Domain Adaptation by Joint Distribution Invariant Projections

Sentao Chen, Mehrtash Harandi, *Member, IEEE*, Xiaona Jin, and Xiaowei Yang

***Abstract*—Domain adaptation addresses the learning problem where the training data are sampled from a source joint distribution (source domain), while the test data are sampled from a different target joint distribution (target domain). Because of this joint distribution mismatch, a discriminative classifier naively trained on the source domain often generalizes poorly to the target domain. In this article, we therefore present a Joint Distribution Invariant Projections (JDIP) approach to solve this problem. The proposed approach exploits linear projections to directly match the source and target joint distributions under the L^2-distance. Since the traditional kernel density estimators for distribution estimation tend to be less reliable as the dimensionality increases, we propose a least square method to estimate the L^2-distance without the need to estimate the two joint distributions, leading to a quadratic problem with analytic solution. Furthermore, we introduce a kernel version of JDIP to account for inherent nonlinearity in the data. We show that the proposed learning problems can be naturally cast as optimization problems defined on the product of Riemannian manifolds. To be comprehensive, we also establish an error bound, theoretically explaining how our method works and contributes to reducing the target domain generalization error. Extensive empirical evidence demonstrates the benefits of our approach over state-of-the-art domain adaptation methods on several visual data sets.**

***Index Terms*—L^2-distance, dimensionality reduction, domain adaptation, joint distribution matching, Riemannian optimization.**

I. Introduction

TRADITIONAL supervised learning assumes that the training (source) and test (target) data are drawn from the same joint probability distribution [1]. In practical applications, however, this assumption often does not hold: the training data are sampled from a source joint distribution $P^s(\boldsymbol{x}, y)$, whilst the test data are sampled from a different target joint distribution $P^t(\boldsymbol{x}, y)$ [2]. For example, in object recognition, the training data may be collected with a different camera viewpoint, background, or lighting condition from that of the test data. In remote sensing image analysis, the training and test data may be acquired from different corrections for atmospheric scattering, daylight conditions or chemical composition of the materials. Under such circumstances, a discriminative classifier naively trained on the source data often generalizes poorly to the target data.

Domain adaptation aims at addressing this joint distribution mismatch problem and learning a discriminative model that performs well on the target domain [3], [4]. Recent years have witnessed the successful applications of domain adaptation in computer vision [5]–[12], natural language processing [13]–[17], and more. Generally speaking, domain adaptation can be categorized into the semi-supervised and the unsupervised settings, depending on whether there are labeled data in the target domain. Semi-supervised domain adaptation assumes that the labeled source data, the unlabeled and a small number of labeled target data are available during training [12], [14], whereas unsupervised domain adaptation only presumes that the labeled source data and the unlabeled target data are accessible. In these settings, the source and target joint distributions are assumed to be defined on the same product space $\mathcal{X} \times \mathcal{Y}$ of feature and label. By relaxing this assumption, the original problem can be morphed to new ones such as heterogeneous domain adaptation [18], open set domain adaptation [19], and partial domain adaptation [20]. Our focus in this work is the semi-supervised and unsupervised settings of the original problem.

From a statistical viewpoint, majority of existing works can be categorized into three groups: statistic matching, marginal distribution matching, marginal and class-conditional distribution matching. In each group, methods make use of various loss functions and hypothesis spaces to achieve the adaptation. After matching the statistical properties, a classifier trained on the source samples would be suitable for classifying the target samples. Statistic matching mainly involves matching the source and target covariance matrices [13], [14], [17], [21], [22] under some divergence measures such as the Stein divergence or the Euclidean distance that is a special case of the Bregman divergence. Since a covariance matrix inherently encodes a zero-mean Gaussian distribution, matching the covariance matrices in this way can therefore be interpreted as matching the marginal distributions of the two domains,

Manuscript received February 29, 2020; revised July 10, 2020; accepted July 26, 2020. Date of publication August 5, 2020; date of current version August 13, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61906069; in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011411 and Grant 2019A1515011700; in part by the Project Funded by the China Postdoctoral Science Foundation under Grant 2019M662912; in part by the Science and Technology Program of Guangzhou under Grant 202002030355; and in part by the Fundamental Research Funds for the Central Universities under Grant 2019MS088. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Raja Bala. *(Corresponding authors: Sentao Chen; Xiaowei Yang.)*

Sentao Chen, Xiaona Jin, and Xiaowei Yang are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: sentaochen@yahoo.com).

Mehrtash Harandi is with the Department of Electrical and Computer Systems Engineering, Monash University, Clayton, VIC 3800, Australia.

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TIP.2020.3013167

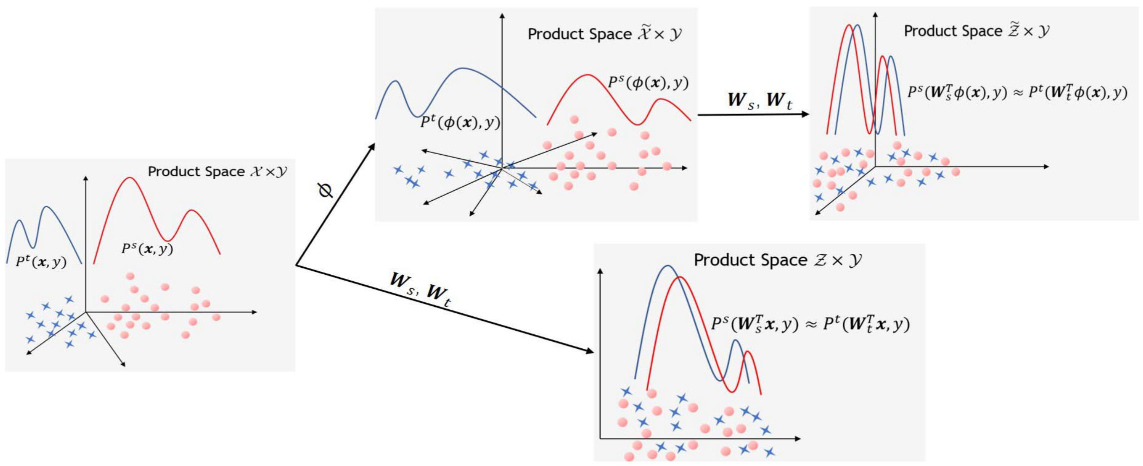
Product Space $\mathcal{X} \times \mathcal{Y}$
$P^t(\boldsymbol{x}, y)$
$P^s(\boldsymbol{x}, y)$
ϕ
Product Space $\tilde{\mathcal{X}} \times \mathcal{Y}$
$P^t(\phi(\boldsymbol{x}), y)$
$P^s(\phi(\boldsymbol{x}), y)$
$\boldsymbol{W}_s, \boldsymbol{W}_t$
Product Space $\tilde{\mathcal{Z}} \times \mathcal{Y}$
$P^s(\boldsymbol{W}_s^T\phi(\boldsymbol{x}), y) \approx P^t(\boldsymbol{W}_t^T\phi(\boldsymbol{x}), y)$
$\boldsymbol{W}_s, \boldsymbol{W}_t$
Product Space $\mathcal{Z} \times \mathcal{Y}$
$P^s(\boldsymbol{W}_s^T\boldsymbol{x}, y) \approx P^t(\boldsymbol{W}_t^T\boldsymbol{x}, y)$

Fig. 1. Illustration of our approach. It is directly motivated by the joint distribution mismatch problem in domain adaptation. Our goal in this work is to learn linear or nonlinear projections to make the source and target joint distributions similar, under the L^2-distance. In the figure, the bottom route illustrates the case of linear projections (Section III-D), and the upper one demonstrates the nonlinear case (Section III-E).

which both take the form of a Gaussian distribution. However, this does not necessarily mean that the source and target joint distributions are matched as well.

Marginal distribution matching matches the source distribution $P^s(\boldsymbol{x})$ and the target distribution $P^t(\boldsymbol{x})$, which mainly contains two lines of works. One line of works make the covariate shift assumption [23]: the source distribution is different from the target distribution, but the source labeling distribution $P^s(y|\boldsymbol{x})$ and the target labeling distribution $P^t(y|\boldsymbol{x})$ are the same. Under this assumption, domain adaptation is achieved via reweighting the source distribution by $w(\boldsymbol{x})$ to mimic the target distribution, and training a classifier with respect to the reweighted source joint distribution $w(\boldsymbol{x})P^s(\boldsymbol{x}, y)$ [24]–[26]. Learning the reweighting function $w(\boldsymbol{x})$, which is the estimate of the density ratio $P^t(\boldsymbol{x})/P^s(\boldsymbol{x})$, is key to the distribution matching here. Generally, this can be accomplished by minimizing the Maximum Mean Discrepancy (MMD) distance [24], the KL divergence [25], or the logistic loss [26]. However, as noted in [27], [28], the assumption that these methods rely on is strict and may not hold in real world applications. Considering this, the other line of works attempt to match the marginal distributions via learning a new feature representation. The core assumption behind this school of thought is that there exists a feature mapping $\mathcal{T}$ (or a set of mappings $\mathcal{T}_s$ and $\mathcal{T}_t$), under which the marginal distributions are similar, *ie*, $P^s(\mathcal{T}(\boldsymbol{x})) \approx P^t(\mathcal{T}(\boldsymbol{x}))$, and the labeling distributions are similar too, *ie*, $P^s(y|\mathcal{T}(\boldsymbol{x})) \approx P^t(y|\mathcal{T}(\boldsymbol{x}))$. Building on this assumption, domain adaptation is achieved via learning the feature mapping $\mathcal{T}$ to match the two marginal distributions, and training a discriminative classifier with respect to the matched source joint distribution $P^s(\mathcal{T}(\boldsymbol{x}), y)$ [27], [29]–[31]. Specifically, the feature mapping can take the form of a dimension reduction matrix [27], [31], [32], a transportation plan [28], or a neural network [5], [10], [29]. It can be learned by minimizing the discrepancy between the source and target distributions, which is measured by the L^2-distance [32], the MMD distance [27], [29], the Hellinger distance [31], the Central Moment Discrepancy (CMD) [10], or the adversarial loss [5], [9], [30]. Although matching the marginal distributions in this way has been justified theoretically [33], as mentioned in [4], [34], [35], this does not imply that the labeling distributions are matched as well.

The idea of marginal and class-conditional distribution matching is pioneered in [36]. Generally, it is assumed that there exists a dimensionality reduction matrix $\boldsymbol{W}$ (or again two projections $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$), such that the marginal distributions are similar, $P^s(\boldsymbol{W}^\top\boldsymbol{x}) \approx P^t(\boldsymbol{W}^\top\boldsymbol{x})$, and the class-conditional distributions are also similar, $P^s(\boldsymbol{W}^\top\boldsymbol{x}|y) \approx P^t(\boldsymbol{W}^\top\boldsymbol{x}|y)$. The projection can be learned by jointly minimizing the marginal distribution discrepancy and the class-conditional distribution discrepancy, under the Jensen-Shannon divergence [36] or the MMD distance [8], [11], [37]–[39]. Although the effectiveness of these methods has been shown empirically, it is not very clear why respectively matching the marginal distributions and the class-conditional distributions should eventually lead to matching the joint distributions. In other words, the goal of matching the source and target joint distributions may not be well achieved in this way since a joint distribution $P(\boldsymbol{x}, y)$ is decomposed into the product of the marginal distribution and the labeling distribution, *ie*, $P(\boldsymbol{x}, y) = P(\boldsymbol{x})P(y|\boldsymbol{x})$, and not the product of the marginal distribution and the class-conditional distribution, *ie*, $P(\boldsymbol{x}, y) \neq P(\boldsymbol{x})P(\boldsymbol{x}|y)$.

In light of the above discussion, in this article we propose to directly match the source and target joint distributions for domain adaptation, which is more intuitive and closely related to the nature of the problem. To this end, we propose a Joint Distribution Invariant Projections (JDIP) approach, which seeks linear projections to make the source and target joint distributions similar. In particular, we exploit the L^2-distance [32], [40] to measure the similarity between joint distributions. Generally, estimating this distance involves respectively estimating the two probability distributions, which is known to be a difficult problem in statistical learning [1], [15]. To overcome this issue, we propose a least square approach to estimate the distance, which avoids going through joint distribution estimation, and leads to a quadratic optimization problem with analytic solution. Seeking the linear projections can then be accomplished by minimizing the estimated L^2-distance of the projected source and target data. Since linear projections may not be expressive enough to model the complex transformations between domains, we therefore introduce a kernel extension of our JDIP approach, which enables us to match the joint distributions nonlinearly. From a geometrical point of view, learning the linear and nonlinear projections can be naturally formulated as optimization problems defined on the product manifolds. The manifold based optimization problems are then solved by the Riemannian Gradient Descent (RGD) method [22], [41]. Fig. 1 illustrates our approach, both for the linear and nonlinear cases. We demonstrate the benefits of our approach over exiting techniques on several visual classification tasks, including object recognition, satellite scene classification and face recognition. In short, our contributions can be summarized as follows:

1) We introduce a novel least square method to estimate the L^2-distance between the source and target joint distributions, which circumvents the difficult distribution estimation problem and simplifies the estimation process.
2) To solve the joint distribution mismatch problem in domain adaptation, we propose the JDIP approach, which learns linear projections to make the source and target joint distributions similar under the L^2-distance. Compared with the previous studies, this is more intuitive and closely related to the nature of the domain adaptation problem.
3) To further account for the complex transformations between domains, we show that our JDIP approach can be kernelized, and introduce its kernel extension: Kernel Joint Distribution Invariant Projections (KJDIP), such that the two joint distributions can be matched nonlinearly.
4) Based on the L^2-distance, we derive a generalization error bound for the proposed approach, explaining how it works from a theoretical perspective.

The rest of this article is organized as follows. In Section II, we briefly review the previous works on domain adaptation. Section III defines the domain adaptation problem, describes our motivation, and introduces the proposed approach. The learning algorithm is designed in Section IV. Section V provides the theoretical analysis of our method. In Section VI, we present comprehensive evaluation results and empirical analysis for our approach. Finally, Section VII concludes the paper with possible future research directions.

II. Related Work

In this section, we focus on discussing the domain adaptation techniques that benefit from the statistical approach, as they are most relevant to our research.

As mentioned previously, the domain adaptation techniques are developed via matching the statistical properties such as the covariance matrix or the probability distribution. Sun *et al.* [21] proposed the CORrelation ALignment (CORAL) that matches the source and target covariance matrices for domain adaptation. This method is then extended by [42] to its end-to-end neural network counterpart, and by [13], [17] to the infinite-dimensional Reproducing Kernel Hilbert Space (RKHS). Li *et al.* [14] introduced the covariance matching approach (DACoM) for semi-supervised domain adaptation, which exploits a couple of dimension reduction matrices to match the source and target second-order moments under the Euclidean distance, and simultaneously preserves the local geometric structure of the data and the discriminative information for classification.

In general, most domain adaptation methods that rely on probability distribution matching attempt to either match the source and target marginal distributions [24]–[27], [29]–[32], or the source and target marginal distributions and class-conditional distributions [8], [11], [36]–[39], [43], [44]. To match the marginal distributions, Chen *et al.* [16] introduced two subspace distribution adaptation frameworks, both of which exploit a subspace distribution adaptation function to make the source distribution similar to the target distribution, and simultaneously learn the adaptive classifier by the structural risk minimization principle. Kumagai and Iwatat [15] transformed the source feature representation via a linear matrix function to make the source distribution similar to the target distribution under the MMD distance. Instead of only modifying the source distribution, Baktashmotlagh *et al.* [31] proposed to make use of a projection matrix to change the feature representation, and matched the source and target distributions under the Hellinger distance. In the context of deep neural networks, distribution matching is achieved via explicitly minimizing the divergences between the source and target activation distributions [10], [29], or implicitly maximizing the adversarial losses [5], [9], [30], which in fact corresponds to divergences such as the $\mathcal{H}$-divergence [30] or the Jensen-Shannon divergence [18]. For example, [10] minimizes the domain discrepancy by matching higher-order moments of the activation distributions under the CMD metric, while [29] encodes the MMD criteria into multiple layers of AlexNet [45], where the network parameters as well as the parameters of the RBF kernel in MMD are jointly optimized. Instead of merely matching the marginal distributions, a string of works explore matching the marginal distributions and class-conditional distributions concurrently [11], [36]–[38], [44]. Roughly speaking, most of these works are based on the Joint Distribution Adaptation (JDA) method [37], which learns a dimension reduction matrix to jointly match the source and target marginal distributions and class-conditional distributions under the MMD distance. Specifically, this basic method is extended in several ways, such as further considering the class discriminative information [8], [39], [43], or the locality structure of the data [44].

Unfortunately, except the work of Courty *et al.* [35] that minimizes the optimal transport loss to reduce the joint distribution mismatch, directly matching the source joint distribution $P^s(\boldsymbol{x}, y)$ and the target joint distribution $P^t(\boldsymbol{x}, y)$ is rarely explored among the statistical matching approaches. This is important since the joint distribution mismatch is actually a crucial problem in domain adaptation [2]–[4], [6]. Therefore, in this research we are dedicated to solving this problem. To be specific, we aim at directly matching the source and target joint distributions under the L^2-distance. Besides, we propose a least square method to estimate the distance, which is free from the difficult distribution estimation, and leads to a simple optimization problem with analytic solution.

Of course, many other important domain adaptation techniques have been proposed in the past [46], [47]. Notable examples include, but are not limited to, methods based on linear discriminant analysis [7], reconstruction [48]–[50], and metric learning [51]. While discussing these approaches in details goes beyond the scope of this article, our experiments on several visual data sets demonstrate the benefits of the proposed approach by comparing with them.

III. Proposed Method

A. Problem Definition

Let $\mathcal{X} \subseteq \mathbb{R}^D$ be an input feature space and $\mathcal{Y}$ be an output label space. $\mathcal{Y}$ is either a discrete set for classification or a continuous set for regression. A domain is a joint probability distribution $P(\boldsymbol{x}, y)$ defined on $\mathcal{X} \times \mathcal{Y}$ [6]. We interchangeably use the terms domain and joint probability distribution throughout this article. The marginal distribution of a domain is defined as $P(\boldsymbol{x}) = \int_y P(\boldsymbol{x}, y)dy$. Let $h : \mathcal{X} \to \mathcal{Y}$ be a hypothesis from a hypothesis space $\mathcal{H}$, and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ a loss function. Then the generalization error of h with respect to a domain $P(\boldsymbol{x}, y)$ is defined as the expected loss $\mathbb{E}_{(\boldsymbol{x},y)\sim P(\boldsymbol{x},y)}[\ell(h(\boldsymbol{x}), y)] = \int_{(\boldsymbol{x},y)} \ell(h(\boldsymbol{x}), y)P(\boldsymbol{x}, y)d\boldsymbol{x}dy$. Besides, we denote an i.i.d set of m samples drawn from a joint distribution as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m \sim P(\boldsymbol{x}, y)$. According to [2], [4], [6], we formally define the domain adaptation problem as follows:

Definition 1: Let $P^s(\boldsymbol{x}, y)$ *be a source domain and* $P^t(\boldsymbol{x}, y)$ *be a target domain,* $P^s(\boldsymbol{x}, y) \neq P^t(\boldsymbol{x}, y)$. *In the semi-supervised setting, the labeled source data* $\mathcal{D}^s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{m_s} \sim P^s(\boldsymbol{x}, y)$, *the unlabeled target data* $\mathcal{D}^u = \{\boldsymbol{x}_i^t\}_{i=1}^{m_u} \sim P^t(\boldsymbol{x})$, *and the labeled target data* $\mathcal{D}^{tl} = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{m_{tl}} \sim P^t(\boldsymbol{x}, y)$ $(m_{tl} \ll m_s)$ *are available during training. In the unsupervised setting, only the labeled source data* $\mathcal{D}^s$ *and the unlabeled target data* $\mathcal{D}^u$ *are accessible. Given these data, the semi-supervised or unsupervised domain adaptation problem consists of picking a hypothesis* $h^* \in \mathcal{H}$ *such that its generalization error with respect to the target domain is minimized, i.e.* $h^* = \text{argmin}_{h\in\mathcal{H}}\mathbb{E}_{(\boldsymbol{x},y)\sim P^t(\boldsymbol{x},y)}[\ell(h(\boldsymbol{x}), y)]$.

B. Motivation

According to Definition 1, a crucial problem in domain adaptation is the joint distribution difference, *ie*, $P^s(\boldsymbol{x}, y) \neq P^t(\boldsymbol{x}, y)$, since it hinders a discriminative classifier trained on the source domain from generalizing well to the target domain:

$$\underset{h\in\mathcal{H}}{\text{argmin}}\, \mathbb{E}_{(\boldsymbol{x},y)\sim P^s(\boldsymbol{x},y)}[\ell(h(\boldsymbol{x}), y)] \neq \underset{h\in\mathcal{H}}{\text{argmin}}\, \mathbb{E}_{(\boldsymbol{x},y)\sim P^t(\boldsymbol{x},y)}[\ell(h(\boldsymbol{x}), y)]. \tag{1}$$

However, if there exist mappings φ_s, φ_t, such that $P^s(\varphi_s(\boldsymbol{x}), y) \approx P^t(\varphi_t(\boldsymbol{x}), y)$, then it leads to

$$\underset{h\in\mathcal{H}}{\text{argmin}}\, \mathbb{E}_{(\varphi_s(\boldsymbol{x}),y)\sim P^s(\varphi_s(\boldsymbol{x}),y)}[\ell(h(\varphi_s(\boldsymbol{x}), y)] \approx \underset{h\in\mathcal{H}}{\text{argmin}}\, \mathbb{E}_{(\varphi_t(\boldsymbol{x}),y)\sim P^t(\varphi_t(\boldsymbol{x}),y)}[\ell(h(\varphi_t(\boldsymbol{x}), y)]. \tag{2}$$

In other words, under the mappings φ_s, φ_t, we can expect a good source discriminative model to achieve low prediction error in the target domain. Formally, we introduce the following assumption for our approach:

Assumption 1: For a domain adaptation problem with $P^s(\boldsymbol{x}, y) \neq P^t(\boldsymbol{x}, y)$, *there exist mappings* φ_s *and* φ_t, *such that* $P^s(\varphi_s(\boldsymbol{x}), y) \approx P^t(\varphi_t(\boldsymbol{x}), y)$.

In the following sections, we will show that φ_s and φ_t together can take two kinds of different forms, corresponding to our linear and nonlinear projection techniques. Before diving into the specific details, it is worth mentioning that the existence of such mappings is reasonable. For instance, in object recognition tasks such as [52] and [53], the image feature vector often contains irrelevant information (*e.g*, background, lighting condition) that causes the joint distribution to change across domains. By mapping the feature to a new space with certain characteristic, such redundant information can probably be reduced, making the source and target joint distributions similar in the resulting space.

C. L^2-Distance Estimation

Under Assumption 1, we aim to search for mappings φ_s, φ_t that minimize the L^2-distance between the source and target joint distributions. This distance is also known as a special Bregman divergence whose seed function is set to the quadratic function [32]. To be specific, the L^2-distance between the source joint distribution $P^s(\boldsymbol{x}, y)$ and the target joint distribution $P^t(\boldsymbol{x}, y)$ is expressed as:

$$L^2\big(P^s(\boldsymbol{x}, y), P^t(\boldsymbol{x}, y)\big) = \int_{(\boldsymbol{x},y)} \big(P^s(\boldsymbol{x}, y) - P^t(\boldsymbol{x}, y)\big)^2 d\boldsymbol{x}dy. \tag{3}$$

Since the joint distributions are not directly accessible in practice, we need to estimate the distance from discrete samples, In particular, let $\mathcal{D}^s_{\varphi_s} = \{(z_i^s, y_i^s)\}_{i=1}^{m_s} \sim P^s(z, y)$ and $\mathcal{D}^t_{\varphi_t} = \{(z_i^t, y_i^t)\}_{i=1}^{m_t} \sim P^t(z, y)$, where z is the new feature vector induced by φ_s, φ_t: $z_i^s = \varphi_s(\boldsymbol{x}_i^s)$ and $z_i^t = \varphi_t(\boldsymbol{x}_i^t)$. With these notations, we present the empirical estimate of the L^2-distance in the following proposition.

Proposition 1: Given the source samples $\mathcal{D}^s_{\varphi_s}$ *and the target samples* $\mathcal{D}^t_{\varphi_t}$, *the empirical estimate of the* L^2*-distance* $L^2\big(P^s(z, y), P^t(z, y)\big)$ *is expressed as*

$$\widehat{L^2}\big(\mathcal{D}^s_{\varphi_s}, \mathcal{D}^t_{\varphi_t}\big) = \frac{1}{m_s}\sum_{i=1}^{m_s}\big(P^s(z_i^s, y_i^s) - P^t(z_i^s, y_i^s)\big) - \frac{1}{m_t}\sum_{i=1}^{m_t}\big(P^s(z_i^t, y_i^t) - P^t(z_i^t, y_i^t)\big). \tag{4}$$

Proof: Please see the supplementary material. □

Obviously, Eq. (4) still involves the unknown joint distribution difference function $P^s(z, y) - P^t(z, y)$. A traditional way to estimate it would be first estimating the two distributions separately via Kernel Density Estimation (KDE) [54], and then computing the difference of the two estimates. However, KDE is known to be less reliable as the dimensionality of the data increases [36], [54]. Inspired by [55], we therefore propose a least square approach to directly estimate this difference function without the need of going through joint distribution estimation. The idea is to use a function $r(z, y; \boldsymbol{\theta})$ to approximate the true difference function $P^s(z, y) - P^t(z, y)$ under the square error. Considering that our focus in this work is the classification problem, let the function $r(z, y; \boldsymbol{\theta})$ be

$$r(z, y; \boldsymbol{\theta}) = \sum_{i=1}^{m_{st}} \theta_i \exp\Big(-\frac{\|z - z_i\|^2}{2\sigma^2}\Big)\delta(y = y_i), \quad (5)$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{m_{st}})^\top$, $m_{st} = m_s + m_t$, and $\sigma > 0$ is the Gaussian kernel width. The function centers of $r(z, y; \boldsymbol{\theta})$ are $\big((z_1, y_1), \ldots, (z_{m_s}, y_{m_s}), (z_{m_s+1}, y_{m_s+1}), \ldots, (z_{m_{st}}, y_{m_{st}})\big) = \big((z_1^s, y_1^s), \ldots, (z_{m_s}^s, y_{m_s}^s), (z_1^t, y_1^t), \ldots (z_{m_t}^t, y_{m_t}^t)\big)$. $\delta(y = y_i)$ is a function which evaluates 1 if $y = y_i$ is true and 0 otherwise. Note that the function model in Eq. (5) is a popular choice for approximating unknown target functions in statistical machine learning [56]–[58]. It is expressive and linear in its parameter $\boldsymbol{\theta}$, which, as shown below, can lead to a simple optimization problem. Similar design of such function form for $r(z, y; \boldsymbol{\theta})$ can also be found in [59].

The optimal parameter $\boldsymbol{\theta}^*$ of the function $r(z, y; \boldsymbol{\theta})$ is learned via:

$$\begin{aligned}\boldsymbol{\theta}^* &= \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^{m_{st}}} \int_{(z,y)} \Big(r(z, y; \boldsymbol{\theta}) - \big(P^s(z, y) - P^t(z, y)\big)\Big)^2 dzdy \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^{m_{st}}} \int_{(z,y)} r(z, y; \boldsymbol{\theta})^2 dzdy \\ &\quad -2\int_{(z,y)} r(z, y; \boldsymbol{\theta})\big(P^s(z, y) - P^t(z, y)\big)dzdy \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^{m_{st}}} \Big[\boldsymbol{\theta}^\top \boldsymbol{H}\boldsymbol{\theta} - 2\boldsymbol{b}^\top\boldsymbol{\theta}\Big], \end{aligned} \quad (6)$$

where $\boldsymbol{H}$ is the $m_{st} \times m_{st}$ matrix and $\boldsymbol{b}$ is the m_{st}-dimensional vector. The (i, j)-th element of $\boldsymbol{H}$ and the i-th element of $\boldsymbol{b}$ are respectively defined as

$$\begin{aligned} h_{i,j} &= \int_z \exp\Big(-\frac{\|z - z_i\|^2}{2\sigma^2}\Big)\exp\Big(-\frac{\|z - z_j\|^2}{2\sigma^2}\Big)dz \\ &\quad \times \sum_y \delta(y = y_i)\delta(y = y_j) \\ &= (\pi\sigma^2)^{d/2}\exp\Big(-\frac{\|z_i - z_j\|^2}{4\sigma^2}\Big)\delta(y_i = y_j), \quad (7) \\ b_i &= \int_{(z,y)} \exp\Big(-\frac{\|z - z_i\|^2}{2\sigma^2}\Big)\delta(y = y_i)P^s(z, y)dzdy \\ &\quad -\int_{(z,y)} \exp\Big(-\frac{\|z - z_i\|^2}{2\sigma^2}\Big)\delta(y = y_i)P^t(z, y)dzdy. \quad (8)\end{aligned}$$

According to the law of large numbers [54], the expectations in Eq. (8) can be approximated by empirical averages. Therefore, $\boldsymbol{b}$ can be estimated as $\widehat{\boldsymbol{b}}$:

$$\begin{aligned}\widehat{b_i} &= \frac{1}{m_s}\sum_{n=1}^{m_s} \exp\Big(-\frac{\|z_n^s - z_i\|^2}{2\sigma^2}\Big)\delta(y_n^s = y_i) \\ &\quad -\frac{1}{m_t}\sum_{n=1}^{m_t} \exp\Big(-\frac{\|z_n^t - z_i\|^2}{2\sigma^2}\Big)\delta(y_n^t = y_i). \end{aligned} \quad (9)$$

Then, by adding an ℓ_2-regularizer to the objective function in (6), we arrive at the following unconstrained quadratic optimization problem:

$$\widehat{\boldsymbol{\theta}} = \operatorname*{argmin}_{\boldsymbol{\theta}\in\mathbb{R}^{m_{st}}} \Big[\boldsymbol{\theta}^\top\boldsymbol{H}\boldsymbol{\theta} - 2\widehat{\boldsymbol{b}}^\top\boldsymbol{\theta} + \lambda\boldsymbol{\theta}^\top\boldsymbol{\theta}\Big], \quad (10)$$

where $\lambda \geq 0$ is the regularization parameter. Obviously, the solution to this problem can be analytically obtained as

$$\widehat{\boldsymbol{\theta}} = (\boldsymbol{H} + \lambda\mathbf{I})^{-1}\widehat{\boldsymbol{b}}. \quad (11)$$

Eventually, the empirical estimate of the L^2-distance is given by

$$\begin{aligned}\widehat{L^2}(\mathcal{D}_{\varphi_s}^s, \mathcal{D}_{\varphi_t}^t) &= \frac{1}{m_s}\sum_{i=1}^{m_s} r(z_i^s, y_i^s; \widehat{\boldsymbol{\theta}}) - \frac{1}{m_t}\sum_{i=1}^{m_t} r(z_i^t, y_i^t; \widehat{\boldsymbol{\theta}}) \\ &= \widehat{\boldsymbol{b}}^\top\widehat{\boldsymbol{\theta}}. \end{aligned} \quad (12)$$

Later in Section VI-D.1, we experimentally verify the effectiveness of this L^2-distance estimation method.

D. Joint Distribution Invariant Projections

With the L^2-distance estimated in Eq. (12), we are now ready to learn our joint distribution invariant projections. However, no matter in semi-supervised or unsupervised domain adaptation, the target set $\mathcal{D}_{\varphi_t}^t$ is always incomplete. Namely, the target data are not all labeled. To solve this problem, we make use of the strategy in [38], which first trains a classifier on all the labeled data, and then uses it to predict labels for the unlabeled target data. Specifically, in semi-supervised domain adaptation, we set $\mathcal{D}_{\varphi_t}^t = \mathcal{D}_{\varphi_t}^{tl} \cup \{(\varphi_t(\boldsymbol{x}_i^t), y_i^t)\}_{i=1}^{m_u}$ with the number of its samples $m_t = m_{tl} + m_u$, where y_i^t is the label estimated by the classifier. In unsupervised domain adaptation, we set $\mathcal{D}_{\varphi_t}^t = \{(\varphi_t(\boldsymbol{x}_i^t), y_i^t)\}_{i=1}^{m_u}$ with the number of its samples $m_t = m_u$, where y_i^t is predicted by the source classifier.

In this section, we first study the simple linear mappings, which is illustrated in the bottom route of Fig. 1. In particular, let the two mappings φ_s, φ_t be two projection matrices: $\boldsymbol{W}_s, \boldsymbol{W}_t : \mathcal{X} \to \mathcal{Z} \subseteq \mathbb{R}^d (d < D)$, where d is the dimensionality of the projected feature. Furthermore, we enforce orthogonality constraints on $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$, such that $\boldsymbol{W}_s^\top\boldsymbol{W}_s = \mathbf{I}$ and $\boldsymbol{W}_t^\top\boldsymbol{W}_t = \mathbf{I}$. These constraints on the one hand can avoid degeneracy, such as having all the projected samples collapsing to the origin [31], and on the other hand guarantee that there is no redundant information across dimensions of the projected feature [34]. As a matter of fact, $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ are now points on the Stiefel manifold [22], [41], *ie*, $\boldsymbol{W}_s, \boldsymbol{W}_t \in \mathrm{St}(d, D)$. Since $\varphi_s = \boldsymbol{W}_s$ and $\varphi_t = \boldsymbol{W}_t$, the new source and target feature vector are respectively expressed as $z_i^s = \boldsymbol{W}_s^\top\boldsymbol{x}_i^s$ and $z_j^t = \boldsymbol{W}_t^\top\boldsymbol{x}_j^t$. As such, the matrix and vectors $\boldsymbol{H}, \widehat{\boldsymbol{b}}, \widehat{\boldsymbol{\theta}}$ defined in (7), (9) and (11) in turn become dependent on $\boldsymbol{W}_s, \boldsymbol{W}_t$. We explicitly write these matrix and vector functions as $\boldsymbol{H} = \boldsymbol{H}(\boldsymbol{W}_s, \boldsymbol{W}_t), \widehat{\boldsymbol{b}} = \widehat{\boldsymbol{b}}(\boldsymbol{W}_s, \boldsymbol{W}_t)$, and $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}(\boldsymbol{W}_s, \boldsymbol{W}_t)$. Consequently, the estimated L^2-distance in Eq. (12) is expressed as:

$$\widehat{L^2}\big(\mathcal{D}^s_{\boldsymbol{W}_s}, \mathcal{D}^t_{\boldsymbol{W}_t}\big) = \Big(\widehat{\boldsymbol{b}}(\boldsymbol{W}_s, \boldsymbol{W}_t)\Big)^\top \widehat{\boldsymbol{\theta}}(\boldsymbol{W}_s, \boldsymbol{W}_t). \tag{13}$$

Finding linear projections $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ that minimize the empirical L^2-distance can thus be concisely formulated as the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{W}_s, \boldsymbol{W}_t}\ & \Big(\widehat{\boldsymbol{b}}(\boldsymbol{W}_s, \boldsymbol{W}_t)\Big)^\top \widehat{\boldsymbol{\theta}}(\boldsymbol{W}_s, \boldsymbol{W}_t)\\ \text{s.t.}\ \ & (\boldsymbol{W}_s, \boldsymbol{W}_t) \in \mathcal{M}_{prod.},\end{aligned} \tag{14}$$

where $\mathcal{M}_{prod.} = \mathrm{St}(d, D) \times \mathrm{St}(d, D)$ is the product manifold discussed in the supplementary material. Working on this manifold not only helps to conveniently handle the orthogonality constraints, but also allows us to simultaneously obtain the projection matrices $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$.

Remark 1: The joint distribution mismatch can be large at the very beginning. This boils down in incorrect estimation of the target labels, which results in inaccurate evaluation of the L^2-distance, and eventually suboptimal feature projections. Therefore, we iteratively refine the labels for the unlabeled target data: at each iteration, once the data are projected by the learned projections, the labels are updated by a new classifier trained on all the truly labeled data. Note that in domain adaptation, iteratively refining the target labels is common, and it usually improves the quality of the labels, as shown in [37]–[39], [60].

E. Kernel Joint Distribution Invariant Projections

We now show that our approach can be kernelized to handle nonlinearity in the data, which is often encountered in computer vision tasks. Following the common practice when converting a linear algorithm to a nonlinear one (*e.g*, from PCA to Kernel PCA [61]), we first map the source and target data into a high-dimensional RKHS $\widetilde{\mathcal{X}}$, and then perform linear projections in that space to match the joint distributions. This two-step procedure is demonstrated in the upper route of Fig. 1.

Specifically, let $\varphi_s = \boldsymbol{W}_s \circ \phi$ and $\varphi_t = \boldsymbol{W}_t \circ \phi$ be two composite mappings. $\phi : \mathcal{X} \to \widetilde{\mathcal{X}} \subseteq \mathbb{R}^{\widetilde{D}}$ is the function mapping an input vector $\boldsymbol{x}$ to the RKHS $\widetilde{\mathcal{X}}$ with corresponding kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_j)$. $\boldsymbol{W}_s, \boldsymbol{W}_t : \widetilde{\mathcal{X}} \to \widetilde{\mathcal{Z}} \subseteq \mathbb{R}^{\widetilde{d}}$ are the projection matrices on the Stiefel manifold $\mathrm{St}(\widetilde{d}, \widetilde{D})$, where $\widetilde{d}$ is the dimensionality of the projected feature. Note that for universal kernel functions, such as the Gaussian kernel, the RKHS $\widetilde{\mathcal{X}}$ is infinite-dimensional, *ie*, $\widetilde{D} \to \infty$, making it impractical to directly learn the matrices $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$. Therefore, we make use of the Riesz representer theorem [61] and respectively express the projection matrices as a linear combination of the examples in $\widetilde{\mathcal{X}}$. Namely, let $\boldsymbol{W}_s = \phi(\boldsymbol{X}^{st})\boldsymbol{A}_s$ and $\boldsymbol{W}_t = \phi(\boldsymbol{X}^{st})\boldsymbol{A}_t$, where $\phi(\boldsymbol{X}^{st}) = (\phi(\boldsymbol{x}^s_1), \ldots, \phi(\boldsymbol{x}^t_{m_t})) \in \mathbb{R}^{\widetilde{D}\times m_{st}}$ is the matrix containing all the unlabeled source and target data mapped to $\widetilde{\mathcal{X}}$, and $\boldsymbol{A}_s, \boldsymbol{A}_t \in \mathbb{R}^{m_{st}\times \widetilde{d}}$ are the representation matrices of $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$. Now learning the projection matrices $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ immediately boils down to learning the representation matrices $\boldsymbol{A}_s$ and $\boldsymbol{A}_t$.

Thanks to the kernel trick, the orthogonality constraints on $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ can be written as $\boldsymbol{A}_s^\top \boldsymbol{K}^{st} \boldsymbol{A}_s = \mathbf{I}$ and $\boldsymbol{A}_t^\top \boldsymbol{K}^{st} \boldsymbol{A}_t = \mathbf{I}$, where $\boldsymbol{K}^{st} = \phi(\boldsymbol{X}^{st})^\top \phi(\boldsymbol{X}^{st})$ is a $m_{st} \times m_{st}$ positive definite kernel matrix. Besides, the new source and target feature vector are respectively expressed as

$$z^s_i = \boldsymbol{W}_s^\top \phi(\boldsymbol{x}^s_i) = \boldsymbol{A}_s^\top k(\boldsymbol{X}^{st}, \boldsymbol{x}^s_i), \tag{15}$$
$$z^t_j = \boldsymbol{W}_t^\top \phi(\boldsymbol{x}^t_j) = \boldsymbol{A}_t^\top k(\boldsymbol{X}^{st}, \boldsymbol{x}^t_j), \tag{16}$$

where the vector $k(\boldsymbol{X}^{st}, \boldsymbol{x}) = \big(k(\boldsymbol{x}^s_1, \boldsymbol{x}), \ldots, k(\boldsymbol{x}^t_{m_t}, \boldsymbol{x})\big)^\top$. As a result, learning nonlinear projections $\boldsymbol{W}_s = \phi(\boldsymbol{X}^{st})\boldsymbol{A}_s$ and $\boldsymbol{W}_t = \phi(\boldsymbol{X}^{st})\boldsymbol{A}_t$ that minimize the empirical L^2-distance can be formulated as the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{A}_s, \boldsymbol{A}_t}\ & \Big(\widehat{\boldsymbol{b}}(\boldsymbol{A}_s, \boldsymbol{A}_t)\Big)^\top \widehat{\boldsymbol{\theta}}(\boldsymbol{A}_s, \boldsymbol{A}_t)\\ \text{s.t.}\ \ & \boldsymbol{A}_s^\top \boldsymbol{K}^{st} \boldsymbol{A}_s = \mathbf{I}, \boldsymbol{A}_t^\top \boldsymbol{K}^{st} \boldsymbol{A}_t = \mathbf{I}.\end{aligned} \tag{17}$$

Here, the vector functions $\widehat{\boldsymbol{b}}(\boldsymbol{A}_s, \boldsymbol{A}_t)$ and $\widehat{\boldsymbol{\theta}}(\boldsymbol{A}_s, \boldsymbol{A}_t)$ are dependent on the representation matrices $\boldsymbol{A}_s, \boldsymbol{A}_t$ of the nonlinear projections. We then use a rearrangement trick to transform this problem into a new one that is defined on the product set of two Stiefel manifolds, such that it can be solved by the same algorithm as optimization problem (14). In particular, let

$$\boldsymbol{B}_s = (\boldsymbol{K}^{st})^{\frac{1}{2}} \boldsymbol{A}_s \ \big(\Leftrightarrow \boldsymbol{A}_s = (\boldsymbol{K}^{st})^{-\frac{1}{2}} \boldsymbol{B}_s\big), \tag{18}$$
$$\boldsymbol{B}_t = (\boldsymbol{K}^{st})^{\frac{1}{2}} \boldsymbol{A}_t \ \big(\Leftrightarrow \boldsymbol{A}_t = (\boldsymbol{K}^{st})^{-\frac{1}{2}} \boldsymbol{B}_t\big), \tag{19}$$

then optimization problem (17) can be rewritten as the following problem defined on the product manifold:

$$\begin{aligned}\min_{\boldsymbol{B}_s, \boldsymbol{B}_t}\ & \Big(\widehat{\boldsymbol{b}}\big((\boldsymbol{K}^{st})^{-\frac{1}{2}} \boldsymbol{B}_s, (\boldsymbol{K}^{st})^{-\frac{1}{2}} \boldsymbol{B}_t\big)\Big)^\top\\ & \qquad \times \widehat{\boldsymbol{\theta}}\big((\boldsymbol{K}^{st})^{-\frac{1}{2}} \boldsymbol{B}_s, (\boldsymbol{K}^{st})^{-\frac{1}{2}} \boldsymbol{B}_t\big)\\ \text{s.t.}\ \ & (\boldsymbol{B}_s, \boldsymbol{B}_t) \in \mathrm{St}(\widetilde{d}, m_{st}) \times \mathrm{St}(\widetilde{d}, m_{st}).\end{aligned} \tag{20}$$

Remark 2: The nonlinear projections presented here can be further extended to more expressive ones by exploiting the multiple kernel learning idea [62], [63]. This can be achieved by associating the mapping ϕ to a kernel matrix in the form $\boldsymbol{K}^{st} = \sum_{i=1}^m \beta_i \boldsymbol{K}^{st}_i$, with $\beta_i \geq 0$ being the linear combination weight and $\boldsymbol{K}^{st}_i$ being the kernel matrix for the mapping ϕ_i. Then, the mapping ϕ can be implicitly optimized via optimizing the combination weights $\{\beta_1, \ldots, \beta_m\}$.

IV. Learning Algorithm

In this section, we design learning algorithm for obtaining the linear or nonlinear joint distribution invariant projections. The algorithm involves iteratively training a classifier (*e.g*, SVM) to label the target data and solving the manifold optimization problem (14) or (20) to obtain the projections. In the following, we first discuss learning the linear projections and then the nonlinear ones.

A. Learning the Linear Projections

As discussed in the supplementary material, we can apply the RGD method [22], [64] to find a stationary point of optimization problem (14). For convenience, we denote the objective function in (14) as $f(\boldsymbol{W}_s, \boldsymbol{W}_t)$. The RGD method consists of first calculating the Riemannian gradient through projecting the Euclidean gradient onto the tangent space, *ie*, Eq. (7) in the supplementary material, and then performing a *retraction* to ensure that the solution stays on the manifold, *ie*, Eq. (8) in the supplementary material. Since the Riemannian gradient is derived from the Euclidean gradient, which depends on the specific form of the objective function, we present in the following theorem the Euclidean gradients of $f(\boldsymbol{W}_s, \boldsymbol{W}_t)$ with respect to $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$.

Theorem 1: Let $h_{ij} = h_{ij}(\boldsymbol{W}_s, \boldsymbol{W}_t)$ be the (i, j)-th element of $\boldsymbol{H} = \boldsymbol{H}(\boldsymbol{W}_s, \boldsymbol{W}_t)$, $\widehat{b_i} = \widehat{b_i}(\boldsymbol{W}_s, \boldsymbol{W}_t)$ the i-th element of $\widehat{\boldsymbol{b}} = \widehat{\boldsymbol{b}}(\boldsymbol{W}_s, \boldsymbol{W}_t)$, and $\widehat{\theta}_i = \widehat{\theta}_i(\boldsymbol{W}_s, \boldsymbol{W}_t)$ the i-th element of $\widehat{\boldsymbol{\theta}} = \widehat{\boldsymbol{\theta}}(\boldsymbol{W}_s, \boldsymbol{W}_t)$, then the Euclidean gradients of $f(\boldsymbol{W}_s, \boldsymbol{W}_t)$ with respect to $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ are expressed as

$$\nabla_{\boldsymbol{W}_s}(f) = 2\sum_{i=1}^{m_{st}} \widehat{\theta}_i \nabla_{\boldsymbol{W}_s}(\widehat{b_i}) - \sum_{i,j=1}^{m_{st}} \widehat{\theta}_i\widehat{\theta}_j \nabla_{\boldsymbol{W}_s}(h_{ij}), \quad (21)$$

$$\nabla_{\boldsymbol{W}_t}(f) = 2\sum_{i=1}^{m_{st}} \widehat{\theta}_i \nabla_{\boldsymbol{W}_t}(\widehat{b_i}) - \sum_{i,j=1}^{m_{st}} \widehat{\theta}_i\widehat{\theta}_j \nabla_{\boldsymbol{W}_t}(h_{ij}). \quad (22)$$

Proof: Please see the supplementary material. □

By combining the target data labeling with the RGD method, our Joint Distribution Invariant Projections (JDIP) approach to learn the linear projections $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ can then be summarized by Algorithm 1. Note that the presented algorithm is for the semi-supervised domain adaptation case. For unsupervised domain adaptation, the labeled target set $\mathcal{D}^{tl}$ should be removed, since it is not available in the problem setting. In Algorithm 1, we write $\{y_i^t\}_{i=1}^{m_u} \leftarrow g(\mathcal{D}^u; \mathcal{D}^s \cup \mathcal{D}^{tl})$ to denote that $\{y_i^t\}_{i=1}^{m_u}$ are the labels for $\mathcal{D}^u$, which are predicted by a classifier g trained on $\mathcal{D}^s \cup \mathcal{D}^{tl}$. $(\boldsymbol{W}_s, \boldsymbol{W}_t) \leftarrow (\boldsymbol{W}_{s0}, \boldsymbol{W}_{t0})$ represents that $(\boldsymbol{W}_s, \boldsymbol{W}_t)$ is initialized to $(\boldsymbol{W}_{s0}, \boldsymbol{W}_{t0})$ on the manifold $\mathcal{M}_{prod.}$. Besides, $\mathcal{D}^u_{\boldsymbol{W}_t} = \{\boldsymbol{W}_t^\top \boldsymbol{x}_i^t\}_{i=1}^{m_u}$, $\mathcal{D}^s_{\boldsymbol{W}_s} = \{(\boldsymbol{W}_s^\top \boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{m_s}$, and $\mathcal{D}^{tl}_{\boldsymbol{W}_t} = \{(\boldsymbol{W}_t^\top \boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{m_{tl}}$. To stop iteratively labeling the target data, we can either set a maximum iteration T, or check the convergence of the label discrepancy. To be specific, at iteration i (≥ 1), the label discrepancy $disc_i$ is defined as

$$disc_i = 1 - \frac{\sum_{j=1}^{m_u} \delta(y_{ij}^t = y_{(i-1)j}^t)}{m_u}, \quad (23)$$

where the sets $\{y_{(i-1)j}^t\}_{j=1}^{m_u}$ and $\{y_{ij}^t\}_{j=1}^{m_u}$ are the predicted labels for the target data in the $(i-1)$-th and i-th iterations. The function $\delta(\cdot,\cdot)$ is defined in Eq. (5). When $disc_i \leq \epsilon$, with ϵ being a small positive value, the iteration can be terminated. In Section VI-D.3, we experimentally show that as the iteration proceeds, the label discrepancy will approach zero, meaning that the predicted target labels stop changing and become stable.

The main computational cost of Algorithm 1 is mainly from matrix multiplication and matrix inversion. Specifically, the cost for computing the Riemannian gradient in line 5 is $\mathcal{O}\big(D^2 d(m_s + m_t) + D^2 d^2\big)$, for performing the *retraction* in line 6 is $\mathcal{O}(Dd^2 + d^3)$, and for training a classifier such as SVM in line 8 is $\mathcal{O}\big((m_s + m_{tl})^3\big)$. Assume that the number of iterations is T from line 2 to 9, and T' from line 4 to 7. Then the overall complexity of Algorithm 1 is $\mathcal{O}\Big(TT'\big(D^2 d(m_s+m_t) + D^2d^2 + Dd^2 + d^3\big) + T(m_s+m_{tl})^3\Big)$.

Algorithm 1 Joint Distribution Invariant Projections

```
Input: Data sets: $\mathcal{D}^s, \mathcal{D}^{tl}, \mathcal{D}^u$; Parameters: $d, \sigma, \lambda$.
Output: $(\boldsymbol{W}_s, \boldsymbol{W}_t) \in \mathcal{M}_{prod.}$
1: $\{y_i^t\}_{i=1}^{m_u} \leftarrow g(\mathcal{D}^u; \mathcal{D}^s \cup \mathcal{D}^{tl})$
2: repeat
3:    $(\boldsymbol{W}_s, \boldsymbol{W}_t) \leftarrow (\boldsymbol{W}_{s0}, \boldsymbol{W}_{t0})$
4:    repeat
5:       Compute $\mathrm{grad} f(\boldsymbol{W}_s, \boldsymbol{W}_t)$ by Eq. (7) in the supplementary material
6:       $(\boldsymbol{W}_s, \boldsymbol{W}_t) \leftarrow r_{(\boldsymbol{W}_s, \boldsymbol{W}_t)}\big(-\alpha \mathrm{grad} f(\boldsymbol{W}_s, \boldsymbol{W}_t)\big)$ by Eq. (8) in the supplementary material
7:    until maximum iteration reached or convergence
8:    $\{y_i^t\}_{i=1}^{m_u} \leftarrow g(\mathcal{D}^u_{\boldsymbol{W}_t}; \mathcal{D}^s_{\boldsymbol{W}_s} \cup \mathcal{D}^{tl}_{\boldsymbol{W}_t})$
9: until maximum iteration $T$ reached or convergence
```

B. Learning the Nonlinear Projections

In Section III-E, we have shown that learning the nonlinear projections $\boldsymbol{W}_s$ and $\boldsymbol{W}_t$ can first be transformed into learning their representation matrices $\boldsymbol{A}_s$ and $\boldsymbol{A}_t$, and then into searching two matrices $\boldsymbol{B}_s$ and $\boldsymbol{B}_t$ on the Stiefel manifold $\mathrm{St}(\widetilde{d}, m_{st})$. Algorithm 2 summarizes our Kernel Joint Distribution Invariant Projections (KJDIP) approach to learn these two matrices. In this Algorithm, $\widetilde{f}(\boldsymbol{B}_s, \boldsymbol{B}_t)$ represents the objective function in (20). Additionally, the sets $\widetilde{\mathcal{D}}^u_{\boldsymbol{B}_t}$, $\widetilde{\mathcal{D}}^s_{\boldsymbol{B}_s}$, and $\widetilde{\mathcal{D}}^{tl}_{\boldsymbol{B}_t}$ are respectively defined as

$$\widetilde{\mathcal{D}}^u_{\boldsymbol{B}_t} = \big\{\boldsymbol{B}_t^\top (\boldsymbol{K}^{st})^{-\frac{1}{2}} k(\boldsymbol{X}^{st}, \boldsymbol{x}_i^t)\big\}_{i=1}^{m_u}, \quad (24)$$

$$\widetilde{\mathcal{D}}^s_{\boldsymbol{B}_s} = \big\{\big(\boldsymbol{B}_s^\top (\boldsymbol{K}^{st})^{-\frac{1}{2}} k(\boldsymbol{X}^{st}, \boldsymbol{x}_i^s), y_i^s\big)\big\}_{i=1}^{m_s}, \quad (25)$$

$$\widetilde{\mathcal{D}}^{tl}_{\boldsymbol{B}_t} = \big\{\big(\boldsymbol{B}_t^\top (\boldsymbol{K}^{st})^{-\frac{1}{2}} k(\boldsymbol{X}^{st}, \boldsymbol{x}_i^t), y_i^t\big)\big\}_{i=1}^{m_{tl}}. \quad (26)$$

V. Theoretical Analysis

We derive an error bound for our domain adaptation approach, which explains how the method works and contributes to reducing the target domain generalization error. Different from previous works [33], [65], here, we attempt to directly work on the joint distributions and measure their discrepancy by the L^2-distance. Before presenting the main theorem, we first introduce the following Lemma. It shows that the generalization error of a hypothesis in the target domain can be bounded by its source error and the L^2-distance between the source and target joint distributions.

Lemma 1: Assume that the loss function $\ell(\cdot,\cdot) \leq 1$, and that $M = \inf_{(z,y)\in\mathcal{Z}\times\mathcal{Y}} |P^s(z, y) - P^t(z, y)| > 0$. Then, for

Algorithm 2 Kernel Joint Distribution Invariant Projections

```
Input: Data sets: $\mathcal{D}^s, \mathcal{D}^{tl}, \mathcal{D}^u$; Kernel function: $k(\cdot,\cdot)$; Parameters: $\widetilde{d}, \sigma, \lambda$.
Output: $(\boldsymbol{B}_s, \boldsymbol{B}_t) \in \mathrm{St}(\widetilde{d}, m_{st}) \times \mathrm{St}(\widetilde{d}, m_{st})$
1: $\{y_i^t\}_{i=1}^{m_u} \leftarrow g(\mathcal{D}^u; \mathcal{D}^s \cup \mathcal{D}^{tl})$
2: repeat
3:     $(\boldsymbol{B}_s, \boldsymbol{B}_t) \leftarrow (\boldsymbol{B}_{s0}, \boldsymbol{B}_{t0})$
4:     repeat
5:         Compute $\mathrm{grad}\widetilde{f}(\boldsymbol{B}_s, \boldsymbol{B}_t)$ by Eq. (7) in the supplementary material
6:         $(\boldsymbol{B}_s, \boldsymbol{B}_t) \leftarrow r_{(\boldsymbol{B}_s, \boldsymbol{B}_t)}\big(-\alpha \mathrm{grad}\widetilde{f}(\boldsymbol{B}_s, \boldsymbol{B}_t)\big)$ by Eq. (8) in the supplementary material
7:     until maximum iteration reached or convergence
8:     $\{y_i^t\}_{i=1}^{m_u} \leftarrow g(\widetilde{\mathcal{D}}^u_{\boldsymbol{B}_t}; \widetilde{\mathcal{D}}^s_{\boldsymbol{B}_s} \cup \widetilde{\mathcal{D}}^{tl}_{\boldsymbol{B}_t})$
9: until maximum iteration $T$ reached or convergence
```

any hypothesis $h \in \mathcal{H}$,

$$\mathbb{E}_{(z,y)\sim P^t(z,y)}[\ell(h(z),y)] \le \mathbb{E}_{(z,y)\sim P^s(z,y)}[\ell(h(z),y)] + \frac{1}{M} L^2\big(P^s(z,y), P^t(z,y)\big). \tag{27}$$

Proof: Please see the supplementary material. □

Subsequently, by bounding the source domain generalization error and the L^2-distance respectively by their empirical estimates, we obtain the following main theorem.

Theorem 2: Let $\mathcal{H}$ a hypothesis space of VC-dimension p, $\mathcal{D}^s_{\varphi_s}$ and $\mathcal{D}^t_{\varphi_t}$ be the source and target data sets of equal size m. Let the assumptions be the same as Lemma 1. Then, for any $\delta \in (0,1)$, there exists a positive integer N_δ, with probability at least $1-\delta$, for $m > N_\delta$ and any hypothesis $h \in \mathcal{H}$,

$$\mathbb{E}_{(z,y)\sim P^t(z,y)}[\ell(h(z),y)] \le \frac{1}{m}\sum_{i=1}^{m} \ell(h(z_i^s), y_i^s) + \frac{1}{M}\big(\widehat{L^2}\big(\mathcal{D}^s_{\varphi_s}, \mathcal{D}^t_{\varphi_t}\big) + 1\big) + \sqrt{\frac{4}{m}\Big(p\log\frac{2em}{p} + \log\frac{8}{\delta}\Big)}. \tag{28}$$

Proof: Please see the supplementary material. □

This theorem tells us that under mild assumptions, the target domain generalization error of a hypothesis is mainly bounded by its source training error and the empirical L^2-distance. This means that if we can find the mappings φ_s, φ_t that minimize the empirical L^2-distance, and a classifier g that minimizes the source training error under the new feature representation, then g is probably a good classifier in the target domain. In our approach, we instantiate φ_s, φ_t to the linear or nonlinear joint distribution invariant projections, and minimize the empirical L^2-distance to learn them. Under the new feature representation produced by these projections, we train a discriminative classifier on the labeled source data and then use it to predict labels for the target data. Since our approach coincides well with the theorem, we can expect it to produce a target classifier with low generalization error.

Remark 3: Note that Theorem 2 is mainly for explaining how our approach is related to reducing the target domain generalization error, which is the ultimate goal of domain adaptation. The bound here is not a very tight one, since it holds for all hypotheses and all possible data distributions.

VI. Experiment

In this section, we comprehensively evaluate our approach on a series of real-world classification tasks for both the semi-supervised and unsupervised domain adaptation settings. Our method is implemented in Python 2.7, with the optimization technique RGD implemented by the pymanopt[1] package. The classification tasks include visual object recognition, satellite scene classification, and face recognition. We start by introducing the data sets and the experimental setup, then present the experimental results with statistical test, and finish by conducting empirical analysis of the proposed approach.

A. Data Sets

Office [52] includes images of 31 objects taken from 3 domains, Amazon (**A**, 2817 images downloaded from amazon.com), DSLR (**D**, 498 high resolution images taken by a digital SLR camera) and Webcam (**W**, 795 low-resolution images recorded by a web camera). Some sample images of this data set are shown in Fig. 2(a). In our experiments, we exploit the 4096-dimensional VGG-VD-16-FC$_7$ feature provided by [7] for this data set. Besides, to compare with the deep domain adaptation methods, the 2048-dimensional feature extracted from the ResNet-50 model [66] is also used here.

Satellite-Scene [7] is a cross-place satellite scene data set for domain adaptation. It contains 3 publicly available subsets (domains): Banja Luka (**B**) with 578 samples, UC Merced Land Use (**U**) with 500 samples, and 18-class Satellite Scene (**S**) with 266 samples. Each domain has 5 scene categories: farmland/field, trees/forest, industry, residential, and river. Following [7], the 4096-dimensional VGG-M-FC$_7$ feature is used here.

Office-Caltech [67] is a visual object recognition data set that has been widely adopted in the literature of domain adaptation [15], [22], [50]. It consists of 4 object image subsets (domains): Amazon (**A**), Caltech (**C**), DSLR (**D**), and Webcam (**W**). The four domains share images from 10 categories, some of which are shown in Fig. 2(a). Following [22], we use the 4096-dimensional VGG-FC$_6$ feature in the experiments.

Office-Home [53] is a large visual recognition data set, which is popular among the deep domain adaptation methods [12], [68]. It consists of 4 domains: Art (**A**, artistic depictions of objects), Clipart (**C**, clipart images), Product (**P**, objects without a background) and Real-World (**R**, objects captured with a regular camera). There are 65 categories shared by these domains, and the number of images in each domain is 2421, 4379, 4428, and 4357, respectively. Some representative images are illustrated in Fig. 2(b). We follow

[1]https://pymanopt.github.io/

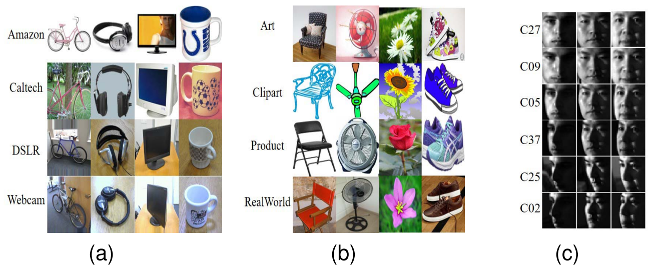

Fig. 2. Image samples from data sets (a) Office (Office-Caltech), (b) Office-Home, and (c) PIE-Multiview. Obviously, the same category in different domains exhibits substantially different visual characteristics.

the protocol in [12] and extract the output of the FC$_7$ layer of the VGG16 [69] model as the deep feature representation.

PIE-Multiview [22] includes face images of 67 individuals captured from different views, illumination conditions, and expressions. This data set has 6 subsets (domains): looking-forward (C27), looking-downward (C09), looking towards left in an increasing angle (C05, C37, C25, C02). See Fig. 2(c) for the sample images. We normalize the images to 32×32 pixels and use the vectorized gray-scale images as features.

B. Experimental Setup

We perform semi-supervised domain adaptation experiments on the Office, Satellite-Scene, Office-Caltech, and Office-Home data sets. On Office and Office-Caltech, we follow the standard protocol in [7], [38], [50], [52] to construct the semi-supervised domain adaptation tasks. To be specific, 20 samples per class are randomly selected from Amazon, and 8 samples per class from other domains if they serve as the source domains. For the target domain, 3 random samples in each class are chosen for training and the others are used for testing. Similar configuration is also applied to the Satellite-Scene data set: the training set is composed of 20 samples per class selected from the source domain and 2 samples per class from the target domain, and the test set includes the remaining target data. For these three data sets, the random sampling procedure is repeated 20 times to calculate the average test accuracy and the standard deviation of the domain adaptation algorithms. Finally on Office-Home, we follow the setting in [12] to construct the training set using all the source data and 3 examples per class from the target data, and the test set using the rest of the target data. By treating a subset as a source domain and then a target domain, we respectively build 6 tasks from Office, 6 tasks from Satellite-Scene, 12 tasks from Office-Caltech, and 12 tasks from Office-Home. Here a domain adaptation task is denoted as S $\rightarrow$ T, where S is the source domain and T is the target domain. Besides, we also perform unsupervised domain adaptation experiments on data sets Office, Office-Caltech, and PIE-Multiview. On Office and Office-Caltech, we follow the full protocol in [7], [22] to construct the unsupervised domain adaptation tasks. On PIE-Multiview, we construct 3 tasks: M1 $\rightarrow$ M2, M2 $\rightarrow$ M3, and M3 $\rightarrow$ M1, where M1, M2 and M3 are mixtures of joint distributions. To be specific, M1 = {C27, C09}, M2 = {C05, C37}, and M3 = {C25, C02}. It is necessary and interesting to study this setting, since in some real world cases, the source and target data may respectively come from several data sources.

In the experiments, we evaluate the performance of our JDIP (Algorithm 1) and KJDIP (Algorithm 2). In particular, the kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in KJDIP is set to the polynomial kernel $(1+\boldsymbol{x}_i^\top \boldsymbol{x}_j)^2$ or the Gaussian kernel $\exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2)$. KJDIP with these two kernels are respectively denoted as KJDIP-poly and KJDIP-rbf. For both JDIP and KJDIP, we fix hyperparameters $\sigma = 0.5, \lambda = 10^{-5}$, and tune the dimensionality d or $\widetilde{d}$ from the range $\{30, 50, 100, 200, 300\}$. The influence of these hyperparameters will be studied in Section VI-D.2. The initialization points $\boldsymbol{W}_{s0}$ and $\boldsymbol{W}_{t0}$ in Algorithm 1 are both set to the PCA projection matrix PCA($\boldsymbol{X}^{st}$), and the ones $\boldsymbol{B}_{s0}$ and $\boldsymbol{B}_{t0}$ in Algorithm 2 the KPCA [61] representation matrix KPCA($\boldsymbol{X}^{st}$). Here, $\boldsymbol{X}^{st} = (\boldsymbol{x}_1^s, \ldots, \boldsymbol{x}_{m_t}^t) \in \mathbb{R}^{D \times m_{st}}$ is the matrix containing all the unlabeled source and target data. In addition, the iteration for updating the target data labels is fixed at a maximum value $T = 10$. We will show in Section VI-D.3 that this value is adequate for the target data labels to converge.

We compare our approach with several kinds of domain adaptation methods: the covariance matching ones, the distribution matching ones (including marginal distribution matching, marginal and class-conditional distribution matching), the reconstruction ones, and others. In particular, methods that are based on covariance matching include ILS [22], KOT [13], and DACoM [14]. Methods that are based on distribution matching consist of TSL [32], DME-H [31], DAUT (distribution matching by unilateral transformations) [15], HCA [38], DANN [30], and JAN [3]. Methods that are based on reconstruction contain MCTL [49] and CRTL [50]. Moreover, we also compare our approach with other important domain adaptation techniques, including LDADA [7], ADR [70], CDAN [68], CADA-P [71], and MME [12]. For model selection of these methods, we follow the standard procedures as explained in the corresponding papers. Note that some of the methods such as TSL, DME-H, and DAUT are mainly proposed for unsupervised domain adaptation. As recommended by [7], here we extend them to the semi-supervised setting by considering labeled target samples in conjunction with source samples during classifier learning. Finally, following [7], [15], [21], we employ a linear SVM as the baseline classifier and the classifier for the domain adaptation methods, except the deep ones which inherently include a probabilistic classifier. The multi-class classification accuracy on the unlabeled target data (test data) is used as the performance measure for all the methods.

C. Experimental Results With Statistical Test

1) Results for Semi-Supervised Domain Adaptation: We report the experimental results on data sets Office, Satellite-Scene, and Office-Caltech in Table I, and the results on Office-Home in Table II. Note that in Table II, the results of the deep neural network methods DANN, ADR, CDAN, and MME are directly cited from [12], since the same experimental setting (VGG16 model and three-shot setting) is used here. For every

TABLE I

MEAN AND STANDARD DEVIATION OF CLASSIFICATION ACCURACY (%) ON DATA SETS OFFICE, SATELLITE-SCENE, AND OFFICE-CALTECH FOR THE SEMI-SUPERVISED DOMAIN ADAPTATION SETTING. THE BEST VALUE IN EACH ROW IS HIGHLIGHTED IN **BOLD**. THE SECOND BEST IS UNDERLINED

Data set	Task	SVM	TSL	DME-H	ILS	LDADA	KOT	DACoM	DAUT	MCTL	HCA	CRTL	JDIP	KJDIP-poly	KJDIP-rbf
Office (VGG-VD-16-FC$_7$)	A → D	86.3(1.6)	85.2(1.9)	85.6(1.9)	74.7(2.2)	87.9(2.0)	80.1(1.2)	85.6(1.9)	87.7(2.2)	86.8(1.7)	88.6(1.6)	88.9(1.9)	**91.1(1.8)**	88.5(1.7)	90.5(3.2)
	A → W	84.1(1.4)	82.7(1.5)	83.1(1.5)	72.6(1.5)	86.7(1.5)	78.6(1.6)	84.7(1.6)	85.3(1.5)	86.1(1.7)	88.4(2.4)	87.2(1.7)	90.8(1.5)	88.6(1.5)	**91.2(1.5)**
	D → A	69.9(1.4)	70.8(1.2)	70.8(1.1)	65.6(1.5)	72.8(1.4)	68.7(1.0)	67.9(1.7)	71.1(1.2)	71.0(1.2)	72.6(1.2)	72.4(1.1)	73.8(1.4)	73.5(0.7)	**73.9(1.0)**
	D → W	96.3(0.9)	95.0(1.0)	95.2(0.9)	91.4(0.9)	93.8(1.0)	92.7(1.2)	89.1(1.1)	95.3(0.9)	96.2(0.6)	94.4(0.7)	96.5(0.8)	96.8(1.0)	96.5(1.0)	**97.0(1.0)**
	W → A	69.2(1.1)	70.4(1.2)	70.5(1.2)	62.6(1.8)	72.7(1.1)	67.6(1.0)	68.4(2.4)	70.9(1.1)	71.0(1.1)	73.1(0.8)	72.0(1.0)	73.6(1.1)	73.8(0.8)	**74.3(0.9)**
	W → D	96.7(1.0)	95.8(0.8)	95.8(0.8)	92.3(1.7)	95.1(0.9)	93.8(1.0)	89.4(1.9)	96.0(1.0)	97.1(0.7)	96.6(0.9)	**97.4(1.0)**	97.2(1.0)	96.0(0.8)	96.4(1.0)
Satellite-Scene (VGG-M-FC$_7$)	B → S	77.3(3.9)	80.4(2.3)	79.3(2.1)	76.2(8.8)	90.1(8.8)	62.3(5.7)	94.4(3.0)	79.1(3.2)	73.7(4.5)	85.6(1.9)	82.4(3.5)	**94.7(2.3)**	93.6(3.2)	93.9(3.0)
	B → U	80.6(2.0)	82.3(2.5)	83.5(3.2)	84.9(5.9)	**94.6(3.7)**	76.2(5.0)	90.8(2.4)	81.9(2.3)	80.9(2.2)	87.4(1.4)	83.0(2.7)	92.2(2.5)	89.8(3.8)	93.3(2.1)
	S → B	75.4(4.5)	79.0(4.5)	78.5(4.6)	72.7(5.2)	86.8(5.2)	56.7(5.3)	88.8(3.0)	78.3(4.5)	79.8(3.5)	81.5(3.3)	79.1(4.2)	88.9(4.3)	89.2(2.7)	**91.2(3.6)**
	S → U	86.5(2.9)	85.9(3.3)	87.3(2.2)	92.0(2.8)	**96.8(0.6)**	89.3(2.7)	93.7(2.2)	87.3(2.6)	86.4(2.3)	92.2(1.6)	88.3(2.3)	94.5(1.6)	95.4(0.8)	95.2(1.8)
	U → B	76.5(4.6)	77.9(5.0)	77.9(3.6)	81.6(4.1)	83.8(3.6)	75.9(3.6)	85.9(3.4)	77.6(5.1)	81.9(3.5)	84.2(2.4)	82.8(3.5)	86.4(4.1)	83.3(2.5)	**87.7(3.4)**
	U → S	88.6(2.8)	88.6(3.2)	88.0(3.7)	93.7(2.1)	95.3(0.7)	90.8(4.1)	95.3(1.8)	87.7(3.2)	89.5(2.4)	86.7(2.8)	90.1(2.6)	95.3(2.6)	**96.3(1.2)**	96.1(2.4)
Office-Caltech (VGG-FC$_6$)	A → C	84.6(0.9)	83.7(1.0)	81.6(1.3)	79.1(1.8)	85.3(1.2)	84.3(1.1)	80.9(2.1)	84.0(0.8)	84.8(0.8)	85.1(1.0)	84.7(1.1)	**85.4(1.5)**	82.7(2.2)	82.5(2.7)
	A → D	86.5(2.7)	74.6(2.5)	86.9(3.7)	79.6(3.7)	88.4(3.2)	81.8(3.7)	89.0(3.8)	88.1(3.5)	87.7(2.6)	86.7(4.0)	89.7(3.0)	**90.3(2.3)**	87.0(4.3)	88.9(4.3)
	A → W	88.1(2.6)	87.2(1.9)	89.2(3.3)	86.0(3.5)	90.1(2.5)	85.0(2.0)	93.1(2.3)	90.3(3.5)	90.5(2.0)	91.9(1.9)	90.3(2.9)	93.2(3.1)	93.1(3.7)	**94.5(3.1)**
	C → A	90.4(1.2)	87.8(1.1)	87.7(1.1)	87.8(1.6)	91.1(0.8)	89.6(0.9)	89.9(1.8)	90.2(0.8)	90.4(1.2)	90.3(1.3)	89.3(1.3)	91.7(0.9)	**91.8(1.0)**	91.4(1.1)
	C → D	87.9(3.9)	79.5(3.9)	87.8(5.0)	82.8(3.7)	87.9(4.9)	85.9(6.8)	**91.2(3.2)**	89.4(4.3)	90.1(3.0)	85.2(3.4)	89.7(2.9)	88.8(5.0)	90.6(3.8)	90.9(4.4)
	C → W	90.8(3.2)	85.7(2.5)	89.0(2.8)	87.7(3.2)	90.5(2.3)	88.9(3.2)	93.9(2.3)	90.9(3.1)	91.2(1.8)	89.8(2.0)	91.2(3.0)	**94.7(2.9)**	94.0(3.0)	94.4(3.3)
	D → A	88.0(1.2)	83.6(1.9)	83.3(1.8)	83.2(3.9)	91.0(0.7)	86.4(1.7)	90.5(1.4)	88.1(1.0)	89.5(1.4)	89.7(1.6)	89.6(1.0)	**91.8(0.8)**	91.8(0.8)	91.4(0.9)
	D → C	77.0(2.2)	76.4(1.5)	78.8(1.3)	75.5(1.8)	80.5(1.3)	77.5(2.0)	81.5(2.0)	80.0(1.6)	79.8(1.5)	82.4(1.6)	80.5(1.8)	83.8(2.6)	**84.2(1.4)**	83.7(1.6)
	D → W	95.8(1.5)	96.7(1.5)	93.5(2.2)	91.6(2.8)	97.0(0.8)	94.8(1.4)	95.5(2.0)	96.3(1.3)	96.8(1.3)	96.9(1.2)	96.8(1.2)	**98.0(1.4)**	97.2(1.6)	97.2(1.3)
	W → A	89.2(1.5)	86.7(1.3)	86.6(1.2)	86.6(1.2)	91.4(0.7)	87.7(1.0)	90.4(1.5)	88.5(1.5)	89.8(1.1)	90.6(1.7)	89.5(1.0)	**91.8(1.1)**	91.8(1.2)	91.4(1.1)
	W → C	80.5(1.4)	81.8(1.7)	79.5(2.1)	77.1(2.7)	81.0(1.8)	80.6(1.6)	80.7(2.0)	81.6(1.3)	81.8(1.3)	82.9(1.7)	82.7(1.4)	84.1(2.7)	**85.0(0.6)**	84.5(1.0)
	W → D	94.0(3.3)	93.7(3.2)	92.8(2.9)	92.2(3.3)	93.0(2.6)	92.0(3.2)	91.2(1.9)	93.9(3.8)	95.2(2.9)	94.8(2.8)	95.3(2.2)	**96.2(2.5)**	95.5(3.9)	95.0(4.2)
	Avg.	85.0	83.8	84.7	82.1	88.5	82.0	87.6	85.8	86.2	87.4	87.1	90.2	89.6	**90.3**
	z-value	-4.3	-4.3	-4.3	-4.3	-3.5	-4.3	-3.9	-4.3	-4.1	-4.3	-4.1	–	-1.8	-0.3

TABLE II

CLASSIFICATION ACCURACY (%) ON OFFICE-HOME WITH VGG16 FEATURES FOR THE SEMI-SUPERVISED DOMAIN ADAPTATION SETTING

Task	SVM	DANN	ADR	CDAN	MME	JDIP	KJDIP-poly	KJDIP-rbf
A → C	47.4	50.0	49.3	46.0	**54.9**	52.2	49.5	52.0
A → P	70.0	69.5	69.9	74.7	75.7	77.0	74.9	**77.4**
A → R	72.8	72.3	73.3	71.4	75.3	75.9	73.4	**76.3**
C → A	57.0	56.4	56.3	52.9	61.1	61.8	60.2	**62.3**
C → P	70.6	69.8	71.4	71.2	**76.3**	75.7	74.4	75.4
C → R	70.3	68.7	69.3	65.9	72.9	73.2	72.3	**73.3**
P → A	56.6	56.3	55.8	50.3	59.2	61.3	60.5	**61.5**
P → C	46.5	52.4	47.8	45.1	**53.6**	51.5	49.2	50.9
P → R	73.7	73.6	73.6	70.8	**76.7**	76.5	75.6	76.1
R → A	63.6	63.7	62.8	62.1	65.7	66.9	66.4	**67.3**
R → C	48.4	56.1	49.0	50.2	**56.9**	52.5	51.2	52.3
R → P	79.0	77.9	78.1	80.9	**82.9**	81.6	80.1	80.7
Avg.	63.0	63.9	63.1	61.8	**67.6**	67.2	65.6	67.1

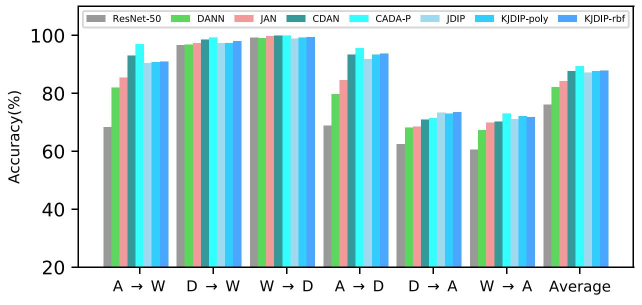

Fig. 3. Classification accuracy (%) on Office with ResNet-50 features for the unsupervised domain adaptation setting.

row in the table, the best result is highlighted in **bold**, and the second best is underlined.

To be strict in a statistical sense, we conduct the Wilcoxon signed-ranks test [16], [72] on the tasks from Table I to check whether the proposed approach is significantly better than the other shallow methods. The test uses a statistic z to compare the performance of two algorithms over multiple tasks. Specifically, in each task the mean classification accuracy is adopted as the performance measure of the algorithms. We fix JDIP as a control algorithm, and conduct 13 pairs of tests: SVM versus JDIP, …, and JDIP-rbf versus JDIP. The detailed description of the test procedure is presented in the supplementary material, and the resulting 13 z-values are reported in the last row of Table I. From Table I, we observe that the z-values for the first 11 pairs are all below the critical value -1.96. According to [16], [72], this indicates that with a significance level 0.05, JDIP is statistically better than the other shallow methods. Additionally, we also notice that the last two z-values -1.8 and -0.3 are both above -1.96, which shows that JDIP does not statistically outperform JDIP-poly and JDIP-rbf. For the comparison with the deep domain adaptation methods in Table II, we observe that our JDIP and KJDIP-rbf, as a shallow approach, perform on par with the state-of-the-art deep method MME [12] on this large Office-Home data set. To a certain extent, this demonstrates that our shallow solution, when accompanied with appropriate deep features, can better address the semi-supervised domain adaptation problem with less resources.

2) Results for Unsupervised Domain Adaptation: We report the experimental results on data sets Office (VGG-VD-16 features), Office-Caltech, and PIE-Multiview in Table III, and the results on Office with ResNet-50 features in Fig. 3. Note that, here, the results of the deep methods ResNet-50, DANN, JAN, CDAN, and CADA-P in Fig. 3 are directly cited from [71].

For the tasks from Table III, we repeat the Wilcoxon signed-ranks test to check whether JDIP is statistically better than the other shallow methods. The values of the test statistic z are reported in the last row of Table III. According to these values, we conclude that in a statistical sense, JDIP performs better than its shallow competitors, but does not outperform its kernel extensions JDIP-poly and JDIP-rbf. For the interesting

TABLE III

CLASSIFICATION ACCURACY (%) ON DATA SETS OFFICE, OFFICE-CALTECH, AND PIE-MULTIVIEW FOR THE UNSUPERVISED DOMAIN ADAPTATION SETTING

Data set	Task	SVM	TSL	DME-H	ILS	LDADA	KOT	DAUT	HCA	JDIP	KJDIP-poly	KJDIP-rbf
Office (VGG-VD-16-FC_7)	A → D	71.6	72.2	70.8	65.4	76.7	76.3	70.2	74.9	78.9	**81.7**	79.9
	A → W	68.2	70.4	65.8	65.3	78.1	74.3	66.4	76.0	78.9	79.0	**79.9**
	D → A	57.5	60.7	61.7	59.0	**68.3**	60.3	59.5	65.0	66.0	67.2	67.6
	D → W	95.2	95.6	96.6	90.1	91.1	90.2	**96.7**	96.2	96.2	95.8	96.5
	W → A	57.2	58.6	59.8	61.6	66.9	58.5	58.2	65.2	**69.3**	67.6	67.0
	W → D	98.4	98.4	98.8	95.8	96.3	95.0	99.4	**99.6**	98.2	98.2	98.0
Office-Caltech (VGG-FC_6)	A → C	84.8	84.0	81.6	82.5	**86.6**	84.8	84.6	85.9	85.5	86.4	85.8
	A → D	69.4	63.7	68.2	68.2	79.6	71.3	69.4	68.8	**89.2**	86.0	87.9
	A → W	77.3	81.7	72.2	83.7	85.8	76.9	77.3	89.2	90.2	**94.2**	91.2
	C → A	89.7	88.1	88.0	89.0	90.6	88.7	89.8	91.6	91.6	91.0	**92.4**
	C → D	76.4	70.1	73.2	72.6	86.0	72.6	77.1	75.8	**90.4**	86.0	90.4
	C → W	80.7	81.7	79.0	77.6	89.5	81.4	81.0	87.5	**90.2**	89.2	89.5
	D → A	72.8	75.8	63.0	75.9	87.7	76.5	72.8	85.4	89.0	**89.6**	89.4
	D → C	66.8	71.1	67.0	66.9	77.6	70.7	66.7	79.4	84.1	**84.6**	78.5
	D → W	96.9	97.3	91.2	95.3	95.3	94.6	96.9	98.0	**98.3**	96.6	97.6
	W → A	82.3	83.6	78.1	86.0	90.7	85.0	82.3	89.8	90.8	88.9	**92.1**
	W → C	75.2	79.9	71.2	79.6	**84.3**	78.2	74.8	83.7	80.8	80.9	83.5
	W → D	94.9	96.2	94.3	93.6	92.4	93.0	94.9	97.5	**99.4**	98.7	96.8
PIE-Multiview (Pixel)	M1 → M2	48.3	79.0	63.5	61.1	11.0	38.3	51.5	75.4	79.4	**81.4**	80.4
	M2 → M3	40.0	55.2	46.6	28.5	7.7	25.8	40.1	53.5	54.5	**56.5**	55.5
	M3 → M1	16.0	36.8	24.5	23.3	5.7	17.5	16.3	40.7	45.7	46.7	**47.7**
	Avg.	72.4	76.2	72.1	72.4	73.7	71.9	72.7	80.0	**83.2**	83.2	83.2
	z-value	-4.0	-3.8	-3.9	-4.0	-3.1	-4.0	-3.9	-3.1	–	-0.1	-1.0

but challenging comparison with the deep domain adaptation methods in Fig. 3, our method JDIP-rbf ranks the second in average performance (87.9%), which is next to the best performing deep method CADA-P (89.4%) [71] on this Office data set. Note that the base network structure for all the methods here is the ResNet-50 model. The comparison shows that our approach, as a shallow solution for the unsupervised domain adaptation problem, has very good competitiveness.

With an appropriate kernel and kernel parameters, generally the kernel model KJDIP is more expressive than the linear one JDIP. However, selecting the appropriate kernel and its parameters is not straightforward. We notice that in the experimental results, the performance of KJDIP with polynomial or Gaussian kernel in some cases is behind the JDIP. This is probably because that in those cases, KJDIP with that specific kernel and kernel parameters does not capture the relationship between the source and target data better than JDIP, thus resulting in a decrease in the accuracy of the model. Overall, the results for the semi-supervised and unsupervised settings on the one hand confirm that our domain adaptation assumption (*ie*, Assumption 1) is realistic for real world problems, and on the other hand reveal that our L^2-distance based joint distribution matching solution is advantageous in addressing the domain adaptation problem.

D. Empirical Analysis

Below, we empirically justify our L^2-distance estimation method, and analyze the properties of our JDIP method, including its hyperparameter sensitivity and convergence behavior.

1) Justification of L^2-Distance Estimation: Here we check whether the estimated L^2-distance is close to the true distance. Recall that in domain adaptation the joint distribution and the domain represent the same thing. Therefore, we consider two settings for the joint distribution $P(\boldsymbol{x}, y)$: $P(\boldsymbol{x}, y) = \text{Clipart}$ or $P(\boldsymbol{x}, y) = \text{Product}$, where the domains come from the Office-Home data set. Let $P^s(\boldsymbol{x}, y) = P^t(\boldsymbol{x}, y) = P(\boldsymbol{x}, y)$,

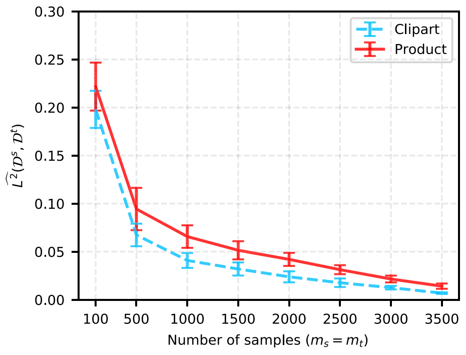

Fig. 4. Mean and standard deviation of the estimated L^2-distance $\widehat{L^2}(\mathcal{D}^s, \mathcal{D}^t)$, where $\mathcal{D}^s \sim P(\boldsymbol{x}, y), \mathcal{D}^t \sim P(\boldsymbol{x}, y)$, and $P(\boldsymbol{x}, y) = \text{Clipart}$ or $P(\boldsymbol{x}, y) = \text{Product}$.

then in either setting we have $L^2\big(P^s(\boldsymbol{x}, y), P^t(\boldsymbol{x}, y)\big) = 0$. Now we only need to draw random samples $\mathcal{D}^s \sim P(\boldsymbol{x}, y), \mathcal{D}^t \sim P(\boldsymbol{x}, y)$, and verify if $\widehat{L^2}(\mathcal{D}^s, \mathcal{D}^t)$ obtained from Eq. (12) approaches zero as the number of samples increases. In particular, we draw equal number of source samples and target samples, *ie*, $m_s = m_t$, and let the sample size vary in the range $\{100, 500, 1000, \ldots, 3500\}$. For each size, we repeat the random sampling procedure for 10 times to calculate the mean and standard deviation of the estimated value $\widehat{L^2}(\mathcal{D}^s, \mathcal{D}^t)$. The results are illustrated in Fig. 4. We observe from this figure that for both settings, $\widehat{L^2}(\mathcal{D}^s, \mathcal{D}^t)$ is close to zero from the very beginning. As the sample size increases, it approaches zero very fast. This confirms that our L^2-distance estimation method is reliable and reasonable.

2) Hyperparameter Sensitivity: We then investigate the sensitivity of JDIP with respect to different choices of its hyperparameters: d, λ, and σ. Specifically, we run the experiments on the tasks S → U (Satellite-Scene), C → W (Office-Caltech), and D → C (Office-Caltech) with the parameter ranges $d \in \{50, 100, \ldots, 300\}$, $\lambda \in \{10^{-6}, 10^{-5}, \ldots, 10^{-2}\}$, and $\sigma \in \{0.4, 0.5, \ldots, 0.8\}$. Following [16], we use a heat

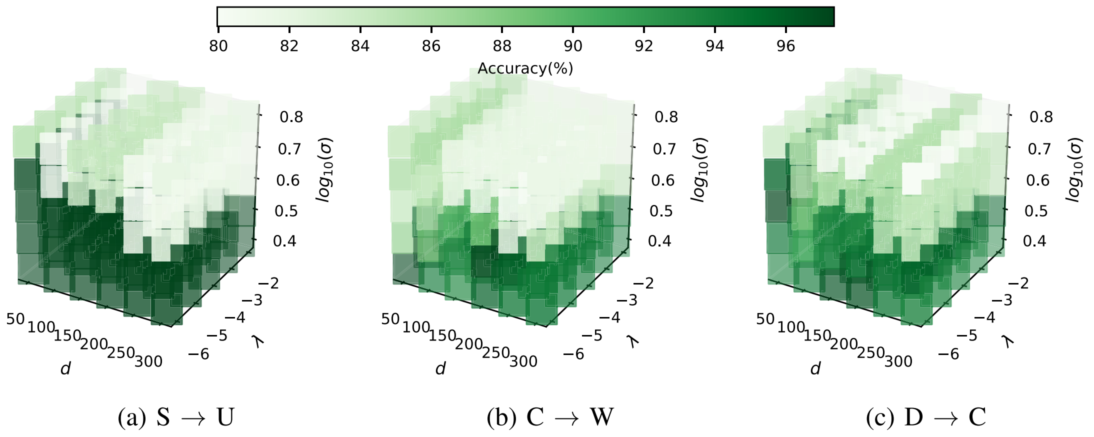

Fig. 5. Hyperparameter sensitivity of JDIP on three domain adaptation tasks.

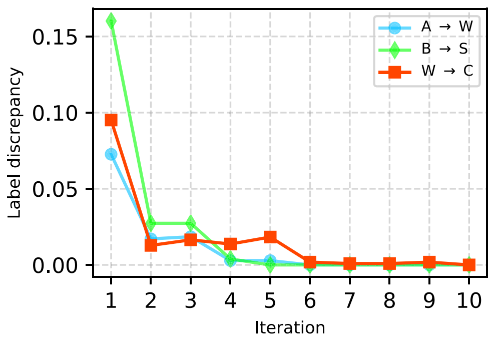

Fig. 6. Empirical convergence analysis with respect to the change of the predicted target labels over 3 domain adaptation tasks as the number of iterations increases.

map to demonstrate the sensitivity result. Fig. 5 shows the classification accuracy of JDIP versus different parameter pairs in terms of (d, λ, σ) on the three domain adaptation tasks. From Fig. 5, we observe that on the tested evaluations JDIP attains its superior performance when $d \in [100, 300]$ and $\lambda \in [10^{-6}, 10^{-3}]$. Besides, we also notice that JDIP is sensitive to its parameter σ: smaller values for σ are significantly better than larger ones. Therefore, as a general guideline for choosing the hyperparameters, we would suggest picking $d \in [100, 300]$, $\lambda \in [10^{-6}, 10^{-3}]$, and $\sigma \in [0.4, 0.5]$.

3) Convergence Analysis: Finally we experimentally show the convergence of the predicted target data labels in Fig. 6 on three domain adaptation tasks: A → W (Office), B → S (Satellite-Scene), and W → C (Office-Caltech). From Fig. 6, we observe that for each adaptation task the label discrepancy decreases as the iteration proceeds, and eventually converges to zero after 6 rounds. According to the definition of the label discrepancy in Eq. (23), this means that the target labels stop changing and become stable after 6 iterations, which is a very desirable property here.

VII. Conclusion and Future Work

In this research, we introduce a novel approach to address the semi-supervised and unsupervised domain adaptation problems. Our approach exploits linear or nonlinear projections to match the source and target joint distributions. Specifically, we make use of the L^2-distance for comparing the joint distributions, and propose a least square method to estimate it without distribution estimation. Additionally, we also derive an error bound based on the L^2-distance for theoretically explaining our approach. Experimental results on several representative domain adaptation data sets confirm that our solution is advantageous in addressing the domain adaptation problem.

The proposed approach can be further studied in several directions. For example, the most crucial and difficult one is to explore the theoretical convergence of our algorithm. Besides, from the perspective of hypothesis space, we can also explore the deep neural networks to serve as the feature mappings, which, as evidence by [12], [68], have their advantages in handling large-scale visual domain adaptation tasks.

Acknowledgment

The authors would like to thank the associate editor and the reviewers for their valuable comments and suggestions.

References

[1] V. N. Vapnik, *Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 1998.

[2] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 819–827.

[3] M. Long, J. Wang, and M. Jordan, "Deep transfer learning with joint adaptation networks," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 2208–2217.

[4] W. M. Kouw and M. Loog, "A review of domain adaptation without target labels," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Oct. 7, 2019, doi: 10.1109/TPAMI.2019.2945942.

[5] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 7167–7176.

[6] M. Ghifary, D. Balduzzi, W. B. Kleijn, and M. Zhang, "Scatter component analysis: A unified framework for domain adaptation and domain generalization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 7, pp. 1414–1430, Jul. 2017.

[7] H. Lu, C. Shen, Z. Cao, Y. Xiao, and A. van den Hengel, "An embarrassingly simple approach to visual domain adaptation," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3403–3417, Jun. 2018.

[8] S. Li, S. Song, G. Huang, Z. Ding, and C. Wu, "Domain invariant and class discriminative feature learning for visual domain adaptation," *IEEE Trans. Image Process.*, vol. 27, no. 9, pp. 4260–4273, Sep. 2018.

[9] A. Chadha and Y. Andreopoulos, "Improved techniques for adversarial discriminative domain adaptation," *IEEE Trans. Image Process.*, vol. 29, pp. 2622–2637, 2020.

[10] W. Zellinger, B. Moser, T. Grubinger, E. Lughofer, T. Natschläger, and S. Saminger-Platz, "Robust unsupervised domain adaptation for neural networks via moment alignment," *Inf. Sci.*, vol. 483, pp. 174–191, May 2019.

[11] J. Liang, R. He, Z. Sun, and T. Tan, "Aggregating randomized clustering-promoting invariant projections for domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1027–1042, May 2019.

[12] K. Saito, D. Kim, S. Sclaroff, T. Darrell, and K. Saenko, "Semi-supervised domain adaptation via minimax entropy," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 8050–8058.

[13] Z. Zhang, M. Wang, Y. Huang, and A. Nehorai, "Aligning infinite-dimensional covariance matrices in reproducing kernel Hilbert spaces for domain adaptation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 3437–3445.

[14] L. Li and Z. Zhang, "Semi-supervised domain adaptation by covariance matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 11, pp. 2724–2739, Nov. 2019.

[15] A. Kumagai and T. Iwata, "Unsupervised domain adaptation by matching distributions based on the maximum mean discrepancy via unilateral transformations," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 4106–4113.

[16] S. Chen, L. Han, X. Liu, Z. He, and X. Yang, "Subspace distribution adaptation frameworks for domain adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jan. 24, 2020, doi: 10.1109/TNNLS.2020.2964790.

[17] Z. Zhang, M. Wang, and A. Nehorai, "Optimal transport in reproducing kernel Hilbert spaces: Theory and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, pp. 1741–1754, Jul. 2020.

[18] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Heterogeneous domain adaptation via nonlinear matrix factorization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 984–996, Mar. 2020.

[19] P. P. Busto and J. Gall, "Open set domain adaptation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 754–763.

[20] J. Zhang, Z. Ding, W. Li, and P. Ogunbona, "Importance weighted adversarial nets for partial domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Mar. 2018, pp. 8156–8164.

[21] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2058–2065.

[22] S. Herath, M. Harandi, and F. Porikli, "Learning an invariant Hilbert space for domain adaptation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3956–3965.

[23] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *J. Statist. Planning Inference*, vol. 90, no. 2, pp. 227–244, 2000.

[24] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 601–608.

[25] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Büenau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," *Ann. Inst. Statist. Math.*, vol. 60, no. 4, pp. 699–746, 2008.

[26] S. Bickel, M. Brückner, and T. Scheffer, "Discriminative learning under covariate shift," *J. Mach. Learn. Res.*, vol. 10, pp. 2137–2155, Sep. 2009.

[27] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[28] N. Courty, R. Flamary, D. Tuia, and A. Rakotomamonjy, "Optimal transport for domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 9, pp. 1853–1865, Sep. 2017.

[29] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 97–105.

[30] Y. Ganin *et al.*, "Domain-adversarial training of neural networks," *J. Mach. Learn. Res.*, vol. 17, no. 59, pp. 1–35, 2016.

[31] M. Baktashmotlagh, M. Harandi, and M. Salzmann, "Distribution-matching embedding for visual domain adaptation," *J. Mach. Learn. Res.*, vol. 17, no. 108, pp. 1–30, Jul. 2016.

[32] S. Si, D. Tao, and B. Geng, "Bregman divergence-based regularization for transfer subspace learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 7, pp. 929–942, Jul. 2010.

[33] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 137–144.

[34] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf, "Domain adaptation with conditional transferable components," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2839–2848.

[35] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3730–3739.

[36] B. Quanz, J. Huan, and M. Mishra, "Knowledge transfer with low-quality data: A feature extraction issue," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 10, pp. 1789–1802, Oct. 2012.

[37] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2013, pp. 2200–2207.

[38] Y. Liu, W. Tu, B. Du, L. Zhang, and D. Tao, "Homologous component analysis for domain adaptation," *IEEE Trans. Image Process.*, vol. 29, no. 1, pp. 1074–1089, Jul. 2019.

[39] Y. Chen, S. Song, S. Li, and C. Wu, "A graph embedding framework for maximum mean discrepancy-based domain adaptation algorithms," *IEEE Trans. Image Process.*, vol. 29, pp. 199–213, 2020.

[40] B. Póczos, L. Xiong, and J. Schneider, "Nonparametric divergence estimation with applications to machine learning on distributions," in *Uncertainty in Artificial Intelligence*. Arlington, VA, USA: AUAI Press, 2011.

[41] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2009.

[42] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2016, pp. 443–450.

[43] J. Zhang, W. Li, and P. Ogunbona, "Joint geometrical and statistical alignment for visual domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5150–5158.

[44] J. Li, M. Jing, K. Lu, L. Zhu, and H. T. Shen, "Locality preserving joint transfer for domain adaptation," *IEEE Trans. Image Process.*, vol. 28, no. 12, pp. 6103–6115, Dec. 2019.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[46] L. Zhang and D. Zhang, "Robust visual knowledge transfer via extreme learning machine-based domain adaptation," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4959–4973, Oct. 2016.

[47] Z. Ding, N. M. Nasrabadi, and Y. Fu, "Semi-supervised deep domain adaptation via coupled neural networks," *IEEE Trans. Image Process.*, vol. 27, no. 11, pp. 5214–5224, Nov. 2018.

[48] L. Zhang, W. Zuo, and D. Zhang, "LSDT: Latent sparse domain transfer learning for visual adaptation," *IEEE Trans. Image Process.*, vol. 25, no. 3, pp. 1177–1191, Mar. 2016.

[49] L. Zhang, S. Wang, G. Huang, W. Zuo, J. Yang, and D. Zhang, "Manifold criterion guided transfer learning via intermediate domain generation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3759–3773, Dec. 2019.

[50] S. Wang, L. Zhang, W. Zuo, and B. Zhang, "Class-specific reconstruction transfer learning for visual recognition across domains," *IEEE Trans. Image Process.*, vol. 29, pp. 2424–2438, 2020.

[51] Z. Ding and Y. Fu, "Robust transfer metric learning for image classification," *IEEE Trans. Image Process.*, vol. 26, no. 2, pp. 660–670, Feb. 2017.

[52] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 213–226.

[53] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proc. CVPR*, 2017, pp. 5018–5027.

[54] L. Wasserman, *All of Statistics: A Concise Course in Statistical Inference*. New York, NY, USA: Springer, 2013.

[55] M. Sugiyama, T. Kanamori, T. Suzuki, M. D. Plessis, S. Liu, and I. Takeuchi, "Density-difference estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 683–691.

[56] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *J. Mach. Learn. Res.*, vol. 10, pp. 1391–1445, Jul. 2009.

[57] Q. Que and M. Belkin, "Back to the future: Radial basis function networks revisited," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2016, pp. 1375–1383.

[58] M. Sugiyama, *Introduction to Statistical Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 2015.

[59] S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer, "Multi-task learning for HIV therapy screening," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 56–63.

[60] L. Zhang, J. Fu, S. Wang, D. Zhang, Z. Dong, and C. L. P. Chen, "Guide subspace learning for unsupervised domain adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 1, 2019, doi: 10.1109/TNNLS.2019.2944455.

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

[62] Y.-Y. Lin, T.-L. Liu, and C.-S. Fuh, "Multiple kernel learning for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1147–1160, Jun. 2011.

[63] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 2211–2268, Jul. 2011.

[64] T. Zhang and Y. Yang, "Robust PCA by manifold optimization," *J. Mach. Learn. Res.*, vol. 19, no. 80, pp. 1–39, 2018.

[65] S. Kurokiyz, N. Charoenphakdeey, H. Baoyz, J. Hondayz, I. Satoyz, and M. Sugiyama, "Unsupervised domain adaptation based on source-guided discrepancy," in *Proc. 33rd AAAI Conf. Artif. Intell.*, 2019, pp. 4122–4129.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[67] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. CVPR*, Jun. 2012, pp. 2066–2073.

[68] M. Long, Z. Cao, J. Wang, and M. Jordan, "Conditional adversarial domain adaptation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1640–1650.

[69] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–14.

[70] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Adversarial dropout regularization," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–15.

[71] V. Kurmi, S. Kumar, and V. Namboodiri, "Attending to discriminative certainty for domain adaptation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 491–500.

[72] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

Sentao Chen received the B.S. degree in statistics from the Guangdong University of Technology, Guangzhou, China, in 2012, and the Ph.D. degree in software engineering from the South China University of Technology, Guangzhou, in 2020. His current research interests include statistical machine learning, domain adaptation, and domain generalization.

Mehrtash Harandi (Member, IEEE) is currently a Senior Lecturer with the Department of Electrical and Computer Systems Engineering, Monash University. He is also a contributing Research Scientist with Machine Learning Research Group (MLRG), Data61/CSIRO, and an Associated Investigator with Australian Center for Robotic Vision (ACRV). His current research interest includes theoretical and computational methods in machine learning, computer vision, signal processing, and Riemannian geometry.

Xiaona Jin received the B.S. degree in information and computing science from the South China University of Technology, Guangzhou, China, where she is currently pursuing the M.Sc. degree in software engineering with the School of Software Engineering. Her research interests include domain adaptation and computer vision.

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