Pattern Recognition 149 (2024) 110295

Contents lists available at ScienceDirect

Pattern Recognition

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

Multi-Source Domain Adaptation with Mixture of Joint Distributions

Sentao Chen

Department of Computer Science, Shantou University, China

ARTICLE INFO

Keywords:
Statistical machine learning
Multi-source domain adaptation
Mixture joint distribution
Kernel method

ABSTRACT

The goal of Multi-Source Domain Adaptation (MSDA) is to train a model (*e.g.*, neural network) with minimal target loss, utilizing training data from multiple source domains (source joint distributions) and a target domain (target joint distribution). The challenge in this problem is that the multiple source joint distributions are different from the target joint distribution. In this paper, we develop a theory that shows a neural network's target loss is upper bounded by both its source mixture loss (*i.e.*, the loss concerning the source mixture joint distribution) and the Pearson χ^2 divergence between the source mixture joint distribution and the target joint distribution. Here, the source mixture joint distribution is the mixture of multiple source joint distributions with mixing weights. Accordingly, we propose an algorithm that optimizes both the mixing weights and the neural network to minimize the estimated source mixture loss and the estimated Pearson χ^2 divergence. To estimate the Pearson χ^2 divergence, we rewrite it as the maximal value of a quadratic functional, exploit a linear-in-parameter function as the functional's input, and solve the resultant optimization problem with an analytic solution. This analytic solution allows us to explicitly express the estimated divergence as a loss of the mixing weights and the network's feature extractor. Finally, we conduct experiments on popular image classification datasets, and the results show that our algorithm statistically outperforms the comparison algorithms. PyTorch code is available at https://github.com/sentaochen/Mixture-of-Joint-Distributions.

1. Introduction

Multi-Source Domain Adaptation (MSDA) is an important problem in pattern recognition and machine learning, and emerges in a wide range of real-world applications, including text classification [1], object recognition [2–4], and medical image segmentation [5]. In MSDA, the training data consist of n ($n \geq 2$) labeled datasets drawn from n source domains (source joint distributions) $p^1(\boldsymbol{x}, y), \ldots, p^n(\boldsymbol{x}, y)$, and an unlabeled dataset from the marginal distribution $p^t(\boldsymbol{x})$ of a target domain (target joint distribution) $p^t(\boldsymbol{x}, y)$, where $\boldsymbol{x}$ denotes the raw features and y represents the class label. In this problem, the challenge is that the source joint distributions are relevant but different from the target joint distribution. The goal of MSDA is to train a model (*e.g.*, neural network) that achieves minimal target loss.

Many studies have been conducted to address the MSDA problem [6,7]. Some works propose to match the source marginal distributions $p^1(\boldsymbol{x}), \ldots, p^n(\boldsymbol{x})$ and the target marginal distribution $p^t(\boldsymbol{x})$ by minimizing the moment distance [2], the discrepancy distance [1], the $\mathcal{H}$-divergence [8], or the mutual information [3]. For example, Wen et al. [1] optimized the domain weights and the networks (including a main network and n auxiliary subnetworks) to minimize the weighted estimated source losses and the weighted estimated discrepancy distances. Park et al. [3] minimized the estimated source combined loss and the estimated mutual information. These works follow the early theoretical works [9,10], factorize $p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y|\boldsymbol{x})$, and presume the class-posterior distribution $p(y|\boldsymbol{x})$ to be a deterministic labeling function. However, as noted by several recent works [11–13], this presumption may not hold for most datasets. Besides, the disparities between the source deterministic labeling functions and the target one are not addressed in these works. Other works propose to align the source class-conditional distributions $p^1(\boldsymbol{x}|y), \ldots, p^n(\boldsymbol{x}|y)$ and the target class-conditional distribution $p^t(\boldsymbol{x}|y)$ by minimizing the Maximum Mean Discrepancy (MMD) [14–16], in addition to aligning the marginal distributions. For example, Li et al. [14] trained the networks to minimize the weighted estimated source losses, the estimated MMDs, and several other loss terms. Although these works have achieved promising empirical results, it is not clear from a theoretical perspective why they can reduce the target loss. Very recently, Chen et al. [13] proposed to match each of the source joint distributions to the target joint distribution with minimal Hellinger distance. However, this work assigns the same weight to the source domains, which is sub-optimal since the sources are not equally important to the target [1].

In this paper, we introduce an innovative solution to the MSDA problem that overcomes the above limitations. We employ a neural network containing a feature extractor and a probabilistic classifier as our model, consider the source joint distributions $p^1(\boldsymbol{x}, y), \ldots, p^n(\boldsymbol{x}, y)$ and the target joint distribution $p^t(\boldsymbol{x}, y)$ without factorizing them, and

E-mail address: sentaochenmail@gmail.com.

https://doi.org/10.1016/j.patcog.2024.110295
Received 13 September 2023; Received in revised form 31 December 2023; Accepted 17 January 2024
Available online 19 January 2024

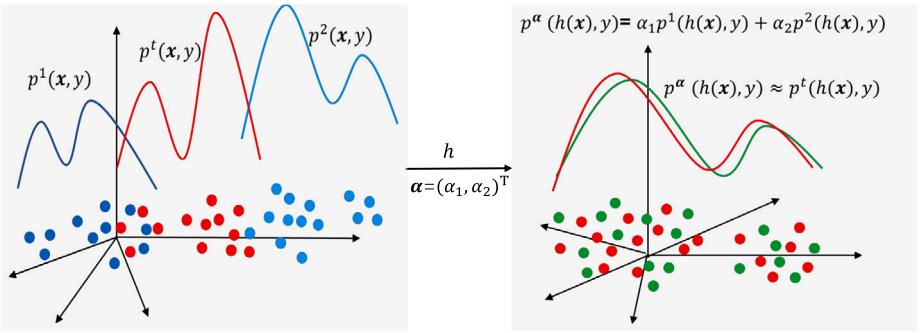


Fig. 1. Our Mixture of Joint Distributions (MJD) for the MSDA problem with two source joint distributions $p^1(\boldsymbol{x}, y)$ and $p^2(\boldsymbol{x}, y)$. MJD optimizes the network's feature extractor h and the mixing weights $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^{\mathrm{T}}$ to align the source mixture joint distribution $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y) = \alpha_1 p^1(\boldsymbol{x}, y) + \alpha_2 p^2(\boldsymbol{x}, y)$ and the target joint distribution $p^t(\boldsymbol{x}, y)$ in the latent feature space.

develop a theoretical loss bound to guide the design of our algorithm. Our theory shows that the network's target loss is upper bounded by both its source mixture loss (*i.e.*, the loss concerning the source mixture joint distribution) and the Pearson χ^2 divergence between the source mixture joint distribution and the target joint distribution. The source mixture joint distribution is defined as the mixture of the n source joint distributions with mixing weights, which are introduced to reflect the importance of each of the sources to the target. Besides, the Pearson χ^2 divergence is used for measuring the disparity between joint distributions. This divergence has the following advantages that drive us to use it instead of other metrics (*e.g.*, $\mathcal{H}$-divergence, MMD, Hellinger distance). (i) Unlike the $\mathcal{H}$-divergence and MMD, it can directly quantify the disparity between joint distributions; (ii) It has shown its advantages in the related problems of domain adaptation [17] and domain generalization [18]. (iii) Unlike the Hellinger distance, it can be estimated in an analytic manner (see Eq. (14)), which is pivotal for learning the feature extractor and the mixing weights in our algorithm.

According to the theoretical loss bound, we propose an algorithm named Mixture of Joint Distributions (MJD). MJD optimizes both the mixing weights and the network (comprising the feature extractor and the probabilistic classifier) to minimize the estimated source mixture loss and the estimated Pearson χ^2 divergence. To estimate the Pearson χ^2 divergence, we first rewrite it as the maximal value of a quadratic functional, then replace the expectations in the functional with sample averages, employ a linear-in-parameter function as the functional's input, and finally solve the resultant quadratic maximization problem via an analytic solution. The analytic solution is pivotal, since it can explicitly express the estimated Pearson χ^2 divergence as a loss of the mixing weights and the network's feature extractor. Note that, here, we adopt the terminology from the book of Schölkopf and Smola [19], and refer to a functional as a function that takes function(s) as its input(s). Finally, we conduct extensive experiments on several popular image classification datasets. From the experimental results, we find that our algorithm statistically outperforms the comparison algorithms. Altogether, our theory, algorithm, and experimental results provide a clear and effective solution to the MSDA problem. For clarity, we illustrate our MJD concept in Fig. 1, and summarize our contributions in the following.

- We develop a theory for the MSDA problem, which shows that a neural network's target loss is constrained by its source mixture loss and the Pearson χ^2 divergence between the source mixture joint distribution and the target joint distribution.
- We propose an MJD algorithm according to the theory, which optimizes the mixing weights and the network to minimize the estimated source mixture loss and the estimated Pearson χ^2 divergence.
- We derive the estimated Pearson χ^2 divergence by solving a quadratic maximization problem with an analytic solution, which allows us to explicitly express the estimated divergence as a loss of the mixing weights and the network's feature extractor.
- We conduct experiments and statistical tests to verify the effectiveness of our algorithm.

2. Related work

We discuss the Domain Adaptation (DA) and Multi-Source Domain Adaptation (MSDA) works, with a focus on the ones that align distributions to address the problems.

2.1. Domain adaptation

DA is intimately connected with MSDA, and trains a model by using data from a source domain and a target domain. Over the last decade, lots of theories [10,12,17,20] and algorithms [21–25] have been proposed for the problem. For example, Ganin et al. [21] matched the source marginal $p^s(\boldsymbol{x})$ and target marginal $p^t(\boldsymbol{x})$ in the latent feature space under the $\mathcal{H}$-divergence. Acuna et al. [17] analyzed the target loss bound using the $\mathrm{D}^{\phi}_{\mathcal{H}}$ discrepancy between marginal distributions, and minimized the estimated source loss and the estimated $\mathrm{D}^{\phi}_{\mathcal{H}}$ discrepancy. Nguyen et al. [12] bounded the target loss by two KL divergences, one between marginals and the other one between class-posteriors, and minimized the estimated source loss and the estimated KL divergence between marginals. Damodaran et al. [23] learned the network embeddings to achieve minimal Wasserstein distance between the source joint distribution and target joint distribution. For a comprehensive review of the DA works, we refer interested readers to the surveys [26–28].

Given the relationship between MSDA and DA, one may consider combining multiple source datasets into a single one to turn the MSDA problem into the DA problem. As a result, the DA algorithms can be employed to address the MSDA problem. Unfortunately, as noted by several works [1,13,29,30], such a naive combination may not be a good solution, since it neglects the differences among the source domains.

2.2. Multi-source domain adaptation

In comparison, MSDA is a less studied problem in theory [9–11] and algorithm [4,6,7,29,31]. Wen et al. [1] proved that the target loss is bounded by the weighted source losses and the weighted discrepancy distances, and optimized the domain weights and the networks to minimize the estimates of the losses and distances. Each discrepancy distance measures the disparity between each source marginal $p^s(\boldsymbol{x})$ $(s = 1, \ldots, n)$ and the target marginal $p^t(\boldsymbol{x})$, and is expressed as the logistic loss of a newly added subnetwork, following previous adversarial algorithms [8,21]. Park et al. [3] optimized the networks to minimize the estimated source combined loss and the estimated mutual information. Similar to the work of Chen et al. [32], the mutual information in [3] also measures the dependency between the latent features and the domain labels. Minimization of mutual information forces the latent features to be independent of the domain labels, and therefore

aligns the source and target marginals simultaneously. Yao et al. [15] quantified the importance of different source domains and aligned the source class-conditionals $p^1(\boldsymbol{x}|y), \ldots, p^n(\boldsymbol{x}|y)$ and target class-conditional $p^t(\boldsymbol{x}|y)$ by minimizing the MMD. Liu et al. [16] integrated multiple source domains into a single domain to match the distribution with the target domain, and matched each source marginal to the target marginal. Chen et al. [13] trained the neural network to minimize the estimates of the source loss and the Hellinger distance between source and target joint distributions.

Our work is different from the above ones in several key aspects. (i) Our model is a simple neural network with a feature extractor and a probabilistic classifier, which contains fewer parameters than the models with multiple networks in most works (*e.g.*, [1,8,16]). (ii) Our theory and algorithm consider the single source mixture joint distribution, rather than the multiple source marginals (*e.g.*, [4,8]), the multiple source class-conditionals (*e.g.*, [14,15]), or the multiple source joint distributions (*e.g.*, [13]). (iii) Our distribution disparity metric is the Pearson χ^2 divergence, rather than the $\mathcal{H}$-divergence (*e.g.*, [8]), the MMD (*e.g.*, [15]), or the Hellinger distance (*e.g.*, [13]). As aforementioned in the introduction section, the Pearson χ^2 divergence is more suitable for our work since it can directly measure the disparity between joint distributions. More importantly, its estimate can be explicitly expressed as a loss of the network's feature extractor and the mixing weights (see Eq. (14)), which is crucial to our algorithm. (iv) Our mixing weights are introduced to reflect the importance of the multiple source joint distributions to the target joint distribution, rather than the importance of the multiple source marginals to the target marginal (*e.g.*, [1,8]). Notably, since our algorithm aligns the source mixture joint distribution to the target joint distribution, the sources similar to the target will be assigned large mixing weights and the dissimilar ones will be assigned small weights.

Of course, there are other works worth mentioning, including but not limited to [30,33–36]. For example, Wang et al. [37] aggregated the knowledge learned from multiple source domains to boost the prediction for query samples, and performed the class-relation-aware domain alignment. Venkat et al. [35] exploited the pseudo-labeled target samples, enforced a classifier agreement on the pseudo-labels, and determined the training convergence by the classifier agreement. Nguyen et al. [36] constructed a multi-source teacher expert network, reduced the gap between the source and target domains in the latent feature space, and trained a student network to imitate the predictions of the teacher expert network. For completeness, in Section 5 we experimentally compare our MJD algorithm against some of these algorithms.

3. Theory

We define the MSDA problem and develop a theoretical loss bound for the problem.

3.1. Problem definition

Following prior MSDA works [11,13], we define a domain as a joint distribution $p(\boldsymbol{x}, y)$. The marginal distribution of a domain is written as $p(\boldsymbol{x}) = \int p(\boldsymbol{x}, y)dy$. We use $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m \sim p(\boldsymbol{x}, y)$ to denote that the samples in D are independently drawn from joint distribution $p(\boldsymbol{x}, y)$. We use $\mathcal{F}$ to denote the classification model space, and L to denote the loss function. Using these notations, we define the MSDA problem as follows.

Definition 1 (*MSDA*)**.** Let $p^1(\boldsymbol{x}, y), \ldots, p^n(\boldsymbol{x}, y)$ be n ($n \geq 2$) source domains (source joint distributions) and $p^t(\boldsymbol{x}, y)$ be a target domain (target joint distribution). The source joint distributions are relevant but different from the target joint distribution. Given the training data that contain n labeled datasets $D^1 = \{(\boldsymbol{x}_i^1, y_i^1)\}_{i=1}^{m_1} \sim p^1(\boldsymbol{x}, y)$, …, $D^n = \{(\boldsymbol{x}_i^n, y_i^n)\}_{i=1}^{m_n} \sim p^n(\boldsymbol{x}, y)$ and an unlabeled dataset $D^u = \{\boldsymbol{x}_i^t\}_{i=1}^{m_t} \sim p^t(\boldsymbol{x}) = \int p^t(\boldsymbol{x}, y)dy$, the goal of MSDA is to train a classification model $f \in \mathcal{F}$, such that its target loss $\mathbb{E}_{p^t(\boldsymbol{x},y)}[L(f(\boldsymbol{x}), y)]$ is minimized. In other words, the trained model is expected to have a good performance in the target domain and well predict the target labels.

3.2. Loss bound

Before presenting the loss bound, we first elaborate on the source mixture joint distribution, the Pearson χ^2 divergence, and the neural network model.

We introduce mixing weights for the source joint distributions and define the source mixture joint distribution as

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

where the mixing weights $\alpha_1, \ldots, \alpha_n \geq 0$ and $\sum_{s=1}^n \alpha_s = 1$. For convenience, we use a mixing weight vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\top \in \Delta = \{\boldsymbol{\alpha} | \alpha_s \geq 0, \sum_{s=1}^n \alpha_s = 1\}$ to collect the n weights. We exploit the Pearson χ^2 divergence to measure the disparity between the source mixture joint distribution $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y)$ and the target joint distribution $p^t(\boldsymbol{x}, y)$. The Pearson χ^2 divergence is defined as

$$\mathrm{D}_{\chi^2}(p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y) \parallel p^t(\boldsymbol{x}, y)) = \int \left[\left(\frac{p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y)}{p^t(\boldsymbol{x}, y)} \right)^2 - 1 \right] p^t(\boldsymbol{x}, y) d\boldsymbol{x} dy. \tag{2}$$

It compares the two joint distributions based on their ratio $\frac{p^{\boldsymbol{\alpha}}(\boldsymbol{x},y)}{p^t(\boldsymbol{x},y)}$. It is non-negative and reaches zero when $p^{\boldsymbol{\alpha}}(\boldsymbol{x}, y) = p^t(\boldsymbol{x}, y)$. We use a neural network as the classification model f, which contains a feature extractor h and a probabilistic classifier g, *i.e.*, $f = g \circ h$. The feature extractor h takes $\boldsymbol{x}$ in the raw feature space and transforms it into $h(\boldsymbol{x})$ in the latent feature space. Then, the probabilistic classifier g takes $h(\boldsymbol{x})$ in the latent feature space and transforms it into $g(h(\boldsymbol{x}))$ in the output space.

Below, we theoretically show that, the network's target loss is upper bounded by both its source mixture loss and the Pearson χ^2 divergence between the source mixture joint distribution and the target joint distribution.

Theorem 1 (*Loss Bound*)**.** *Assume that the loss L is upper bounded by some $M > 0$ and that the Pearson χ^2 divergence $\mathrm{D}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \parallel p^t(h(\boldsymbol{x}), y))$ is lower bounded by some $\epsilon > 0$. Then, for any network classification model $f \in \mathcal{F}$,*

$$\begin{aligned} &\mathbb{E}_{p^t(\boldsymbol{x},y)}[L(f(\boldsymbol{x}), y)] \\ &\leq \mathbb{E}_{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})), y)] + \frac{2M}{\sqrt{\epsilon}} \mathrm{D}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \parallel p^t(h(\boldsymbol{x}), y)) \qquad (3) \\ &= \sum_{s=1}^n \alpha_s \mathbb{E}_{p^s(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})), y)] + \frac{2M}{\sqrt{\epsilon}} \mathrm{D}_{\chi^2}(\sum_{s=1}^n \alpha_s p^s(h(\boldsymbol{x}), y) \parallel p^t(h(\boldsymbol{x}), y)). \qquad (4) \end{aligned}$$

We present the proof in Appendix. According to Theorem 1, an algorithm aiming to train a network model with minimal target loss should optimize both the network $f = g \circ h$ and the mixing weights $\alpha_1, \ldots, \alpha_n$ to minimize two losses: (i) the source mixture loss $\sum_{s=1}^n \alpha_s \mathbb{E}_{p^s(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})), y)]$, and (ii) the Pearson χ^2 divergence $\mathrm{D}_{\chi^2}(\sum_{s=1}^n \alpha_s p^s(h(\boldsymbol{x}), y) \parallel p^t(h(\boldsymbol{x}), y))$. Since minimizing the Pearson χ^2 divergence matches the source mixture and target joint distributions, the sources similar to the target will be assigned large mixing weights and the dissimilar ones will receive small weights.

Remark 1. To derive the loss bound, it is common to assume that the loss L is upper bounded [9,11,12,17]. Moreover, since in real-world problems the source mixture joint distribution may not be perfectly matched to the target joint distribution, it is reasonable to assume that the Pearson χ^2 divergence is lower bounded by a positive constant.

Remark 2. Our target loss bound is different from those presented in prior works [1,2,8]. Specifically, these works focus on binary classification, factorize joint distribution $p(\boldsymbol{x}, y) = p(\boldsymbol{x})p(y|\boldsymbol{x})$, and assume the class-posterior distribution $p(y|\boldsymbol{x})$ to be a deterministic labeling function. Subsequently, they propose to bound the target loss via the divergence or distance between the source and target marginal distributions. Unfortunately, most classification problems like image classification [38,39] and document classification [15,32] are multi-class problems. Besides, the assumption that the class-posterior distribution is a deterministic labeling function may not hold for most datasets [11–13]. Although a recent work [13] manages to overcome these limitations, it assigns the same weight to the source joint distributions, which is sub-optimal since the source joint distributions differ in similarity to the target one. By contrast, our theoretical analysis applies to multi-class classification, bounds the target loss by the divergence between the source mixture and target joint distributions, and assigns different weights to the source joint distributions.

4. Algorithm

We propose the Mixture of Joint Distributions (MJD) algorithm according to the theory in Theorem 1. As aforementioned, the MJD algorithm should optimize both the network and the mixing weights to minimize the source mixture loss and the Pearson χ^2 divergence. Since these two loss terms are unknown in practice, the algorithm operates by minimizing their empirical estimates. In particular, the estimated source mixture loss is expressed as $\sum_{s=1}^{n} \frac{\alpha_s}{m_s} \sum_{i=1}^{m_s} L(g(h(\boldsymbol{x}_i^s)), y_i^s)$, and the estimated Pearson χ^2 divergence is derived in detail in the following subsection. After that, we present the optimization problem of our algorithm, which seeks to jointly minimize the estimated source mixture loss and the estimated Pearson χ^2 divergence.

4.1. Estimated divergence

We derive the estimated Pearson χ^2 divergence between the source mixture joint distribution and the target joint distribution. To this end, we rewrite the Pearson χ^2 divergence as

$$\begin{aligned}
&\mathrm{D}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \,\|\, p^t(h(\boldsymbol{x}), y)) \\
&= \int \left[\left(\frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)}{p^t(h(\boldsymbol{x}), y)} \right)^2 - 1 \right] p^t(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y && (5) \\
&= \int \max_r \left(2 \frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)}{p^t(h(\boldsymbol{x}), y)} r(h(\boldsymbol{x}), y) - r(h(\boldsymbol{x}), y)^2 - 1 \right) p^t(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y && (6) \\
&= \max_r \int \left(2 \frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y)}{p^t(h(\boldsymbol{x}), y)} r(h(\boldsymbol{x}), y) - r(h(\boldsymbol{x}), y)^2 - 1 \right) p^t(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y && (7) \\
&= \max_r \Big(2 \sum_{s=1}^{n} \alpha_s \int p^s(h(\boldsymbol{x}), y) r(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y \\
&\qquad - \int r(h(\boldsymbol{x}), y)^2 p^t(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y \Big) - 1. && (8)
\end{aligned}$$

Eq. (6) uses the equation $u^2 - 1 = \max_v (2uv - v^2 - 1)$, where the ratio function $\frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}{p^t(h(\boldsymbol{x}),y)}$ is regarded as u and the function $r(h(\boldsymbol{x}), y)$ is regarded as v. In Eq. (7), the objective function is a quadratic functional that takes the function $r(h(\boldsymbol{x}), y)$ as its input. The maximal value of the quadratic functional is reached at the function $r(h(\boldsymbol{x}), y) = \frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}{p^t(h(\boldsymbol{x}),y)}$, and the maximal value is Eq. (5). Eq. (8) expands the source mixture joint distribution as $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) = \sum_{s=1}^{n} \alpha_s p^s(h(\boldsymbol{x}), y)$.

Since the joint distributions are linear in the integrals in Eq. (8), according to the knowledge of probability and statistics, we can use sample averages to replace the integrals and approximate the Pearson χ^2 divergence as

$$\begin{aligned}
&\mathrm{D}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \,\|\, p^t(h(\boldsymbol{x}), y)) \\
&\approx \max_r \Big(2 \sum_{s=1}^{n} \frac{\alpha_s}{m_s} \sum_{i=1}^{m_s} r(h(\boldsymbol{x}_i^s), y_i^s) - \frac{1}{m_t} \sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t), y_i^t)^2 \Big) - 1. && (9)
\end{aligned}$$

Eq. (9) makes use of the labeled source datasets $\mathcal{D}^1 = \{(\boldsymbol{x}_i^1, y_i^1)\}_{i=1}^{m_1}, \ldots, \mathcal{D}^n = \{(\boldsymbol{x}_i^n, y_i^n)\}_{i=1}^{m_n}$ generated by the source joint distributions $p^1(\boldsymbol{x}, y), \ldots, p^n(\boldsymbol{x}, y)$ (see Definition 1), and the labeled target dataset $\mathcal{D}^t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{m_t}$ generated by the target joint distribution $p^t(\boldsymbol{x}, y)$. We assume that $\mathcal{D}^t$ is available in this subsection such that we can focus on deriving the estimated divergence. In the next subsection, we discuss how it is obtained from the unlabeled target dataset.

We then design function $r(h(\boldsymbol{x}), y)$ as the following linear-in-parameter function

$$r(h(\boldsymbol{x}), y; \boldsymbol{\theta}) = \sum_{i=1}^{m} \theta_i k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i)) k_y(y, y_i), \tag{10}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)^\top$ are the parameters, $k_x(h(\boldsymbol{x}), h(\boldsymbol{x}_i)) = \exp(- \| h(\boldsymbol{x}) - h(\boldsymbol{x}_i) \|^2 / \sigma)$ is the Gaussian kernel with kernel width σ (>0),[1] and $k_y(y, y_i)$ is the delta kernel that evaluates 1 if $y = y_i$ and 0 otherwise. In addition, $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)\} = \{(\boldsymbol{x}_1^1, y_1^1), \ldots, (\boldsymbol{x}_{m_t}^t, y_{m_t}^t)\} = \mathcal{D}^1 \cup \cdots \cup \mathcal{D}^n \cup \mathcal{D}^t$ are the $m = m_1 + \cdots + m_n + m_t$ function centers. Note that, the function $r(h(\boldsymbol{x}), y)$ is designed as the model in Eq. (10), *i.e.*, a linear combination of kernel functions, since such a model is flexible for function approximation and has been demonstrated to be effective in approximating various functions, *e.g.*, the ratio function $\frac{p^s(\boldsymbol{x},y)}{p^t(\boldsymbol{x},y)}$, the difference function $p^s(\boldsymbol{x}, y) - p^t(\boldsymbol{x}, y)$ [32,40,41]. Importantly, such a model (linear in its parameters) enables us to analytically solve the maximization problem in Eq. (9).

Plugging the model in Eq. (10) into Eq. (9), we obtain the estimated Pearson χ^2 divergence as

$$\begin{aligned}
&\widehat{\mathrm{D}}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}), y) \,\|\, p^t(h(\boldsymbol{x}), y)) \\
&= \max_{\boldsymbol{\theta}} \Big(2 \sum_{s=1}^{n} \frac{\alpha_s}{m_s} \sum_{i=1}^{m_s} r(h(\boldsymbol{x}_i^s), y_i^s; \boldsymbol{\theta}) - \frac{1}{m_t} \sum_{i=1}^{m_t} r(h(\boldsymbol{x}_i^t), y_i^t; \boldsymbol{\theta})^2 \Big) - 1 && (11) \\
&= \max_{\boldsymbol{\theta}} \Big(2 \sum_{s=1}^{n} \alpha_s [\frac{1}{m_s} \mathbf{1}^\top \boldsymbol{K}^s] \boldsymbol{\theta} - \boldsymbol{\theta}^\top [\frac{1}{m_t} (\boldsymbol{K}^t)^\top \boldsymbol{K}^t] \boldsymbol{\theta} \Big) - 1 && (12) \\
&= \max_{\boldsymbol{\theta}} \Big(2 \big(\sum_{s=1}^{n} \alpha_s \boldsymbol{b}^s \big)^\top \boldsymbol{\theta} - \boldsymbol{\theta}^\top \boldsymbol{H} \boldsymbol{\theta} \Big) - 1 && (13) \\
&= 2 \big(\sum_{s=1}^{n} \alpha_s \boldsymbol{b}^s \big)^\top \widehat{\boldsymbol{\theta}} - \widehat{\boldsymbol{\theta}}^\top \boldsymbol{H} \widehat{\boldsymbol{\theta}} - 1. && (14)
\end{aligned}$$

Eq. (12) introduces several notations, where $\mathbf{1}$ is an m_s-dimensional column vector of ones, and $\boldsymbol{K}^s \in \mathbb{R}^{m_s \times m}$, $\boldsymbol{K}^t \in \mathbb{R}^{m_t \times m}$ are two matrices. The (i, j)th elements of $\boldsymbol{K}^s$ and $\boldsymbol{K}^t$ are defined as $k_x(h(\boldsymbol{x}_i^s), h(\boldsymbol{x}_j)) k_y(y_i^s, y_j)$ and $k_x(h(\boldsymbol{x}_i^t), h(\boldsymbol{x}_j)) k_y(y_i^t, y_j)$. Eq. (13) uses notations

$$\boldsymbol{b}^s = \frac{1}{m_s} (\boldsymbol{K}^s)^\top \mathbf{1} \tag{15}$$

$$\boldsymbol{H} = \frac{1}{m_t} (\boldsymbol{K}^t)^\top \boldsymbol{K}^t \tag{16}$$

to highlight the quadratic maximization problem. Eq. (14) solves the problem with the analytic solution

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{H} + \epsilon \mathbf{I})^{-1} \big(\sum_{s=1}^{n} \alpha_s \boldsymbol{b}^s \big), \tag{17}$$

where $\epsilon \mathbf{I}$ is a diagonal matrix added to $\boldsymbol{H}$ to ensure that the inversion of the matrix is always feasible in practice. Here, ϵ is a small positive value[2] and $\mathbf{I}$ is the identity matrix. For clarity, we summarize the procedure for calculating the estimated Pearson χ^2 divergence in Algorithm 1. Obviously, the divergence is dependent on both the mixing weights $\alpha_1, \ldots, \alpha_n$ and the feature extractor h, since matrix $\boldsymbol{H}$ and vectors $\boldsymbol{b}^1, \ldots, \boldsymbol{b}^n, \widehat{\boldsymbol{\theta}}$ are dependent on h.

[1] In the experiments, σ is set to the median pairwise squared distances on the unlabeled data following the practice in [18].

[2] In the experiments, ϵ is set to 10^{-3}.

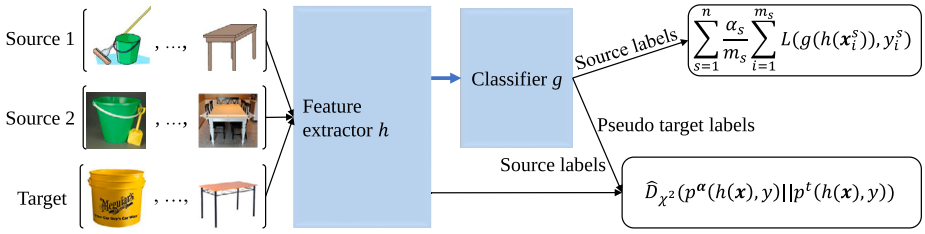


Fig. 2. Illustration of the input data (*e.g.*, images from dataset Office-Home [39]), the network model, and the loss terms in our Mixture of Joint Distributions (MJD) algorithm.

Algorithm 1 Estimated Divergence Algorithm

Input: Datasets $\mathcal{D}^1,\cdots,\mathcal{D}^n,\mathcal{D}^t$.
Output: Divergence $\widehat{\mathrm{D}}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))$.
1: Construct matrices $\boldsymbol{K}^s(s=1,\cdots,n)$, $\boldsymbol{K}^t$, where their (i,j)-th elements are $k_x(h(\boldsymbol{x}_i^s),h(\boldsymbol{x}_j))k_y(y_i^s,y_j)$, $k_x(h(\boldsymbol{x}_i^t),h(\boldsymbol{x}_j))k_y(y_i^t,y_j)$.
2: Compute vectors $\boldsymbol{b}^s(s=1,\cdots,n)$, matrix $\boldsymbol{H}$, and vector $\widehat{\boldsymbol{\theta}}$ via Eqs. (15)-(17).
3: Obtain $\widehat{\mathrm{D}}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\|p^t(h(\boldsymbol{x}),y))$ via Eq. (14).

Remark 3. Since the Pearson χ^2 divergence is asymmetric, we explain the reason why we opt for the current direction $\mathrm{D}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\ \|\ p^t(h(\boldsymbol{x}),y))$, rather than the other direction $\mathrm{D}_{\chi^2}(p^t(h(\boldsymbol{x}),y)\ \|\ p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y))$. In Eq. (10), the function $r(h(\boldsymbol{x}),y;\boldsymbol{\theta})$ aims to approximate the ratio function $\frac{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}{p^t(h(\boldsymbol{x}),y)}$. If we use the other direction, $r(h(\boldsymbol{x}),y;\boldsymbol{\theta})$ will instead approximate the other function $\frac{p^t(h(\boldsymbol{x}),y)}{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}$. However, for certain values of $\alpha_1,\ldots,\alpha_n$, the denominator $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)$ may be very small and consequently $\frac{p^t(h(\boldsymbol{x}),y)}{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}$ may go to infinity, which makes the approximation difficult. For this reason, we prefer the current direction.

4.2. MJD algorithm

We present the optimization problem of our Mixture of Joint Distributions (MJD) algorithm as follows

$$\min_{g,h}\min_{\boldsymbol{\alpha}\in\Delta}\sum_{s=1}^{n}\frac{\alpha_s}{m_s}\sum_{i=1}^{m_s}L(g(h(\boldsymbol{x}_i^s)),y_i^s)+\lambda\widehat{\mathrm{D}}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\ \|\ p^t(h(\boldsymbol{x}),y)),\tag{18}$$

where $\boldsymbol{\alpha}=(\alpha_1,\ldots,\alpha_n)^\top\in\Delta=\{\boldsymbol{\alpha}|\alpha_s\geq 0,\sum_{s=1}^n\alpha_s=1\}$ is the mixing weight vector, L is the cross-entropy loss, and λ is a tradeoff parameter. Recall that the estimated Pearson χ^2 divergence involves the labeled target dataset $\mathcal{D}^t$. In the MSDA problem, we construct it from the unlabeled target dataset $\mathcal{D}^u=\{\boldsymbol{x}_i^t\}_{i=1}^{m_t}$ by utilizing the popular pseudo labeling technique [14,16,25]. Specifically, we assign pseudo labels to $\mathcal{D}^u$ and set the labeled target dataset $\mathcal{D}^t=\{(\boldsymbol{x}_i^t,y_i^t)\}_{i=1}^{m_t}$, where y_i^t is the pseudo label produced by the source trained network. Clearly, our MJD algorithm optimizes the network $f=g\circ h$ and the mixing weight vector $\boldsymbol{\alpha}$ to jointly minimize the estimated source mixture loss and the estimated Pearson χ^2 divergence. For clarity, we illustrate in Fig. 2 the input data (*e.g.*, images from Office-Home [39]), the network model, and the loss terms in our algorithm. By minimizing the loss $\widehat{\mathrm{D}}_{\chi^2}(p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\ \|\ p^t(h(\boldsymbol{x}),y))$ in (18), the source mixture joint distribution and the target joint distribution are aligned such that $p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)\approx p^t(h(\boldsymbol{x}),y)$. As a result, the following relationship holds

$$\sum_{s=1}^{n}\frac{\alpha_s}{m_s}\sum_{i=1}^{m_s}L(g(h(\boldsymbol{x}_i^s)),y_i^s)\approx\mathbb{E}_{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})),y)]\tag{19}$$

$$\approx\mathbb{E}_{p^t(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})),y)]\tag{20}$$

$$=\mathbb{E}_{p^t(\boldsymbol{x},y)}[L(f(\boldsymbol{x}),y)].\tag{21}$$

This implies that by minimizing the other loss $\sum_{s=1}^n\frac{\alpha_s}{m_s}\sum_{i=1}^{m_s}L(g(h(\boldsymbol{x}_i^s)),y_i^s)$ in (18), the target loss $\mathbb{E}_{p^t(\boldsymbol{x},y)}[L(f(\boldsymbol{x}),y)]$ of the network $f=g\circ h$ will become small and the goal of MSDA can be achieved (see Definition 1).

We solve minimization problem (18) by first solving the inner minimization given g,h and then the outer minimization given $\boldsymbol{\alpha}$, which is inspired by the work of Wen et al. [1]. For the inner minimization with respect to $\boldsymbol{\alpha}$, we use the softmax function to absorb the constraint Δ, *i.e.*, $\alpha_s\leftarrow\frac{\exp(\alpha_s)}{\sum_{i=1}^n\exp(\alpha_i)}$, and solve the resultant unconstrained problem via the Limited memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [42]. This algorithm is mature and available in many optimization packages (*e.g.*, pytorch-minimize[3]). For the outer minimization with respect to g,h, we use the minibatch SGD algorithm. In every SGD iteration, $n+1$ minibatches are drawn from the $n+1$ source and target datasets. The objective function in (18) is evaluated on these minibatches. We also update and refine the pseudo target labels throughout the iterations. This is due to the concern that in the early stage, the source mixture joint distribution and the target joint distribution are not aligned in the latent feature space. Consequently, the source trained network may not be able to produce accurate pseudo target labels. When the network is better trained during the iterations, the alignment between joint distributions enhances, and the pseudo target labels will be better updated and become more accurate. Later in Section 5.5, we perform experiments to verify this point. For clarity, we summarize the above procedure in Algorithm 2.

5. Experiments

We exploit Pytorch to implement our algorithm, and contrast its performance against other algorithms on several image classification datasets, which are popular in the field [34,35,43].

5.1. Experimental datasets

PACS [38] contains 7 classes and 4 domains: ArtPainting, Cartoon, Photo, and Sketch. The numbers of images in the 4 domains are 2048, 2344, 1670, and 3929, respectively. **Office-Home** [39] contains 65 classes and 4 domains: Art, Clipart, Product, and RealWorld. The numbers of images in the 4 domains are 2421, 4379, 4428, and 4357, respectively. **miniDomainNet** [43] contains 126 classes and 4 domains: Clipart, Painting, Real, and Sketch. The numbers of images in the 4 domains are 18 703, 31 202, 65 609, and 24 492, respectively. **DomainNet** [2] is proposed for large scale MSDA experiments. It has around 0.6 million images and contains 345 classes and 6 domains: Clipart, Infograph, Painting, Quickdraw, Real, and Sketch.

5.2. Experimental setup

Comparison algorithms. We compare our MJD algorithm against the Baseline, DA, and MSDA algorithms. (i) The Baseline algorithm trains the neural network on the combined source dataset without any domain adaptation operation. (ii) The DA algorithms encompass DANN [21] and f-DAL [17]. These algorithms are representative adversarial methods that align distributions via solving minimax problems. They also train the networks on the combined source dataset. (iii) The

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

Algorithm 2 Mixture of Joint Distributions Algorithm

Input: Labeled source datasets $\mathcal{D}^1, \cdots, \mathcal{D}^n$, unlabeled target dataset $\mathcal{D}^u$.
Output: Network $f = g \circ h$.
1: Train the network on $\mathcal{D}^1, \cdots, \mathcal{D}^n$.
2: Use the network to label $\mathcal{D}^u$ and obtain pseudo labeled target dataset $\mathcal{D}^t$.
3: **while** training does not end **do**
4: **for** k in $1 : K$ **do**
5: Sample minibatches $\mathcal{D}_k^1, \cdots, \mathcal{D}_k^n, \mathcal{D}_k^t$ from datasets $\mathcal{D}^1, \cdots, \mathcal{D}^n, \mathcal{D}^t$.
6: Run the L-BFGS algorithm [42] to return the optimal mixing weight vector $\boldsymbol{\alpha}$ in objective function (18), which is calculated on $\mathcal{D}_k^1, \cdots, \mathcal{D}_k^n, \mathcal{D}_k^t$.
7: Take a gradient step to update the parameters of the feature extractor h and probabilistic classifier g.
8: **end for**
9: Update the pseudo labels in $\mathcal{D}^t$ by using the network.
10: **end while**

Table 1
Mean and standard deviation (std) of classification accuracy (%) with ResNet50 model on PACS (7 classes).

Algorithm	→ArtPainting	→Cartoon	→Photo	→Sketch	Avg
Baseline	83.50 (0.46)	76.88 (0.73)	98.20 (0.92)	70.84 (0.80)	82.35
DANN [21]	87.16 (0.92)	84.55 (0.80)	97.18 (0.64)	81.68 (1.21)	87.64
f-DAL [17]	90.82 (0.42)	89.77 (0.65)	98.74 (0.56)	70.69 (0.87)	87.50
MDAN [8]	85.16 (0.70)	82.17 (0.67)	98.38 (0.88)	73.68 (1.05)	84.84
M3SDA [2]	90.82 (0.76)	85.57 (0.62)	98.50 (0.73)	85.36 (0.90)	90.06
DARN [1]	88.19 (0.81)	85.22 (0.92)	97.95 (0.50)	80.83 (0.86)	88.05
MIAN [3]	86.72 (0.57)	84.25 (0.83)	98.80 (0.34)	77.30 (0.66)	86.76
WJDOT [44]	87.35 (0.67)	84.77 (0.62)	98.05 (0.45)	76.48 (0.80)	86.66
RRL [13]	92.48 (0.46)	84.34 (0.70)	98.98 (0.58)	85.35 (0.82)	90.29
LTC [37]	88.38 (0.53)	86.38 (0.84)	98.52 (0.38)	**86.42 (0.59)**	89.93
SImpAl [35]	85.38 (0.87)	82.56 (0.75)	98.50 (0.63)	74.28 (0.50)	85.18
MJD (ours)	**93.90 (0.65)**	**92.32 (0.62)**	**99.34 (0.51)**	86.05 (0.72)	**92.90**

MSDA algorithms include MDAN [8], M3SDA [2], DARN [1], MIAN [3], WJDOT [44], RRL [13], LTC [37], and SImpAl [35]. Among these algorithms, the first six ones address MSDA by aligning the marginal distributions or joint distributions under various metrics, while the last two ones tackle the problem through knowledge aggregation or implicit alignment.

Evaluation protocol. We construct the MSDA task by following prior works [13,37]. That is, for a dataset (*e.g.*, PACS) with several domains, we select one domain as the target domain and the others as the source domains. Such a task is denoted as "→ArtPainting", where ArtPainting indicates the selected target domain. We measure the performance of an algorithm by its target classification accuracy (%). On each task, since different random seeds may lead to slightly different classification accuracy, we repeat the experiment 5 times to report the mean and standard deviation (std) of the accuracy.

Implementation details. On PACS, Office-Home, and miniDomainNet, we implement the network model using both ResNet50 and ResNet101 [45]. On the very large DomainNet, we implement the model using ResNet101. Note that, these networks are pretrained on ImageNet and are common choices in MSDA for image classification [2,3,13,14,35]. On each dataset, we use the pretrained feature extractor of the network and append a probabilistic classifier to it. We optimize the mixing weight vector and the network (comprising the feature extractor and the probabilistic classifier) respectively by the L-BFGS algorithm [42] and the minibatch SGD algorithm (again see Algorithm 2). In each SGD iteration, $n+1$ minibatches of identical size are sampled from the $n+1$ source and target datasets. The learning rates for the feature extractor and the probabilistic classifier are set to 0.001 and 0.01, respectively. Moreover, we follow the strategy in Ganin et al. [21] and gradually change the tradeoff parameter λ using the formula $\lambda_p = \frac{2}{1+e^{-10p}} - 1$, where p is the training progress linearly changing from 0 to 1. This strategy has been empirically verified to be effective in quite a few works [13,14,16,18].

5.3. Experimental results

We report in Tables 1–6 the classification results (ResNet50 and ResNet101) on PACS, Office-Home, and miniDomainNet, and in Table 7 the classification results (ResNet101) on DomainNet. In the last column of each table, the average classification accuracy (Avg) over all the tasks is computed. For every column in the table, the best result is highlighted in **bold**. Note that, here, we cite some comparison algorithms' results from [3,13,35,37], since our experimental setup aligns with these works. For the comparison algorithms that do not report results on a certain dataset, we run them using their released source codes, and report their best classification results on that dataset. Given the complete results of the comparison algorithms on all the datasets, we can compare our algorithm against them comprehensively.

From Tables 1 to 7, our MJD algorithm outperforms the Baseline, DA, and other MSDA algorithms by a significant margin on most of the MSDA tasks. For instance, in Table 1, our algorithm surpasses the Baseline algorithm by more than 10% (92.90% versus 82.35%), and the second best algorithm RRL by more than 2% (92.90% versus 90.29%) in terms of average classification accuracy. The results indicate that on the tested datasets with a small and large number of classes (7, 65, 126, and 345), our algorithm with the deep ResNet50 and deeper ResNet101 models, is superior to the comparison algorithms. Although some algorithms including M3SDA, DARN, RRL, and LTC, have also achieved good classification results, these algorithms fail to tackle the joint distribution disparity problem in MSDA, or overlook the different similarities of the multiple source domains to the target domain. We conjecture that to a certain extent, these limitations have resulted in the less superior performance of the comparison algorithms. To summarize, the experimental results testify that our MJD algorithm, which aligns the source mixture joint distribution and the target joint distribution, is more advantageous than the comparison algorithms, which rely on marginal distribution alignment [1,3,8], knowledge aggregation [37], or implicit alignment [35].

Table 2
Mean and std of classification accuracy (%) with ResNet101 model on PACS (7 classes).

Algorithm	→ArtPainting	→Cartoon	→Photo	→Sketch	Avg
Baseline	86.52 (0.67)	79.05 (0.48)	98.56 (0.73)	72.56 (0.89)	84.18
DANN [21]	90.77 (0.70)	85.53 (0.96)	98.02 (1.32)	82.60 (0.56)	89.23
f-DAL [17]	88.67 (0.45)	91.22 (0.70)	99.16 (0.42)	82.53 (0.72)	90.40
MDAN [8]	88.01 (0.33)	83.43 (0.68)	98.43 (0.90)	83.90 (0.98)	88.44
M3SDA [2]	94.17 (0.73)	89.74 (0.52)	99.28 (0.52)	88.70 (0.60)	92.97
DARN [1]	91.27 (0.96)	85.78 (0.50)	98.73 (0.89)	82.13 (0.74)	89.48
MIAN [3]	90.54 (0.82)	85.39 (0.78)	99.21 (0.12)	79.41 (0.45)	88.64
WJDOT [44]	89.28 (0.21)	87.41 (0.56)	98.82 (0.60)	79.66 (0.77)	88.54
RRL [13]	95.02 (0.34)	87.93 (0.52)	99.16 (0.08)	85.09 (0.74)	91.80
LTC [37]	92.37 (0.50)	87.57 (0.93)	99.34 (0.24)	87.27 (0.60)	91.64
SImpAl [35]	87.84 (1.24)	85.30 (0.62)	**99.45 (0.38)**	78.48 (0.77)	87.77
MJD (ours)	**95.65 (0.56)**	**92.58 (0.41)**	99.40 (0.20)	**89.64 (0.66)**	**94.32**

Table 3
Mean and std of classification accuracy (%) with ResNet50 model on Office-Home (65 classes).

Algorithm	→Art	→Clipart	→Product	→RealWorld	Avg
Baseline	67.40 (0.39)	52.38 (0.38)	77.95 (0.28)	80.70 (0.17)	69.61
DANN [21]	69.19 (0.58)	57.08 (0.19)	79.18 (0.70)	82.63 (0.40)	72.02
f-DAL [17]	72.30 (0.47)	64.26 (0.39)	83.15 (0.41)	82.81 (0.30)	75.63
MDAN [8]	69.57 (0.36)	55.22 (0.46)	79.71 (0.29)	81.48 (0.19)	71.49
M3SDA [2]	66.22 (0.52)	58.55 (0.62)	79.45 (0.52)	81.35 (0.19)	71.39
DARN [1]	68.93 (0.44)	55.64 (0.61)	79.72 (0.27)	82.45 (0.31)	71.68
MIAN [3]	69.88 (0.35)	64.20 (0.68)	80.87 (0.37)	81.49 (0.24)	74.11
WJDOT [44]	69.46 (0.44)	59.65 (0.35)	81.14 (0.70)	82.61 (0.61)	73.22
RRL [13]	75.02 (0.18)	68.93 (0.42)	85.61 (0.19)	84.05 (0.36)	78.40
LTC [37]	70.44 (0.21)	64.26 (0.43)	82.19 (0.55)	81.75 (0.43)	74.66
SImpAl [35]	70.80 (0.20)	56.30 (0.20)	80.20 (0.30)	81.50 (0.30)	72.20
MJD (ours)	**75.88 (0.31)**	**71.25 (0.35)**	**87.31 (0.23)**	**85.58 (0.32)**	**80.01**

Table 4
Mean and std of classification accuracy (%) with ResNet101 model on Office-Home (65 classes).

Algorithm	→Art	→Clipart	→Product	→RealWorld	Avg
Baseline	72.48 (0.41)	58.80 (0.15)	81.84 (0.40)	83.09 (0.11)	74.05
DANN [21]	72.81 (0.25)	59.42 (0.66)	81.34 (0.45)	83.09 (0.38)	74.16
f-DAL [17]	75.13 (0.51)	65.66 (0.64)	84.08 (0.38)	83.95 (0.47)	77.20
MDAN [8]	71.27 (0.25)	61.19 (0.57)	81.07 (0.30)	82.58 (0.39)	74.03
M3SDA [2]	72.77 (0.63)	60.96 (0.45)	80.79 (0.50)	82.74 (0.32)	74.32
DARN [1]	71.25 (0.35)	61.33 (0.58)	80.12 (0.36)	82.60 (0.25)	73.82
MIAN [3]	72.08 (0.42)	60.02 (0.73)	81.25 (0.46)	83.16 (0.25)	74.13
WJDOT [44]	71.88 (0.53)	60.60 (0.27)	81.76 (0.65)	83.07 (0.74)	74.33
RRL [13]	76.92 (0.22)	70.12 (0.38)	86.69 (0.33)	85.40 (0.27)	79.78
LTC [37]	71.85 (0.37)	65.66 (0.55)	83.05 (0.30)	82.78 (0.50)	75.84
SImpAl [35]	73.40 (0.40)	62.40 (0.10)	81.00 (0.20)	82.70 (0.20)	74.80
MJD (ours)	**78.24 (0.45)**	**73.08 (0.53)**	**87.86 (0.36)**	**86.55 (0.46)**	**81.43**

Table 5
Mean and std of classification accuracy (%) with ResNet50 model on miniDomainNet (126 classes).

Algorithm	→Clipart	→Painting	→Real	→Sketch	Avg
Baseline	70.76 (0.66)	66.31 (0.72)	71.29 (0.82)	63.04 (0.25)	67.85
DANN [21]	68.23 (0.74)	66.40 (0.60)	69.65 (0.47)	64.78 (0.62)	67.26
f-DAL [17]	74.14 (0.56)	70.78 (0.95)	76.83 (0.38)	66.79 (0.73)	72.14
MDAN [8]	70.17 (0.81)	68.33 (0.76)	78.83 (0.80)	66.17 (0.64)	70.87
M3SDA [2]	74.45 (0.35)	70.85 (0.56)	76.54 (0.74)	67.51 (0.70)	72.34
DARN [1]	74.67 (0.92)	68.41 (0.70)	72.83 (0.36)	67.05 (0.64)	70.74
MIAN [3]	72.67 (0.60)	65.17 (0.65)	78.33 (0.52)	62.17 (0.76)	69.58
WJDOT [44]	73.44 (0.76)	66.89 (0.50)	74.89 (0.63)	64.90 (0.88)	70.03
RRL [13]	79.50 (0.43)	74.28 (0.27)	80.78 (0.53)	74.03 (0.55)	77.15
LTC [37]	75.64 (0.30)	71.23 (0.35)	78.05 (0.40)	69.80 (0.65)	73.68
SImpAl [35]	73.01 (0.71)	69.83 (0.60)	76.55 (0.67)	65.97 (0.74)	71.34
MJD (ours)	**82.03 (0.37)**	**75.54 (0.58)**	**83.24 (0.46)**	**75.51 (0.67)**	**79.08**

Table 6
Mean and std of classification accuracy (%) with ResNet101 model on miniDomainNet (126 classes).

Algorithm	→Clipart	→Painting	→Real	→Sketch	Avg
Baseline	72.98 (0.74)	68.56 (0.58)	73.42 (0.62)	64.98 (0.41)	69.99
DANN [21]	70.95 (0.68)	68.61 (0.42)	70.98 (0.36)	65.65 (0.58)	69.05
f-DAL [17]	75.35 (0.40)	72.90 (0.72)	78.89 (0.76)	67.07 (0.73)	73.55
MDAN [8]	73.05 (0.64)	69.81 (0.70)	80.03 (0.74)	69.97 (0.80)	73.21
M3SDA [2]	78.06 (0.76)	73.65 (0.74)	80.16 (0.34)	70.48 (0.62)	75.59
DARN [1]	76.60 (0.51)	71.63 (0.45)	79.01 (0.68)	71.47 (0.53)	74.68
MIAN [3]	74.00 (0.40)	66.67 (0.67)	79.83 (0.50)	63.50 (0.72)	71.00
WJDOT [44]	75.33 (0.65)	67.32 (0.28)	77.89 (0.36)	66.20 (0.60)	71.69
RRL [13]	82.31 (0.38)	75.50 (0.60)	81.89 (0.83)	**76.04 (0.74)**	78.94
LTC [37]	76.98 (0.54)	74.32 (0.72)	79.60 (0.51)	70.27 (0.86)	75.29
SImpAl [35]	75.31 (0.58)	72.47 (0.80)	78.50 (0.47)	69.29 (0.73)	73.89
MJD (ours)	**83.80 (0.48)**	**77.13 (0.53)**	**84.16 (0.56)**	75.95 (0.40)	**80.26**

Table 7
Mean and std of classification accuracy (%) with ResNet101 model on DomainNet (345 classes).

Algorithm	→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
DANN [21]	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
f-DAL [17]	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
MDAN [8]	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 [2]	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 [1]	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 [3]	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 [44]	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 [13]	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 [37]	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 [35]	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
MJD (ours)	**70.86 (0.40)**	28.01 (0.58)	**61.05 (0.57)**	**30.78 (0.63)**	**70.70 (0.70)**	61.98 (0.42)	**53.90**

Table 8
Statistical test results of other algorithms versus our MJD algorithm.

Algorithm	Baseline	DANN	f-DAL	MDAN	M3SDA	DARN	MIAN	WJDOT	RRL	LTC	SImpAl
T value	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	8.00	5.00	1.00

5.4. Statistical test

To be strict in a statistical sense, we conduct the Wilcoxon signed-ranks test [13] to check if the proposed MJD algorithm is statistically better than its comparison algorithms on the tasks from Tables 1–7. Briefly speaking, the test contrasts two algorithms' performance on N tasks by using a statistic T. In our case, there are $N = 30$ tasks and 11 comparison algorithms. Therefore, we conduct 11 tests: Baseline versus MJD, …, and SImpAl versus MJD, and report the resulting T values in Table 8. From Table 8, we find that with the number of tasks $N = 30$ and a confidence level $\alpha = 0.05$, all the T values are smaller than the critical value 137 for Wilcoxon's test. Accordingly, we conclude that our MJD algorithm is statistically better than its competitors.

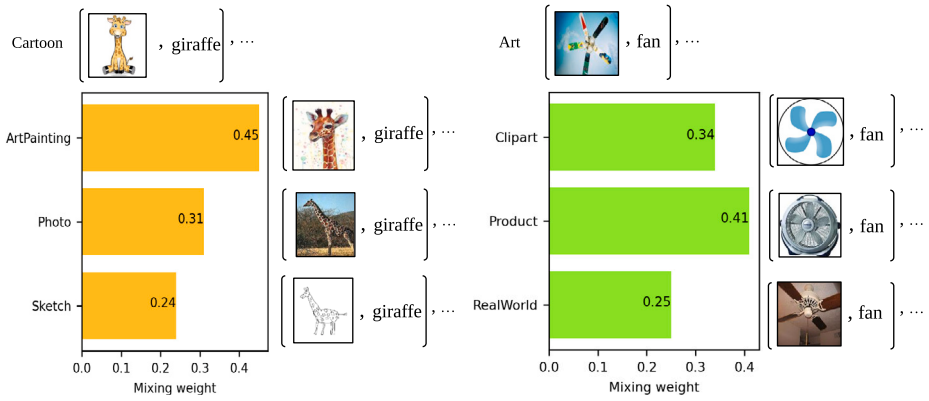
5.5. Experimental analysis

Weight visualization. We visualize in Figs. 3(a) and 3(b) the mixing weights of the source domains (source joint distributions) learned by our MJD algorithm (ResNet50) on tasks "→Cartoon" (PACS) and "→Art" (Office-Home). Interestingly, we observe from Fig. 3(a) that in terms of visual effect, the source domain ArtPainting, which is most similar to the target domain Cartoon, is assigned a large mixing weight of 0.45, and the least similar source domain Sketch receives a small weight of 0.24. From Fig. 3(b), we observe that the source domains more similar to the target domain are also assigned larger weights. These results show that our algorithm well reflects the similarity/importance of each of the source domains to the target domain.

Weight contribution. We study the contribution of the mixing weights by comparing our MJD against its variant MJD ($\alpha_s = \frac{1}{n}$) with the fixed same weight $\alpha_s = \frac{1}{n}$ for the n source domains (joint distributions). Both algorithms utilize the ResNet50 model. Figs. 4(a) and 4(b) plot the classification results of MJD and MJD ($\alpha_s = \frac{1}{n}$) on Office-Home and miniDomainNet, respectively. We find that MJD, which dynamically optimizes the weights according to the source and target data, consistently outperforms MJD ($\alpha_s = \frac{1}{n}$) with the fixed same weight. This verifies that the mixing weights are indispensable and contribute to the performance of our algorithm.

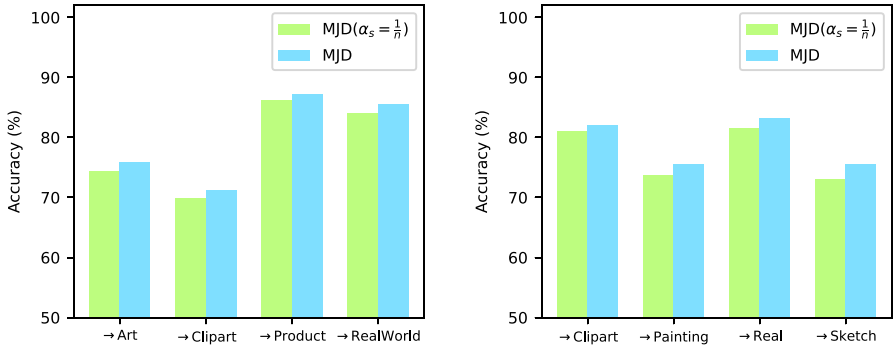
Label updating. We demonstrate that iteratively updating the pseudo target labels in our MJD algorithm (ResNet50 and ResNet101) can make the labels become more accurate. To validate this point, we record our algorithm's target classification accuracy during the iterations and plot the variations of classification accuracy in Figs. 5(a) and 5(b). We observe from the 2 figures that on all the tasks, the target classification accuracy first gradually increases as the iteration proceeds, and then reaches a plateau after a sufficient number of iterations. This indicates that the pseudo labels are improved during the iterations and become more accurate in the end.

Comparison with more SOTAs. We compare our MJD algorithm with more recent State-Of-The-Art (SOTA) methods published in top conferences to reinforce the advantage of our algorithm. Table 9 reports the comparison results on Office-Home. Clearly, our algorithm significantly outperforms iMSDA [46] and DINE [47]. Besides, on DomainNet, the average classification results of 53.90% versus 51.70% also indicate that our algorithm is much better than the method EIDCo



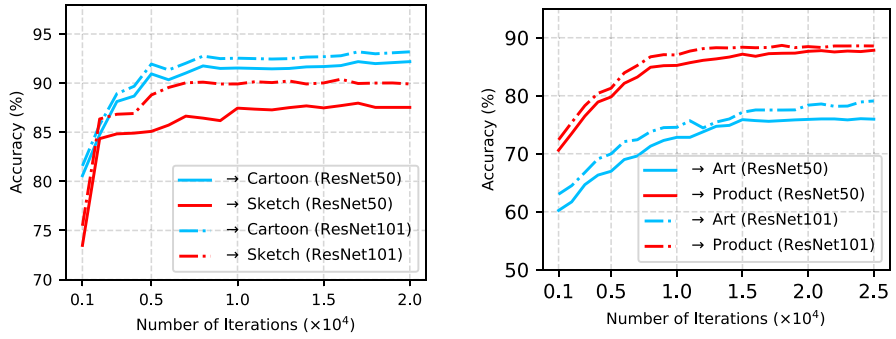
(a) PACS (b) Office-Home

Fig. 3. Mixing weights of the source domains (source joint distributions) learned by the MJD algorithm (ResNet50). (a) The sources are ArtPainting, Photo, and Sketch, and the target is Cartoon (PACS). (b) The sources are Clipart, Product, and RealWorld, and the target is Art (Office-Home).



(a) Office-Home (b) miniDomainNet

Fig. 4. Classification accuracy of MJD and MJD ($\alpha_s = \frac{1}{n}$) on Office-Home and miniDomainNet. MJD ($\alpha_s = \frac{1}{n}$) is the variant of MJD with the fixed same weight $\alpha_s = \frac{1}{n}$ for the n source domains.



(a) PACS (b) Office-Home

Fig. 5. Target classification accuracy of MJD (ResNet50 and ResNet101) over the iterations on PACS and Office-Home.

(ICCV23) [48]. Based on these comparison results, we believe that our algorithm is more advantageous and effective than the SOTAs.

Multi-source versus single-source. We explore whether the result from all the source domains is better than the result from a single source domain. For this purpose, we compare our MJD against its variant MJD ($n = 1$), which selects a single source domain to produce the best target classification accuracy. Table 10 shows the results of MJD and MJD ($n = 1$) using the ResNet50 model on the Office-Home dataset. We observe that MJD unambiguously outperforms MJD ($n = 1$) across all tasks. This testifies that appropriately incorporating multiple source domains into adaptation indeed leads to better result than using a single source domain.

Table 9
Results of MJD and SOTAs on Office-Home.

Algorithm	→Art	→Clipart	→Product	→RealWorld	Avg
iMSDA (ICML22) [46]	75.40 (0.86)	61.40 (0.73)	83.50 (0.22)	84.47 (0.38)	76.19
DINE (CVPR22) [47]	74.80 (0.00)	64.10 (0.00)	85.00 (0.00)	84.60 (0.00)	77.10
MJD (ours)	**75.88 (0.31)**	**71.25 (0.35)**	**87.31 (0.23)**	**85.58 (0.32)**	**80.01**

Table 10
Results of MJD and MJD ($n = 1$) on Office-Home.

Algorithm	→Art	→Clipart	→Product	→RealWorld	Avg
MJD	**75.88 (0.31)**	**71.25 (0.35)**	**87.31 (0.23)**	**85.58 (0.32)**	**80.01**
MJD ($n = 1$)	72.80 (0.27)	61.32 (0.36)	85.78 (0.20)	81.56 (0.37)	75.37

Table 11
Results of MJD, MJD-W, and MJD-H on miniDomainNet.

Algorithm	→Clipart	→Painting	→Real	→Sketch	Avg
MJD	**82.03 (0.37)**	75.54 (0.58)	**83.24 (0.46)**	**75.51 (0.67)**	**79.08**
MJD-W	76.83 (0.58)	72.03 (0.72)	79.74 (0.29)	72.80 (0.72)	75.35
MJD-H	80.41 (0.41)	**76.45 (0.64)**	82.38 (0.35)	74.61 (0.66)	78.46

Other metrics. We study the impact of other metrics on the performance of our MJD algorithm. Considering the applicability of the Wasserstein distance and Hellinger distance in quantifying joint distribution disparity [13,23], we use these metrics to replace the original Pearson χ^2 divergence and derive 2 variants of our algorithm: MJD with the Wasserstein distance (MJD-W) and MJD with the Hellinger distance (MJD-H). Table 11 reports the classification results of MJD, MJD-W, and MJD-H on the miniDomainNet dataset. We find that MJD notably outperforms MJD-W across all tasks and exhibits superior performance compared to MJD-H in three out of the four tasks. These findings imply that, for our algorithm, the Pearson χ^2 divergence proves more effective than other metrics such as the Wasserstein distance and Hellinger distance.

6. Conclusion

In this article, we develop a theory for the MSDA problem by introducing the concepts of mixture joint distribution and Pearson χ^2 divergence. Grounded on the developed theory, we propose the Mixture of Joint Distributions (MJD) algorithm, which optimizes both the mixing weights and the network to minimize the estimated source mixture loss and the estimated Pearson χ^2 divergence. We analytically solve a quadratic maximization problem to derive the estimated Pearson χ^2 divergence, so that it can be explicitly expressed as a loss of the mixing weights and the network's feature extractor. Our experiments on several object recognition datasets show that with the deep ResNet50 and deeper ResNet101 models, our MJD algorithm statistically outperforms other comparison algorithms.

It is worth mentioning that in this work, we implicitly assume that the source mixture label distribution $p^{\boldsymbol{\alpha}}(y)$ and the target label distribution $p^t(y)$ are not very different, such that the source mixture joint distribution and the target joint distribution can be aligned. In the case where this assumption is violated, our theory and algorithm may not work very well. As a future work, we therefore intend to design and learn a weight function to overcome such a limitation. Additionally, we also plan to explore the variant of MSDA, where the source and target domains possess different label spaces.

CRediT authorship contribution statement

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

Declaration of competing interest

Authors declare that they have no conflict of interest. The submitted manuscript is an original work, which has not been published elsewhere and is not currently under consideration for another journal, conference or workshop.

Data availability

Data will be made available on request.

Acknowledgments

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

Appendix. Proof of Theorem 1

Proof. First, we have

$$\begin{aligned}
&\mathbb{E}_{p^t(\boldsymbol{x},y)}[L(f(\boldsymbol{x}),y)]\\
&=\mathbb{E}_{p^{\boldsymbol{\alpha}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x}),y)]+\mathbb{E}_{p^t(\boldsymbol{x},y)}[L(f(\boldsymbol{x}),y)]-\mathbb{E}_{p^{\boldsymbol{\alpha}}(\boldsymbol{x},y)}[L(f(\boldsymbol{x}),y)] &\text{(A.1)}\\
&\le\mathbb{E}_{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})),y)]+\left|\mathbb{E}_{p^t(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})),y)]\right.\\
&\qquad\left.-\mathbb{E}_{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})),y)]\right| &\text{(A.2)}\\
&\le\mathbb{E}_{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}[L(g(h(\boldsymbol{x})),y)]+M\int\left|p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)-p^t(h(\boldsymbol{x}),y)\right|dh(\boldsymbol{x})dy, &\text{(A.3)}
\end{aligned}$$

where Eq. (A.3) utilizes the assumption that loss $L \le M$. Then, we also have

$$\begin{aligned}
&\int\left|p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)-p^t(h(\boldsymbol{x}),y)\right|dh(\boldsymbol{x})dy\\
&=\sqrt{\left(\int\left|p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)-p^t(h(\boldsymbol{x}),y)\right|dh(\boldsymbol{x})dy\right)^2} &\text{(A.4)}\\
&\le\left[\int\left(\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}+\sqrt{p^t(h(\boldsymbol{x}),y)}\right)^2dh(\boldsymbol{x})dy\right]^{\frac{1}{2}}\\
&\times\left[\int\left(\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}-\sqrt{p^t(h(\boldsymbol{x}),y)}\right)^2dh(\boldsymbol{x})dy\right]^{\frac{1}{2}} &\text{(A.5)}\\
&=\left[2+2\int\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)p^t(h(\boldsymbol{x}),y)}dh(\boldsymbol{x})dy\right]^{\frac{1}{2}}\\
&\times\left[2-2\int\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)p^t(h(\boldsymbol{x}),y)}dh(\boldsymbol{x})dy\right]^{\frac{1}{2}} &\text{(A.6)}\\
&\le 2\left[2-2\int\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)p^t(h(\boldsymbol{x}),y)}dh(\boldsymbol{x})dy\right]^{\frac{1}{2}} &\text{(A.7)}\\
&=2\left[2\int\left(1-\frac{\sqrt{p^t(h(\boldsymbol{x}),y)}}{\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}}\right)p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy\right]^{\frac{1}{2}} &\text{(A.8)}\\
&\le 2\left[2\int\left(-\log\frac{\sqrt{p^t(h(\boldsymbol{x}),y)}}{\sqrt{p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)}}\right)p^{\boldsymbol{\alpha}}(h(\boldsymbol{x}),y)dh(\boldsymbol{x})dy\right]^{\frac{1}{2}} &\text{(A.9)}
\end{aligned}$$

$$= 2\Big[\int \Big(\log \frac{p^{\alpha}(h(\boldsymbol{x}), y)}{p^{t}(h(\boldsymbol{x}), y)}\Big) p^{\alpha}(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y\Big]^{\frac{1}{2}} \tag{A.10}$$

$$\leq 2\Big[\int \Big(\frac{p^{\alpha}(h(\boldsymbol{x}), y)}{p^{t}(h(\boldsymbol{x}), y)} - 1\Big) p^{\alpha}(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y\Big]^{\frac{1}{2}} \tag{A.11}$$

$$= 2\Big[\int \Big[\Big(\frac{p^{\alpha}(h(\boldsymbol{x}), y)}{p^{t}(h(\boldsymbol{x}), y)}\Big)^{2} - 1\Big] p^{t}(h(\boldsymbol{x}), y) d h(\boldsymbol{x}) d y\Big]^{\frac{1}{2}} \tag{A.12}$$

$$\leq \frac{2}{\sqrt{\epsilon}} \mathrm{D}_{\chi^2}(p^{\alpha}(h(\boldsymbol{x}), y) \parallel p^{t}(h(\boldsymbol{x}), y)). \tag{A.13}$$

Eq. (A.5) applies the Cauchy–Schwarz inequality. Eq. (A.7) holds since

$$2 \int \sqrt{p^{\alpha}(h(\boldsymbol{x}), y) p^{t}(h(\boldsymbol{x}), y)} d h(\boldsymbol{x}) d y$$

$$\leq \int [p^{\alpha}(h(\boldsymbol{x}), y) + p^{t}(h(\boldsymbol{x}), y)] d h(\boldsymbol{x}) d y = 2. \tag{A.14}$$

Eqs. (A.9) and (A.11) apply inequalities $1 - u \leq -\log u$ and $\log u \leq u - 1$, $\forall u > 0$, respectively. Eq. (A.13) uses the assumption that $\mathrm{D}_{\chi^2}(p^{\alpha}(h(\boldsymbol{x}), y) \parallel p^{t}(h(\boldsymbol{x}), y))$ is lower bounded by $\epsilon > 0$, leading to $2\sqrt{\epsilon}\sqrt{\mathrm{D}_{\chi^2}(p^{\alpha}(h(\boldsymbol{x}), y) \parallel p^{t}(h(\boldsymbol{x}), y))/\epsilon} \leq \frac{2}{\sqrt{\epsilon}} \mathrm{D}_{\chi^2}(p^{\alpha}(h(\boldsymbol{x}), y) \parallel p^{t}(h(\boldsymbol{x}), y))$. Finally, combining Eqs. (A.3) and (A.13), we obtain the result

$$\mathbb{E}_{p^{t}(\boldsymbol{x}, y)}[L(f(\boldsymbol{x}), y)]$$

$$\leq \mathbb{E}_{p^{\alpha}(h(\boldsymbol{x}), y)}[L(g(h(\boldsymbol{x})), y)] + \frac{2M}{\sqrt{\epsilon}} \mathrm{D}_{\chi^2}(p^{\alpha}(h(\boldsymbol{x}), y) \parallel p^{t}(h(\boldsymbol{x}), y)) \tag{A.15}$$

$$= \sum_{s=1}^{n} \alpha_s \mathbb{E}_{p^{s}(h(\boldsymbol{x}), y)}[L(g(h(\boldsymbol{x})), y)] + \frac{2M}{\sqrt{\epsilon}} \mathrm{D}_{\chi^2}$$

$$\times \Big(\sum_{s=1}^{n} \alpha_s p^{s}(h(\boldsymbol{x}), y) \parallel p^{t}(h(\boldsymbol{x}), y)\Big). \quad \square \tag{A.16}$$

References

[1] J. Wen, R. Greiner, D. Schuurmans, Domain aggregation networks for multi-source domain adaptation, in: International Conference on Machine Learning, Vol. 119, 2020, pp. 10214–10224.

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

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

[4] Y.-H. Liu, C.-X. Ren, A two-way alignment approach for unsupervised multi-source domain adaptation, Pattern Recognit. 124 (2022) 108430.

[5] M. Kang, P. Chikontwe, D. Won, M. Luna, S.H. Park, Structure-preserving image translation for multi-source medical image domain adaptation, Pattern Recognit. 144 (2023) 109840.

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

[7] S. Zhao, B. Li, P. Xu, K. Keutzer, Multi-source domain adaptation in the deep learning era: A systematic survey, 2020, pp. 1–7, arXiv preprint arXiv:2002.12169.

[8] H. Zhao, S. Zhang, G. Wu, J.M.F. Moura, J.P. Costeira, G.J. Gordon, Adversarial multiple source domain adaptation, in: Advances in Neural Information Processing Systems, Vol. 31, 2018.

[9] Y. Mansour, M. Mohri, A. Rostamizadeh, Domain adaptation with multiple sources, in: Advances in Neural Information Processing Systems, 2009, pp. 1041–1048.

[10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J.W. Vaughan, A theory of learning from different domains, Mach. Learn. 79 (1) (2010) 151–175.

[11] J. Hoffman, M. Mohri, N. Zhang, Algorithms and theory for multiple-source adaptation, in: Advances in Neural Information Processing Systems, 2018, pp. 8246–8256.

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

[13] S. Chen, L. Zheng, H. Wu, Riemannian representation learning for multi-source domain adaptation, Pattern Recognit. 137 (2023) 109271.

[14] K. Li, J. Lu, H. Zuo, G. Zhang, Multi-source contribution learning for domain adaptation, IEEE Trans. Neural Netw. Learn. Syst. 33 (10) (2022) 5293–5307.

[15] Y. Yao, X. Li, Y. Zhang, Y. Ye, Multisource heterogeneous domain adaptation with conditional weighting adversarial network, IEEE Trans. Neural Netw. Learn. Syst. 34 (4) (2023) 2079–2092.

[16] Z.-G. Liu, L.-B. Ning, Z.-W. Zhang, A new progressive multisource domain adaptation network with weighted decision fusion, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–11.

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

[18] S. Chen, L. Wang, Z. Hong, X. Yang, Domain generalization by joint-product distribution alignment, Pattern Recognit. 134 (2023) 109086.

[19] B. Schölkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and beyond, MIT Press, 2001.

[20] Y. Mansour, M. Mohri, A. Rostamizadeh, Domain adaptation: Learning bounds and algorithms, in: Conference on Learning Theory, 2009.

[21] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. 17 (59) (2016) 1–35.

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

[23] B. Bhushan Damodaran, B. Kellenberger, R. Flamary, D. Tuia, N. Courty, Deepjdot: Deep joint distribution optimal transport for unsupervised domain adaptation, in: European Conference on Computer Vision, 2018, pp. 447–463.

[24] S. Chen, L. Han, X. Liu, Z. He, X. Yang, Subspace distribution adaptation frameworks for domain adaptation, IEEE Trans. Neural Netw. Learn. Syst. 31 (12) (2020) 5204–5218.

[25] S. Chen, Z. Hong, M. Harandi, X. Yang, Domain neural adaptation, IEEE Trans. Neural Netw. Learn. Syst. 34 (11) (2023) 8630–8641.

[26] W.M. Kouw, M. Loog, A review of domain adaptation without target labels, IEEE Trans. Pattern Anal. Mach. Intell. 43 (3) (2021) 766–785.

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

[28] L. Zhang, X. Gao, Transfer adaptation learning: A decade survey, IEEE Trans. Neural Netw. Learn. Syst. (2022) 1–22.

[29] R. Xu, Z. Chen, W. Zuo, J. Yan, 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.

[30] R. Li, X. Jia, J. He, S. Chen, 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.

[31] Z. Luo, X. Zhang, S. Lu, S. Yi, Domain consistency regularization for unsupervised multi-source domain adaptive classification, Pattern Recognit. 132 (2022) 108955.

[32] S. Chen, H. Wu, C. Liu, Domain invariant and agnostic adaptation, Knowl.-Based Syst. 227 (2021) 107192.

[33] S. Zhao, G. Wang, S. Zhang, Y. Gu, Y. Li, Z. Song, P. Xu, R. Hu, H. Chai, K. Keutzer, Multi-source distilling domain adaptation, in: AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 12975–12983.

[34] L. Yang, Y. Balaji, S.-N. Lim, A. Shrivastava, Curriculum manager for source selection in multi-source domain adaptation, in: European Conference on Computer Vision, 2020, pp. 608–624.

[35] N. Venkat, J.N. Kundu, D. Singh, A. Revanur, et al., Your classifier can secretly suffice multi-source domain adaptation, in: Advances in Neural Information Processing Systems, Vol. 33, 2020, pp. 4647–4659.

[36] V.-A. Nguyen, T. Nguyen, T. Le, Q.H. Tran, D. Phung, Stem: An approach to multi-source domain adaptation with guarantees, in: IEEE International Conference on Computer Vision, 2021, pp. 9352–9363.

[37] H. Wang, M. Xu, B. Ni, W. Zhang, Learning to combine: Knowledge aggregation for multi-source domain adaptation, in: European Conference on Computer Vision, 2020, pp. 727–744.

[38] D. Li, Y. Yang, Y.-Z. Song, T.M. Hospedales, Deeper, broader and artier domain generalization, in: IEEE International Conference on Computer Vision, 2017, pp. 5542–5550.

[39] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5385–5394.

[40] S. Chen, Decomposed adversarial domain generalization, Knowl.-Based Syst. 263 (2023) 110300.

[41] S. Chen, M. Harandi, X. Jin, X. Yang, Domain adaptation by joint distribution invariant projections, IEEE Trans. Image Process. 29 (2020) 8264–8277.

[42] J. Nocedal, S.J. Wright, Numerical Optimization, Springer, 1999.

[43] K. Zhou, Y. Yang, Y. Qiao, T. Xiang, Domain adaptive ensemble learning, IEEE Trans. Image Process. 30 (2021) 8008–8018.

[44] R. Turrisi, R. Flamary, A. Rakotomamonjy, M. Pontil, Multi-source domain adaptation via weighted joint distributions optimal transport, in: Conference on Uncertainty in Artificial Intelligence, 2022.

[45] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[46] L. Kong, S. Xie, W. Yao, Y. Zheng, G. Chen, P. Stojanov, V. Akinwande, K. Zhang, Partial identifiability for domain adaptation, in: International Conference on Machine Learning, 2022, pp. 11455–11472.
[47] J. Liang, D. Hu, J. Feng, R. He, Dine: Domain adaptation from single and multiple black-box predictors, in: IEEE Conference on Computer Vision and Pattern Recognition, 2022, pp. 8003–8013.
[48] Y. Zhang, Z. Wang, J. Li, J. Zhuang, Z. Lin, Towards effective instance discrimination contrastive loss for unsupervised domain adaptation, in: IEEE International Conference on Computer Vision, 2023, pp. 11388–11399.

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