdots,$

[1]According to the book [2], a functional takes function(s) as its input(s) and outputs a scalar value.

$\mathcal{D}^n = \{(\boldsymbol{x}_i^n, y_i^n)\}_{i=1}^{m_n}$ respectively generated by n joint source distributions $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y)$, and an unlabeled target dataset $\mathcal{D}^u = \{\boldsymbol{x}_i^t\}_{i=1}^{m_t}$ generated by a marginal target distribution $p^t(\boldsymbol{x}) = \int p^t(\boldsymbol{x}, y)dy$. The n joint source distributions are relevant but distinct from the joint target distribution, *i.e.*, $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$ for $s = 1, \cdots, n$. The objective of MSDA is to train a neural network $f(\boldsymbol{x})$ that can well estimate the target data labels. Namely, the network $f(\boldsymbol{x})$ should have 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 the target data label unknown during training.

B. Problem Analysis

The MSDA objective. Starting from the MSDA objective, we demonstrate how the problem can be solved. We first exploit a neural network $f = g \circ h$, where the feature extractor h generates the network's feature space and the classifier g produces the probabilistic prediction. We then define a weighted joint source distribution as

$$p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y) = \sum_{s=1}^{n} \alpha^s p^s(\boldsymbol{x}, y), \tag{1}$$

where $\boldsymbol{\alpha} = (\alpha^1, \cdots, \alpha^n)^\top \in \Delta = \{\boldsymbol{\alpha} | \alpha^s \geq 0, \sum_{s=1}^n \alpha^s = 1\}$ are the relevance weights of the n joint source distributions. Since the sources are relevant to the target, we can align the weighted joint source distribution $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y)$ to the joint target distribution $p^t(\boldsymbol{x}, y)$ in the network's feature space generated by h, such that $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \approx p^t(h(\boldsymbol{x}), y)$. As a result, we have the following relationship

$$\begin{aligned}
&\frac{1}{m_t}\sum_{i=1}^{m_t}\mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t) \\
&\approx \mathbb{E}_{p^t(h(\boldsymbol{x}), y)}[\mathcal{L}(g(h(\boldsymbol{x})), y)] &(2)\\
&\approx \mathbb{E}_{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)}[\mathcal{L}(g(h(\boldsymbol{x})), y)] &(3)\\
&= \sum_{s=1}^{n}\alpha^s \mathbb{E}_{p^s(h(\boldsymbol{x}), y)}[\mathcal{L}(g(h(\boldsymbol{x})), y)] &(4)\\
&\approx \sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s), &(5)
\end{aligned}$$

where the two expectations in Eq. (2) and Eq. (4) are estimated by the average of target samples and the weighted averages of source samples, respectively. Evidently, the relationship from Eq. (2) to Eq. (5) shows that, the objective of training a neural network with small target loss $\frac{1}{m_t}\sum_{i=1}^{m_t}\mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t)$, can be achieved by minimizing the network's weighted source loss $\sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}\mathcal{L}(g(h(\boldsymbol{x}_i^s)), y_i^s)$.

The relative entropy. To achieve the desired joint distribution alignment $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \approx p^t(h(\boldsymbol{x}), y)$, we utilize relative entropy (Kullback-Leibler divergence) as the alignment loss, which is non-negative and reduces to zero when $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) = p^t(h(\boldsymbol{x}), y)$ [24]. The relative entropy has proven effective in the related DA and DG works [7], [17], [25], [26]. Moreover, it strongly connects to the geodesic distance on statistical manifold and hence can well characterize the statistical distance between probability distributions [25], [27]. Mathematically, the relative entropy is defined as

$$\begin{aligned}
&\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \| p^t(h(\boldsymbol{x}), y)) \\
&= \int p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)\log\Big(\frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)}{p^t(h(\boldsymbol{x}), y)}\Big)dh(\boldsymbol{x})dy \geq 0. &(6)
\end{aligned}$$

This is an asymmetric statistical distance, which means that we do not have $\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \| p^t(h(\boldsymbol{x}), y)) = \mathrm{RE}(p^t(h(\boldsymbol{x}), y) \| p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y))$ in general [24]. The reason for choosing the current direction will be explained later in Remark 1. For practical application, we estimate the relative entropy from samples, and optimize both the relevance weights and the feature extractor to achieve minimal estimated relative entropy. During distribution alignment, the joint source distributions more relevant to the joint target distribution will receive large weights, and the less relevant ones will receive small weights.

Semi-supervised learning. Once the joint distributions are aligned, the data tend to follow the i.i.d. assumption, and the MSDA problem turns into a semi-supervised learning problem with labeled source data and unlabeled target data. Following the entropy regularization principle in semi-supervised learning [28], we minimize the entropy for target probabilistic predictions to enhance the neural network's target performance.

Below, we introduce our kernel relative entropy estimation in Subsection II-C, describe the target entropy regularization in Subsection II-D, and present our joint entropies optimization in Subsection II-E.

C. Kernel Relative Entropy Estimation

Variational representation. For estimating the relative entropy, we first rewrite its original expression in Eq. (6) as

$$\begin{aligned}
&\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \| p^t(h(\boldsymbol{x}), y)) \\
&= -\min_r \Big(\int \exp(r(h(\boldsymbol{x}), y) - 1)p^t(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy \\
&\qquad - \int r(h(\boldsymbol{x}), y)p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy\Big) &(7)\\
&= -\min_r \Big(\int \exp(r(h(\boldsymbol{x}), y) - 1)p^t(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy \\
&\qquad - \sum_{s=1}^{n}\alpha^s \int r(h(\boldsymbol{x}), y)p^s(h(\boldsymbol{x}), y)dh(\boldsymbol{x})dy\Big). &(8)
\end{aligned}$$

Eq. (7) applies the variational representation of the relative entropy in [6], and expresses the relative entropy as the negative of the minimal value of a functional. The functional will reach its minimal value when the input function $r(h(\boldsymbol{x}), y) = 1 + \log\Big(\frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)}{p^t(h(\boldsymbol{x}), y)}\Big)$, and the negative of that minimal value is Eq. (6). Later in Remark 2, we include more discussions on Eq. (6) and Eq. (7) to support our motivation in reformulating the relative entropy. Eq. (8) expands the weighted joint source distribution as $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) = \sum_{s=1}^{n}\alpha^s p^s(h(\boldsymbol{x}), y)$.

Empirical approximation. Then, we replace the expectations in Eq. (8) with sample averages, and approximate the relative entropy as

$$\begin{aligned}&\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))\\&\approx -\min_r\Big(\frac{1}{m_t}\sum_{i=1}^{m_t}\exp(r(h(\boldsymbol{x}_i^t),y_i^t)-1)\\&\qquad-\sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}r(h(\boldsymbol{x}_i^s),y_i^s)\Big).\end{aligned}\tag{9}$$

Eq. (9) uses the labeled source datasets $\mathcal{D}^1=\{(\boldsymbol{x}_i^1,y_i^1)\}_{i=1}^{m_1}$, $\cdots,\mathcal{D}^n=\{(\boldsymbol{x}_i^n,y_i^n)\}_{i=1}^{m_n}$, and the labeled target dataset $\mathcal{D}^t=\{(\boldsymbol{x}_i^t,y_i^t)\}_{i=1}^{m_t}$, which is assumed to be available here so that we can focus on the estimation. In Subsection II-E, we will discuss how $\mathcal{D}^t$ is obtained.

RKHS and Representer Theorem. We exploit a function r from the Reproducing Kernel Hilbert Space (RKHS) [2] $\mathcal{R}$ as the functional's input in Eq. (9). The reason for choosing the RKHS is explained in Remark 3. The RKHS $\mathcal{R}$ is associated with a product kernel $k(h(\boldsymbol{x}_i),h(\boldsymbol{x}_j))\delta(y_i,y_j)$, where $k(h(\boldsymbol{x}_i),h(\boldsymbol{x}_j))=\exp(-\|h(\boldsymbol{x}_i)-h(\boldsymbol{x}_j)\|^2/\sigma)$ is the Gaussian kernel with kernel width $\sigma(>0)$, and $\delta(y_i,y_j)$ is the delta kernel that evaluates 1 if $y_i=y_j$ and 0 otherwise. Based on Eq. (9), we select the optimal function by solving the following optimization problem

$$\begin{aligned}\min_{r\in\mathcal{R}}\Big(&\frac{1}{m_t}\sum_{i=1}^{m_t}\exp(r(h(\boldsymbol{x}_i^t),y_i^t)-1)\\&-\sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}r(h(\boldsymbol{x}_i^s),y_i^s)+\lambda\|r\|_{\mathcal{R}}^2\Big),\end{aligned}\tag{10}$$

where a regularization term $\lambda\|r\|_{\mathcal{R}}^2$ with regularization parameter $\lambda(>0)$ is introduced to avoid overfitting. According to the Representer Theorem [2], the optimal function can be expressed as

$$r(h(\boldsymbol{x}),y;\boldsymbol{\theta})=\sum_{i=1}^{m}\theta_i k(h(\boldsymbol{x}),h(\boldsymbol{x}_i))\delta(y,y_i),\tag{11}$$

where $\boldsymbol{\theta}=(\theta_1,\cdots,\theta_m)^\top$ are the function parameters, and $\{(\boldsymbol{x}_1,y_1),\cdots,(\boldsymbol{x}_m,y_m)\}=\{(\boldsymbol{x}_1^1,y_1^1),\cdots,(\boldsymbol{x}_{m_t}^t,y_{m_t}^t)\}=\mathcal{D}^1\cup\cdots\cup\mathcal{D}^n\cup\mathcal{D}^t$ are the $m=m_1+\cdots+m_n+m_t$ samples.

The convex problem. Plugging the expression in Eq. (11) into Eq. (10), we learn the optimal function parameters by solving the following convex problem

$$\begin{aligned}\widehat{\boldsymbol{\theta}}=\operatorname*{argmin}_{\boldsymbol{\theta}}\Big(&\frac{1}{m_t}\sum_{i=1}^{m_t}\exp(r(h(\boldsymbol{x}_i^t),y_i^t;\boldsymbol{\theta})-1)\\&-\sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}r(h(\boldsymbol{x}_i^s),y_i^s;\boldsymbol{\theta})+\lambda\boldsymbol{\theta}^\top\boldsymbol{P}\boldsymbol{\theta}\Big),\end{aligned}\tag{12}$$

where $\boldsymbol{P}$ is a product kernel matrix whose (i,j)-th element is $k(h(\boldsymbol{x}_i),h(\boldsymbol{x}_j))\delta(y_i,y_j)$. Note that, Eq. (12) is a convex problem, since the objective function is convex in $\boldsymbol{\theta}$. In fact, we can verify this point by observing the three components $\exp(r(h(\boldsymbol{x}),y;\boldsymbol{\theta})-1)$, $-r(h(\boldsymbol{x}),y;\boldsymbol{\theta})$, and $\boldsymbol{\theta}^\top\boldsymbol{P}\boldsymbol{\theta}$ of the objective function, which are all convex in $\boldsymbol{\theta}$. According to the knowledge of convex optimization, the objective function is also convex in $\boldsymbol{\theta}$. To find the global optimal solution $\widehat{\boldsymbol{\theta}}$, we use the mature optimization algorithm named L-BFGS [29] implemented in the pytorch-minimize[2] package. Finally, we obtain the estimated relative entropy as

$$\begin{aligned}&\widehat{\mathrm{RE}}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))\\&=-\Big(\frac{1}{m_t}\sum_{i=1}^{m_t}\exp(r(h(\boldsymbol{x}_i^t),y_i^t;\widehat{\boldsymbol{\theta}})-1)\\&\qquad-\sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}r(h(\boldsymbol{x}_i^s),y_i^s;\widehat{\boldsymbol{\theta}})\Big).\end{aligned}\tag{13}$$

This estimated relative entropy serves as the loss function for the relevance weights $\boldsymbol{\alpha}=(\alpha^1,\cdots,\alpha^n)^\top$ and the feature extractor h. For clarity, we summarize this Kernel Relative Entropy Estimation (KREE) method in Algorithm 1. Like other estimation techniques, the estimation error of our KREE method, *i.e.*, the absolute difference between the relative entropy and its estimated value, is also dependent on the size of the samples. Later in Subsection V-D "**Estimation error**", we study the estimation error with different numbers of samples. Besides, in Subsection V-D "**KREE loss**", we also include an ablation study to evaluate the impact of the KREE loss.

Algorithm 1 Kernel Relative Entropy Estimation (KREE)

Input: Datasets $\mathcal{D}^1,\cdots,\mathcal{D}^n,\mathcal{D}^t$.
Output: Estimated value $\widehat{\mathrm{RE}}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))$.
1: Construct convex optimization problem in Eq. (12).
2: Run optimization algorithm (*e.g.*, L-BFGS [29]) to return the optimal solution $\widehat{\boldsymbol{\theta}}$ to the problem in Eq. (12).
3: Obtain $\widehat{\mathrm{RE}}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))$ via Eq. (13).

Remark 1. *The relative entropy is an asymmetric statistical distance [24]. Here, we explain why we choose the current direction* $\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))$*, rather than the reverse direction* $\mathrm{RE}(p^t(h(\boldsymbol{x}),y)\|p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y))$*. According to the explanations right below Eq. (8),* $r(h(\boldsymbol{x}),y;\boldsymbol{\theta})$ *in Eq. (11) is intended to approximate* $1+\log\left(\frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}{p^t(h(\boldsymbol{x}),y)}\right)$*. If we select the reverse direction,* $r(h(\boldsymbol{x}),y;\boldsymbol{\theta})$ *will instead approximate* $1+\log\left(\frac{p^t(h(\boldsymbol{x}),y)}{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}\right)$*. For certain values of* $\alpha^1,\cdots,\alpha^n$*, however, it is possible that* $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)=\sum_{s=1}^n\alpha^s p^s(h(\boldsymbol{x}),y)\approx 0$ *and hence* $1+\log\left(\frac{p^t(h(\boldsymbol{x}),y)}{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}\right)$ *may diverge. This makes it difficult for the function approximation. Given such consideration, we choose the current direction* $\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))$.

Remark 2. *It is also feasible to estimate the relative entropy based on its original expression in Eq. (6). Specifically, following the two-step method in [25], [27], one can first estimate the two distributions separately, and then use the two estimated distributions to express the relative entropy. However, distribution estimation is known to be a hard problem in high dimensional spaces (*e.g.*, the network's feature space here) [1], [3]. Moreover, due to the two-step procedure, the estimation error introduced in the first step can be magnified in the second step. To overcome these limitations, we therefore*

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

propose to rewrite the relative entropy as Eq. (7), and design the single-step method KREE for relative entropy estimation.

Remark 3. *In our estimation, we exploit a function r from the RKHS due to the following considerations. (1) The RKHS is a common and popular function space in machine learning [2], [30], [31]. (2) Importantly, the RKHS can lead to the convex optimization problem in Eq. (12). The convexity allows us to easily obtain the global optimal solution to the problem. With other function spaces (*e.g.*, neural networks), however, the resulting problems may not be convex.*

D. Target Entropy Regularization

From the semi-supervised learning perspective, we perform entropy regularization [28] for the class-posterior predictions $\hat{p}(y=j|\boldsymbol{x}_i^t; f=g\circ h)=g_j(h(\boldsymbol{x}_i^t))$ on the unlabeled target dataset $\mathcal{D}^u=\{\boldsymbol{x}_i^t\}_{i=1}^{m_t}$ to improve the network's performance. This entropy regularization term is defined as

$$\frac{1}{m_t}\sum_{i=1}^{m_t}\mathrm{H}(g(h(\boldsymbol{x}_i^t)))=\frac{-1}{m_t}\sum_{i=1}^{m_t}\sum_{j=1}^{c}g_j(h(\boldsymbol{x}_i^t))\log g_j(h(\boldsymbol{x}_i^t)), \tag{14}$$

where c is the number of classes. Fundamentally, minimizing Eq. (14) drives the class-posterior distribution $\hat{p}(y|\boldsymbol{x}; f=g\circ h)$ to become a deterministic function $y=\phi(\boldsymbol{x})$ on the unlabeled target dataset, *i.e.*, $\hat{p}(y=\phi(\boldsymbol{x})|\boldsymbol{x}; f=g\circ h)=1$. Therefore, the unlabeled target data can be classified with high confidence and the resulting target labels will be reliable. In Subsection V-D "**Entropy regularization**", we conduct an ablation study to evaluate the contribution of this term to our approach.

E. Joint Entropies Optimization

JDWA optimization problem. Combining the classification loss in Subsection II-B, the alignment loss in Subsection II-C, and the regularization loss in Subsection II-D, the optimization problem of our Joint Distribution Weighted Alignment (JDWA) approach is formulated as

$$\begin{aligned}\min_{g,h}\min_{\boldsymbol{\alpha}}&\sum_{s=1}^{n}\frac{\alpha^s}{m_s}\sum_{i=1}^{m_s}\mathcal{L}(g(h(\boldsymbol{x}_i^s)),y_i^s)+\frac{\lambda_{\mathrm{E}}}{m_t}\sum_{i=1}^{m_t}\mathrm{H}(g(h(\boldsymbol{x}_i^t)))\\&+\lambda_{\mathrm{RE}}\widehat{\mathrm{RE}}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y)),\end{aligned} \tag{15}$$

where $\boldsymbol{\alpha}=(\alpha^1,\cdots,\alpha^n)^\top\in\Delta=\{\boldsymbol{\alpha}|\alpha^s\geq 0,\sum_{s=1}^{n}\alpha^s=1\}$ are the relevance weights, and $\lambda_{\mathrm{E}},\lambda_{\mathrm{RE}}$ are the tradeoff parameters. We use cross entropy as the classification loss and define $\mathcal{L}(g(h(\boldsymbol{x}_i^s)),y_i^s)=-\sum_{j=1}^{c}\delta(y_i^s,j)\log g_j(h(\boldsymbol{x}_i^s))$, where c is the number of classes and $\delta(y_i^s,j)$ evaluates 1 if $y_i^s=j$ and 0 otherwise. The other two losses in (15) are the target entropy and the estimated relative entropy. Recall that the estimated relative entropy requires the labeled target dataset $\mathcal{D}^t$. In MSDA, we create this dataset by utilizing both the unlabeled target dataset $\mathcal{D}^u=\{\boldsymbol{x}_i^t\}_{i=1}^{m_t}$ and the popular pseudo labeling technique [22], [23], [30], [32]. Specifically, we use the technique to generate pseudo labels for $\mathcal{D}^u$ and create the labeled target dataset as $\mathcal{D}^t=\{(\boldsymbol{x}_i^t,y_i^t)\}_{i=1}^{m_t}$. Here, y_i^t represents the pseudo label estimated by the network trained by minimizing the cross entropy loss on source datasets. Clearly, our approach in (15) optimizes the weights $\boldsymbol{\alpha}$ and the network $f=g\circ h$ to jointly minimize the relative entropy for joint distribution weighted alignment, the cross entropy for classification, and the entropy for semi-supervised learning. In Fig. 1, we depict the input data, the neural network architecture, and the losses in our JDWA approach.

Optimization algorithm. To solve problem (15), we first solve the inner minimization concerning $\boldsymbol{\alpha}$ by fixing g,h and then the outer minimization concerning g,h by fixing $\boldsymbol{\alpha}$. Such alternating optimization is common and can also be found in the related work [13]. The optimization procedure is performed in an iterative manner. In each iteration, $n+1$ batches of data with identical size are drawn from datasets $\mathcal{D}^1,\cdots,\mathcal{D}^n$, $\mathcal{D}^t$, respectively. (i) For the minimization problem concerning $\boldsymbol{\alpha}$, we enforce the constraint Δ by utilizing the softmax function, *i.e.*, $\alpha^s\leftarrow\frac{\exp(\alpha^s)}{\sum_{i=1}^{n}\exp(\alpha^i)}$, and solve the unconstrained minimization problem via the L-BFGS optimization algorithm [29], with $\boldsymbol{\alpha}$ being initialized as $\boldsymbol{\alpha}=(\frac{1}{n},\cdots,\frac{1}{n})^\top$. Note that when optimizing $\boldsymbol{\alpha}$, the optimization of $\boldsymbol{\theta}$ in Eq. (12), which also involves executing the L-BFGS algorithm (see Algorithm 1), is nested in the optimization of $\boldsymbol{\alpha}$. Such kind of nested optimization problem can also be found in the work of Gu *et al.* [33]. There, the authors constructed the nested problem to learn the weights of source data for the Partial Domain Adaptation (PDA) problem. To reduce the computational cost, following [33], we only perform 5 iterations for each of the two L-BFGSs, which can already yield satisfactory results in the experiments. (ii) For the minimization problem concerning g,h, we leverage the minibatch SGD algorithm. In addition, considering that in the beginning the pseudo labels are inaccurate, we update the pseudo labels during the optimization procedure. The updated labels become more accurate due to the distribution alignment [22], [30], [32], [34] and entropy regularization [30]. Algorithm 2 summarizes the above procedure. Later in Subsection V-D "**Convergence behavior**", we study the convergence behavior of the algorithm.

Remark 4. *In deep domain adaptation literature (*e.g.*, [7], [14], [30], [35], [36]), it is common to compute the estimated distribution distance on a minibatch of data, instead of all the data. During optimization, our estimated relative entropy in (15), is also calculated on a minibatch of data. In Subsection V-D "**Batch size**", we find that generally a larger batch size leads to a better performance.*

III. Related Work

A. Domain Adaptation

DA by distribution alignment. Domain Adaptation (DA) involves first training a model on data from a single source domain and a target domain, and then leveraging the trained model to estimate the target data labels. This concept holds close ties to the MSDA problem. During the past ten years, significant efforts have been dedicated to addressing the DA challenge [5], [6], [35], [37]–[39]. For example, Baktashmotlagh *et al.* [25] proposed to learn a projection matrix to align the marginal source and target distributions in the projection space under the relative entropy, which is estimated by Kernel Density Estimation (KDE) [3]. In the deep learning context,

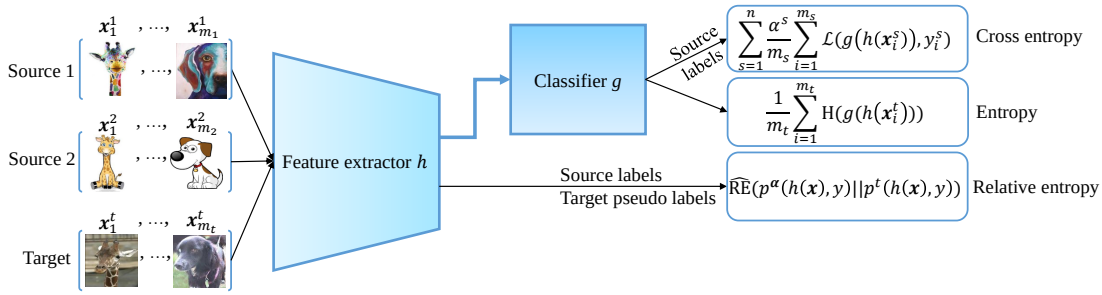


Fig. 1: The input data, neural network architecture, and losses in our Joint Distribution Weighted Alignment (JDWA) approach.

Algorithm 2 Joint Distribution Weighted Alignment (JDWA)

Input: Labeled source datasets $\mathcal{D}^1, \cdots, \mathcal{D}^n$, unlabeled target dataset $\mathcal{D}^u$.
Output: Trained network $f = g \circ h$.

Train the network on $\mathcal{D}^1, \cdots, \mathcal{D}^n$.
Construct the pseudo labeled target dataset $\mathcal{D}^t$ by using the trained network to label $\mathcal{D}^u$.
while training does not end **do**
  for k in $1 : K$ **do**
    Sample $n+1$ minibatches of data with identical size from $\mathcal{D}^1, \cdots, \mathcal{D}^n$, $\mathcal{D}^t$.
    Compute the first and third terms of the objective function (15) on the minibatches, where the third term is computed by executing Algorithm 1.
    Run the optimization algorithm L-BFGS [29] to return the optimal weights $\boldsymbol{\alpha}$, where the constraint Δ is reflected by the softmax function.
    Calculate the whole objective function (15) on the minibatches.
    Take a gradient step to update the parameters of feature extractor h and classifier g.
  end for
  Update pseudo labels in $\mathcal{D}^t$ by using the network.
end while

Nguyen *et al*. [7] used two relative entropies to upper bound the target loss, one between the marginal distributions and the other between the class-posterior distributions, and minimized the relative entropy to align the marginal distributions in the network's representation space. Zhu *et al*. [32] proposed to optimize the features to minimize the Local MMD for matching the class-conditional source distribution and class-conditional target distribution in the network's feature space. Ren *et al*. [30] reduced the Conditional Kernel Bures metric to mitigate the shift between the class-conditional distributions, and also preserved the domain intrinsic structures. Long *et al*. [40] minimized the Joint MMD to match two joint distributions of the network features from multiple layers, one for the source and the other one for the target. Nicolas *et al*. [41] optimized for a coupling between the joint distributions of two domains, and a classification model that estimates the target data labels. Damodaran *et al*. [42] extended the work to deep networks, and used optimal transport to minimize the disparity between joint source distribution and joint target distribution. In addition to distribution alignment, interested readers can refer to a recent survey [43] for more DA solutions.

Discussion. Since MSDA and DA are closely related problems, it may seem intuitive to combine multiple source datasets into a single dataset. As such, the MSDA problem can be simplified to the DA problem and the DA methods can be applied to MSDA. However, this naive approach is flawed because it fails to account for the disparities among the multiple source domains.

B. Multi-Source Domain Adaptation

MSDA by distribution alignment. As discussed in Section I "**Existing MSDA works**", most MSDA works can be divided into three lines: marginal alignment, class-conditional alignment, and joint alignment. In the first line of works, marginal alignment, Zhao *et al*. [11] learned feature representations that are invariant among the marginal source and target distributions in the sense of $\mathcal{H}$-divergence, and also ensured that the representations are discriminative for the supervised task. Peng *et al*. [14] minimized the moment distance to align the marginal source distributions with the marginal target distribution, and also the marginal sources with each other. Wen *et al*. [13] used a convex combination of both the source errors and the discrepancy distances to bound the target error, and proposed to optimize the combination weights and the neural networks to minimize the convex combination. In particular, the authors utilized the discrepancy distance to measure the distance between each marginal source distribution and the marginal target distribution, and followed previous adversarial methods [11], [44] to express the discrepancy distance as a newly added subnetwork's logistic loss. Besides, through adversarial training, Park *et al*. [15] aligned the multiple marginal source distributions and the marginal target distribution concurrently by minimizing the mutual information between the network's features and the domain labels. In the second line of works, class-conditional alignment, Li *et al*. [20] proposed to adapt the source and target domains using marginal and class-conditional alignments under the MMD, and developed a weight adjustment strategy to utilize the best source domain since the sources are different from each other. Additionally, Yao *et al*. [19] introduced a conditional weighting strategy to compute the source domain weights, and aligned the class-conditional distributions among domains by minimizing the MMD. In the third line of works, joint alignment, Chen *et al*. [22] proposed to learn the neural network features to

separately match each joint source distribution to the joint target distribution in the sense of Hellinger distance, where all the sources are associated with the fixed same weight. Moreover, Turrisi *et al*. [45] extracted neural network features using multi-task learning, and minimized the Wasserstein distance between an estimated joint target distribution and a convex combination of the joint source distributions in the fixed feature space.

Discussion. Our work differs from the above discussed studies in several key aspects. (i) Our neural network model $f = g \circ h$ is simple and has fewer parameters than the complex neural network models in prior works (*e.g*., [11], [13], [20]). Our feature extractor is optimized to minimize the loss for the MSDA task, rather than the loss for the multi-task learning task in Turrisi *et al*. [45]. (ii) Our approach considers a single weighted joint source distribution (the weighted sum of multiple joint source distributions), instead of the multiple marginal/class-conditional/joint source distributions (*e.g*., [11], [14], [20]–[22]). (iii) Our alignment loss is the relative entropy, instead of the $\mathcal{H}$-divergence (*e.g*., [11], [16]) or MMD (*e.g*., [19]–[21]). In our work, the relative entropy is more suitable as it is well defined for the distance between joint (features and label) distributions. To estimate the relative entropy, we develop a kernel-based method and solve a convex problem in Eq. (12). Our work is also different from [46], which estimates and minimizes another statistical distance named the χ^2-divergence and is not aware of semi-supervised learning.

Other MSDA works. Besides the three lines of works mentioned above, there are also other important works that are worth mentioning and discussing [47]–[52]. For example, Li *et al*. [49] integrated tensor singular value decomposition into training the network, constructed a prototypical similarity matrix to model the category-wise relations, and developed an uncertainty-aware weighting mechanism to combine multiple data distributions. Yuan *et al*. [51] employed the self-supervised tasks of classical pretext classification and domain classification to improve the target performance, and used a graph neural network as a bridge between these tasks. Zhou *et al*. [52] designed and refined the multiple source-specific networks and the domain-ensemble network in a cycle manner to combine the transferable knowledge in each source domain for adaptation, where the transferable knowledge is discovered by an instance-level ensemble strategy. The instance-level ensemble and cycle self-refinement learning strategies together are shown to be effective for MSDA. Different from these works, our work tackles the problem from the statistical learning perspective and aims to align joint distributions to solve the MSDA challenge. In the experiments in Section V, we compare our approach with some of these methods.

C. Few-Shot Domain Adaptation

Few-Shot Domain Adaptation (FSDA) focuses on a few-shot domain adaptation setting, where a very limited number of source data are labeled, while all the remaining source and target data are unlabeled. The goal of FSDA is to train a model on labeled source data, unlabeled source and target data, and utilize the trained model to estimate the target data labels [53], [54]. To address the FSDA problem, Yue *et al*. [53] performed prototypical self-supervised learning, and learned an adaptive prototypical classifier. Xiong *et al*. [54] transferred source knowledge from high-confidence data to low-confidence data by cross-domain visual dispersal, and improved the learning of hard unlabeled target data by intra-domain visual dispersal. Similarly, Multi-Source Few-Shot Domain Adaptation (MSFSDA) focuses on the multi-source domain adaptation scenario with limited labeled data from each source domain and unlabeled data from the target domain [55]. Yue *et al*. [55] proposed a Multi-Source Few-Shot Adaptation Network (MSFSAN) that performs multi-domain prototypical self-supervised learning, support-set based cross-domain similarity consistency, and multi-domain prototypical classifier learning. By contrast, Zhu *et al*. [56] considered the setting where multiple labeled source domains are available but the target domain only contains very limited labeled data, and proposed a progressive mix-up solution that creates an intermediate mix-up domain and adapts the mix-up ratio to mitigate the domain shift.

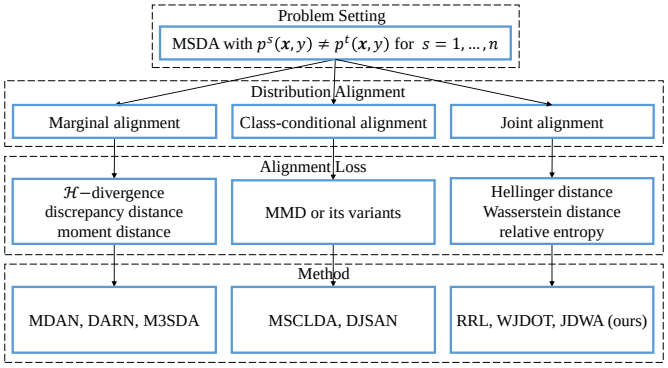


Fig. 2: Illustration of our JDWA approach and other MSDA methods from the perspectives of distribution alignment and alignment loss.

Discussion. Our work aims to address the MSDA problem, which is different from the FSDA and MSFSDA problems and requires sufficient labeled source data for training the model. Under such a setting, our joint distribution weighted alignment can be performed on the sufficient labeled data that well reflect the joint distributions. When facing FSDA and MSFSDA with very limited labeled source data, it may be more effective to exploit the strategies in [53]–[56] to address these problems, including prototypical self-supervised learning, prototypical contrastive learning, and mix-up.

IV. Further Discussions

We discuss our JDWA approach and other MSDA methods in further detail from the perspectives of distribution alignment and alignment loss. These methods include MDAN [11], DARN [13], M3SDA [14], MSCLDA [20], DJSAN [21], RRL [22], and WJDOT [45]. For clarity, we illustrate the two perspectives in Fig. 2 and discuss each one below.

Distribution alignment. MSDA considers the problem where the multiple joint source distributions are relevant but

distinct from the joint target distribution, *i.e.*, $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$ for $s = 1, \cdots, n$. To tackle the problem, MDAN, DARN, and M3SDA align the marginal source distributions and marginal target distribution $p^1(\boldsymbol{x}), \cdots, p^n(\boldsymbol{x}), p^t(\boldsymbol{x})$. MSCLDA and DJSAN match the marginals, and the class-conditionals $p^1(\boldsymbol{x}|y), \cdots, p^n(\boldsymbol{x}|y), p^t(\boldsymbol{x}|y)$. However, these methods fail to fully address the MSDA challenge, and the joint distribution disparities $p^s(\boldsymbol{x}, y) \neq p^t(\boldsymbol{x}, y)$ for $s = 1, \cdots, n$ may still exist. As a result, minimizing the weighted source loss $\sum_{s=1}^{n} \frac{\alpha^s}{m_s} \sum_{i=1}^{m_s} \mathcal{L}(f(\boldsymbol{x}_i^s), y_i^s)$ in these methods may not lead to the desired small target loss $\frac{1}{m_t} \sum_{i=1}^{m_t} \mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t)$ and the good estimation of target labels. This is because the weighted source loss is connected to the joint source distributions $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y)$ via $\sum_{s=1}^{n} \frac{\alpha^s}{m_s} \sum_{i=1}^{m_s} \mathcal{L}(f(\boldsymbol{x}_i^s), y_i^s) \approx \sum_{s=1}^{n} \alpha^s \int \mathcal{L}(f(\boldsymbol{x}), y) p^s(\boldsymbol{x}, y) d\boldsymbol{x} dy$ and the target loss is connected to the joint target distribution $p^t(\boldsymbol{x}, y)$ via $\frac{1}{m_t} \sum_{i=1}^{m_t} \mathcal{L}(f(\boldsymbol{x}_i^t), y_i^t) \approx \int \mathcal{L}(f(\boldsymbol{x}), y) p^t(\boldsymbol{x}, y) d\boldsymbol{x} dy$. By contrast, RRL, WJDOT, and our JDWA address the MSDA challenge by matching joint distributions $p^1(\boldsymbol{x}, y), \cdots, p^n(\boldsymbol{x}, y), p^t(\boldsymbol{x}, y)$. Within the joint alignment, RRL overlooks the varying relevance of the sources to the target, and naively assigns the same weight to each source, which can hurt the target performance. WJDOT does not learn the features to align joint distributions, and instead uses fixed features extracted from multi-task learning, which can suffer from the problem of under alignment. In comparison, our JDWA optimizes both the relevance weights and network features to match the weighted joint source distribution to the joint target distribution.

Alignment loss. To align distributions, MDAN, DARN, and M3SDA exploit the $\mathcal{H}$-divergence, discrepancy distance, and moment distance as their alignment losses, respectively. MSCLDA and DJSAN utilize the MMD and joint semantic MMD. By contrast, RRL, WJDOT, and our JDWA make use of the Hellinger distance, Wasserstein distance, and relative entropy, which can directly handle the joint (features and label) distribution disparity. Note that some statistical distances like $\mathcal{H}$-divergence, moment distance, and MMD are only defined for the distance between marginal distributions or class-conditional distributions. Compared with the Hellinger distance and Wasserstein distance, the relative entropy is a popular one, and has proven effective in several related problems, including domain adaptation [6], [25], [26], domain generalization [17], and representation learning [57]. Later in Subsection V-D "**Statistical distances**", we study the performance of other statistical distances to justify our choice of the relative entropy in this work. While prior works [6], [17], [25], [57] have also made use of the relative entropy, different from them, we develop a kernel-based method named KREE for relative entropy estimation, and solve a convex problem to derive the estimate.

V. Experiments

A. Experimental Datasets

We conduct extensive experiments on 6 object recognition datasets. Among these datasets, the number of classes varies from a few (7 in PACS) to many (345 in DomainNet), and the number of samples varies from small (1800 in ImageCLEF) to large (0.6 million in DomainNet). We briefly introduce each dataset as follows. **(1) PACS** [58] includes 4 domains and 7 classes. The 4 domains are Photo, ArtPainting, Cartoon, and Sketch, which contain 1670, 2048, 2344, and 3929 images. **(2) ImageCLEF**[3] consists of 3 domains and 12 classes. The 3 domains are Caltech, ImageNet, and Pascal, and each domain contains 600 images. **(3) Office** [59] is comprised of 3 domains and 31 classes. The 3 domains are DSLR, Amazon, and Webcam, which contain 795, 2817, and 498 images. **(4) Office-Home** [60] includes 4 domains and 65 classes. The 4 domains are Art, Clipart, Product, and RealWorld, which contain 2421, 4379, 4428, and 4357 images. **(5) miniDomainNet** [61] has 4 domains and 126 classes. The 4 domains are Clipart, Painting, Real, and Sketch, which contain 18703, 31202, 65609, and 24492 images. **(6) DomainNet** [14] contains 6 domains and 345 classes. The 6 domains are Infograph, Painting, Clipart, Quickdraw, Real, and Sketch. This dataset has around 0.6 million images, and is specifically proposed for large-scale MSDA experiments.

B. Experimental Setup

We describe the experimental settings in the following.

Comparison methods. We compare the proposed JDWA approach with the Baseline, DA, and MSDA methods. (i) The Baseline method combines multiple source datasets into a single dataset, and minimizes the cross entropy loss on the resulting dataset to train the network. (ii) The DA methods contain f-DAL [6], DANN [44], and DAN [62]. These methods also use the combined source dataset to train the networks, since they are designed for the single source problem. (iii) The MSDA methods are MDAN [11], M3SDA [14], DARN [13], WJDOT [45], MIAN [15], RRL [22], SImpAl [63], LTC [47], T-SVDNet [49], and CSR [52]. Among these 10 comparison methods, the first 6 ones address MSDA by distribution alignment, and the last 4 ones tackle the problem via other strategies (implicit alignment, knowledge aggregation, tensor singular value decomposition, cycle self-refinement). It is worth noting that if the comparison methods' experimental results on some datasets are available in prior works (*e.g.*, [15], [22], [47], [52], [63]), we directly cite the available results of the methods. Otherwise, we report the best results of the comparison methods on the datasets by running their official source codes and using their suggested hyperparameters. With the complete results on every dataset, we are able to comprehensively contrast our approach with its competitors.

Evaluation protocol. We follow [13], [15], [47], [52] to build the MSDA task. For a dataset like PACS, we first designate one domain, for instance, ArtPainting, as target domain, and then utilize the remaining domains as source domains. This particular MSDA task is written as "→ArtPainting". Our evaluation of a method's performance is based on its target classification accuracy (%). Since the classification accuracy may vary due to different random seeds, each experiment is

[3]https://www.imageclef.org/2014/adaptation

TABLE I: Classification results with ResNet50 on the PACS dataset (7 classes).

Method	→ArtPainting	→Cartoon	→Photo	→Sketch	Avg
Baseline	83.50 (0.47)	76.88 (0.73)	98.20 (0.53)	70.84 (0.52)	82.35
f-DAL [6]	90.82 (0.42)	89.77 (0.65)	98.74 (0.56)	70.69 (0.87)	87.50
DANN [44]	87.16 (0.92)	84.55 (0.80)	97.18 (0.64)	81.68 (1.21)	87.64
DAN [62]	87.45 (0.66)	84.32 (0.58)	98.54 (0.72)	78.63 (0.76)	87.24
MDAN [11]	85.16 (0.70)	82.17 (0.67)	98.38 (0.88)	73.68 (1.05)	84.84
M3SDA [14]	90.82 (0.76)	85.57 (0.62)	98.50 (0.73)	85.36 (0.90)	90.06
DARN [13]	88.19 (0.81)	85.22 (0.92)	97.95 (0.50)	80.83 (0.86)	88.05
MIAN [15]	86.72 (0.57)	84.25 (0.83)	98.80 (0.34)	77.30 (0.66)	86.76
WJDOT [45]	87.35 (0.67)	84.77 (0.62)	98.05 (0.45)	76.48 (0.80)	86.66
RRL [22]	92.48 (0.46)	84.34 (0.70)	98.98 (0.58)	85.35 (0.82)	90.29
LTC [47]	88.38 (0.53)	86.38 (0.84)	98.52 (0.38)	86.42 (0.59)	89.93
SImpAl [63]	85.38 (0.87)	82.56 (0.75)	98.50 (0.63)	74.28 (0.50)	85.18
T-SVDNet [49]	89.01 (0.67)	84.92 (0.79)	98.83 (0.53)	85.25 (0.88)	89.50
CSR [52]	92.30 (0.00)	90.20 (0.00)	98.50 (0.00)	87.70 (0.00)	92.20
JDWA (ours)	**94.85 (0.57)**	**92.67 (0.66)**	**99.25 (0.38)**	**89.55 (0.68)**	**94.08**

repeated 5 times to obtain the classification accuracy's mean and standard deviation.

Implementation details. We implement our JDWA approach by using the toolboxes of Pytorch and pytorch-minimize. The neural network in our approach includes both the ResNet50 and ResNet101 [64] models. In particular, on each of the 5 datasets PACS, $\cdots$, miniDomainNet, following [15], [20], [22], [63], the network is the pretrained ResNet50 model, whose final classification layer is rebuilt such that the number of network outputs is the same as the number of classes in the dataset (7 for PACS, $\cdots$, 126 for miniDomainNet). Moreover, on the remaining very large dataset DomainNet, following [14], [47], [49], the network is the pretrained ResNet101 model, whose final classification layer is reconstructed to fit the dataset with 345 classes. We optimize the relevance weights and the network respectively by the L-BFGS and the minibatch SGD algorithms (again see Algorithm 2). For the hyperparameters of our approach, we set the width σ of the Gaussian kernel to the median pairwise squared distances on the training data following the practice in [32], [35], [65], [66], and set the tradeoff parameter λ_{RE} following the practice in [44], [46], [65], [66]. That is, we change λ_{RE} from 0 to 1 by using the formula $\lambda_p = \frac{2}{1+e^{-10p}} - 1$, where p is the training progress linearly changing from 0 to 1. Moreover, we empirically set the regularization parameter to $\lambda = 0.01$, and the other tradeoff parameter to $\lambda_{\mathrm{E}} = 0.1$. Later in Subsection V-D "Hyperparameter sensitivity", we justify the settings of these 2 parameters by conducting sensitivity analysis. For the hyperparameters of the L-BFGS algorithm, following the practice in [33], we set the maximum iteration times to a small number of 5 for the optimization of $\boldsymbol{\alpha}$ to reduce the computational cost brought by the nested optimization. In addition, we run the algorithm until convergence for the optimization of $\boldsymbol{\theta}$ in Eq. (12), with the optimal $\boldsymbol{\alpha}$ being fixed. For the hyperparameters of minibatch SGD, we use 0.001 as the feature extractor's learning rate and 0.01 as the classifier's learning rate, and sample $n+1$ minibatches of data from the $n+1$ domain datasets per iteration.

C. Experimental Results with Statistical Tests

We first present the classification results, and then run the statistical tests to verify if our approach performs better than its competitors in a statistical sense.

TABLE II: Classification results with ResNet50 on the ImageCLEF dataset (12 classes).

Method	→Caltech	→ImageNet	→Pascal	Avg
Baseline	91.96 (0.24)	88.23 (0.10)	77.42 (0.13)	85.87
f-DAL [6]	95.24 (0.21)	93.71 (0.20)	78.05 (0.18)	89.00
DANN [44]	93.78 (0.11)	91.70 (0.26)	77.85 (0.32)	87.78
DAN [62]	93.15 (0.17)	92.76 (0.16)	77.47 (0.08)	87.79
MDAN [11]	93.61 (0.35)	91.45 (0.32)	77.27 (0.24)	87.44
M3SDA [14]	94.25 (0.20)	91.90 (0.31)	77.72 (0.10)	87.96
DARN [13]	93.85 (0.42)	90.63 (0.24)	77.02 (0.16)	87.17
MIAN [15]	95.14 (0.28)	91.45 (0.17)	77.56 (0.40)	88.05
WJDOT [45]	95.47 (0.16)	92.67 (0.35)	77.82 (0.25)	88.65
RRL [22]	97.63 (0.23)	**96.86 (0.15)**	79.98 (0.30)	91.50
LTC [47]	96.02 (0.32)	94.14 (0.37)	78.60 (0.42)	89.59
SImpAl [63]	**99.30 (0.30)**	91.00 (0.40)	77.50 (0.30)	89.27
T-SVDNet [49]	95.46 (0.30)	93.67 (0.22)	78.87 (0.35)	89.33
CSR [52]	96.20 (0.00)	95.00 (0.00)	79.20 (0.00)	90.10
JDWA (ours)	98.21 (0.28)	96.54 (0.25)	**80.12 (0.18)**	**91.62**

TABLE III: Classification results with ResNet50 on the Office dataset (31 classes).

Method	→Amazon	→DSLR	→Webcam	Avg
Baseline	66.92 (0.45)	99.52 (0.16)	96.22 (0.29)	87.56
f-DAL [6]	73.40 (0.28)	99.46 (0.10)	98.25 (0.38)	90.37
DANN [44]	67.97 (0.38)	99.71 (0.37)	96.64 (0.26)	88.10
DAN [62]	67.69 (0.56)	99.60 (0.26)	96.68 (0.44)	87.99
MDAN [11]	66.12 (0.32)	99.68 (0.16)	97.86 (0.15)	87.88
M3SDA [14]	69.41 (0.82)	99.64 (0.19)	99.30 (0.31)	89.45
DARN [13]	66.36 (0.43)	99.84 (0.08)	98.67 (0.19)	88.29
MIAN [15]	76.17 (0.24)	99.22 (0.35)	98.39 (0.76)	91.26
WJDOT [45]	73.72 (0.39)	99.70 (0.27)	98.87 (0.30)	90.76
RRL [22]	77.45 (0.26)	99.92 (0.10)	99.42 (0.38)	92.26
LTC [47]	69.22 (0.28)	99.57 (0.34)	99.60 (0.45)	89.46
SImpAl [63]	70.60 (0.60)	99.20 (0.20)	97.40 (0.10)	89.00
T-SVDNet [49]	74.07 (0.38)	99.42 (0.61)	**99.63 (0.18)**	91.04
CSR [52]	78.60 (0.00)	**100.00 (0.00)**	99.60 (0.00)	**92.70**
JDWA (ours)	**78.67 (0.32)**	99.95 (0.18)	99.35 (0.23)	92.66

Classification results. We report in Tables I-VI the classification results on the 6 datasets, respectively. For every column in the table, we highlight the best result in **bold**, and underline the second best result. Again note that, for the comparison methods (especially the most related MSDA ones) whose results are not available on some datasets, we run their official source codes to produce their best results on those datasets, such that we can comprehensively contrast our approach with them. In Table I for the PACS dataset, our JDWA approach consistently outperforms the comparison methods on all the 4 tasks, and yields the best average classification accuracy of 94.08%. This result improves the Baseline method by a large margin of 10% (94.08% versus 82.35%), and the second best method CSR by a margin of 1.88% (94.08% versus 92.20%). In Tables II and III for the ImageCLEF and Office datasets, our approach is among the top performing methods. On the total 6 tasks, it outperforms the comparison methods on 2 tasks (*i.e.*,"→Pascal", "→Amazon"), and yields comparable performance to the best methods on the remaining 4 tasks. In Tables IV and V for the Office-Home and miniDomainNet datasets, our approach delivers much better results than its

TABLE IV: Classification results with ResNet50 on the Office-Home dataset (65 classes).

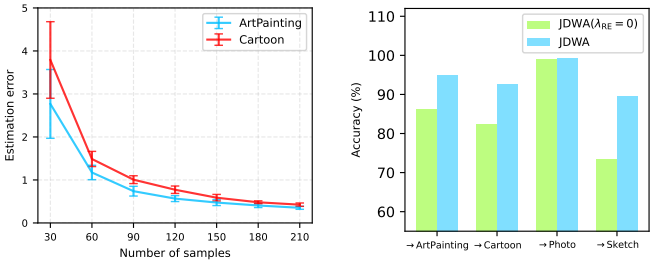
Method	→Art	→Clipart	→Product	→RealWorld	Avg
Baseline	67.40 (0.39)	52.38 (0.38)	77.95 (0.28)	80.70 (0.17)	69.61
f-DAL [6]	72.30 (0.47)	64.26 (0.39)	83.15 (0.41)	82.81 (0.30)	75.63
DANN [44]	69.19 (0.58)	57.08 (0.19)	79.18 (0.70)	82.63 (0.40)	72.02
DAN [62]	69.62 (0.87)	56.50 (1.36)	79.66 (0.65)	83.16 (0.40)	72.23
MDAN [11]	69.57 (0.36)	55.22 (0.46)	79.71 (0.29)	81.48 (0.19)	71.49
M3SDA [14]	66.22 (0.52)	58.55 (0.62)	79.45 (0.52)	81.35 (0.19)	71.39
DARN [13]	68.93 (0.44)	55.64 (0.61)	79.72 (0.27)	82.45 (0.31)	71.68
MIAN [15]	69.88 (0.35)	64.20 (0.68)	80.87 (0.37)	81.49 (0.24)	74.11
WJDOT [45]	69.46 (0.44)	59.65 (0.35)	81.14 (0.70)	82.61 (0.61)	73.22
RRL [22]	75.02 (0.18)	68.93 (0.42)	85.61 (0.19)	84.05 (0.36)	78.40
LTC [47]	70.44 (0.21)	64.26 (0.43)	82.19 (0.55)	81.75 (0.43)	74.66
SImpAl [63]	70.80 (0.20)	56.30 (0.20)	80.20 (0.30)	81.50 (0.30)	72.20
T-SVDNet [49]	71.94 (0.30)	65.06 (0.59)	82.59 (0.35)	81.79 (0.03)	75.34
CSR [52]	76.70 (0.00)	71.40 (0.00)	86.80 (0.00)	85.50 (0.00)	80.10
JDWA (ours)	**77.00 (0.24)**	**72.36 (0.43)**	**87.29 (0.30)**	**86.14 (0.28)**	**80.70**

TABLE V: Classification results with ResNet50 on the miniDomainNet dataset (126 classes).

Method	→Clipart	→Painting	→Real	→Sketch	Avg
Baseline	70.76 (0.66)	66.31 (0.72)	71.29 (0.82)	63.04 (0.25)	67.85
f-DAL [6]	74.14 (0.56)	70.78 (0.95)	76.83 (0.38)	66.79 (0.73)	72.14
DANN [44]	68.23 (0.74)	66.40 (0.60)	69.65 (0.47)	64.78 (0.62)	67.26
DAN [62]	68.50 (0.62)	66.86 (0.47)	69.22 (0.54)	64.38 (0.50)	67.24
MDAN [11]	70.17 (0.81)	68.33 (0.76)	78.83 (0.80)	66.17 (0.64)	70.87
M3SDA [14]	74.45 (0.35)	70.85 (0.56)	76.54 (0.74)	67.51 (0.70)	72.34
DARN [13]	74.67 (0.92)	68.41 (0.70)	72.83 (0.36)	67.05 (0.64)	70.74
MIAN [15]	72.67 (0.60)	65.17 (0.65)	78.33 (0.52)	62.17 (0.76)	69.58
WJDOT [45]	73.44 (0.76)	66.89 (0.50)	74.89 (0.63)	64.90 (0.88)	70.03
RRL [22]	79.50 (0.43)	74.28 (0.27)	80.78 (0.53)	74.03 (0.55)	77.15
LTC [47]	75.64 (0.30)	71.23 (0.35)	78.05 (0.40)	69.80 (0.65)	73.68
SImpAl [63]	73.01 (0.71)	69.83 (0.60)	76.55 (0.67)	65.97 (0.74)	71.34
T-SVDNet [49]	75.02 (0.76)	72.03 (0.50)	78.47 (0.72)	69.55 (0.62)	73.77
CSR [52]	77.90 (0.00)	73.50 (0.00)	79.43 (0.00)	73.50 (0.00)	76.10
JDWA (ours)	**82.45 (0.42)**	**76.30 (0.50)**	**83.86 (0.54)**	**76.75 (0.65)**	**79.84**

competitors on all the 8 tasks, and reaches the best average classification accuracy of 80.70% and 79.84%, respectively. Finally in Table VI for the DomainNet dataset, our approach yields the best results on half of the tasks, and the competitive results on the remaining tasks. These classification results imply that on the tested few-class/many-class datasets and small/large-scale datasets, our JDWA approach is superior to the comparison methods. We also notice that some methods like f-DAL, RRL, T-SVDNet, and CSR, have obtained good results on the datasets. However, these methods overlook the multi-source problem (f-DAL), do not address the joint distribution discrepancy (f-DAL, T-SVDNet, CSR), or disregard the various degrees of relevance of the sources to the target (RRL). These limitations may have contributed to the less satisfactory results of the comparison methods.

Statistical tests. To evaluate whether our JDWA approach statistically outperforms the other methods, we conduct the Wilcoxon signed-rank test [67] based on the results presented in Tables I-VI. This non-parametric test compares the performance of two methods across N tasks using a statistic T. In this study, we have $N = 24$ tasks and 14 comparison methods, leading to 14 separate tests: Baseline vs JDWA, $\cdots$, and CSR vs JDWA. The computed T values are shown in Table VII. According to the table, with $N = 24$ and a significance level of $\alpha = 0.05$, no T value exceeds the critical value of 81 for the Wilcoxon test. This indicates that, statistically, our JDWA approach, which integrates joint distribution weighted alignment and entropy regularization, significantly outperforms the comparison methods in addressing the MSDA problem.



(a) Estimation error (b) KREE loss

Fig. 3: Estimation error of our KREE method and contribution of KREE loss to our JDWA approach. (a) Mean and standard deviation of estimation error with various numbers of samples. (b) Classification accuracy of JDWA and JDWA ($\lambda_{\mathrm{RE}} = 0$).

D. Experimental Analysis

Below, we experimentally analyze our KREE method and JDWA approach.

Estimation error. We study the estimation error of our KREE method with the varying numbers of samples. Specifically, we consider the relative entropy from weighted joint source distribution $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) = \frac{1}{3}\sum_{s=1}^{3} p^s(h(\boldsymbol{x}), y)$ to joint target distribution $p^t(h(\boldsymbol{x}), y)$, where $p^s(h(\boldsymbol{x}), y) = p^t(h(\boldsymbol{x}), y) = p(h(\boldsymbol{x}), y)$ $(\forall s)$ and h is a fixed feature extractor here. In this case, we know the value of relative entropy, *i.e.*, $\mathrm{RE}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \| p^t(h(\boldsymbol{x}), y)) = 0$. Therefore, we can measure the estimation error of our method, *i.e.*, the absolute difference between the relative entropy and its estimated value. Let $p(h(\boldsymbol{x}), y)$ be the domains ArtPainting and Cartoon from PACS, respectively. We sample the same number of source and target samples from $p(h(\boldsymbol{x}), y)$ to compute the estimated relative entropy. The sampling procedure is repeated 10 times to calculate the mean and standard deviation of the estimation error. Fig. 3(a) plots the results by varying the number of samples. We find that for both domains, the estimation errors decrease with the increase in sample size, and approach 0 as the sample size becomes sufficiently large. This suggests that to a certain extent, our KREE method is a reasonable and reliable estimator. In addition, it also implies that in a probabilistic sense, a larger sample size can lead to a smaller estimation error of our method.

KREE loss. We investigate the contribution of the KREE loss to the performance of our JDWA approach, by considering a variant of our approach without this loss, *i.e.*, JDWA ($\lambda_{\mathrm{RE}} = 0$). Fig. 3(b) shows the classification results of JDWA and JDWA ($\lambda_{\mathrm{RE}} = 0$) on PACS. We find that without the KREE loss, the performance of our approach significantly degrades on most of the evaluated tasks. This indicates that the KREE loss, which is responsible for joint distribution weighted alignment under the relative entropy, is indispensable for the superior performance of our approach.

Entropy regularization. We evaluate the contribution of the entropy regularization term, by considering a variant of

TABLE VI: Classification results with ResNet101 on the DomainNet dataset (345 classes).

Method	→Clipart	→Infograph	→Painting	→Quickdraw	→Real	→Sketch	Avg
Baseline	47.60 (0.52)	13.10 (0.41)	38.10 (0.45)	13.30 (0.39)	51.90 (0.85)	33.70 (0.54)	32.95
f-DAL [6]	64.20 (0.31)	24.50 (0.53)	53.10 (0.70)	14.60 (0.62)	61.20 (0.67)	53.20 (0.51)	45.13
DANN [44]	60.60 (0.42)	25.80 (0.34)	50.40 (0.51)	7.70 (0.68)	62.00 (0.66)	51.70 (0.19)	43.00
DAN [62]	60.23 (0.29)	26.52 (0.48)	51.14 (0.75)	8.22 (0.47)	61.01 (0.56)	50.23 (0.84)	42.89
MDAN [11]	60.30 (0.41)	25.00 (0.43)	50.30 (0.36)	8.20 (1.92)	61.50 (0.46)	51.30 (0.58)	42.80
M3SDA [14]	58.60 (0.53)	26.00 (0.89)	52.30 (0.55)	6.30 (0.58)	62.70 (0.51)	49.50 (0.76)	42.60
DARN [13]	59.24 (0.32)	24.78 (0.56)	51.25 (0.66)	8.91 (0.72)	61.88 (0.63)	51.55 (0.38)	42.94
MIAN [15]	60.57 (0.54)	24.45 (0.46)	55.25 (0.58)	14.88 (0.64)	62.78 (0.52)	50.76 (0.48)	44.78
WJDOT [45]	61.34 (0.67)	23.65 (0.67)	53.79 (0.39)	13.24 (0.50)	62.80 (0.83)	50.76 (0.57)	44.26
RRL [22]	69.54 (0.59)	28.35 (0.53)	59.21 (0.41)	29.48 (0.44)	69.44 (0.64)	**62.33 (0.57)**	53.06
LTC [47]	63.10 (0.50)	**28.70 (0.70)**	56.10 (0.50)	16.30 (0.50)	66.10 (0.60)	53.80 (0.60)	47.40
SImpAl [63]	66.40 (0.80)	26.50 (0.50)	56.60 (0.70)	18.90 (0.80)	68.00 (0.50)	55.50 (0.30)	48.60
T-SVDNet [49]	66.10 (0.40)	25.00 (0.80)	54.30 (0.70)	16.50 (0.90)	65.40 (0.50)	54.60 (0.60)	47.00
CSR [52]	**73.00** (0.00)	28.10 (0.00)	58.80 (0.00)	26.00 (0.00)	71.10 (0.00)	60.70 (0.00)	52.90
JDWA (ours)	71.45 (0.45)	27.76 (0.36)	**60.88 (0.62)**	**31.82 (0.55)**	**72.34 (0.68)**	61.42 (0.56)	**54.28**

TABLE VII: Results of Wilcoxon's test of each comparison method vs our JDWA approach.

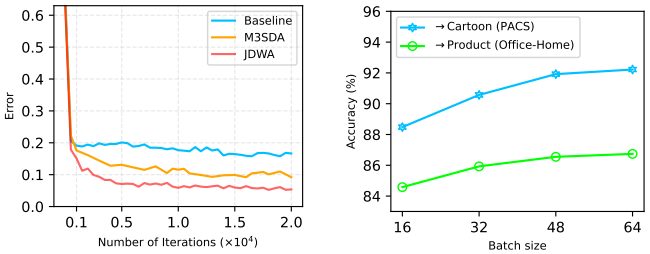
Method	Baseline	f-DAL	DAN	MDAN	DANN	M3SDA	MIAN	DARN	WJDOT	CSR	RRL	LTC	SImpAl	T-SVDNet
T value	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	36.00	22.00	5.00	3.00	1.00

TABLE VIII: Results of JDWA and JDWA ($\lambda_{\mathrm{E}} = 0$) on PACS.

Method	→ArtPainting	→Cartoon	→Photo	→Sketch	Avg
JDWA	94.85 (0.57)	92.67 (0.66)	99.25 (0.38)	89.55 (0.68)	94.08
JDWA ($\lambda_{\mathrm{E}} = 0$)	93.98 (0.45)	90.78 (0.42)	98.73 (0.40)	88.88 (0.74)	93.09

TABLE IX: Results of JDWA and JDWA ($\alpha^s = \frac{1}{n}$) on PACS.

Method	→ArtPainting	→Cartoon	→Photo	→Sketch	Avg
JDWA	94.85 (0.57)	92.67 (0.66)	99.25 (0.38)	89.55 (0.68)	94.08
JDWA ($\alpha^s = \frac{1}{n}$)	92.43 (0.31)	91.29 (0.95)	99.19 (0.16)	86.89 (0.82)	92.45



(a) Convergence behavior (b) Batch size

Fig. 4: Convergence behavior of our approach and impact of batch size on our approach. (a) Convergence behavior on the task "→ArtPainting" (PACS). (b) Classification accuracy with various batch sizes, where the batch size is for each domain.

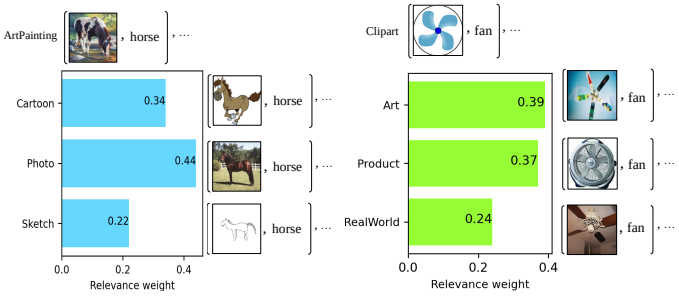
our JDWA approach without this term, *i.e.*, JDWA ($\lambda_{\mathrm{E}} = 0$). Table VIII presents the classification results of JDWA and JDWA ($\lambda_{\mathrm{E}} = 0$) on PACS. We observe that JDWA performs slightly better than JDWA ($\lambda_{\mathrm{E}} = 0$) on the tested evaluations. This suggests that adding the entropy regularization term, which reduces the uncertainty of target predictions and ensures reliable target pseudo labels, is beneficial for improving JDWA's performance.

Weight contribution. We evaluate the relevance weights' contribution by contrasting our JDWA against its variant JDWA ($\alpha^s = \frac{1}{n}$) with the fixed same weight $\alpha^s = \frac{1}{n}$ for each source domain (joint distribution). Table IX shows the classification results of JDWA and JDWA ($\alpha^s = \frac{1}{n}$) on PACS. The results demonstrate that JDWA, which adjusts the weights based on the source and target data, performs better than its variant JDWA ($\alpha^s = \frac{1}{n}$). This indicates that the relevance weights are crucial and significantly contribute to the improved performance of our approach.

Convergence behavior. We study the convergence behavior of our JDWA approach on the task "→ArtPainting" (PACS) as an example. In addition, we include the Baseline and an MSDA method named M3SDA for comparison. Fig. 4(a) illustrates the convergence results. We find that the target error of our approach gradually decreases with each iteration, and converges after a sufficient number of iterations. Ultimately, our approach achieves a lower target error than the Baseline and the M3SDA methods.

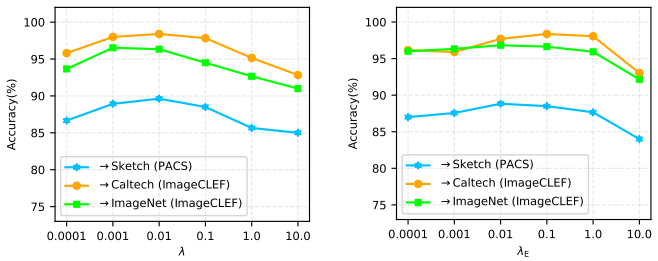
Batch size. We analyze the impact of batch size on our approach. Fig. 4(b) shows the classification results on 2 tasks "→Cartoon" (PACS) and "→Product" (Office-Home), with various batch sizes. Note that the batch size here is for each domain, and the same batch size of data is sampled from all domains during the minibatch SGD. We observe that our approach tends to benefit from a larger batch size. We believe that this is because a larger batch size encourages our KREE method to better estimate the relative entropy (see the previous paragraph "**Estimation error**"). Consequently, minimization of our objective (15), which contains the estimated relative entropy, better reduces the joint distribution disparity and increases the classification accuracy. Therefore, we recommend increasing the batch size of our approach whenever possible.

Weight visualization. We plot in Fig. 5(a) and Fig. 5(b) the relevance weights associated with the source domains (joint source distributions) on the tasks where the target domains are ArtPainting (PACS) and Clipart (Office-Home), respectively. We observe from Fig. 5(a) a very interesting phenomenon:



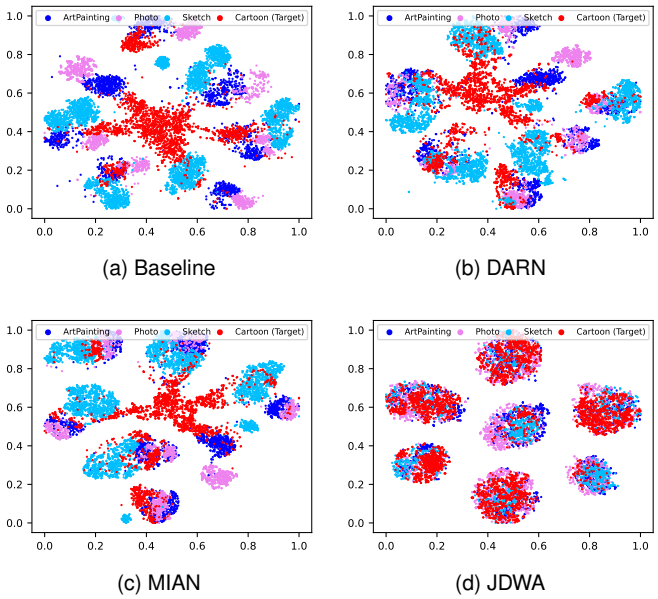
(a) PACS (b) Office-Home

Fig. 5: Relevance weights associated with the source domains. (a) ArtPainting (PACS) is the target domain and the remaining ones are the source domains. (b) Clipart (Office-Home) is the target domain and the remaining ones are the source domains.



(a) Accuracy w.r.t. λ ($\lambda_{\mathrm{E}} = 0.1$) (b) Accuracy w.r.t. λ_{E} ($\lambda = 0.01$)

Fig. 6: Classification accuracy of our approach with respect to (w.r.t.) different values of parameters λ and λ_{E}.



(a) Baseline (b) DARN

(c) MIAN (d) JDWA

Fig. 7: T-SNE embeddings of source data (ArtPainting, Photo, and Sketch) and target data (Cartoon) in the feature spaces of Baseline, DARN, MIAN, and JDWA.

The source domain Photo, which visually resembles the target domain ArtPainting, receives a significant weight of 0.44, while the least relevant domain, Sketch, is assigned a smaller weight of 0.22. Additionally, in Fig. 5(b), we observe that the source domains Art and Product, which are more relevant to the target domain Clipart, receive higher weights as well. These findings indicate that our approach effectively captures the varying degrees of relevance between the source domains and the target domain.

Hyperparameter sensitivity. We examine the sensitivity of our approach to different settings of the regularization parameter λ and tradeoff parameter λ_{E}. To do this, we plot the classification accuracy of our approach in Fig. 6(a) and Fig. 6(b), by respectively varying the values of λ and λ_{E}. While testing one parameter, the other is fixed at $\lambda_{\mathrm{E}} = 0.1$ and $\lambda = 0.01$. From Fig. 6(a), we find that on all 3 tasks, the best accuracy is attained when the regularization parameter λ is around 0.01, and the accuracy decreases when λ is too small or too large, *i.e.*, $\lambda < 0.001$ or $\lambda > 0.1$. This implies the importance of a proper regularization strength to our approach. Besides, Fig. 6(b) shows that setting the tradeoff parameter λ_{E} within the range $[0.01, 1.0]$ well balances between the entropy and the other two losses in our objective function, and leads to the best performance of our approach. According to the above analysis, we therefore recommend setting the parameters $\lambda = 0.01$ and λ_{E} within the range $[0.01, 1.0]$, as we have done in the main experiments.

Feature visualization. We visualize in Fig. 7(a)-Fig. 7(d) the t-SNE embeddings [68] of the source data and target data in the feature spaces of Baseline, DARN, MIAN, and JDWA. Specifically, the t-SNE technique we use is implemented in the sklearn library[4] with the default parameters. The source data are from three domains: ArtPainting, Photo, and Sketch, and the target data are from the remaining domain, Cartoon, where the domains are from the PACS dataset. We observe that our JDWA approach effectively aligns the multiple source domain data to the target data in the network's feature space, and outperforms its competitors (Baseline, DARN, and MIAN) in terms of alignment. This demonstrates the effectiveness and power of our approach in tackling the MSDA problem.

Statistical distances. We study the performance of other statistical distances to justify our choice of the relative entropy. Since the Hellinger distance, Wasserstein distance, and Jensen-Shannon divergence can also measure the joint distribution distance [22], [45], we replace relative entropy with each of these distances and derive 3 variants of our JDWA approach: JDWA with Hellinger distance (JDWA-H), JDWA with Wasserstein distance (JDWA-W), and JDWA with Jensen-Shannon divergence (JDWA-JS). Table X presents the comparison results of JDWA, JDWA-H, JDWA-W, and JDWA-JS on the miniDomainNet dataset. The results show that JDWA consistently outperforms JDWA-W and JDWA-JS, and demonstrates better performance than JDWA-H. These findings suggest that the relative entropy is more suitable and effective for our approach compared to other statistical distances.

[4]https://scikit-learn.org/0.16/modules/generated/sklearn.manifold.TSNE.html

TABLE X: Results of JDWA, JDWA-H, JDWA-W, and JDWA-JS on miniDomainNet.

Method	→Clipart	→Painting	→Real	→Sketch	Avg
JDWA	**82.45 (0.42)**	76.30 (0.50)	**83.86 (0.54)**	**76.75 (0.65)**	**79.84**
JDWA-H	80.41 (0.41)	**76.45 (0.64)**	82.38 (0.35)	74.61 (0.66)	78.46
JDWA-W	76.83 (0.58)	72.03 (0.72)	79.74 (0.29)	72.80 (0.72)	75.35
JDWA-JS	78.53 (0.37)	74.29 (0.60)	82.73 (0.48)	73.79 (0.55)	77.34

VI. Conclusion

In this article, we introduce the JDWA approach to tackle the challenge of disparate joint distributions between sources and target in MSDA. Our JDWA approach aligns the weighted sum of multiple joint source distributions to the joint target distribution under the relative entropy, and performs entropy regularization to improve the performance. Besides, we design the KREE method to estimate the relative entropy from samples, enabling the estimated value to serve as the practical alignment loss. Our KREE method solves a convex optimization problem to achieve the estimation. The experimental results on various benchmark datasets demonstrate the statistical superiority of our approach over the comparison methods. Our next research plan is to explore new neural network architectures to further improve our approach. For example, in the new architecture, we can try configuring different classifiers for different domains. We also plan to extend our approach to explore the Multi-Source Open-Set Domain Adaptation (MS-OSDA) problem. In this problem, one of the challenges is also the disparities between the multiple joint source distributions and the joint target distribution.

References

[1] V. N. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[2] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.

[3] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. New York, NY: Springer, 2004.

[4] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. Cambridge, MA: MIT Press, 2008.

[5] S. Chen, L. Han, X. Liu, Z. He, and X. Yang, "Subspace distribution adaptation frameworks for domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 12, pp. 5204–5218, 2020.

[6] D. Acuna, G. Zhang, M. T. Law, and S. Fidler, "f-domain adversarial learning: Theory and algorithms," in *International Conference on Machine Learning*, vol. 139, 2021, pp. 66–75.

[7] A. T. Nguyen, T. Tran, Y. Gal, P. H. Torr, and A. G. Baydin, "Kl guided domain adaptation," in *International Conference on Learning Representations*, 2022, pp. 1–12.

[8] Y. Zheng, X. Wang, G. Zhang, B. Xiao, F. Xiao, and J. Zhang, "Multi-kernel coupled projections for domain adaptive dictionary learning," *IEEE Transactions on Multimedia*, vol. 21, no. 9, pp. 2292–2304, 2019.

[9] S. Chen, M. Harandi, X. Jin, and X. Yang, "Domain adaptation by joint distribution invariant projections," *IEEE Transactions on Image Processing*, vol. 29, pp. 8264–8277, 2020.

[10] Y. Lu, D. Li, W. Wang, Z. Lai, J. Zhou, and X. Li, "Discriminative invariant alignment for unsupervised domain adaptation," *IEEE Transactions on Multimedia*, vol. 24, pp. 1871–1882, 2022.

[11] H. Zhao, S. Zhang, G. Wu, J. M. F. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domain adaptation," in *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[12] R. Xu, Z. Chen, W. Zuo, J. Yan, and L. Lin, "Deep cocktail network: Multi-source unsupervised domain adaptation with category shift," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3964–3973.

[13] J. Wen, R. Greiner, and D. Schuurmans, "Domain aggregation networks for multi-source domain adaptation," in *International Conference on Machine Learning*, vol. 119, 2020, pp. 10 214–10 224.

[14] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *IEEE International Conference on Computer Vision*, 2019, pp. 1406–1415.

[15] G. Y. Park and S. W. Lee, "Information-theoretic regularization for multi-source domain adaptation," in *IEEE International Conference on Computer Vision*, 2021, pp. 9214–9223.

[16] D. Pernes and J. S. Cardoso, "Tackling unsupervised multi-source domain adaptation with optimism and consistency," *Expert Systems with Applications*, vol. 194, pp. 1–13, 2022.

[17] S. Zhao, M. Gong, T. Liu, H. Fu, and D. Tao, "Domain generalization via entropy regularization," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 3118–3129.

[18] A. T. Nguyen, T. Tran, Y. Gal, and A. G. Baydin, "Domain invariant representation learning with domain density transformations," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[19] Y. Yao, X. Li, Y. Zhang, and Y. Ye, "Multisource heterogeneous domain adaptation with conditional weighting adversarial network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 4, pp. 2079–2092, 2023.

[20] K. Li, J. Lu, H. Zuo, and G. Zhang, "Multi-source contribution learning for domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 10, pp. 5293–5307, 2022.

[21] Z. Cheng, S. Wang, D. Yang, J. Qi, M. Xiao, and C. Yan, "Deep joint semantic adaptation network for multi-source unsupervised domain adaptation," *Pattern Recognition*, vol. 151, pp. 1–11, 2024.

[22] S. Chen, L. Zheng, and H. Wu, "Riemannian representation learning for multi-source domain adaptation," *Pattern Recognition*, vol. 137, p. 109271, 2023.

[23] L. Wen, S. Chen, M. Xie, C. Liu, and L. Zheng, "Training multi-source domain adaptation network by mutual information estimation and minimization," *Neural Networks*, vol. 171, pp. 353–361, 2024.

[24] A. Basu, H. Shioya, and C. Park, *Statistical Inference: The Minimum Distance Approach*. New York: CRC press, 2011.

[25] M. Baktashmotlagh, M. Harandi, and M. Salzmann, "Learning domain invariant embeddings by matching distributions," in *Domain adaptation in computer vision applications*, 2017, pp. 95–114.

[26] S. Chen, H. Wu, and C. Liu, "Domain invariant and agnostic adaptation," *Knowledge-Based Systems*, vol. 227, p. 107192, 2021.

[27] M. Harandi, M. Salzmann, and M. Baktashmotlagh, "Beyond gauss: Image-set matching on the riemannian manifold of pdfs," in *IEEE International Conference on Computer Vision*, 2015, pp. 4112–4120.

[28] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in Neural Information Processing Systems*, 2004, pp. 529–536.

[29] J. Nocedal and S. Wright, *Numerical Optimization*. New York, NY: Springer, 1999.

[30] C.-X. Ren, Y.-W. Luo, and D.-Q. Dai, "Buresnet: Conditional bures metric for transferable representation learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 4198–4213, 2023.

[31] S. Chen, "Joint weight optimization for partial domain adaptation via kernel statistical distance estimation," *Neural Networks*, vol. 180, p. 106739, 2024.

[32] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, and Q. He, "Deep subdomain adaptation network for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 4, pp. 1713–1722, 2021.

[33] X. Gu, X. Yu, J. Sun, Z. Xu *et al.*, "Adversarial reweighting for partial domain adaptation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 860–14 872, 2021.

[34] L. Wen, S. Chen, Z. Hong, and L. Zheng, "Maximum likelihood weight estimation for partial domain adaptation," *Information Sciences*, vol. 676, p. 120800, 2024.

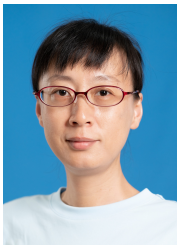
[35] S. Chen, Z. Hong, M. Harandi, and X. Yang, "Domain neural adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 11, pp. 8630–8641, 2023.

[36] S. Chen, "Decomposed adversarial domain generalization," *Knowledge-Based Systems*, vol. 263, p. 110300, 2023.

[37] P. Wang, Y. Yang, Y. Xia, K. Wang, X. Zhang, and S. Wang, "Information maximizing adaptation network with label distribution priors for unsupervised domain adaptation," *IEEE Transactions on Multimedia*, vol. 25, pp. 6026–6039, 2023.

[38] F. Ding, J. Li, W. Tian, S. Zhang, and W. Yuan, "Unsupervised domain adaptation via risk-consistent estimators," *IEEE Transactions on Multimedia*, vol. 26, pp. 1179–1187, 2024.
[39] H. Zhang, J. Tang, Y. Cao, Y. Chen, Y. Wang, and Q. M. J. Wu, "Cycle consistency based pseudo label and fine alignment for unsupervised domain adaptation," *IEEE Transactions on Multimedia*, vol. 25, pp. 8051–8063, 2023.
[40] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *International Conference on Machine Learning*, 2017, pp. 2208–2217.
[41] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Advances in Neural Information Processing Systems*, 2017, pp. 3730–3739.
[42] B. Bhushan Damodaran, B. Kellenberger, R. Flamary, D. Tuia, and N. Courty, "Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation," in *European Conference on Computer Vision*, 2018, pp. 447–463.
[43] S. Zhao, X. Yue, S. Zhang, B. Li, H. Zhao, B. Wu, R. Krishna, J. E. Gonzalez, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and K. Keutzer, "A review of single-source deep unsupervised visual domain adaptation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 473–493, 2022.
[44] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.
[45] R. Turrisi, R. Flamary, A. Rakotomamonjy, and M. Pontil, "Multi-source domain adaptation via weighted joint distributions optimal transport," in *Conference on Uncertainty in Artificial Intelligence*, 2022.
[46] S. Chen, "Multi-source domain adaptation with mixture of joint distributions," *Pattern Recognition*, vol. 149, p. 110295, 2024.
[47] H. Wang, M. Xu, B. Ni, and W. Zhang, "Learning to combine: Knowledge aggregation for multi-source domain adaptation," in *European Conference on Computer Vision*, 2020, pp. 727–744.
[48] L. Yang, Y. Balaji, S.-N. Lim, and A. Shrivastava, "Curriculum manager for source selection in multi-source domain adaptation," in *European Conference on Computer Vision*, 2020, pp. 608–624.
[49] R. Li, X. Jia, J. He, S. Chen, and Q. Hu, "T-svdnet: Exploring high-order prototypical correlations for multi-source domain adaptation," in *IEEE International Conference on Computer Vision*, 2021, pp. 9971–9980.
[50] C. Zhou, Z. Wang, X. Zhang, and B. Du, "Domain complementary adaptation by leveraging diversity and discriminability from multiple sources," *IEEE Transactions on Multimedia*, vol. 26, pp. 4490–4501, 2024.
[51] J. Yuan, F. Hou, Y. Yang, Y. Zhang, Z. Shi, X. Geng, J. Fan, Z. He, and Y. Rui, "Domain-aware graph network for bridging multi-source domain adaptation," *IEEE Transactions on Multimedia*, vol. 26, pp. 7210–7224, 2024.
[52] C. Zhou, Z. Wang, B. Du, and Y. Luo, "Cycle self-refinement for multi-source domain adaptation," in *AAAI Conference on Artificial Intelligence*, vol. 38, no. 15, 2024, pp. 17 096–17 104.
[53] X. Yue, Z. Zheng, S. Zhang, Y. Gao, T. Darrell, K. Keutzer, and A. S. Vincentelli, "Prototypical cross-domain self-supervised learning for few-shot unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 834–13 844.
[54] Y. Xiong, H. Chen, Z. Lin, S. Zhao, and G. Ding, "Confidence-based visual dispersal for few-shot unsupervised domain adaptation," in *IEEE International Conference on Computer Vision*, 2023, pp. 11 621–11 631.
[55] X. Yue, Z. Zheng, H. P. Das, K. Keutzer, and A. S. Vincentelli, "Multi-source few-shot domain adaptation," *arXiv preprint arXiv:2109.12391*, 2021.
[56] R. Zhu, X. Yu, and S. Li, "Progressive mix-up for few-shot supervised multi-source domain transfer," in *International Conference on Learning Representations*, 2023, pp. 1–12.
[57] M. I. Belghazi, A. Baratin, S. Rajeshwar, S. Ozair, Y. Bengio, A. Courville, and D. Hjelm, "Mutual information neural estimation," in *International Conference on Machine Learning*, 2018, pp. 531–540.
[58] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, "Deeper, broader and artier domain generalization," in *IEEE International Conference on Computer Vision*, 2017, pp. 5542–5550.
[59] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European Conference on Computer Vision*, 2010, pp. 213–226.
[60] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5385–5394.
[61] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, "Domain adaptive ensemble learning," *IEEE Transactions on Image Processing*, vol. 30, pp. 8008–8018, 2021.
[62] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International Conference on Machine Learning*, 2015, pp. 97–105.
[63] N. Venkat, J. N. Kundu, D. Singh, A. Revanur *et al.*, "Your classifier can secretly suffice multi-source domain adaptation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4647–4659, 2020.
[64] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
[65] S. Chen, L. Wang, Z. Hong, and X. Yang, "Domain generalization by joint-product distribution alignment," *Pattern Recognition*, vol. 134, p. 109086, 2023.
[66] S. Chen and Z. Hong, "Domain generalization by distribution estimation," *International Journal of Machine Learning and Cybernetics*, vol. 14, no. 10, pp. 3457–3470, 2023.
[67] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
[68] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

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 Associate Professor with the Department of Computer Science and Technology, School of Mathematics and Computer Science, Shantou University, Shantou, China. His research interests include statistical machine learning, probabilistic modeling, domain adaptation, and domain generalization.



Ping Xuan received the Ph.D. degree in Computer Science and Technology from Harbin Institute of Technology, Harbin, China, in 2012. She is currently a Researcher and Professor with the Department of Computer Science and Technology, Shantou University, Shantou, China. Her research interests include machine learning, deep learning, graph learning, and medical image processing.

Zhifeng Hao received the B.S. degree in Mathematics from Sun Yat-Sen University in 1990, and the Ph.D. degree in Mathematics from Nanjing University in 1995. He is currently a Professor with the Department of Mathematics, School of Mathematics and Computer Science, Shantou University, Shantou, China. His research interests involve various aspects of algebra, machine learning, data mining, and evolutionary algorithms.