Pattern Recognition 174 (2026) 113013

Contents lists available at ScienceDirect

Pattern Recognition

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

Open-Set Domain Adaptation by Joint Distribution Alignment and Unknown Risk Minimization

Lisheng Wen [a], Sentao Chen [b,*], Lin Zheng [b], Ping Xuan [c]

[a] School of Microelectronics and Communication Engineering, Chongqing University, China
[b] Department of Computer Science and Technology, Shantou University, China
[c] School of Cyberspace Security, Hainan University, China

ARTICLE INFO

Keywords:
Statistical machine learning
Open-set domain adaptation
Statistical distance minimization
Empirical risk minimization

ABSTRACT

Open-Set Domain Adaptation (OSDA) aims to generalize a classification model from a source domain (source joint distribution) to a different target domain (target joint distribution), where the target joint distribution contains open-set (unknown) classes. To achieve this OSDA goal, a good target model should be learned by minimizing the target risk. In this paper, we show that the model's target risk is upper bounded by its source risk, the χ^2-divergence between the source joint distribution and the target known joint distribution, and the target unknown risk. Based on this theoretical risk bound, we propose a **J**oint Distribution **A**lignment and **U**nknown Risk **M**inimization (JAUM) algorithm, which optimizes the model by minimizing the empirical source risk, the empirical χ^2-divergence, and the empirical target unknown risk. Experimental results show that our method statistically outperforms other methods on benchmark datasets. The introductory video and PyTorch code are available on GitHub (https://github.com/sentaochen/Joint-Distribution-Alignment-and-Unknown-Risk-Minimization). Interested readers can also visit https://github.com/sentaochen for more source code on the related topics, including (multi-source, partial) domain adaptation and domain generalization. These resources provide a series of practical kernel-based techniques for estimating the KL divergence (mutual information), Hellinger distance, χ^2-divergence, and other f-divergences from empirical data, which may be applied to a wider range of machine learning and pattern recognition problems.

1. Introduction

Traditional supervised learning generally trains a classification model by minimizing the empirical training risk on the training dataset, which is generated by the training joint distribution $P^{tr}(x, y)$ of the input features x and class label y [1]. Under the assumption that the training joint distribution $P^{tr}(x, y)$ is identical to the testing joint distribution $P^{te}(x, y)$, *i.e.*, $P^{tr}(x, y) = P^{te}(x, y)$, minimizing the empirical training risk leads to a model with low testing risk and good performance [2,3]. However, this assumption is often violated in practical applications, resulting in the trained model performing poorly. To overcome this limitation, Domain Adaptation (DA) has been introduced, enabling the trained model to generalize well from a training (source) joint distribution to a different testing (target) joint distribution [4–6].

This paper focuses on a more realistic DA scenario, called Open-Set Domain Adaptation (OSDA) [7–9], where the target domain includes open-set (unknown) classes. In OSDA, a labeled source dataset and an unlabeled target dataset are provided, which are sampled from the source domain (source joint distribution) $P^s(x, y)$ and the marginal $P^t(x)$ of the target domain (target joint distribution) $P^t(x, y)$, respectively. The target joint distribution contains both the source closed-set (known) classes and the target open-set (unknown) classes. The goal of OSDA is to train a classification model that (i) recognizes the target open-set classes data as a uniform unknown class, and (ii) classifies other target data into the correct known classes. That is, the model should achieve a good target performance with a small target risk.

To achieve the OSDA goal, the primary challenges arise from the joint distribution mismatch, and the mixed classes of the target joint distribution. Traditional DA methods attempt to optimize a feature extractor h, which is a component of the classification model f, to align the joint distributions such that $P^t(h(x), y) \approx P^s(h(x), y)$ [5,10]. However, since the target joint distribution is with respect to both the known and unknown classes and the source joint distribution is with respect to only the known classes, aligning the source joint distribution and the target joint distribution is unreasonable and infeasible in OSDA (see more details in Section 2.1 "**Domain Adaptation**"). In statistical learning, solving the OSDA problem requires addressing (i) the target mixture

* Corresponding author.
E-mail addresses: lishengwenmail@126.com (L. Wen), sentaochenmail@gmail.com (S. Chen).

https://doi.org/10.1016/j.patcog.2025.113013
Received 19 April 2025; Received in revised form 5 November 2025; Accepted 25 December 2025
Available online 29 December 2025

distribution of the known and unknown classes, and (ii) the cross-domain joint distribution mismatch with respect to the known classes.

Existing OSDA works have been proposed to tackle the above two challenges. For the challenge of target mixture distribution of the known and unknown classes, these works separate the target unknown distribution from the target known distribution using open-set recognition [8,11], threshold-based discrimination [7], or weighting mechanisms [12,13]. For convenience, we refer to the target distribution with respect to the known and unknown classes as target known distribution and target unknown distribution, respectively. For the challenge of cross-domain joint distribution mismatch with respect to the known classes, these works align the source distribution to the target known distribution using adversarial learning [7,14], projection matrices [11,15], or rotation recognition [16]. Note that, separating the target unknown distribution helps recognize the target unknown class data, and aligning the distributions with respect to the known classes enhances the model's classification ability for the target known classes data. Finally, a good target model is learned with a small target risk.

Although these existing OSDA methods have proven practically effective, they suffer from two main shortcomings. First, some works separate the target unknown distribution using heuristic methods, such as fixing a threshold to build a boundary to separate the target unknown class data [7], or introducing a weighting network to estimate the likelihood that data belongs to the unknown class [12,13]. Since these works lack a theoretical foundation for separating the target unknown distribution, they may be suboptimal in handling the target risk on the unknown class and recognizing the target unknown class data. Second, by the theory of Ben-David et al. [17], reducing the difference between marginal distributions is considered beneficial for minimizing the target risk. Therefore, some works optimize the feature extractor h to align the marginal distributions $P^s(h(x)) \approx P^t(h(x))$ [7,12] or align the class-conditional distributions $P^s(h(x)|y) \approx P^t(h(x)|y)$ [18] on the known classes. However, from a statistical perspective, the target risk is closely related to the mismatch between the joint distributions, not the marginal or the class-conditional distributions [5,10,19]. Since these works do not match the joint distributions with respect to the known classes, they may be sub-optimal in handling the target risk on the known classes and classifying the target known classes data.

In this work, we overcome the above shortcomings and provide theory and algorithm to solve the OSDA problem. We start from the model's target risk on both the known and unknown classes. For the target risk on the known classes, we show that it can be controlled by the source risk and the χ^2-divergence between the source joint distribution and the target known joint distribution. For the target risk on the unknown class, we represent it as the target unknown risk with the unknown probability. By combining these two parts, we know that the model's target risk is upper bounded by its source risk, the χ^2-divergence, and the target unknown risk. Motivated by this theoretical risk bound, we propose a **J**oint Distribution **A**lignment and **U**nknown Risk **M**inimization (JAUM) algorithm to learn a good target model by minimizing (i) the empirical source risk, (ii) the empirical χ^2-divergence, and (iii) the empirical target unknown risk. Specifically, the empirical χ^2-divergence is derived by solving an unconstrained quadratic optimization problem which has an analytic solution. Minimization of the empirical χ^2-divergence matches the cross-domain joint distributions with respect to the known classes. The empirical target unknown risk is obtained from the sample average with the unknown probability. Minimization of the empirical target unknown risk separates the target unknown distribution. Clearly, our work differs from the aforementioned OSDA works in three key ways. First, we focus on target risk minimization, which trains a good target model in a statistical sense. Second, we employ unknown risk minimization to solve the target mixture distribution challenge, and empirically illustrate its usefulness in Section 4.4 "**Feature Visualization**". Third, we align the joint distributions to solve the joint distribution mismatch challenge, which is theoretically and experimentally better than aligning the marginal distributions or the class-conditional distributions (see more details in Section 2.1 "**Domain Adaptation**" and Section 4.4 "**Analysis of χ^2-divergence-based Joint Distribution Alignment.** "). Finally, we conduct comparative experiments to demonstrate the effectiveness of our algorithm. Below, we summarize the contributions of this paper:

- We provide a new theoretical risk bound for OSDA, which shows that the classification model's target risk is upper bounded by its source risk, the χ^2-divergence between the source joint distribution and the target known joint distribution, and the target unknown risk.
- Based on the theoretical risk bound, we propose a JAUM algorithm to optimize the model by minimizing the empirical source risk, the empirical χ^2-divergence, and the empirical target unknown risk.
- We utilize joint distribution alignment and unknown risk minimization to address the two key challenges in OSDA.
- We conduct comparative and analytical experiments on three datasets, demonstrating the effectiveness of our algorithm.

2. Related work

2.1. Domain adaptation

DA aims to train a classification model f with a small target risk, by leveraging data sampled from a labeled source domain and an unlabeled target domain [4–6]. The challenge in DA is the source and target domains are described by different joint distributions, *i.e.*, $P^s(x,y) \neq P^t(x,y)$, which causes the trained model to suffer from a large target risk and consequently a weak target performance. To tackle the challenge of cross-domain joint distribution mismatch, most existing DA works propose to optimize a feature extractor h, which is a component of the classification model f, to align the marginal distributions such that $P^s(h(x)) \approx P^t(h(x))$ [6,20,21], the class-conditional distributions such that $P^s(h(x)|y) \approx P^t(h(x)|y)$ [22,23], or the joint distributions such that $P^s(h(x),y) \approx P^t(h(x),y)$ [5,10,24]. For instance, among the marginal distribution alignment works [6,20,21], Nguyen et al. [21] first proposed a theory to show that reducing the marginal distributions difference helps reduce the target risk, and then minimized the estimated Kullback-Leibler (KL) divergence between marginal distributions. Among the class-conditional distribution alignment works [22,23], Ge et al. [23] minimized a metric called Conditional Maximum Mean Discrepancy (CMMD) to match the source class-conditional distribution $P^s(h(x)|y)$ and the target class-conditional distribution $P^t(h(x)|y)$. Note that, since the CMMD metric fixes the class label y and measures the disparity between $P^s(h(x)|y)$ and $P^t(h(x)|y)$, which is fundamentally equivalent to measuring the disparity between $P^t(h(x))$ and $P^s(h(x))$, the class-conditional distribution alignment can therefore be regarded as a fine-grained marginal distribution alignment. Among the joint distribution alignment works [5,10], Chen et al. [5] first provided a theoretical risk bound and employed the Relative Chi-Square (RCS) divergence for measuring the joint distributions difference. They then minimized the estimated RCS-divergence to align the joint distributions, ultimately provided an algorithmic solution with powerful experimental results.

Our JAUM algorithm is related to the above joint distribution alignment works, which align the joint distributions to reduce the target risk. This is because the target risk is closely related to the discrepancies between the joint distributions, not the marginal distributions or the class-conditional distributions [5,10,19]. According to probability theory [2], we know that joint distribution $P(x,y)$ can be decomposed into the product of marginal distribution $P(x)$ and class-posterior distribution $P(y|x)$, *i.e.*, $P(x,y) = P(x)P(y|x)$. Since $P(y|x)$ may vary across domains, aligning the marginal or the class-conditional distribution therefore does not solve the core joint distribution mismatch challenge, and is sub-optimal in minimizing the target risk [19]. Besides, several joint distribution alignment works for DA [5,10,19] have brought strong experimental results, which reinforce the importance of joint distribution alignment. Note that, different from the above joint distribution alignment works for DA, our work for OSDA aligns the

source joint distribution to the target known joint distribution, because aligning the joint distributions with respect to both the known and unknown classes is infeasible in OSDA. To be specific, suppose we optimize the feature extractor h to align the joint distributions such that $P^s(h(x), y) \approx P^t(h(x), y)$, then by marginalizing both sides of the equation over $h(x)$, *i.e.*, $\int P^s(h(x), y)dh(x) = \int P^t(h(x), y)dh(x)$, we will derive a new equation $P^s(y) \approx P^t(y)$. Since the source joint distribution is with respect to only the known classes and the target joint distribution is with respect to both the known and unknown classes, the above new equation clearly violates the OSDA problem definition. Hence, aligning joint distributions with respect to the known and unknown classes is not reasonable and infeasible for tackling the OSDA problem.

Besides the above distribution alignment approaches, recent works [25,26] explore other strategies. For example, Farhadi et al. [25] integrated meta-learning and optimized the weights of the selecting subset, which utilizes previously acquired knowledge to adapt models to a new domain. He et al. [26] injected physics-informed priors into the adaptation pipeline by constraining feature extraction and loss terms with domain-specific physical priors to improve interpretability.

2.2. Open-set domain adaptation

OSDA is a more realistic DA scenario, where the target joint distribution contains open-set (unknown) classes. Under the mainstream problem setup [7,14,27,28], the model is trained to (i) recognize the target open-set classes data as a uniform unknown class, and (ii) classify other target data into the correct known classes. Here, it is worth noting that distinguishing multiple target unknown classes (*e.g.*, classifying one unknown class versus another unknown class) is a more complex problem and is beyond the focus of mainstream OSDA methods, including ours.

The challenges in OSDA are (i) the target mixture distribution of the known and unknown classes, and (ii) the cross-domain joint distribution mismatch with respect to the known classes. To tackle the first challenge, prior OSDA works [7,12,13] separate the target unknown distribution from the target known distribution by utilizing heuristic methods, thereby enabling the trained model to recognize the target unknown class data with a small target risk on the unknown class. To tackle the second challenge, prior OSDA works [7,11,14] align the marginal or the class-conditional distributions with respect to the known classes, thereby enabling the trained model to classify the target known classes data with a small target risk on the known classes. For example, Saito et al. [7] first fixed a classification boundary to separate the target unknown distribution, and then aligned the source marginal distribution with the target known marginal distribution using adversarial learning. Liu et al. [12] trained a multi-binary classifier to assign weights to target data, where the weights represent the similarities between the known classes and the target data, and aligned the source marginal distribution with the weighted target marginal distribution. Jang et al. [14] first used entropy as an indicator and combined posterior inference to estimate the probability of target data belonging to the unknown class. Then, the authors aligned the source marginal with target known marginal distributions and separated the target unknown marginal distribution.

Besides the above works, Fang et al. [8] introduced the open set difference for measuring target risk on the unknown class, and provided a theoretical risk bound for OSDA. According to the risk bound, they optimized a shallow network model to minimize the source risk, the Maximum Mean Discrepancy (MMD) between marginal distributions, and the open set difference. Unfortunately, when extending the shallow network to deep neural network, the open set difference converges to a negative value, which degrades the model's performance in recognizing the unknown class data. Considering this limitation, Zhong et al. [18] introduced a threshold boundary to circumvent the negative value in the open set difference, and trained a deep neural network model with a small target risk.

Our work differs from the aforementioned OSDA works in three aspects. (i) We provide a new theoretical risk bound to address the OSDA problem, enabling us to train a good target model in a statistical sense. (ii) Unlike existing works that rely on heuristic methods to separate and classify the target unknown data, our work classifies the unknown data via unknown risk minimization, which is consistent with the established Empirical Risk Minimization (ERM) principle [1] in statistical learning. (iii) Unlike existing works that align the marginal or class-conditional distributions, our work aligns the joint distributions, which is theoretically and experimentally better than them. In other related problems, joint distribution alignment has been proven more advantageous [24,29–31]. Of course, we notice that Zhong et al. [18] have also provided a theoretical risk bound for the OSDA problem. However, their risk bound suggests aligning the class-conditional distributions for OSDA, not the joint distributions. Moreover, they introduced a threshold boundary to the target risk, which may be sub-optimal for minimizing the target risk and learning a good target model.

2.3. Other domain adaptation settings

Beyond the conventional DA and OSDA scenarios, several related DA settings are worth mentioning. For example, Domain Generalization (DG) assumes multiple source domains but no access to the target domain [30,32,33]. Chen et al. [30] aligned multiple source joint distributions via a joint-product alignment objective to reduce domain shift and enhance generalization to unseen domains. Multi-Source DA (MSDA) extends conventional DA to multiple labeled sources [34,35]. Cai et al. [35] employed a diffusive adversarial generation model together with a causal-guided re-weighting scheme and a density-constraint regularization to align multiple source domains while preserving label-conditional semantics. In Source-Free DA, the source model is available but source data are not (*e.g.*, for privacy reasons). Tian et al. [36] trained a more robust source model using augmentation and label smoothing techniques, then adapted it to the unlabeled target domain. Universal DA is similar to OSDA but does not assume prior knowledge of the label-set overlap between the source and target domains, meaning that either domain may contain private classes. The goal of Universal DA is to classify the target data into shared classes or label them unknown class if they belong to target-private classes. You et al. [37] proposed the Universal Adaptation Network (UAN), which estimates each data's transferability to identify the shared label subset and rejects unknown classes accordingly. Cai et al. [38] introduced a more relaxed setting called More Universal DA, and proposed an attention cycle-consistent universal network that aligns multiple modalities and reduces distribution shift among source modalities.

3. Algorithm

3.1. Problem formulation

In this paper, we focus on OSDA problem. We denote a domain as a joint distribution $P(x, y)$ over input features $x \in \mathcal{X}$ and class label $y \in \mathcal{Y}$, where $\mathcal{X}$ is the input feature space and $\mathcal{Y}$ is the class label set. In OSDA, a labeled source dataset $\mathcal{D}^s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and an unlabeled target dataset $\mathcal{D}^u = \{x_i^t\}_{i=1}^{n_t}$ are provided. These datasets are drawn from the source joint distribution $P^s(x, y)$ and the marginal $P^t(x)$ of the target joint distribution $P^t(x, y)$, respectively. The source label set has K known classes, $\mathcal{Y}^s = \{1, \dots, K\}$, and the target label set has the K known classes and an aggregated unknown class *unk*, *i.e.*, $\mathcal{Y}^t = \mathcal{Y}^s \cup \{unk\}$. Note that in this paper, the multiple target unknown classes are combined as an aggregated unknown class by following the common practice in mainstream OSDA literature [7,8,14,16,28]. The goal of OSDA is to train a classification model $f : \mathcal{X} \to \mathcal{Y}^t$ with minimal target risk $\int P^t(x, y)\ell(f(x), y)dxdy$, where ℓ is the classification loss for measuring the prediction error.

3.2. Theoretical risk bound

We define the classification model f as the composition of a feature extractor h and a probabilistic classifier g, *i.e.*, $f = g \circ h$. The target risk can be decomposed into the risk on the known classes and the risk on the unknown class as follows

$$\begin{aligned}
&\int P^t(x,y)\ell(f(x),y)dxdy \\
&= \int P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \qquad (1)\\
&= \int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + \int_{\mathcal{X}\times unk} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy. \qquad (2)
\end{aligned}$$

Next, we define the probability of target data belonging to the unknown class unk as the unknown probability $P^t(unk|h(x))$, and call the target risk on the unknown class as the target unknown risk. The target unknown risk can be expressed as

$$\begin{aligned}
&\int_{\mathcal{X}\times unk} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&= \int_{\mathcal{X}} P^t(h(x),unk)\ell(g(h(x)),unk)dh(x) \qquad (3)\\
&= \int_{\mathcal{X}} P^t(h(x))\ell(g(h(x)),unk)P^t(unk|h(x))dh(x). \qquad (4)
\end{aligned}$$

Eq. (3) is obtained by marginalizing over the class label y. Eq. (4) decomposes the target unknown joint distribution $P^t(h(x),unk)$ into the product of $P^t(h(x))$ and $P^t(unk|h(x))$: $P^t(h(x),unk) = P^t(h(x))P^t(unk|h(x))$. For convenience, we denote $\mathcal{R}^t(f) = \int P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy$, $\mathcal{R}^{t,k}(f) = \int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy$, and finally $\mathcal{R}^{t,u}(f) = \int_{\mathcal{X}} P^t(h(x))\ell(g(h(x)),unk)P^t(unk|h(x))dh(x)$. Following prior works [5, 21], we adopt a bounded-loss assumption below.

Assumption 1 (Bounded loss). *Assume ℓ is upper bounded,* i.e., *there exists a constant $C > 0$ such that $\forall x \in \mathcal{X},\ \forall y \in \mathcal{Y}, \ell\big(f(x),y\big) \le C$.*

While the commonly used cross-entropy loss for multi-class classification is not upper bounded, we can employ the strategy in Nguyen et al. [21] to ensure that the loss meets Assumption 1. Based on the assumption, we develop the following upper bound for the target risk.

Theorem 1. *Assume the loss $\ell \le C$ for some constant $C > 0$. Then, for any model f, we have*

$$\mathcal{R}^t(f) \le \underbrace{\mathcal{R}^s(f)}_{\text{source risk}} + \underbrace{2CC_{\chi^2}\chi^2\big(P^s,P^{t,k}\big)}_{\chi^2\text{-divergence}} + \underbrace{\mathcal{R}^{t,u}(f)}_{\text{target unknown risk}}. \qquad (5)$$

Here, $\chi^2\big(P^s,P^{t,k}\big) = \chi^2_{\mathcal{X}\times\mathcal{Y}^s}\big(P^s(h(x),y),P^t(h(x),y)\big)$ is the χ^2-divergence for measuring the discrepancy between the joint distributions with respect to the known classes, and C_{χ^2} is a constant associated with the χ^2-divergence.

Proof. Please see Appendix A. □

3.3. Empirical estimation

Eq. (5) indicates that to train a good target model with minimal target risk, we should minimize the source risk, the χ^2-divergence, and the target unknown risk. Because these three terms are unknown in practice, we therefore train the model by minimizing the empirical source risk, the empirical χ^2-divergence, and the empirical target unknown risk, all of which can be estimated from the available data samples.

For the empirical source risk, we compute it as $\frac{1}{n_s}\sum_{i=1}^{n_s}\ell(g(h(x_i^s)),y_i^s)$ by using the labeled source data. For the empirical χ^2-divergence, we obtain it through the following derivations. First, we present the definition of the χ^2-divergence between two joint distributions $P^s(h(x),y)$ and $P^t(h(x),y)$ as follows:

$$\begin{aligned}
&\chi^2_{\mathcal{X}\times\mathcal{Y}^s}\big(P^s(h(x),y),P^t(h(x),y)\big) \\
&= \int_{\mathcal{X}\times\mathcal{Y}^s}\left[\left(\frac{P^s(h(x),y)}{P^t(h(x),y)}\right)^2 - 1\right]P^t(h(x),y)dh(x)dy. \qquad (6)
\end{aligned}$$

This divergence is non-negative and reduces to zero if and only if the two joint distributions are equal on $\mathcal{X}\times\mathcal{Y}^s$.

Then, we reformulate the χ^2-divergence in Eq. (6) as

$$\begin{aligned}
&\chi^2_{\mathcal{X}\times\mathcal{Y}^s}\big(P^s(h(x),y),P^t(h(x),y)\big) \\
&= \max_r \int_{\mathcal{X}\times\mathcal{Y}^s}\left(2\frac{P^s(h(x),y)}{P^t(h(x),y)}r(h(x),y) - r(h(x),y)^2 - 1\right)P^t(h(x),y)dh(x)dy \qquad (7)\\
&= \max_r \int_{\mathcal{X}\times\mathcal{Y}^s}\Big(2P^s(h(x),y)r(h(x),y) - P^t(h(x),y)r(h(x),y)^2\Big)dh(x)dy - 1. \qquad (8)
\end{aligned}$$

Eq. (7) is obtained from Eq. (6) through the equation $p^2 = \max_r(2pr - r^2)$, where $\frac{P^s(h(x),y)}{P^t(h(x),y)}$ is treated as p and the function $r(h(x),y)$ represents r. When $r(h(x),y) = \frac{P^s(h(x),y)}{P^t(h(x),y)}$, Eq. (7) reaches its maximum value, *i.e.*, Eq. (6). Eq. (8) is derived from simple mathematical calculations.

Building on Eq. (8), we can estimate the χ^2-divergence by sample averages.

$$\begin{aligned}
&\chi^2_{\mathcal{X}\times\mathcal{Y}^s}\big(P^s(h(x),y),P^t(h(x),y)\big) \\
&\approx \max_r\left(\frac{2}{n_s}\sum_{i=1}^{n_s} r(h(x_i^s),y_i^s) - \frac{1}{n_t'}\sum_{i=1}^{n_t'} r(h(x_i^t),y_i^t)^2\right) - 1. \qquad (9)
\end{aligned}$$

Here, we presume the labeled target dataset $\mathcal{D}^t = \{(x_i^t,y_i^t)\}_{i=1}^{n_t}$ is available, where $\mathcal{D}^t$ contains n_t' target known classes data and n_t'' target unknown class data, *i.e.*, $n_t = n_t' + n_t''$. We will explain how $\mathcal{D}^t$ is obtained in Section 3.4 "**JAUM Algorithm**". Note that since the χ^2-divergence is with respect to only the known classes here, we use known classes dataset $\{(x_i,y_i)\}_{i=1}^m$, which contains n_s source data and n_t' target known classes data, *i.e.*, $m = n_s + n_t'$, to estimate the empirical χ^2-divergence.

Thirdly, we design the input function $r(h(x),y)$ as a linear-in-parameter function

$$r(h(x),y;\theta) = \sum_{i=1}^{m}\theta_i k(h(x),h(x_i))\delta(y,y_i), \qquad (10)$$

where $\theta = (\theta_1,\cdots,\theta_m)^\top$ represent the function parameters. Here, $k(h(x),h(x_i)) = \exp(-\|h(x)-h(x_i)\|^2/\sigma)$ is the Gaussian kernel with kernel width $\sigma(>0)$, and $\delta(y,y_i)$ is the delta kernel, taking the value 1 when $y = y_i$ and 0 otherwise. One may notice that the function $r(h(x),y;\theta)$ is nonlinear with respect to its input variables $(h(x),y)$, but linear with respect to its parameters θ. This structure provides two key advantages: (i) the nonlinearity enables good approximation of the optimal function $\frac{P^s(h(x),y)}{P^t(h(x),y)}$, and (ii) the linearity turns Eq. (9) into an unconstrained quadratic optimization problem, as detailed in subsequent derivation.

Finally, by plugging the linear-in-parameter function from Eq. (10) into Eq. (9), we derive the empirical χ^2-divergence

$$\begin{aligned}
&\hat{\chi}^2_{\mathcal{X}\times\mathcal{Y}^s}\big(P^s(h(x),y),P^t(h(x),y)\big) \\
&= \max_\theta\left(\frac{2}{n_s}\sum_{i=1}^{n_s} r(h(x_i^s),y_i^s;\theta) - \frac{1}{n_t'}\sum_{i=1}^{n_t'} r(h(x_i^t),y_i^t;\theta)^2\right) - 1 \qquad (11)\\
&= \max_\theta\left(\frac{2}{n_s}1^\top G^s\theta - \frac{1}{n_t'}\theta^\top G^{t\top}G^t\theta\right) - 1 \qquad (12)\\
&= \max_\theta\left(2b^\top\theta - \theta^\top H\theta\right) - 1 \qquad (13)\\
&= 2b^\top\hat{\theta} - \hat{\theta}^\top H\hat{\theta} - 1. \qquad (14)
\end{aligned}$$

In Eq. (12), $1^\top$ is an n_s dimensional row vector with elements 1, G^s is an $n_s\times m$ matrix whose (i,j)-th element $g_{ij}^s = k(h(x_i^s),h(x_j))\delta(y_i^s,y_j)$, and G^t is an $n_t'\times m$ matrix whose (i,j)-th element $g_{ij}^t = k(h(x_i^t),h(x_j))\delta(y_i^t,y_j)$. Eq. (13) introduces two symbols $b = \frac{1}{n_s}G^{s\top}1$, $H = \frac{1}{n_t'}G^{t\top}G^t$. By solving the unconstrained quadratic optimization problem in Eq. (13), we obtain

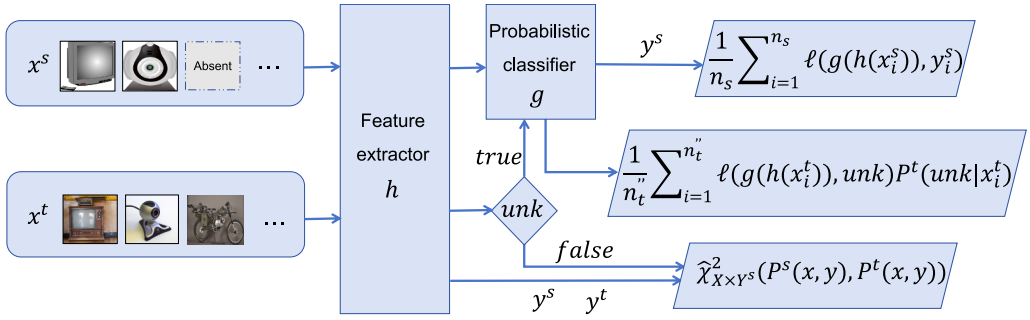


Fig. 1. Illustration of the proposed JAUM algorithm with a feature extractor h and a probabilistic classifier g. Source data x^s are processed by h and g, together with source label y^s, to compute the empirical source risk $\frac{1}{n_s}\sum_{i=1}^{n_s} \ell(g(h(x_i^s)), y_i^s)$. Target data x^t are processed by h to obtain target features $h(x^t)$. The target features $h(x^t)$ whose label are belong to unknown class are further processed by g to computed the empirical target unknown risk $\frac{1}{n_t''}\sum_{i=1}^{n_t''} \ell(g(h(x_i^t)), unk)P^t(unk|x_i^t)$, where the target unknown probability $P^t(unk|x^t)$ is obtained from $g(h(x^t))$. The target features $h(x^t)$ whose label belong to known classes, together with the source features $h(x^s)$, source labels y^s and target labels y^t, are used to estimate the empirical χ^2-divergence $\hat{\chi}^2_{X\times Y^s}\left(P^s(h(x), y), P^t(h(x), y)\right)$.

an analytical solution $\hat{\theta} = (H + \epsilon I)^{-1} b$. Here, to guarantee numerical invertibility of H, we introduce a regularization term ϵI, where $\epsilon = 0.01$ is a small positive value and I denotes the identity matrix.

Algorithm 1 Empirical Estimation of χ^2-divergence.

Input: Labeled source dataset D^s, labeled target dataset D^t.
Output: Estimated value $\hat{\chi}^2_{X\times Y^s}(P^s(h(x), y), P^t(h(x), y))$.
1: Construct the kernel matrices G^s, G^t from D^s, D^t.
2: Compute b, H, and $\hat{\theta}$.
3: Obtain $\hat{\chi}^2_{X\times Y^s}\left(P^s(h(x), y), P^t(h(x), y)\right)$ via Eq. (14).

For clarity, we outline the empirical estimation of χ^2-divergence in Algorithm 1. The main computational cost of Algorithm 1 comes from the construction of kernel matrices and the subsequent matrix inverse. Concretely, computing the kernel matrices G^s and G^t costs $O(n_s m + n_t' m)$, computing the vector b and the matrix H costs $O(n_s m + n_t' m^2)$, and solving $\hat{\theta} = (H + \varepsilon I)^{-1} b$ costs $O(m^3)$, where $m = n_s + n_t'$. Hence, the overall time complexity of Algorithm 1 is $O(m^3)$.

For the empirical target unknown risk, we use n_t'' target unknown class data to compute it as $\frac{1}{n_t''}\sum_{i=1}^{n_t''} \ell(g(h(x_i^t)), unk)P^t(unk|h(x_i^t))$. Since the probability that the target data x_i^t belongs to the unknown class *unk*, *i.e.*, $P^t(unk|h(x_i^t))$, is still unknown, we propose to compute it as follows. We know that the model f includes a probabilistic classifier g, which takes the extracted features as input and produces $K+1$ outputs $\{P(y = k|h(x))\}_{k=1}^{K+1}$. The first $1 \sim K$ outputs reflect the probability that the target data belongs to the $1 \sim K$ known classes, and the last $K+1$ output $P(y = K+1|h(x))$ reflects the probability that the target data belongs to the unknown class. Hence, we propose to compute $P^t(unk|h(x_i^t))$ as $P(y = K+1|h(x_i^t))$, which we believe is reasonable.

Remark 1. *Notably, there are two main reasons for choosing the χ^2-divergence in our algorithm. First, the χ^2-divergence is a suitable metric for measuring the joint probability distribution discrepancy, which has been successfully adopted in previous transfer learning works [30,31]. Second, the estimation of χ^2-divergence can be reformulated as an unconstrained quadratic optimization problem with an analytic solution, which enables stable estimation and yields superior empirical performance. Concretely, we discuss the benefits of joint distribution alignment in "Related work" section and empirically validate them in "**Analysis of χ^2-divergence-based Joint Distribution Alignment**" from Subsection 4.4 "Analysis". Metrics such as MMD, which are typically used for measuring the marginal or conditional distribution discrepancy, are therefore not suitable in our algorithm for joint distribution alignment. Among metrics that are applicable for measuring the joint distribution discrepancy, the Wasserstein distance requires solving an optimal transport problem and thus incurs high computational cost. The KL and Jensen-Shannon (JS) divergences do not admit convenient analytic solutions in our method and typically require complex adversarial training processes that may introduce instability [30]. Besides, in "**Analysis of χ^2-divergence-based Joint Distribution Alignment**" from Subsection 4.4 "Analysis", we implement two variants: JAUM (KL) and JAUM (JS), which replace the χ^2-divergence with the KL-divergence and JS-divergence. Experiments show that the χ^2-divergence is more effective than the two divergences in our algorithm. Based on these considerations, we adopt the χ^2-divergence.*

3.4. JAUM algorithm

Combining the empirical source risk, the empirical χ^2-divergence, and the empirical target unknown risk, we formulate the optimization problem of JAUM as

$$\min_{g,h} \frac{1}{n_s}\sum_{i=1}^{n_s} \ell(g(h(x_i^s)), y_i^s) + \lambda_1 \hat{\chi}^2_{X\times Y^s}\left(P^s(h(x), y), P^t(h(x), y)\right) + \frac{\lambda_2}{n_t''}\sum_{i=1}^{n_t''} \ell(g(h(x_i^t)), unk)P^t(unk|x_i^t). \quad (15)$$

Here, ℓ is the cross-entropy loss, and $\lambda_1, \lambda_2 (>0)$ are trade-off parameters.

We solve optimization problem (15) by minibatch SGD algorithm. In each iteration, we draw two minibatches D_b^s, D_b^t from the labeled source dataset $D^s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and the labeled target dataset $D^t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$, and then use these two minibatches to compute the objective function in (15). Notably, the labeled target dataset D^t is a pseudo labeled dataset. This dataset is obtained by first pre-training a model and then using the pre-trained model to predict pseudo labels $\{y_i^t\}_{i=1}^{n_t}$ for the unlabeled target dataset $D^u = \{x_i^t\}_{i=1}^{n_t}$. Since the initial pseudo labels may not be reliable, they are iteratively updated by the model in the formal training phase to enhance their reliability. Such iterative pseudo-labeling practice is common [24,27,39,40], and theoretical justification for this practice is provided by the work of Chen et al. [41]. In Section 4.4 "**Analysis of Pseudo Labels**", we analytically assess the feasibility of the pseudo labels in our algorithm. For clarity, we illustrate the network architecture of our JAUM algorithm in Fig. 1.

We now elaborate on the pre-training phase for generating the initial pseudo labels, including both the known and unknown classes. Inspired by the work of Jang et al. [14], target data with entropy values exceeding the maximum entropy observed in the source data are regarded as the unknown class data. By combining these target data (*i.e.*, unknown class data) with the source data (*i.e.*, known classes data), a model is trained to recognize the unknown class data and classify the known classes data through ERM. The optimization objective is

$$\min_{g,h} \frac{1}{n_s}\sum_{i=1}^{n_s} \ell(g(h(x_i^s)), y_i^s) + \frac{\lambda_3}{n_t''}\sum_{i=1}^{n_t''} \ell(g(h(x_i^t)), unk), \quad (16)$$

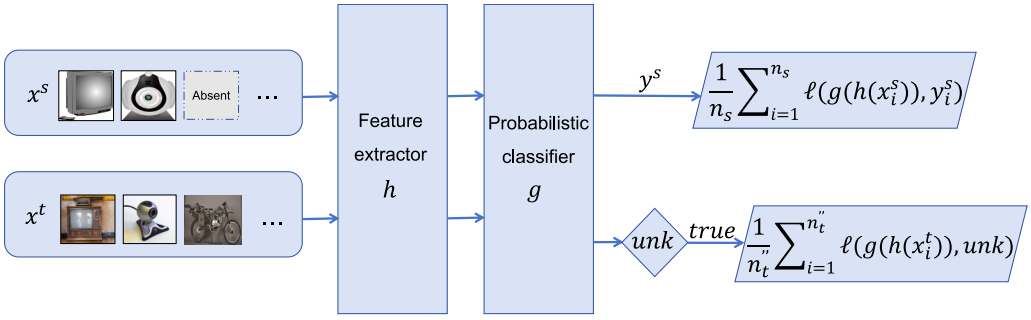


Fig. 2. Illustration of the pre-training phase with a feature extractor h and a probabilistic classifier g. Source data x^s are passed through h and g, together with source label y^s, to compute the empirical source risk $\frac{1}{n_s}\sum_{i=1}^{n_s} \ell(g(h(x_i^s)), y_i^s)$. Target data x^t are processed by h and g, and the entropy-based discrimination is applied to identify the target data that belong to the unknown class, which are then used to estimate $\frac{1}{n_t''}\sum_{i=1}^{n_t''} \ell(g(h(x_i^t)), unk)$.

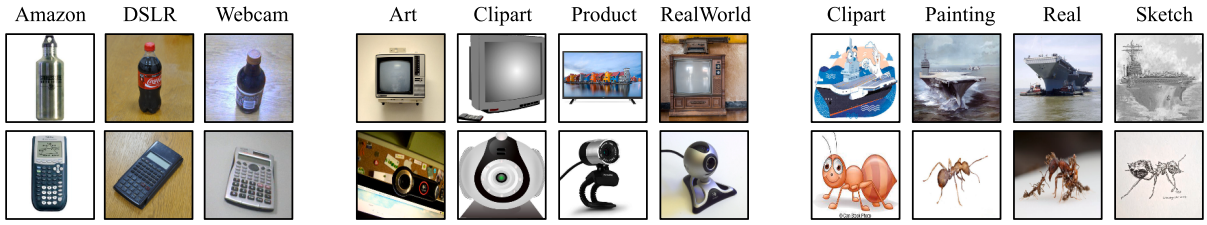


Fig. 3. Example images from Office [42], OfficeHome [43], and DomainNet [44] datasets. The same category across different domains shows significant variations in visual appearance.

where $\lambda_3(>0)$ is a trade-off parameter. Notably, since the entropy-based method is coarse-grained, the pre-trained model may produce sub-optimal pseudo labels. However, as demonstrated in Section 4.4 "**Analysis of Pseudo Labels**", our algorithm exhibits robustness to these pseudo label variations, and these pseudo labels are refined as training progresses. For clarity, we illustrate the pre-training phase in Fig. 2. Algorithm 2 provides a detailed procedure for our JAUM algorithm. Algorithm 2 contains the pre-training stage and the formal training stage. The pre-training stage uses a standard training routine and its complexity is $O(E(n_s + n_t''))$, where E is the number of pre-training epochs. The formal training stage relies on mini-batch SGD. Each iteration computes the empirical χ^2-divergence, the empirical source risk, and the empirical target unknown risk. The formal training cost is $O(Tm^3)$, where T is the number of formal training iterations. Consequently, the overall time complexity of Algorithm 2 is $O\big(E(n_s + n_t'') + Tm^3\big)$.

Remark 2. *Our core training objective is theory-driven. We minimize the empirical source risk, the empirical χ^2-divergence, and the empirical target unknown risk. These terms are grounded in the theoretical risk bound in Theorem 1. In practice, computing the empirical objectives requires the initial target pseudo labels, which are obtained via an entropy-based pre-training procedure. The pre-training relies on a heuristic approach inspired by the work of Jang et al. [14]. Consequently, the full algorithm is a hybrid algorithm: a principled, theory-driven formal training phase built on the heuristic initialization. This hybrid design makes our algorithm more theoretical than approaches that rely solely on heuristic methods to separate the target unknown distribution, but also exposes our algorithm to the pseudo label drift. In the paragraph "**Analysis of Pseudo Labels**" from Section 4.4, we perform feasibility analysis of the pseudo labels to show that our algorithm is robust to different initial pseudo labels and can refine them as training progresses.*

4. Experiments

We evaluate the performance of our JAUM algorithm on three standard benchmark datasets. The following subsections detail the datasets, implementation setups, and experimental results.

4.1. Datasets

Fig. 3 summarizes the three benchmark datasets used in our experiments. **Office** [42] includes 3 domains: Amazon (A), DSLR (D), and Webcam (W) with 31 classes. Following Saito et al. [7], we define the first 10 classes, listed alphabetically, as known classes and the last 11 classes as the uniform unknown class. **OfficeHome** [43] contains 4 domains: Art (A), Clipart (C), Product (P), and RealWorld (R) with 65 classes. Following Liu et al. [12], we define the first 25 classes, listed alphabetically, as known classes and the remaining classes as the uniform unknown class. **DomainNet** [44] contains 6 domains with 345 classes.

Algorithm 2 JAUM Algorithm.

Input: Labeled source dataset D^s and unlabeled target dataset D^u.
Output: The trained model $f = g \circ h$.

Pre-training Phase
for b in $1:B$ **do**
  Sample minibatches D_b^s, D_b^u from datasets D^s, D^u.
  Use entropy to recognize the unknown class data.
  Calculate the objective function in Eq. (16).
  Take a gradient step to update the model's parameters.
end for
Predict initial pseudo labels to obtain labeled target dataset D^t.
Formal Training Phase
while training does not end **do**
  for b in $1:B$ **do**
    Draw minibatches D_b^s, D_b^t from datasets D^s, D^t.
    Use these minibatches to run Algorithm 1 to obtain the empirical χ^2-divergence, and compute the empirical source risk and empirical target unknown risk in Eq. (15).
    Take a gradient step to update the model's parameters.
  end for
  Update D^t by current model.
end while

Table 1
Hyperparameter values used in our experiments.

Dataset	λ_1	λ_2	λ_3	η_0 of h	η_0 of g
Office31	1.0	1.0	0.1	0.0001	0.001
OfficeHome	0.1	0.1	0.1	0.001	0.01
DomainNet	0.1	0.1	0.1	0.001	0.01

Since some domains and classes are noisy, we follow Saito et al. [45] to select 4 domains: Clipart (C), Painting (P), Real (R), and Sketch (S) with 126 classes. Inspired by the practice in prior works [31,40], we define the first 40 classes, listed alphabetically, as known classes and the remaining classes as the uniform unknown class.

4.2. Setup

Experimental Configurations. Our implementation is built on PyTorch and runs on an NVIDIA GeForce RTX 3090. Following the mainstream OSDA works [14,16,27,28], we employ ResNet50 [3] as the feature extractor h, and add a probabilistic classifier g after h. The probabilistic classifier produces $K+1$ probabilistic outputs, where the first $1 \sim K$ outputs are associated with the $1 \sim K$ known classes and the last $K+1$ output is associated with the unknown class (for example, 26 outputs for OfficeHome, with the first 1∼25 outputs for the known classes and the last 26 output for the unknown class). We optimize model parameters using minibatch SGD algorithm with a weight decay of 0.0005, a momentum of 0.9, and a batch size of 96. The learning rate η of feature extractor h and classifier g follows an adaptive schedule used in prior works [4,19]: $\eta = \frac{\eta_0}{(1+\alpha p)^\beta}$, where $\alpha = 10$, $\beta = 0.75$, p denotes the training progress, and η_0 is the initial learning rate. The pre-training phase runs for 30 epochs and the formal training phase runs for up to 30,000 steps. Other detailed hyperparameter choices are reported in Table 1.

Comparison Methods. We compare our JAUM algorithm against OSBP [7], STA [12], UAN [37], ROS [16], MTS [9], UADAL [14], and ANNA [28]. Since our experimental setup aligns with these methods, we directly use the experimental results documented in their original papers. For methods lacking published results on specific datasets, we conduct experiments using their official source codes under the experimental setup described above, and report their best results.

4.3. Results

The results for datasets Office, OfficeHome, and DomainNet are presented in Tables 2–4, respectively. For clarity and ease of comparison, the highest HOS on each OSDA task is highlighted in **bold**, and the second-highest is underlined. Observing from Tables 2–4, it is evident that our algorithm obtains nice results. It achieves the highest average HOS of 89.08% on Office, 67.33% on DomainNet, and obtains the second-highest average HOS of 70.05% on OfficeHome. Notably, on the large-scale and challenging DomainNet dataset, our algorithm surpasses the second-highest method, ANNA, in terms of average HOS by nearly 5% (67.33% vs 62.31%). We also find that some methods like UADAL and ANNA achieve commendable experimental results. However, since our algorithm solves the two key challenges in OSDA by the joint distribution alignment and the unknown risk minimization, instead of the marginal or class-conditional distribution alignment and the heuristic methods, the experimental results of our JAUM algorithm are better than those of the comparison methods. To sum up, the results indicate that our algorithm is more effective in addressing the OSDA problem than the comparison methods.

4.4. Analysis

Statistical Test. To check whether our algorithm significantly outperforms other methods, we conduct the Wilcoxon signed-rank tests [19] using the HOS results from Tables 2–4. The test statistic $\mathcal{T}$ is computed between our JAUM and comparison methods, with results summarized in Table 5. At a significance level of $\alpha = 0.05$ with $N = 30$ tasks, all $\mathcal{T}$ values are below the critical threshold of 137. This shows that our JAUM algorithm is statistically superior to the comparison methods.

Feature Visualization. We employ t-SNE [46] to visualize the extracted features from OSBP [7], UADAL [14], ANNA [28], and our JAUM on the task: Clipart → Product, and show the results in Fig. 4. In the figures, the source data are represented in red, the target known classes data in purple, and the target unknown class data in blue. Clearly, compared with other methods, our JAUM algorithm better aligns the source data to the target known classes data, benefiting from our joint distribution alignment. Furthermore, our algorithm also better separates the target unknown class data, benefiting from our unknown risk minimization. To some extent, this explains why our algorithm outperforms the compared methods.

Analysis of χ^2-divergence-based Joint Distribution Alignment. To demonstrate the advantage of χ^2-divergence-based joint distribution alignment for OSDA, we design four variants of our algorithm: JAUM (marginal), JAUM (class-conditional), JAUM (KL), and JAUM (JS). The first two variants align the marginal and class-conditional distributions respectively, and the latter two replace the χ^2-divergence with KL-divergence and JS-divergence in joint distribution alignment. We conduct experiments on OfficeHome and show the results in Table 6. Clearly, regarding the average HOS, all variants achieve commendable results but perform less superior than the original JAUM with χ^2-divergence-based joint distribution alignment. This empirically demonstrates that, for tackling the OSDA problem, (i) aligning the joint distributions is experimentally better than aligning the marginal or the class-conditional distributions, and (ii) χ^2-divergence is more effective than JS-divergence and KL-divergence in our algorithm.

Ablation Study. We conduct an ablation study on the tasks from OfficeHome to quantify the contribution of each component in our algorithm. We evaluate five components by removing them one at a time: (i) pre-training phase, (ii) entropy-based objective, (iii) χ^2-divergence, (iv) target unknown risk, and (v) pseudo label update strategy. Table 7 lists OS*, UNK and HOS for the full algorithm and for each ablated variant. The results show that removing different components produces distinct failure modes. To be specific, removing the pre-training phase or the entropy-based objective prevents the generation of reliable initial pseudo labels. Consequently, the final trained model classifies the known classes data well but fails to recognize the unknown class data, yielding high OS* but zero UNK and HOS. Removing the χ^2-divergence degrades the model's classification ability for known classes data and biases the model toward predicting unknown class data, which increases UNK while reducing OS*. Excluding the target unknown risk prevents the model from recognizing the unknown class data. Disabling the pseudo label update strategy lowers pseudo label accuracy and reduces the final HOS. Overall, these observations indicate that each algorithmic component is necessary and that the full combination is required to achieve superior OSDA results.

Analysis of Pseudo Labels. We analyze the robustness of our algorithm to different initial pseudo labels and its ability to refine pseudo labels as the training progresses. We use the HOS metric as a proxy for pseudo label accuracy, where a higher HOS value indicates better accuracy. Specifically, we monitor the variation of HOS during (i) the pre-training phase and (ii) the formal training phase. The formal training phase uses different initial pseudo labels obtained from different pre-trained models, which are trained with 20, 25, and 30 epochs, respectively. We select tasks A→C, C→R, and P→R from OfficeHome. Fig. 5 shows that the pseudo label accuracy is very different at different pre-training epochs. Fig. 6 shows that, regardless of the different initial pseudo label accuracy, the HOS tends to improve as the training progresses and eventually converges, achieving comparable final HOS. Note that, the initial drop in Fig. 6 is due to the reason that the pre-training and formal training phases have distinct optimization objec-

Table 2
OS*, UNK, and HOS (%) for the OSDA tasks on Office.

Method	A→D			A→W			D→A			D→W			W→A			W→D			Avg		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [7]	90.50	75.50	82.40	86.80	79.20	82.70	76.10	72.30	75.10	97.70	96.70	97.20	73.00	74.40	73.70	99.10	84.20	91.10	87.20	80.40	83.70
STA [12]	91.00	63.90	75.00	86.70	67.60	75.90	83.10	65.90	73.20	94.10	55.50	69.80	66.20	68.00	66.10	84.90	67.80	75.20	84.30	64.80	72.60
UAN [37]	95.60	24.40	38.90	95.50	31.00	46.80	93.50	53.40	68.00	99.80	52.50	68.80	94.10	38.80	54.90	81.50	41.40	53.00	93.40	40.30	55.10
ROS [16]	87.50	77.80	82.40	88.40	76.70	82.10	74.80	81.20	77.90	99.30	93.00	96.00	69.70	86.60	77.20	100.0	99.40	**99.70**	86.60	85.80	85.90
UADAL [14]	85.60	90.40	**87.90**	85.50	95.10	**90.10**	74.20	87.80	80.50	98.70	97.70	98.20	65.60	87.80	75.10	99.30	99.40	99.40	84.82	93.03	88.53
MTS [9]	96.80	48.40	64.53	98.20	59.70	74.26	99.50	81.90	**89.85**	100.0	87.90	93.56	92.00	65.60	76.59	91.90	67.70	77.97	96.40	68.53	79.46
ANNA [28]	93.20	76.10	83.80	82.80	88.40	85.50	75.40	91.10	82.50	99.40	99.60	**99.50**	76.00	87.90	81.60	100.0	96.80	98.40	87.80	89.98	88.55
JAUM (ours)	82.61	86.70	84.61	83.08	89.14	86.00	81.78	91.30	86.28	93.63	94.78	94.78	78.45	89.62	**83.67**	98.94	99.13	99.13	86.42	91.78	**89.08**

Table 3
OS*, UNK, and HOS (%) for the OSDA tasks on OfficeHome.

Method	A→C			A→P			A→R			C→A			C→P			C→R		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [7]	50.20	61.10	55.10	71.80	59.80	65.20	79.30	67.50	72.90	59.40	70.30	64.30	67.00	62.70	64.70	72.00	69.20	70.60
STA [12]	46.00	72.30	55.80	68.00	48.40	54.00	78.60	60.40	68.30	51.40	65.00	57.40	61.80	59.10	60.40	67.00	66.70	66.80
UAN [37]	62.40	0.00	0.00	81.10	0.00	0.00	88.20	0.00	0.00	70.50	0.00	0.00	74.00	0.00	0.00	80.60	0.10	0.20
ROS [16]	50.60	74.10	60.10	68.40	70.30	69.30	75.80	77.20	76.50	53.60	65.50	58.90	59.80	71.60	65.20	65.30	72.20	68.60
UADAL [14]	54.90	74.70	63.20	69.10	72.50	70.80	81.30	73.70	**77.40**	53.50	80.50	64.20	62.10	78.80	69.50	69.10	78.30	73.40
MTS [9]	58.62	62.10	60.31	63.99	57.39	60.51	67.39	53.09	59.40	56.51	53.29	54.85	50.95	57.74	54.13	74.85	50.63	60.40
ANNA [28]	61.40	78.70	**69.00**	68.30	79.90	73.70	74.10	79.70	76.80	58.00	73.10	64.70	64.20	73.60	68.60	66.90	80.20	73.00
JAUM (ours)	61.17	70.26	65.40	66.89	89.04	**76.39**	84.41	67.79	75.20	60.37	71.45	**65.44**	69.90	74.11	**71.95**	77.27	71.80	**74.43**

Method	P→A			P→C			P→R			R→A			R→C			R→P			Avg		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [7]	59.10	68.10	63.20	44.50	66.30	53.20	76.20	71.70	73.90	66.10	67.30	66.70	48.00	63.00	54.50	76.30	68.60	72.30	64.16	66.30	64.72
STA [12]	54.20	72.40	61.90	44.20	67.10	53.20	76.20	64.30	69.50	67.50	66.70	67.10	49.90	61.10	54.50	77.10	55.40	64.50	61.83	63.24	61.12
UAN [37]	73.70	0.00	0.00	59.10	0.00	0.00	84.00	0.10	0.20	77.50	0.10	0.20	66.20	0.00	0.00	85.00	0.10	0.10	75.19	0.03	0.06
ROS [16]	57.30	64.30	60.60	46.50	71.20	56.30	70.80	78.40	74.40	67.00	70.80	68.80	51.50	73.00	60.40	72.00	80.00	75.70	61.55	72.38	66.23
UADAL [14]	50.50	83.70	63.00	43.40	81.50	56.60	71.60	83.10	76.90	66.70	78.60	**72.10**	51.10	74.50	60.60	77.40	76.20	76.80	62.56	78.01	68.71
MTS [9]	80.31	48.67	60.61	82.22	53.42	**64.76**	57.05	61.40	59.15	72.88	52.51	61.04	81.80	43.25	56.58	70.89	51.95	59.96	68.12	53.79	59.31
ANNA [28]	63.00	70.30	**66.50**	54.60	74.80	63.10	74.30	78.90	76.60	66.10	77.30	71.30	59.70	73.10	**65.70**	76.40	81.00	**78.70**	65.58	76.72	**70.64**
JAUM (ours)	56.55	78.33	65.68	46.10	75.43	57.22	77.76	80.09	**78.91**	68.11	68.31	68.21	58.66	73.20	65.13	82.23	71.85	76.69	67.45	74.30	70.05

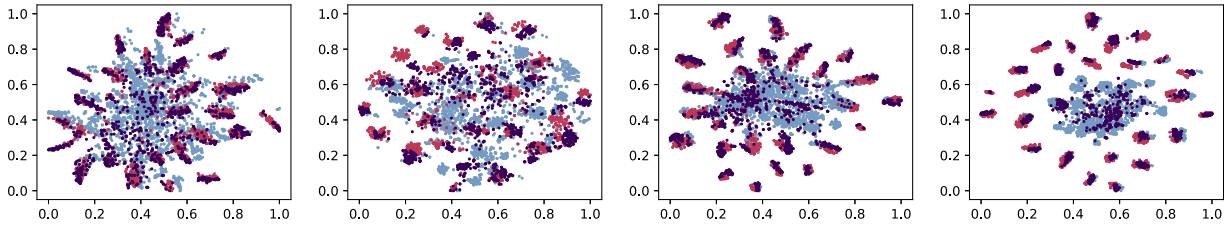
Table 4
OS*, UNK, and HOS (%) for the OSDA tasks on DomainNet.

Method	C→P			C→R			C→S			P→C			P→R			P→S		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [7]	51.78	55.71	53.68	67.52	59.44	63.23	43.25	58.25	49.64	52.06	63.09	57.05	74.21	60.67	66.76	46.48	61.63	53.00
STA [12]	53.56	46.24	49.63	40.46	52.25	45.61	46.86	72.91	57.05	42.29	50.22	45.91	32.24	55.64	40.82	41.18	41.34	41.26
UAN [37]	58.29	10.64	17.99	67.62	10.85	18.70	56.55	10.75	18.06	53.69	7.51	13.17	64.80	8.76	15.44	58.83	6.44	11.61
ROS [16]	44.40	73.11	55.25	64.71	30.52	41.47	39.22	73.26	51.09	52.77	78.65	63.16	66.46	0.00	0.00	43.69	74.67	55.13
UADAL [14]	60.20	65.90	62.90	72.10	75.30	73.70	62.10	62.90	62.50	60.70	71.40	65.60	77.40	73.80	**75.60**	53.60	61.50	57.30
MTS [9]	40.19	62.90	49.05	43.81	63.49	51.84	47.74	62.13	53.99	47.26	56.62	51.52	47.38	58.44	52.33	58.86	56.68	57.75
ANNA [28]	49.45	69.19	57.68	60.79	71.79	65.83	52.36	65.53	58.21	56.47	69.30	62.23	69.73	68.46	69.09	51.53	64.03	57.11
JAUM (ours)	62.63	71.09	**66.59**	75.28	74.13	**74.70**	58.61	71.20	**64.30**	65.54	73.40	**69.25**	78.17	69.05	73.33	56.41	80.61	**66.37**

Method	R→C			R→P			R→S			S→C			S→P			S→R			Avg		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
OSBP [7]	53.93	70.56	61.13	57.53	66.42	61.65	40.97	69.64	51.59	51.88	53.75	52.80	46.60	57.31	51.41	65.22	57.18	60.94	54.29	61.14	56.91
STA [12]	58.71	34.56	43.51	39.28	61.97	48.08	40.88	44.60	42.66	49.68	70.36	58.24	50.45	51.38	50.91	37.53	75.91	50.23	44.43	54.78	47.83
UAN [37]	44.45	25.50	32.41	44.42	24.91	31.92	44.48	22.34	29.74	44.59	19.67	27.29	44.18	28.55	34.69	44.45	25.50	32.41	52.20	16.78	23.62
ROS [16]	61.14	66.86	63.87	68.06	13.53	22.57	46.11	64.76	53.86	51.20	69.53	58.98	43.59	73.25	54.66	56.29	22.32	31.97	53.14	53.37	46.00
UADAL [14]	52.40	40.10	45.40	31.10	91.30	46.30	45.60	41.70	43.60	63.10	72.20	67.30	59.00	62.10	60.50	66.70	76.50	**71.30**	58.67	66.22	61.00
MTS [9]	67.02	45.86	54.46	47.98	60.30	53.44	45.63	64.31	53.38	53.59	70.81	61.01	57.59	53.23	55.32	60.70	50.50	55.13	51.48	58.77	54.10
ANNA [28]	65.68	74.22	69.69	62.19	65.67	63.88	51.77	64.86	**57.58**	58.09	71.03	63.91	53.10	63.26	57.74	63.80	65.68	64.72	57.91	67.75	62.31
JAUM (ours)	66.34	74.68	**70.26**	60.15	74.53	**66.57**	42.54	66.36	51.84	67.43	67.63	**67.53**	61.86	73.76	**67.29**	71.26	68.66	69.93	63.85	72.09	**67.33**

Table 5
Wilcoxon signed-rank test results.

Method	OSBP [7]	STA [12]	UAN [37]	ROS [16]	UADAL [14]	MTS [9]	ANNA [28]
$\mathcal{T}$ value	6	0	0	21	131	7	128



(a) OSBP (b) UADAL (c) ANNA (d) JAUM

Fig. 4. T-SNE visualization of the extracted features from OSBP [7], UADAL [14], ANNA [28], and our JAUM on the task: Clipart → Product. The source data, target known classes data, and target unknown class data are represented in red, purple, and blue, respectively. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 6
Analysis of χ^2-divergence-based Joint Distribution Alignment on OfficeHome.

Method	Avg		
	OS*	UNK	HOS
JAUM (marginal)	55.55	77.97	64.39
JAUM (class-conditional)	65.38	73.37	68.73
JAUM (JS)	61.50	80.41	69.17
JAUM (KL)	62.62	80.77	69.40
JAUM (ours)	67.45	74.30	70.05

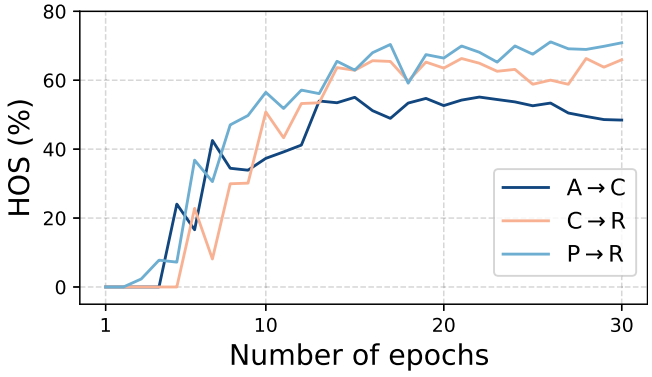


Fig. 5. Variation curves of HOS (%) of the pre-trained phase at different epochs on tasks from OfficeHome.

Table 7
Ablation study on the components of our JAUM algorithm.

Variants	C→P			S→C		
	OS*	UNK	HOS	OS*	UNK	HOS
JAUM	62.63	71.09	**66.59**	67.43	67.63	**67.53**
Remove pre-training phase	70.62	0.00	0.00	77.21	0.00	0.00
Remove entropy-based objective	71.88	0.00	0.00	76.88	0.00	0.00
Remove χ^2-divergence	2.27	99.34	4.43	10.89	98.78	19.61
Remove target unknown risk	72.04	0.00	0.00	77.07	0.00	0.00
Remove pseudo label update strategy	55.75	72.28	62.95	55.64	77.00	64.60

tives. To more precisely quantify the effect of different initial pseudo labels, we report OS*, UNK, and HOS for both the initial pseudo labels and the labels predicted by the final trained models. The results are summarized in Table 8. Although the initial pseudo labels differ in OS*, UNK, and HOS, our algorithm improves most metrics and achieves comparable final HOS across different initializations. Overall, these empirical results confirm that our algorithm is robust to different initial pseudo labels and can maintain stable performance with sub-optimal initializations. Besides, the pseudo labels are iteratively refined during the formal training phase, progressively approximating the true labels.

Hyperparameter Sensitivity. We analyze the sensitivity of our algorithm to hyperparameters $\lambda_1, \lambda_2, \lambda_3$. We perform a grid search over the candidate values $\{0.01, 0.05, 0.1, 0.5, 1.0, 5.0\}$. In preliminary experiments, we observed that λ_1 and λ_2 should be set to the same value. Therefore, we set $\lambda_1 = \lambda_2$ during the search. Concretely, we first fix $\lambda_3 = 0.1$ and search $\lambda_1 = \lambda_2$ over the grid, and then fix $\lambda_1 = \lambda_2 = 0.1$ and search λ_3. We use HOS as the selection criterion and plot HOS for OfficeHome tasks in Fig. 7. We observe that our algorithm performs well when $\lambda_1, \lambda_2 \in \{0.05, 0.1, 0.5, 1.0\}$ and $\lambda_3 = 0.1$. Accordingly, we recommend setting $\lambda_1 = \lambda_2 = \lambda_3 = 0.1$ for OfficeHome. We applied the same procedure to Office31 and DomainNet, and the values for these datasets are summarized in "Experimental Configurations" in Subsection 4.2.

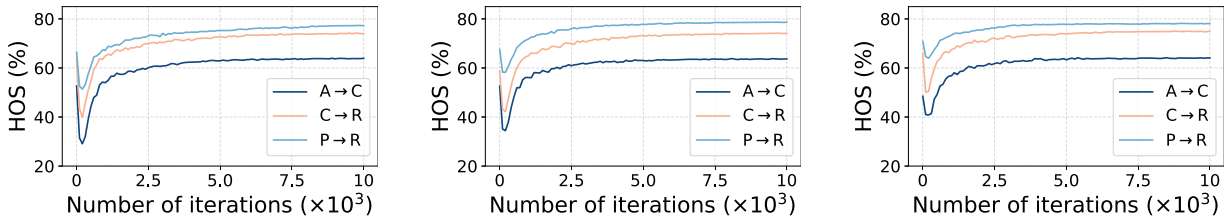
Kernel Sensitivity. To evaluate sensitivity of the kernel bandwidth used in Eq. (10). We sweep the bandwidth over [800, 900, 1000, 1100, 1200] and report the resulting HOS on Office31 tasks in Table 8. Obviously, HOS remains stable across this range, indicating our algorithm is robust to different kernel bandwidths.

Robustness Analysis. We analyze the robustness of our algorithm to varying unknown class distributions. Under the OSDA setting, all unknown classes data are treated as a uniform unknown class. We vary the number of constituent unknown classes within this uniform class, and plot the HOS of our algorithm with comparison methods on task C→P from OfficeHome. Fig. 9 shows that our method consistently outperforms the comparison methods across these scenarios. This suggests that our method is robust to varying unknown class distributions.

Complexity-Performance Analysis. To comprehensively analyze the complexity-performance of our algorithm. We tested the training time and GPU memory of our algorithm compared with OSBP, UADAL and ANNA on task C→P from OfficeHome. To be specific, we measure the average training time of 1000 iterations and record peak GPU memory usage for each method using a batch size of 96. As shown in Table 9, our algorithm outperforms other methods in HOS, requiring similar computing resources and computing time.

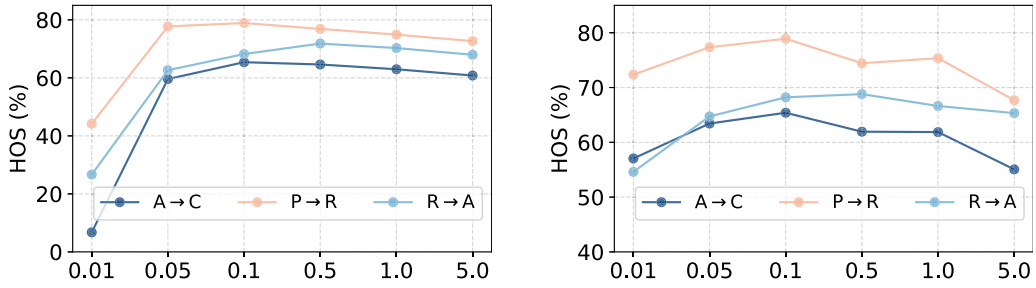
Application to Real-World Datasets. In addition to the benchmark DA datasets, we evaluate our algorithm to the real-world skin disease classification datasets: the PAD-UFES-20 dataset[1] and the ISIC-SkinCancer dataset[2]. The PAD-UFES-20 dataset includes clinical skin lesion images collected via smartphone devices, containing 6 classes. The ISIC-SkinCancer dataset is a skin disease classification dataset collected via professional equipment and has 9 classes. In the OSDA experiment, we use the PAD-UFES-20 dataset as the source domain and the ISIC-SkinCancer dataset as the target domain. The 6 shared classes from both domains are defined as the known classes, and the 3 private classes from the target domain are defined as the uniform unknown class.

[1] https://data.mendeley.com/datasets/zr7vgbcyr2/1
[2] https://www.kaggle.com/datasets/rajivaiml/isic-skin-cancer-dataset



(a) 20 epochs (b) 25 epochs (c) 30 epochs

Fig. 6. Variation curves of HOS (%) of the formal training phase on tasks from OfficeHome. The formal training phase uses different initial pseudo labels obtained from different pre-trained models, which are trained with 20, 25, and 30 epochs, respectively.



(a) λ_1, λ_2 (b) λ_3

Fig. 7. HOS (%) of our algorithm to different hyperparameters λ_1, λ_2, and λ_3.

Table 8
OS*, UNK and HOS (%) for the initial pseudo labels and the labels from the final trained models on tasks from OfficeHome. △ indicates performance gains.

Epochs	A→C						C→R						P→R					
	initial pseudo labels			labels from final trained model			initial pseudo labels			labels from final trained model			initial pseudo labels			labels from final trained model		
	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS	OS*	UNK	HOS
20	46.97	64.16	54.24	58.77	78.07	**63.92**	66.30	60.96	63.52	77.68	78.62	**73.98**	74.38	60.02	66.43	80.76	73.96	**77.20**
△				(+11.80)	(+5.91)	(+9.69)				(+11.39)	(+9.66)	(+10.47)				(+6.38)	(+13.94)	(+10.78)
25	46.30	68.86	52.59	58.16	78.26	**63.64**	70.58	58.43	58.83	76.63	71.88	**74.13**	71.45	64.06	67.55	78.58	78.67	**78.60**
△				(+11.86)	(+9.41)	(+11.05)				(+6.05)	(+21.37)	(+15.31)				(+7.13)	(+14.61)	(+11.07)
30	34.84	79.41	48.43	56.57	73.90	**64.08**	63.72	68.34	65.95	75.03	74.82	**74.92**	64.56	78.55	70.87	75.45	80.91	**80.91**
△				(+21.72)	(-5.50)	(+15.65)				(+11.31)	(+6.48)	(+8.98)				(+10.89)	(+2.36)	(+10.04)

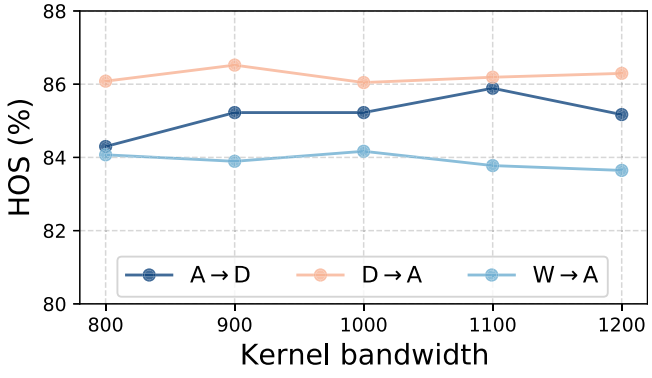


Fig. 8. Kernel sensitivity on tasks from Office31.

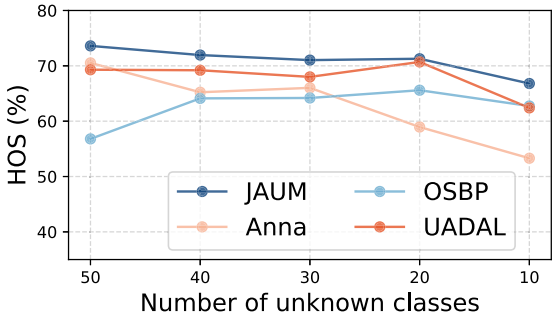


Fig. 9. Robustness analysis to varying unknown classes on task C→P from OfficeHome.

Table 9
Computational efficiency and performance comparison on task C→P from OfficeHome.

Method	speed(s/1000 iter)	Peak GPU Memory (MB)	HOS (%)
OSBP	440	7086	64.70
UADAL	185	15,372	69.50
ANNA	1362	11,264	68.60
JAUM (ours)	306	11,256	**71.95**

Table 10 reports the experimental results. Our findings indicate that the proposed JAUM algorithm is effective on the real-world OSDA task for skin disease classification, and achieves the highest HOS among all the OSDA methods. Compared with our algorithm's results on the benchmark DA datasets in Tables 2–4, we also find that the HOS here is not as high as those in the 3 tables. Based on our preliminary experiments on the two skin disease classification datasets, we conjecture that this is because these two real-world datasets are much more challenging than the

Table 10
OS*, UNK, and HOS (%) for the real-world OSDA task, where the source is the PAD-UFES-20 dataset and the target is the ISIC-SkinCancer dataset.

Method	OSBP [7]	STA [12]	UAN [37]	ROS [16]	UADAL [14]	MTS [9]	ANNA [28]	JAUM (ours)
OS*	29.56	22.28	24.52	17.97	24.10	26.23	42.27	36.15
UNK	22.54	33.55	1.92	46.58	36.10	25.41	26.43	30.37
HOS	25.58	26.78	3.55	25.93	28.90	25.81	30.44	**33.01**

benchmark DA datasets. We believe that by incorporating medical background knowledge, the performance of our algorithm and other OSDA methods can be improved.

5. Conclusion

In this paper, we introduce theory and algorithm to address the OSDA problem. We develop a new theoretical risk bound to show that the model's target risk is constrained by the source risk, the χ^2-divergence between the source joint distribution and the target known joint distribution, and the target unknown risk. Based on this theoretical risk bound, we propose a Joint Distribution Alignment and Unknown Risk Minimization (JAUM) algorithm to train a classification model by minimizing the empirical source risk, the empirical χ^2-divergence, and the empirical target unknown risk. The trained model well recognizes the target unknown classes data as a uniform unknown class, and classifies other target data into the correct known classes. Experimental results demonstrate the effectiveness of our method.

It is worth mentioning that in this work, we follow the mainstream OSDA literature [7,14,28] and regard the (multiple) unknown classes as one uniform unknown class. However, this approach may be limited in large-scale open-set settings that contain many classes and complex intra-unknown semantic diversity (*e.g.*, DomainNet). Thus, in the future work, we plan to extend our method by integrating clustering or specialized classification techniques to distinguish varying numbers of unknown classes. Besides, we plan to employ distributed training to address heavy computational and efficiency challenge in such large-scale open-set scenario. Potential social risks also need to be noted. For instance, in our practical application of skin disease classification, there is a risk of incorrect prediction for the patient. To mitigate this, future work will incorporate medical background knowledge and involve clinicians to improve reliability and ensure safer deployment of our method.

CRediT authorship contribution statement

Lisheng Wen: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Sentao Chen:** Writing – review & editing, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization; **Lin Zheng:** Writing – review & editing, Funding acquisition; **Ping Xuan:** Writing – review & editing, Funding acquisition.

Data availability

Data will be made available on request.

Declaration of competing interest

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

Acknowledgements

We would like to thank our group member, Xiabin Peng, for pointing out some typos in the manuscript during the revision. This work was supported in part by the Guangdong Basic and Applied Basic Research Foundation under Grants 2023A1515012954 and 2023A1515011240, in part by National Natural Science Foundation of China under Grants 62106137 and 62372282, and in part by Shantou University under Grant NTF21035.

Appendix A. Proof of Theorem 1

Proof. First, for the target risk on the known classes, we have

$$\begin{aligned}
&\int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&= \int P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + \int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad - \int_{\mathcal{X}\times\mathcal{Y}^s} P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \qquad (A.1)\\
&\le \int P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + \Big| \int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad - \int_{\mathcal{X}\times\mathcal{Y}^s} P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \Big| \qquad (A.2)\\
&\le \int P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + C\int_{\mathcal{X}\times\mathcal{Y}^s} \left|P^s(h(x),y) - P^t(h(x),y)\right| dh(x)dy \qquad (A.3)\\
&= \int P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy + 2C\mathrm{TV}_{\mathcal{X}\times\mathcal{Y}^s}\left(P^s(h(x),y), P^t(h(x),y)\right) \qquad (A.4)\\
&\le \int P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + 2CC_{\chi^2}\chi^2_{\mathcal{X}\times\mathcal{Y}^s}\left(P^s(h(x),y), P^t(h(x),y)\right). \qquad (A.5)
\end{aligned}$$

Eq. (A.3) is derived from the integration property and the assumption that loss $\ell \le C$ for some constant $C > 0$. Eq. (A.4) introduces the Total Variation (TV) distance

$$\begin{aligned}
&\mathrm{TV}_{\mathcal{X}\times\mathcal{Y}^s}\left(P^s(h(x),y), P^t(h(x),y)\right) \\
&= \frac{1}{2}\int_{\mathcal{X}\times\mathcal{Y}^s} \left|P^s(h(x),y) - P^t(h(x),y)\right| dh(x)dy. \qquad (A.6)
\end{aligned}$$

Eq. (A.5) applies Proposition 6 from the work of Yuan et al. [20], where C_{χ^2} is a constant associated with the χ^2-divergence.

With target unknown risk $\int_{\mathcal{X}} P^t(h(x))\ell(g(h(x)),unk)P^t(unk|h(x))dh(x)$, we derive the following upper bound on the target risk

$$\begin{aligned}
&\int P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&= \int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + \int_{\mathcal{X}\times unk} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy \qquad (A.7)\\
&\le \int P^s(h(x),y)\ell(g(h(x)),y)dh(x)dy \\
&\quad + 2CC_{\chi^2}\chi^2_{\mathcal{X}\times\mathcal{Y}^s}\left(P^s(h(x),y), P^t(h(x),y)\right)
\end{aligned}$$

$$+\int_{\mathcal{X}\times unk} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy. \tag{A.8}$$

Finally, using notations $\mathcal{R}^t(f)=\int P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy$, $\mathcal{R}^{t,k}(f)=\int_{\mathcal{X}\times\mathcal{Y}^s} P^t(h(x),y)\ell(g(h(x)),y)dh(x)dy$, and also $\mathcal{R}^{t,u}(f)=\int_{\mathcal{X}} P^t(h(x))\ell(g(h(x)),unk)P^t(unk|h(x))dh(x)$, we have the bound

$$\mathcal{R}^t(f)\le \underbrace{\mathcal{R}^s(f)}_{\text{source risk}} + \underbrace{2CC_{\chi^2}\chi^2\left(P^s,P^{t,k}\right)}_{\chi^2\text{-divergence}} + \underbrace{\mathcal{R}^{t,u}(f)}_{\text{target unknown risk}}. \tag{A.9}$$

□

References

[1] V.N. Vapnik, Statistical Learning Theory, Wiley-Interscience, 1998.
[2] L. Wasserman, All of Statistics: A Concise Course in Statistical Inference, Vol. 26, Springer, 2004.
[3] 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.
[4] 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.
[5] S. Chen, Z. Hong, M. Harandi, X. Yang, Domain neural adaptation, IEEE Trans. Neural Netw. Learn. Syst. 34 (11) (2023) 8630–8641.
[6] J. Na, H. Jung, H.J. Chang, W. Hwang, Bridging domain spaces for unsupervised domain adaptation, Pattern Recognit. 164 (2025) 111537.
[7] K. Saito, S. Yamamoto, Y. Ushiku, T. Harada, Open set domain adaptation by backpropagation, in: European Conference on Computer Vision, 2018, pp. 153–168.
[8] Z. Fang, J. Lu, F. Liu, J. Xuan, G. Zhang, Open set domain adaptation: theoretical bound and algorithm, IEEE Trans. Neural Netw. Learn. Syst. 32 (10) (2021) 4309–4322.
[9] D. Chang, A. Sain, Z. Ma, Y.-Z. Song, R. Wang, J. Guo, Mind the gap: open set domain adaptation via mutual-to-separate framework, IEEE Trans. Circuits Syst. Video Technol. 34 (6) (2024) 4159–4174.
[10] 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.
[11] J. Liu, H. He, M. Liu, J. Li, K. Lu, Manifold regularized joint transfer for open set domain adaptation, IEEE Trans. Multimed. 25 (2023) 9356–9369.
[12] H. Liu, Z. Cao, M. Long, J. Wang, Q. Yang, Separate to adapt: open set domain adaptation via progressive separation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2922–2931.
[13] T. Shermin, G. Lu, S.W. Teng, M. Murshed, F. Sohel, Adversarial network with multiple classifiers for open set domain adaptation, IEEE Trans. Multimed. 23 (2021) 2732–2744.
[14] J. Jang, B. Na, D.H. Shin, M. Ji, K. Song, I.-c. Moon, Unknown-aware domain adversarial learning for open-set domain adaptation, in: Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 16755–16767.
[15] Q. Wang, F. Meng, T.P. Breckon, Progressively select and reject pseudo-labelled samples for open-set domain adaptation, IEEE Trans. Artif. Intell. 5 (9) (2024) 4403–4414.
[16] S. Bucci, M.R. Loghmani, T. Tommasi, On the effectiveness of image rotation for open set domain adaptation, in: European Conference on Computer Vision, 2020, pp. 422–438.
[17] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J.W. Vaughan, A theory of learning from different domains, Mach. Learn. 79 (2010) 151–175.
[18] L. Zhong, Z. Fang, F. Liu, B. Yuan, G. Zhang, J. Lu, Bridging the theoretical bound and deep algorithms for open set domain adaptation, IEEE Trans. Neural Netw. Learn. Syst. 34 (8) (2023) 3859–3873.
[19] S. Chen, L. Zheng, H. Wu, Riemannian representation learning for multi-source domain adaptation, Pattern Recognit. 137 (2023) 109271.
[20] Z. Yuan, X. Hu, Q. Wu, S. Ma, C.H. Leung, X. Shen, Y. Huang, A unified domain adaptation framework with distinctive divergence analysis, Trans. Mach. Learn. Res. (2022) 1–21.
[21] A.T. Nguyen, T. Tran, Y. Gal, P.H.S. Torr, A.G. Baydin, KL Guided domain adaptation, in: International Conference on Learning Representations, 2022, pp. 1–12.
[22] C.-X. Ren, Y.-W. Luo, D.-Q. Dai, BuresNet: conditional bures metric for transferable representation learning, IEEE Trans. Pattern Anal. Mach. Intell. 45 (4) (2023) 4198–4213.
[23] P. Ge, C.-X. Ren, X.-L. Xu, H. Yan, Unsupervised domain adaptation via deep conditional adaptation network, Pattern Recognit. 134 (2023) 109088.
[24] S. Chen, Multi-source domain adaptation with mixture of joint distributions, Pattern Recognit. 149 (2024) 110295.
[25] A. Farhadi, A. Sharifi, Leveraging meta-learning to improve unsupervised domain adaptation, Comput. J. 67 (5) (2023) 1838–1850.
[26] C. He, H. Shi, X. Liu, J. Li, Interpretable physics-informed domain adaptation paradigm for cross-machine transfer diagnosis, Knowl. Based Syst. 288 (2024) 111499.
[27] S. Chen, P. Xuan, L. He, Open set domain adaptation via known joint distribution matching and unknown classification risk reformulation, IEEE Trans. Neural Netw. Learn. Syst. (2026) 1–14.
[28] W. Li, J. Liu, B. Han, Y. Yuan, Adjustment and alignment for unbiased open set domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2023, pp. 24110–24119.
[29] S. Chen, P. Xuan, Z. Hao, Joint distribution weighted alignment for multi-source domain adaptation via kernel relative entropy estimation, IEEE Trans Multimed. 27 (2025) 6606–6619.
[30] S. Chen, L. Wang, Z. Hong, X. Yang, Domain generalization by joint-product distribution alignment, Pattern Recognit. 134 (2023) 109086.
[31] S. Chen, Joint weight optimization for partial domain adaptation via kernel statistical distance estimation, Neural Netw. 180 (2024) 106739.
[32] S. Chen, Z. Hong, Domain generalization by distribution estimation, Int. J. Mach. Learn. Cybern. 14 (10) (2023) 3457–3470.
[33] S. Chen, L. Chen, Joint-product representation learning for domain generalization in classification and regression, Neural Comput. Appl. 35 (2023) 16509–16526.
[34] 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.
[35] Z. Cai, Y. Huang, T. Zhang, Y. Zheng, D. Yue, Multi-source domain adaptation by causal-guided adaptive multimodal diffusion networks, Int. J. Comput. Vis. 133 (7) (2025) 4623–4645.
[36] J. Tian, J. Zhang, Y. Jiang, S. Wu, H. Luo, S. Yin, A novel generalized source-free domain adaptation approach for cross-domain industrial fault diagnosis, Reliab. Eng. Syst. Saf. 243 (2024) 109891.
[37] K. You, M. Long, Z. Cao, J. Wang, M.I. Jordan, Universal domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2720–2729.
[38] Z. Cai, Y. Huang, T. Zhang, X.-Y. Jing, Y. Zheng, L. Shao, Attention cycle-consistent universal network for more universal domain adaptation, Pattern Recognit. 147 (2024) 110109.
[39] S. Chen, M. Harandi, X. Jin, X. Yang, Domain adaptation by joint distribution invariant projections, IEEE Trans. Image Process. 29 (2020) 8264–8277.
[40] L. Wen, S. Chen, Z. Hong, L. Zheng, Maximum likelihood weight estimation for partial domain adaptation, Inf. Sci. 676 (2024) 120800.
[41] Y. Chen, C. Wei, A. Kumar, T. Ma, Self-training avoids using spurious features under domain shift, in: Advances in Neural Information Processing Systems, vol. 33, 2020, pp. 21061–21071.
[42] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: European Conference on Computer Vision, 2010, pp. 213–226.
[43] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.
[44] 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.
[45] K. Saito, D. Kim, S. Sclaroff, T. Darrell, K. Saenko, Semi-supervised domain adaptation via minimax entropy, in: IEEE International Conference on Computer Vision, 2019, pp. 8050–8058.
[46] L. v. d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Learn. Res. 9 (Nov) (2008) 2579–2605.

Lisheng Wen received the M.S. degree from Shantou University. He is currently pursuing his Ph.D. degree at Chongqing University. His research focuses on generalized domain adaptation problems, including multi-source domain adaptation, partial domain adaptation, and open-set domain adaptation.

Sentao Chen is an Associate Professor at Shantou University and a Researcher in statistical machine learning. His recent research interest is transfer learning. From 2020 to 2025, he has extended the empirical risk minimization principle, proposed the joint distribution matching framework, and developed the kernel statistical distance estimation technique to address the fundamental challenges in transfer learning. These contributions have lead to a series of principled, straightforward, and effective algorithms for various transfer learning problems.

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 at the School of Cyberspace Security, Hainan University, Haikou, China. Her research interests include machine learning, deep learning, graph learning, and medical image processing.

Lin Zheng received his Ph.D. degree in computer science from Wuhan University, China in 2017. From 2017 to 2018, he was a post-doctoral research fellow in the Department of Computer Science at Hong Kong Baptist University. He is currently an Associate Professor in the Department of Computer Science and Technology, Shantou University. His research interests include explainable artificial intelligence and recommender systems. He is a member of both ACM and CCF and has served as a reviewer of some journals such as ACM TIST, TKDD, IEEE TMM and TPAMI.