AIMS Mathematics, 9(3): 6628–6647.
DOI: 10.3934/math.2024323
Received: 18 August 2023
Revised: 24 October 2023
Accepted: 05 November 2023
Published: 06 February 2024

http://www.aimspress.com/journal/Math

Research article

Unsupervised domain adaptation with deep network based on discriminative class-wise MMD

Hsiau-Wen Lin[1], Yihjia Tsai[2], Hwei Jen Lin[2,*], Chen-Hsiang Yu[3] and Meng-Hsing Liu[2]

[1] Department of Information Management, Chihlee University of Technology, Taipei, Taiwan
[2] Department of Computer Science and Information Engineering, Tamkang University, Taipei, Taiwan
[3] Multidisciplinary Graduate Engineering, College of Engineering, Northeastern University, Boston, MA, USA

* **Correspondence:** Email: 086204@mail.tku.edu.tw; Tel: +886226215656; Fax: +886226219749.

Abstract: General learning algorithms trained on a specific dataset often have difficulty generalizing effectively across different domains. In traditional pattern recognition, a classifier is typically trained on one dataset and then tested on another, assuming both datasets follow the same distribution. This assumption poses difficulty for the solution to be applied in real-world scenarios. The challenge of making a robust generalization from data originated from diverse sources is called the domain adaptation problem. Many studies have suggested solutions for mapping samples from two domains into a shared feature space and aligning their distributions. To achieve distribution alignment, minimizing the maximum mean discrepancy (MMD) between the feature distributions of the two domains has been proven effective. However, this alignment of features between two domains ignores the essential class-wise alignment, which is crucial for adaptation. To address the issue, this study introduced a discriminative, class-wise deep kernel-based MMD technique for unsupervised domain adaptation. Experimental findings demonstrated that the proposed approach not only aligns the data distribution of each class in both source and target domains, but it also enhances the adaptation outcomes.

Keywords: maximum mean discrepancy (MMD); unsupervised domain adaptation; transfer learning; reproduced kernel Hilbert space; pseudo labels
Mathematics Subject Classification: 15A06, 15A15, 68T07

1. Introduction

Deep learning techniques have proven successful in various computer vision fields, such as image classification [1], object detection [2], and semantic segmentation [3]. However, the effectiveness of deep learning relies heavily on large, labeled training datasets, which could be labor-intensive to annotate. When dealing with large unlabeled datasets, it is often impractical to label enough data for training a deep learning model. An alternative approach is transfer learning, where labeled data from related domains (source domain) are utilized to enhance the model's performance in the domain of interest (target domain). Transfer learning is the process of applying knowledge learned from a labeled source domain to a target domain, where labeled data may be limited or unavailable.

Pan [4] classified transfer learning into three categories according to labeled data in the two domains used during training. They are (1) inductive transfer learning: When the target domain data is labeled, irrespective of whether the source domain data is labeled or not; (2) transductive transfer learning: When only the source domain data is labeled, while the target domain data remains unlabeled; and (3) unsupervised transfer learning: When both domains lack labels. Transductive transfer learning can be further divided into two types: (1) domain adaptation: When both domains use the same attributes but have different marginal probability distributions; and (2) sample selection bias: When the sample spaces or data types of the two domains are different, such as images in the source domain and text in the target domain. This paper focuses on unsupervised domain adaptation (UDA), which aims to minimize the distribution discrepancy between data from two domains, enabling successful knowledge transfer from the source domain to the target domain.

Currently, numerous domain adaptation methods have been researched and developed [4–23]. These methods fall into three main categories [4]: Instance reweighting methods [5–7], feature extraction methods [8–11], and classifier adaptive approaches [12,13]. Feature extraction methods aim to learn domain-invariant feature representations and are broadly categorized into two types [14]: Adversarial learning-based approaches [15–17] and statistics-based approaches [18–20]. Adversarial learning-based methods seek to achieve domain-invariant features by generating images or feature representations from different domains. For instance, the deep reconstruction classification network (DRCN) [21] establishes a classifier for labeled source domain data and constructs a domain-invariant feature representation shared with unlabeled target domain data. Statistical methods involve defining a suitable measure of difference or distance between two distinct distributions [18,24–29]. Various distance metrics, such as quadratic [30], Kullback-Leibler [31] and Mahalanobis [32], have been proposed over the years. However, these methods are not easily adaptable to different domain adaptation (DA) models and may not effectively describe complex distributions like conditional and joint distributions due to theoretical limitations. In recent years, the MMD [28], initially used for two-sample testing, has been found to be effective in calculating the distance between sample distributions from two domains in feature space. It facilitates alignment between the distributions by minimizing the MMD between them. The method presented in this paper falls under this category. Long et al. [9] introduced the regularization of MMD, utilizing it to reduce the distribution difference between the feature distributions of two domains in hidden layers of deep adaptation networks.

The use of MMD focuses mainly on aligning the overall distribution of two domains, but often falls short in ensuring precise alignment of data within the same category across domains. In response, Long et al. [19] proposed the class-wise maximum mean discrepancy (CWMMD) to enhance robust domain adaptation. The two-domain samples are linearly mapped into a common feature space and the

MMD for each category is calculated, then summed to obtain a CWMMD. Wang et al. [33] highlighted that minimizing the MMD is equivalent to minimizing the overall data variance while simultaneously maximizing the intra-class distances of the source and target domains, leading to a decrease in feature discriminativeness. They adjusted balance parameters to mitigate this issue but were limited to linear transformations in the feature space. However, they used the L2 norm as the MMD estimator in a linearly transformed feature space. It is worth noting that the L2 norm is not well suited for general estimation [34,35], and that linear transformations may not adequately capture complex data relationships, especially if nonlinear mappings are required.

In contrast, deep neural networks, particularly convolutional neural networks (CNNs), learn powerful and expressive nonlinear transformations. This paper proposes a method to improve upon this, which involves training a CNN architecture, so that the model automatically learns feature representations that are well-suited for the task at hand. Furthermore, the loss function used in the domain adaptation process can be efficiently evaluated in a reproduced kernel hilbert space (RKHS). This facilitates effective alignment of data belonging to the same class from both the source and target domains in the shared feature space.

2. Materials and methods

This section presents the related research, including pseudo labels and different variants of MMD.

2.1. Pseudo labels

Computing a class-level MMD during training requires the use of pseudo-labels for unlabeled target domain data. A simple way to generate pseudo labels is directly applying formula (1) to the source domain model [14]; that is, input the target sample x_t into the source domain model $f = \boldsymbol{C}\cdot\boldsymbol{F}$, which comprises a feature extractor $\boldsymbol{F}$ and a classifier $\boldsymbol{C}$, to obtain the softmax result of classification $\delta = (\delta_1, \delta_2, \dots, \delta_C)$, and then set the index of the maximum of components in the output vector δ as the pseudo label for the target sample x_t. However, due to domain shift, the pseudo-label generated by this method may have large bias. Instead, this study adopts another method, the self-supervised pseudo-label strategy proposed by Liang et al. [36]. The strategy first uses the current target domain model to calculate the centroid of each category for the target domain data, which is similar to weighted *K*-means clustering, as shown in formula (2). These centroids robustly represent the distribution of different classes in the target domain data. Next, the category of the nearest centroid for each target domain data is obtained as its pseudo label, as shown in formula (3), where $D_{\cos}(a, b)$ means the cosine distance between a and b. The new pseudo-label is then utilized to recalculate the centroid, as shown in formula (4), and update the pseudo-label again, as shown in formula (5). Finally, with the new pseudo labels, the target model is self-supervised using the cross-entropy loss function, as shown in (6).

$$\hat{y}_t = \arg\max_{1\le k\le C} \delta_k(f(x_t)), \tag{1}$$

$$C_k^{(0)} = \frac{\sum_{x_t\in X_t}\delta_k(f(x_t))\boldsymbol{F}(x_t)}{\sum_{x_t\in X_t}\delta_k(f(x_t))},\ k = 1, 2, \dots, C, \tag{2}$$

$$\hat{y}_t = \arg\min_{1\le k\le C} D_{\cos}(\boldsymbol{F}(x_t), C_k^{(0)}), \tag{3}$$

$$C_k^{(1)} = \frac{\sum_{x_t \in X_t} \mathbf{1}(\hat{y}_t = k) \boldsymbol{F}(x_t)}{\sum_{x_t \in X_t} \mathbf{1}(\hat{y}_t = k)}, \tag{4}$$

$$\hat{y}_t = \arg \min_{1 \le k \le C} D_{\cos}(\boldsymbol{F}(x_t), C_k^{(1)}), \tag{5}$$

$$L_T^{ssl}(f_t; X_t, \hat{Y}_t) = -E_{(x, \hat{y}_t) \in X_s \times \hat{Y}_t} \sum_{k=1}^{K} \mathbf{1}[k = \hat{y}_t] \log \delta_k (f(x)). \tag{6}$$

2.2. MMD from two-sample tests

The MMD is a distance measure between feature means. Gretton et al. [28] introduced an MMD measure, which involves embedding distribution metrics in the RKHS and using it to conduct a two-sample test for detecting differences between two unknown distributions, p and q. The purpose of this test is to draw two sets of samples X and Y from these distributions and to determine whether p and q are different distributions. They applied a kernel-based MMD to two-sample tests on various problems and achieved excellent performance. Furthermore, these kernel-based MMDs have been shown to be consistent, asymptotically normal, robust to model misspecification, and have been successfully applied to various problems, including transfer learning [29], kernel Bayesian inference [37], approximate Bayesian computation [38], two-sample testing [28], optimal degree-of-fit testing [39], generating moment matching networks (GMMN) [40], and autoencoders [41].

Gaussian kernel-based MMDs are commonly used estimators, which have the key property of universality, allowing estimators to converge to the best approximation for generating distribution of the (unknown) data in the model, without making any assumptions about this distribution. In contrast, the L2 norm lacks the above properties, suffers from the curse of dimensionality, and is not suitable for universal estimation [34,35]. Furthermore, Gaussian kernel-based MMDs also serve as an effective measure for domain differences in UDA scenarios, and their computation is streamlined by applying a kernel function directly to the samples.

2.2.1. Formulation of MMD

The squared MMD in an RKHS, denoted as $\|\mu_p - \mu_q\|_{\mathcal{H}}^2$, can be straightforwardly expressed using kernel functions. Additionally, it is possible to easily derive an unbiased estimate for finite samples. Considering independent random variables x and x' from distribution p, as well as independent random variables y and y' from distribution q, let $\mathcal{H}$ be a universal RKHS with unit ball denoted $\mathcal{F}$, and one can give the squared MMD, as shown in (7) [32], where $\phi(\cdot)$ is a function mapping the samples to $\mathcal{H}$, $\mu_p = \mathbb{E}_{x \sim p}[\phi(x)]$ and $\mu_q = \mathbb{E}_{y \sim q}[\phi(y)]$ representing the kernel mean embeddings, and k is set to the commonly used Gaussian kernel, as shown in (8). An unbiased empirical estimate is given in (9), where $X = \{x_1, \ldots, x_m\}$ and $Y = \{y_1, \ldots, y_n\}$ are two sets randomly sampled from two probability distributions p and q, respectively. However, there is no definitive method for selecting the bandwidth σ of the kernel k in (9). Gretton et al. [28] suggested using the median distance between samples as the bandwidth, but did not verify that this choice is optimal.

$$\big(MMD(\mathcal{F}, p, q)\big)^2 = \|\mu_p - \mu_q\|_{\mathcal{H}}^2 = \langle \mu_p - \mu_q, \mu_p - \mu_q \rangle_{\mathcal{H}} = \langle \mu_p, \mu_p \rangle_{\mathcal{H}} + \langle \mu_q, \mu_q \rangle_{\mathcal{H}} - 2\langle \mu_p, \mu_q \rangle_{\mathcal{H}}$$

$$= \mathbb{E}_{x,x'\sim p}[\langle\phi(x),\phi(x')\rangle_{\mathcal{H}}] + \mathbb{E}_{y,y'\sim q}[\langle\phi(y),\phi(y')\rangle_{\mathcal{H}}] - 2\mathbb{E}_{x\sim p,y\sim q}[\langle\phi(x),\phi(y)\rangle_{\mathcal{H}}]$$

$$= \mathbb{E}_{x,x'\sim p}[k(x,x')] + \mathbb{E}_{y,y'\sim q}[k(y,y')] - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)], \tag{7}$$

$$k(a,b) = \exp\left(-\frac{\|a-b\|_2^2}{2\sigma_\phi^2}\right), \tag{8}$$

$$\left(MMD_u(\mathcal{F},X,Y)\right)^2 = \frac{1}{m(m-1)}\sum_{i\neq j}^{m} k(x_i,x_j) + \frac{1}{n(n-1)}\sum_{i\neq j}^{n} k(y_i,y_j) - \frac{2}{mn}\sum_{i,j}^{m,n} k(x_i,y_j). \tag{9}$$

2.2.2. Class-wise MMD based on L2-norm

Although MMD has been commonly used in cross-domain problems, minimizing the MMD between the samples from two domains only narrows their marginal distributions. Long et al. [8] proposed joint distribution adaptation (JDA), which jointly adapts marginal and conditional distributions in a reduced-dimensional principal component space and constructs new feature representations. They adopted a principle component analysis (PCA) transformation and minimized the Euclidean distance between the sample means of the two domains in a reduced-dimensional principal component space. They referred to this method as MMD for marginal distribution, as shown in formula (10), where $X_s \in R^{d\times n_s}$ and $X_t \in R^{d\times n_t}$ are the samples from the source and target domains, respectively. Here, n_s and n_t are the numbers of samples from the source and target domains, d is the sample dimension, and φ represents a linear transformation function. Let A be the $d \times K$ standard matrix of the linear transformation function φ. Formula (10) can then be rewritten as formula (11), where $X_{st} = [X_s|X_t]$ and $M_0 \in R^{n_{st}\times n_{st}}$ are calculated as shown in formula (12).

$$MMD^2 = \left\| \frac{1}{n_s}\sum_{x_i\in X_s}\varphi(x_i) - \frac{1}{n_t}\sum_{x_j\in X_t}\varphi(x_j) \right\|_2^2, \tag{10}$$

$$MMD^2 = \left\| \frac{1}{n_s}\sum_{x_i\in X_s}A^T x_i - \frac{1}{n_t}\sum_{x_j\in X_t}A^T x_j \right\|_2^2 = tr(A^T X_{st} M_0 X_{st}^T A), \tag{11}$$

$$(M_0)_{ij} = \begin{cases} \frac{1}{n_s n_s}, & x_i, x_j \in X_s \\ \frac{1}{n_t n_t}, & x_i, x_j \in X_t \\ \frac{-1}{n_s n_t}, & \text{otherwise} \end{cases}. \tag{12}$$

In addition to the MMD for marginal distribution, they proposed also the MMD for conditional distribution. However, during empirical estimation, to obtain samples for each category, labels for target domain samples that do not exist need to be provided. To address this, they suggested using pseudo labels for the target samples, which can be obtained either from the current classifier trained on the samples from the source domain or through other methods. They named the MMD for conditional distribution as class-wise MMD (CWMMD), as shown in Eq (13). Here, X_s^c and X_t^c represent the data samples of the c^{th} category from the source and target domains, respectively, while n_s^c and n_s^c are the respective sample sizes and the calculation of $M_c \in R^{n_{st}^c\times n_{st}^c}$ is detailed in (14).

$$CWMMD^2 = \sum_{c=1}^{C} || \frac{1}{n_s^c} \sum_{x_i \in X_s^c} A^T x_i - \frac{1}{n_t^c} \sum_{x_j \in X_t^c} A^T x_j ||_2^2 = \sum_{c=1}^{C} tr(A^T X_{st} M_c X_{st}^T A), \quad (13)$$

$$(M_c)_{ij} = \begin{cases} \frac{1}{n_s^c n_s^c}, & x_i, x_j \in X_s^c \\ \frac{1}{n_t^c n_t^c}, & x_i, x_j \in X_t^c \\ \frac{-1}{n_s^c n_t^c}, & x_i \in X_s^c, x_j \in X_t^c \text{ or } x_j \in X_s^c, x_i \in X_t^c \\ 0, & \text{otherwise} \end{cases}. \quad (14)$$

In JDA, both marginal distribution discrepancy and conditional distribution discrepancy across domains are simultaneously minimized. Consequently, the optimization problem for JDA is resolved by combining Eqs (11) and (13), as shown in Eq (15). In (15), the constraint condition $A^T X_{st} H_{st} X_{st}^T A = I_{K \times K}$ limits the overall data variation to a fixed value, ensuring that data information on the subspace is statistically retained to some extent. $\|A\|_F^2$ controls the size of the matrix A, and α is a regularization parameter ensuring a well-defined optimization problem. It is important to note that $H_{st} = I_{n_{st} \times n_{st}} - \frac{1}{n_{st}} 1_{n_{st} \times n_{st}}$ is a centering matrix, where $n_{st}^c = n_s^c + n_s^c$ and $1_{n_{st} \times n_{st}}$ is a matrix of size $n_{st} \times n_{st}$ with all elements being one.

$$\min_A \sum_{c=0}^{C} tr(A^T X_{st} M_c X_{st}^T A) + \alpha \|A\|_F^2 \ s.t. \ A^T X_{st} H_{st} X_{st}^T A = I_{K \times K}. \quad (15)$$

2.2.3. Discriminative CWMMD based on L2-norm

Wang et al. [33] proposed an insight into the working principle of MMD and theoretically revealed its high degree of agreement with human transferable behavior. In Figure 1 [33], when considering a pair of classes labeled "desktop computers", respectively, from the source and target domains, the process of minimizing the MMD between these two distributions involves two key transformations. They are (1) two relatively small red circles (hollow and mesh circles) transformed into larger red ones, which are magnified; i.e., maximizing their specific intra-class distance and (2) two tiny red circles gradually moving closer along their specific arrows; i.e., minimizing their joint variance. This process is analogous to how humans abstract common features to encompass all possible appearances, but the detailed information is heavily decayed. Wang et al. also theoretically demonstrated this insight.

Let $(S(A,X))_{inter}^c = tr(A^T X_{st} M_c X_{st}^T A)$ denote the inter-class distance (i.e., square of MMD) between the cth class data in the source domain and the target domain in the transformation space according to the transformation matrix A, and let $S_{inter} = \sum_{c=1}^{C} (S(A,X))_{inter}^c$, then (15) is written as (16). Wang et al. derived $S_{inter} = S_{var} - S_{intra}$, so (16) is written as (17), where S_{intra} represents the intra-class distance, and S_{var} is the variance of the entire data. Therefore, minimizing the inter-class distance S_{inter} is equivalent to maximizing their variation S_{var}, and maximizing the intra-class distance S_{intra} at the same time, which will reduce feature discriminativeness. To address this, a trade-off parameter is introduced to adjust the hidden intra-class distance in S_{inter}, as shown in Eq (18). They obtained an optimal linear transformation matrix A, thus minimizing the loss evaluated in this transformation space.

$$\min_A [S_{inter} + MMD^2 + \alpha \|A\|_F^2] \ s.t. \ A^T X_{st} H_{st} X_{st}^T A = I_{k \times k}, \tag{16}$$

$$\min_A [S_{var} - S_{intra} + MMD^2 + \alpha \|A\|_F^2] \ s.t. \ A^T X_{st} H_{st} X_{st}^T A = I_{k \times k}, \tag{17}$$

$$\min_A [S_{var} + \beta \cdot S_{intra} + MMD^2 + \alpha \|A\|_F^2] \ s.t. \ A^T X_{st} H_{st} X_{st}^T A = I_{k \times k}. \tag{18}$$

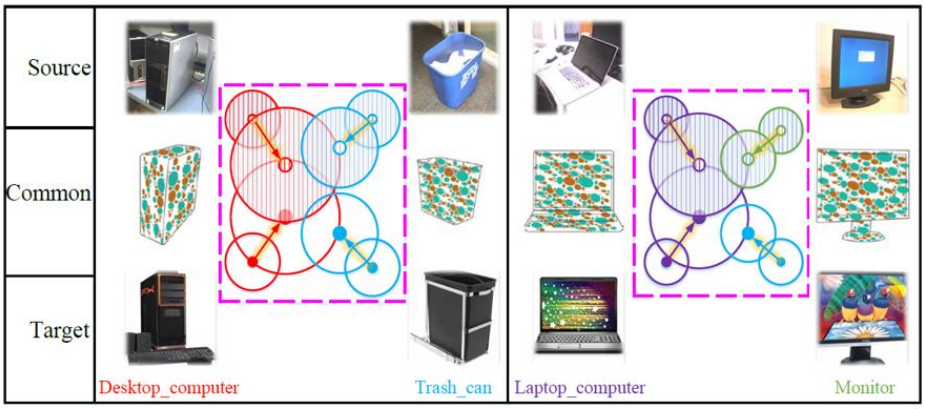

Figure 1. Working principle of the MMD. Different color circles represent various categories, the tiny circles are the means of specific categories and the hollow and meshed circles represent the source and target domains, respectively; the solid arrows denote the DA processes with MMD, and the comparatively larger circles are the transformed data features [33].

3. The proposed method

The unsupervised domain adaptation training proposed in this paper focuses on using discriminative CWMMD (DCWMMD) to align data of the same class between the source and target domains. By alleviating the problem of MMD through reducing feature discriminativeness while minimizing the mean difference between the two domains, the proposed method effectively achieves the goal of unsupervised domain adaptation.

Unlike Wang et al. [33], who used the L2-norm as an MMD estimator in the linearly transformed feature space, this study employed a network to train a feature space. Samples in this space are then projected into an RKHS to efficiently evaluate and minimize the loss function. The Gaussian kernel is commonly used because the RKHS with the Gaussian kernel is guaranteed to be universal [30]. Wang et al. proposed that the inter-class distance is equal to the variation minus the intraclass distance under the MMD they defined. This section reformulates the interclass distance, intra-class distance, and variation as defined by Wang et al. and adopts the MMD with the Gaussian kernel. Moreover, it provides a proof that when using this MMD to measure the distance of distribution of samples from two domains, the interclass distance is indeed equal to the variation minus the intra-class distance.

3.1. Symbols and notations

This study considers only two domains, one source domain and one target domain. X_s^c and X_t^c

represent the sample sets of class c from the source domain and the target domain, respectively, and X^c (or X_{st}^c) represents the union of all sample sets of class c in both domains, i.e., $X^c = X_{st}^c = X_s^c \cup X_t^c$. More symbols and notations are presented in a nomenclature table provided in Table 1.

Table 1. Definitions of symbols and notations.

symbol	meaning
X_s^c	set of samples of class c from source domain
X_t^c	set of samples of class c from target domain
$X^c\ (= X_{st}^c)$	$X^c = X_s^c \cup X_t^c$, set of samples of class c from source and target domains
X_s	$X_s = \bigcup_{c=1}^{C} X_s^c$, set of samples from source domain
X_t	$X_t = \bigcup_{c=1}^{C} X_t^c$, set of samples from target domain
$X\ (= X_{st})$	$X = X_s \cup X_t$, set of samples from source and target domains
n_s^c	$\|X_s^c\|$, number of samples in X_s^c
n_t^c	$\|X_t^c\|$, number of samples in X_t^c
$n^c (= n_{st}^c)$	$\|X^c\| = n_s^c + n_t^c$, number of samples in X^c
n_s	$\|X_s\|$, number of samples in X_s
n_t	$\|X_t\|$, number of samples in X_t
$n\ (= n_{st})$	$\|X\| = \sum_{c=1}^{C} n^c = n_s + n_t$, number of samples in X
m_s^c	$(1/n_s^c) \sum_{x_i \in X_s^c} x_i$, mean of X_s^c
m_t^c	$(1/n_t^c) \sum_{x_i \in X_t^c} x_i$, mean of X_t^c
$m^c\ (= m_{st}^c)$	$(1/n^c) \sum_{x_i \in X^c} x_i$, mean of X^c
m_s	$(1/n_s) \sum_{x_i \in X_s} x_i$, mean of X_s
m_t	$(1/n_t) \sum_{x_i \in X_t} x_i$, mean of X_t
$m\ (= m_{st})$	$(1/n) \sum_{x_i \in X} x_i$, mean of X

3.2. Relations between interclass distance, intra-class distance, and variance

This subsection uses the RKHS-based MMD and leverages the kernel trick to efficiently compute the interclass distance, intra-class distance, and variation between samples from the two domains. Moreover, it demonstrates that when using the Gaussian kernel-based MMD, the inter-class distance can be decomposed into their respective intraclass distances and variations.

Definition 3.1. Interclass distance.

The square of the interclass distance between the samples from the source domain and the target domain is defined as $S_{inter} = \sum_{c=1}^{C} (S)_{inter}^c$, where $(S)_{inter}^c = (S_{st})_{inter}^c$ is the square of the interclass distance (or MMD) between the samples of class c from the source domain and the target domain, as shown in (19), which is derived as the forms in (20) and (21).

$$(S)_{inter}^{c} = k(m_s^c - m_t^c, m_s^c - m_t^c), \tag{19}$$

$$(S)_{inter}^{c} = \left[\frac{n_s^c+n_t^c}{n_t^c} k(m_s^c - m_{st}^c, m_s^c - m_{st}^c) + \frac{n_s^c+n_t^c}{n_s^c} k(m_t^c - m_{st}^c, m_t^c - m_{st}^c\right], \tag{20}$$

$$(S)_{inter}^{c} = \frac{n_s^c+n_t^c}{n_s^c n_t^c} (n_s^c k(m_s^c - m_{st}^c, m_s^c - m_{st}^c) + n_t^c k(m_t^c - m_{st}^c, m_t^c - m_{st}^c). \tag{21}$$

Definition 3.2. Intraclass distance.

The square of the intra-class distance between the samples from the source domain and the target domain is defined as $S_{intra} = \sum_{c=1}^{C} (S)_{intra}^{c}$, where $(S)_{intra}^{c} = (S_{st})_{intra}^{c}$ is the square of the intraclass distance of the samples of class c from the two domains, as defined in (22).

$$(S)_{intra}^{c} = \frac{n_s^c+n_t^c}{n_s^c n_t^c} (\sum_{x_i \in X_s^c} k(x_i - m_s^c, x_i - m_s^c) + \sum_{x_j \in X_t^c} k(x_j - m_t^c, x_j - m_t^c). \tag{22}$$

Definition 3.3. Variance.

The joint variance of the samples from the source domain and the target domain is defined as $S_{var} = \sum_{c=1}^{C} (S)_{var}^{c}$, where $(S)_{var}^{c} = (S_{st})_{var}^{c}$ is the joint variance of the samples of class c from the source domain and the target domain, as shown in (23).

$$(S)_{var}^{c} = (S_{st})_{var}^{c} = \frac{n_s^c+n_t^c}{n_s^c n_t^c} \sum_{x_i \in X_{st}^c} k(x_i - m_{st}^c, x_i - m_{st}^c). \tag{23}$$

The square of the MMD between samples X_s and X_t from the source domain and the target domain, respectively is defined using (24) or derived as (25). Conceptually, this is equivalent to treating the samples from both domains as belonging to the same class and computing the interclass distance. This can be expressed as $(S)_{inter}^{o} = (S_{st})_{inter}^{o} = \left(MMD(X_s, X_t)\right)^2$ or $\left(MMD_u(X_s, X_t)\right)^2$ when unbiased estimation is employed, as demonstrated in formulas (26) and (27), respectively.

$$\begin{aligned}\left(MMD(X_s, X_t)\right)^2 &= k(m_s - m_t, m_s - m_t) \\ &= \frac{n_s+n_t}{n_t} k(m_s - m_{st}, m_s - m_{st}) + \frac{n_s+n_t}{n_s} k(m_t - m_{st}, m_t - m_{st},\end{aligned} \tag{24}$$

$$\left(MMD(X_s, X_t)\right)^2 = \frac{n_s+n_t}{n_s n_t} (n_s k(m_s - m_{st}, m_s - m_{st}) + n_t k(m_t - m_{st}, m_t - m_{st}), \tag{25}$$

$$\begin{aligned}\left(MMD(X_s, X_t)\right)^2 &= \frac{1}{(n_s)^2} \sum_{x_i \in X_s} \sum_{x_j \in X_s} k(x_i, x_j) \\ &+ \frac{1}{(n_t)^2} k(x_i, x_j) - \frac{2}{n_s n_t} \sum_{x_i \in X_s} \sum_{x_j \in X_t} k(x_i, x_j)\end{aligned} \tag{26}$$

$$\begin{aligned}\left(MMD_u(X_s, X_t)\right)^2 &= \frac{1}{n_s(n_s-1)} \sum_{\substack{x_i, x_j \in X_s \\ x_i \neq x_j}} k(x_i, x_j) \\ &+ \frac{1}{n_t(n_t-1)} \sum_{\substack{x_i, x_j \in X_t \\ x_i \neq x_j}} k(x_i, x_j) - \frac{2}{n_s n_t} \sum_{x_i \in X_s} \sum_{x_j \in X_t} k(x_i, x_j).\end{aligned} \tag{27}$$

Theorem 3.1. The square of the interclass distance equals the data variance minus the square of the intra-class distance; that is, $S_{inter} = S_{var} - S_{intra}$.

Proof. For $(S)_{inter}^{c} = \frac{n_s^c+n_t^c}{n_s^c n_t^c} (n_s^c k(m_s^c - m_{st}^c, m_s^c - m_{st}^c) + n_t^c k(m_t^c - m_{st}^c, m_t^c - m_{st}^c))$, $(S)_{var}^{c} =$

$\frac{n_s^c+n_t^c}{n_s^c n_t^c}\sum_{x_i\in X_{st}^c} k(x_i - m_{st}^c, x_i - m_{st}^c)$, and $(S)_{intra}^c = \frac{n_s^c+n_t^c}{n_s^c n_t^c}\Big(\sum_{x_i\in X_s^c} k(x_i - m_s^c, x_i - m_s^c) + \sum_{x_j\in X_t^c} k\big(x_j - m_t^c, x_j - m_t^c\big)\Big)$, it is sufficient to prove that $(S)_{inter}^c + (S)_{intra}^c = (S)_{var}^c$ or that $n_{s_d}^c k\big(m_{s_d}^c - m^c, m_{s_d}^c - m^c\big) + \sum_{x_i\in X_{s_d}^c} k\big(x_i - m_{s_d}^c, x_i - m_{s_d}^c\big) = \sum_{x_i\in X_{s_d}^c} k(x_i - m^c, x_i - m^c)$ for $1 \le c \le C$ and $s_d \in \{s,t\}$. Since $n_{s_d}^c k\big(m_{s_d}^c - m^c, m_{s_d}^c - m^c\big) = \sum_{x_i\in X_{s_d}^c} k\big(m_{s_d}^c - m^c, m_{s_d}^c - m^c\big) = \sum_{x_i\in X_{s_d}^c}\Big(k\big(m_{s_d}^c, m_{s_d}^c\big) + k(m^c, m^c) - 2k\big(m_{s_d}^c, m^c\big)\Big) = \sum_{x_i\in X_{s_d}^c}\Big(k\big(m_{s_d}^c, m_{s_d}^c\big) + k(m^c, m^c) - 2k(x_i, m^c)\Big)$ and $\sum_{x_i\in X_{s_d}^c} k\big(x_i - m_{s_d}^c, x_i - m_{s_d}^c\big) = \sum_{x_i\in X_{s_d}^c}\Big(k(x_i, x_i) + k\big(m_{s_d}^c, m_{s_d}^c\big) - 2k\big(x_i, m_{s_d}^c\big)\Big) = \sum_{x_i\in X_{s_d}^c}\Big(k(x_i, x_i) + k\big(m_{s_d}^c, m_{s_d}^c\big) - 2k\big(m_{s_d}^c, m_{s_d}^c\big)\Big) = \sum_{x_i\in X_{s_d}^c}\Big(k(x_i, x_i) - k\big(m_{s_d}^c, m_{s_d}^c\big)\Big)$, we have $n_{s_d}^c k\big(m_{s_d}^c - m^c, m_{s_d}^c - m^c\big) + \sum_{x_i\in X_{s_d}^c} k\big(x_i - m_{s_d}^c, x_i - m_{s_d}^c\big) = \sum_{x_i\in X_{s_d}^c}\Big(k\big(m_{s_d}^c, m_{s_d}^c\big) + k(m^c, m^c) - 2k(x_i, m^c) + k(x_i, x_i) - k\big(m_{s_d}^c, m_{s_d}^c\big)\Big) = \sum_{x_i\in X_{s_d}^c} k(x_i - m^c, x_i - m^c)$. This completes the proof.

3.3. Discriminative class-wise loss

Theorem 3.1 indicates that minimizing the interclass distance is equivalent to minimizing their variation, while simultaneously maximizing the intra-class distance, thus reducing feature discriminativeness. To address this, the strategy proposed by Wang et al. [33] was adopted with a trade-off parameter β $(-1 \le \beta \le 1)$ introduced to adjust the hidden intra-class distance within S_{inter}, resulting in the formulation of the discriminative class-level loss function, denoted as L_{dcwmmd} in formula (28), and its expansion is given in formula (29).

$$\begin{aligned} L_{dcwmmd} &= S_{var} + \beta \cdot S_{intra} + MMD^2(X_s, X_t) \\ &= \sum_{c=1}^{C}(S_{st})_{var}^c + \beta \cdot \sum_{c=1}^{C}(S_{st})_{intra}^c + (S_{st})_{inter}^0, \end{aligned} \tag{28}$$

$$L_{dcwmmd} = \sum_{c=1}^{C} \frac{n_s^c+n_t^c}{n_s^c n_t^c}\Big(\sum_{x_j\in X_{st}^c}\langle x_j - m_{st}^c, x_j - m_{st}^c\rangle_{\mathcal{H}} + \beta\sum_{x_i\in X_s^c}\langle x_i - m_s^c, x_i - m_s^c\rangle_{\mathcal{H}} + \beta\sum_{x_j\in X_t^c}\langle x_j - m_t^c, x_j - m_t^c\rangle_{\mathcal{H}}\Big) + \frac{n_s+n_t}{n_t} < m_s - m_{st}, m_s - m_{st} > + \frac{n_s+n_t}{n_s} < m_t - m_{st}, m_t - m_{st} >. \tag{29}$$

The terms $(S_{st})_{inter}^c$, $(S_{st})_{intra}^c$, and $(S_{st})_{var}^c$, defined in Definitions 3.1 to 3.3, can be expressed in terms of individual sample representations $x_j's$ using formulas (30) to (32). To ensure unbiased estimation and calculate deviations in the feature space, this study uses the loss function L_{dcwmmd}^u, represented by the feature representations $z_j's$ of the samples $x_j's$, as shown in (28). By setting $\alpha_1 = \frac{(\beta+1)(n_s^c+n_t^c)}{n_s^c n_t^c}$, $\alpha_2 = -\Big(\frac{n_s^c+(n_s^c+n_t^c)\beta}{(n_s^c)^2 n_t^c}\Big)$, $\alpha_3 = -\Big(\frac{n_t^c+(n_s^c+n_t^c)\beta}{n_s^c(n_t^c)^2}\Big)$, $\alpha_4 = -\Big(\frac{2}{n_s^c n_t^c}\Big)$, $\gamma_1 = \frac{1}{n_s(n_s-1)}$, $\gamma_2 = \frac{1}{n_t(n_t-1)}$, and $\gamma_3 = -\frac{2}{n_s n_t}$, the simplified form of L_{dcwmmd}^u is given in formula (34).

During the training process, these scalar values can be precomputed and stored, eliminating the need for subsequent recalculation.

$$(S_{st})^c_{inter} = \frac{1}{(n_s^c)^2}\sum_{x_i,x_j\in X_s^c}\langle x_i,x_j\rangle_{\mathcal{H}} + \frac{1}{(n_t^c)^2}\sum_{x_i,x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}} - \frac{2}{n_s^c n_t^c}\sum_{x_i\in X_s^c,x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}}, \quad (30)$$

$$(S_{st})^c_{intra} = \frac{n_s^c+n_t^c}{n_s^c n_t^c}\left[\sum_{x_i\in X_{st}^c}\langle x_i,x_i\rangle_{\mathcal{H}} - \frac{1}{n_s^c}\sum_{x_i,x_j\in X_s^c}\langle x_i,x_j\rangle_{\mathcal{H}} - \frac{1}{n_t^c}\sum_{x_i,x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}}\right], \quad (31)$$

$$\begin{aligned}(S_{st})^c_{var} = {} & \frac{n_s^c+n_t^c}{n_s^c n_t^c}\sum_{x_j\in X_{st}^c}\langle x_j,x_j\rangle_{\mathcal{H}} - \frac{1}{n_s^c n_t^c}\sum_{x_i,x_j\in X_s^c}\langle x_i,x_j\rangle_{\mathcal{H}} \\ & - \frac{1}{n_s^c n_t^c}\sum_{x_i,x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}} - \frac{2}{n_s^c n_t^c}\sum_{x_i\in X_s^c,x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}}, \end{aligned} \quad (32)$$

$$\begin{aligned}L^u_{dcwmmd}(X_s,X_t) = {} & \sum_{c=1}^{C}\left[\frac{(\beta+1)(n_s^c+n_t^c)}{n_s^c n_t^c}\sum_{x_j\in X_{st}^c}\langle x_j,x_j\rangle_{\mathcal{H}} - \left(\frac{n_s^c+(n_s^c+n_t^c)\beta}{(n_s^c)^2 n_t^c}\right)\sum_{x_i\in X_s^c}\sum_{x_j\in X_s^c}\langle x_i,x_j\rangle_{\mathcal{H}} - \right. \\ & \left.\left(\frac{n_t^c+(n_s^c+n_t^c)\beta}{n_s^c(n_t^c)^2}\right)\sum_{x_i\in X_t^c}\sum_{x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}} - \frac{2}{n_s^c n_t^c}\sum_{x_i\in X_s^c}\sum_{x_j\in X_t^c}\langle x_i,x_j\rangle_{\mathcal{H}}\right] \\ & + \frac{1}{n_s(n_s-1)}\sum_{\substack{x_i,x_j\in X_s\\ x_i\neq x_j}}\langle x_i,x_j\rangle_{\mathcal{H}} + \frac{1}{n_t(n_t-1)}\sum_{\substack{x_i,x_j\in X_t\\ x_i\neq x_j}}\langle x_i,x_j\rangle_{\mathcal{H}} \\ & - \frac{2}{n_s n_t}\sum_{x_i\in X_s}\sum_{x_j\in X_t}\langle x_i,x_j\rangle_{\mathcal{H}}, \end{aligned} \quad (33)$$

$$\begin{aligned}L^u_{dcwmmd}(Z_s,Z_t) = {} & \sum_{c=1}^{C}\left[\alpha_1\sum_{z_j\in Z_{st}^c}\langle z_j,z_j\rangle_{\mathcal{H}} + \alpha_2\sum_{z_i,z_j\in Z_s^c}\langle z_i,z_j\rangle_{\mathcal{H}} + \alpha_3\sum_{z_i,z_j\in Z_t^c}\langle z_i,z_j\rangle_{\mathcal{H}} + \right. \\ & \left.\alpha_4\sum_{x_i\in Z_j\in Z_t^c}\langle z_i,z_j\rangle_{\mathcal{H}}\right] \\ & + \gamma_1\sum_{\substack{z_i,z_j\in Z_s\\ z_i\neq z_j}}\langle z_i,z_j\rangle_{\mathcal{H}} + \gamma_2\sum_{\substack{z_i,z_j\in Z_t\\ z_i\neq z_j}}\langle z_i,z_j\rangle_{\mathcal{H}} + \gamma_3\sum_{z_i\in Z_s,z_j\in Z_t}\langle z_i,z_j\rangle_{\mathcal{H}}. \end{aligned} \quad (34)$$

A categorical cross-entropy, L_{cls}, is commonly used as the error for the classifier's classification results on the source domain data, as shown in formula (35). Here, $\hat{\ell}^c_{sj}$ is the c-th element of $\hat{\boldsymbol{\ell}}_{si} = \mathbf{C}(z_i^s)$ and y^c_{si} is the c-th element of the ground truth one-hot label vector $\boldsymbol{y_{si}}$, where $y^c_{si} = 1$ if the label of the original sample x_i^s corresponding to z_i^s is c, and $y^c_{si} = 0$ otherwise. To encourage the samples to form dense, uniform, and well-separated clusters, the label-smoothing (LS) technique [42] is applied to the cross-entropy loss. This involves substituting the smooth label $(1-\alpha)y^c_{si} + \alpha/C$, a weighted average of y^c_{si} and $1/C$, with y^c_{si} in the categorical cross-entropy to form the smoothed categorical cross-entropy $L^{ls}_{cls}(Z_s,Y_s)$, as shown in (36). Here, α is a smoothing factor generally set to 0.1 for better performance and C represents the number of classes. The goal of LS is to prevent the model from becoming too confident in its predictions and to reduce overfitting. Rafael Müller et al. [42] have shown that LS encourages the representations of training examples from the same class to group in tight clusters.

During training with target samples, the entropy of predicted results for those samples is minimized, as illustrated in formula (37). This strategy is employed because it has been indicated that unlabeled examples are especially beneficial when class overlap is small [43]. Minimizing this entropy encourages the predicted results to be more inclined toward a specific category, making the feature distribution between categories in the target domain more distinct and explicit. The training loss function of the entire network is defined as $\mathcal{L}_{DCWDA}$, as shown in formula (38), where ω_1 and ω_2

are weight parameters.

$$L_{cls}(Z_s, Y_s) = \frac{1}{n_s}\sum_{i=1}^{n_s}\sum_{c=1}^{C} y_{si}^c \log \hat{\ell}_{si}^c, \tag{35}$$

$$L_{cls}^{ls}(Z_s, Y_s) = \frac{1}{n_s}\sum_{i=1}^{n_s}\sum_{c=1}^{C}((1-\alpha)y_{si}^c + \alpha/C)\log \hat{\ell}_{si}^c, \tag{36}$$

$$L_{ent}(Z_t) = \frac{1}{n_t}\sum_{j=1}^{n_t}\sum_{c=1}^{C} \hat{\ell}_{tj}^c \log \hat{\ell}_{tj}^c, \tag{37}$$

$$L_{DCWDA} = L_{dcwmmd}^u + \omega_1 L_{cls}^{ls} + \omega_2 L_{ent}. \tag{38}$$

The training architecture of the proposed discriminative class-wise domain adaptation (DCWDA) system, as shown in Figure 2, consists of a feature extractor (**F**) used for extracting domain-invariant features and a classifier (**C**). The feature extractor **F** and the classifier **C** are duplicated to represent the data paths of the source domain and the target domain, and a dotted line is drawn in the middle to indicate shared weights. During training, the source domain samples x_s and target domain samples x_t are first separately input into the feature extractor **F**, which outputs features $z_s = \mathbf{F}(x_s)$ and $z_t = \mathbf{F}(x_t)$. The discriminative class-wise loss function L_{dcwmmd}^u is then computed for z_s and z_t. Subsequently, z_s and z_t are separately input into the classifier **C**, producing classification results $\hat{\boldsymbol{\ell}}_s = \mathbf{C}(z_s)$ and $\hat{\boldsymbol{\ell}}_t = \mathbf{C}(z_t)$. This allows the calculation of the cross-entropy L_{cls} for the predicted result for the source domain sample x_s and the entropy L_{ent} for the predicted result of the target domain sample x_t. The training algorithm is shown in Algorithm 1, where the batch sizes of both the source sample and the target sample are set to *N*.

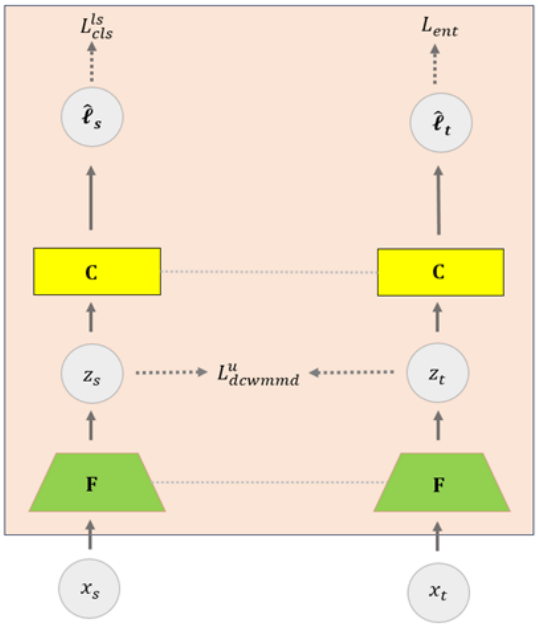

Figure 2. Architecture of DCWDA.

Algorithm 1. Training the DCWDA model.

Input: $\Delta_s, \Delta_t, \alpha, \omega_1, \omega_2, \eta_2$;

Initialize parameters $\theta_{\mathbf{F}}$ and $\theta_{\mathbf{C}}$;

 # train the model parameters $\theta_{\mathbf{F}}$ *and* $\theta_{\mathbf{C}}$ *on* Δ_s *and* Δ_t;

repeat until convergence

$(X_s, Y_s) = \{(x_{s1}, y_{s1}), (x_{s2}, y_{s2}), \dots, (x_{sN}, y_{sN})\} \leftarrow$ mini-batch from Δ_s;

$X_t = \{x_{t1}, x_{t2}, \dots, x_{tN}\} \leftarrow$ mini-batch from Δ_t;

$Z_s \leftarrow \mathbf{F}(X_s)$; $Z_t \leftarrow \mathbf{F}(X_t)$;

 # generate pseudo labels:

$\hat{\mathcal{L}}_t = \{\hat{\boldsymbol{\ell}}_{t1}, \hat{\boldsymbol{\ell}}_{t2}, \dots, \hat{\boldsymbol{\ell}}_{tN}\} \leftarrow \mathbf{C}(\mathbf{F}(X_t))$; *# classifier target sample*

$Y_t = \{y_{t1}, y_{t2}, \dots, y_{tN}\} = \{\mathrm{M}(\hat{\boldsymbol{\ell}}_{t1}), \mathrm{M}(\hat{\boldsymbol{\ell}}_{t2}), \dots, \mathrm{M}(\hat{\boldsymbol{\ell}}_{tN})\}$; *# obtain pseudo labels*

$\#\,\mathrm{M}((v_1, v_2, \dots, v_C)) = \underset{1 \le c \le C}{\operatorname{argmax}} v_c$;

 # evaluate losses:

$L^u_{dcwmmd}(X_s, X_t) = \cdots$; # using (29)

$L^{ls}_{cls} \leftarrow \frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}((1-\alpha)y^c_{si} + \alpha/C)\log \hat{\ell}^c_{sj}$; # using (36)

$L_{ent} \leftarrow \frac{1}{N}\sum_{j=1}^{N}\sum_{c=1}^{C}\hat{\ell}^c_{tj}\log \hat{\ell}^c_{tj}$; # using (37)

$L_{DCWDA} \leftarrow L^u_{dcwmmd} + \omega_1 L^{ls}_{cls} + \omega_2 L_{ent}$;

update $\theta_{\mathbf{F}}$ and $\theta_{\mathbf{C}}$ to minimize $\mathcal{L}_{DCWDA}$;

$\theta_{\mathbf{F}} \leftarrow \theta_{\mathbf{F}} - \eta_2 \nabla_{\theta_{\mathbf{F}}} \mathcal{L}_{DCWDA}$;

$\theta_{\mathbf{C}} \leftarrow \theta_{\mathbf{C}} - \eta_2 \nabla_{\theta_{\mathbf{C}}} \mathcal{L}_{DCWDA}$;

end repeat

4. Experimental results

The proposed method was evaluated using digit datasets and office object data. The digit datasets used in the experiments include modified national institute of standards and technology database (MNIST) [44], U.S. postal service (USPS) [45], and street view house numbers (SVHN) [46]. MNIST and USPS are handwritten datasets. MNIST has 60,000 training samples and 10,000 testing samples, all grayscale images of size 28×28. USPS consists of 9,298 grayscale images of size 16x16. SVHN contains 73,257 training images and 26,032 test images, which are color images of size 32×32 captured from street-view house number photo images. For each image, the digit to be recognized are a single digit in a house number located in the center of the image, surrounded by other digits or distracting

objects. In the experiment, the images are scaled to a size of 32×32 pixels. Figure 2 shows some images from MNIST, USPS, and SVHN, and the image in each blue frame is used as a training sample. Figure 3 displays some images from MNIST, USPS, and SVHN, where the numbers within blue frames in SVHN images are the digits to be recognized. The Office-31 [47] dataset comprises three domains: Amazon (A), DSLR (Digital Single – Lens Reflex) (D), and Webcam (W). Each domain comprises 31 object categories in an office environment, totaling 4,110 images, with varying numbers of images for each category. Figure 4 displays some images from Webcam, DSLR, and Amazon.

In the training process, the batch sizes for the digit dataset and Office-31 dataset are set to 128 and 64, respectively. Resnet-18 and Resnet-50 [1] are adopted as the network architectures for the feature extractors on the digit dataset and the Office-31 dataset, respectively. Both architectures undergo fine-tuning with pre-trained ImageNet network parameters. In addition, the pseudo-labels of all target domain training data are updated with the current classifier parameters at the beginning of each epoch.

Figure 3. Digital data: (a) MNIST, (b) USPS, (c) SVHN.

Figure 4. Office-31 data: (a) Webcam, (b) DSLR, (c) Amazon.

The accuracies of various combinations of source domain and target domain were evaluated. The combinations for digital datasets include: MNIST to USPS (M → U), USPS to MNIST (U → M), and SVHN to MNIST (S → M). The combinations for Office-31 datasets include: Amazon to DSLR (A → D), Amazon to Webcam (A → W), DSLR to Amazon (D → A), DSLR to Webcam (D → W), Webcam to Amazon (W → A), and Webcam to DSLR (W → D). Table 1 compares the proposed method with various unsupervised domain adaptation methods on the digit datasets, including adversarial discriminative domain adaptation (ADDA) [17], adversarial dropout regularization (ADR) [48],

conditional adversarial domain adaptation (CDAN) [49], cycle-consistent adversarial domain adaptation (CyCADA) [50], sliced wasserstein discrepancy (SWD) [51] and source hypothesis transfer (SHOT) [36]. Table 2 compares the proposed method with various unsupervised domain adaptation methods on the Office-31dataset, including: Wang et al. [33], deep adaptation networks (DAN) [18], domain-adversarial neural network (DANN) [16], ADDA [17], multi-adversarial domain adaptation (MADA) [52], SHOT [36], collaborative and adversarial network (CAN) [14] and mini-batch dynamic geometric embedding (MDGE) [23]. Each accuracy represents the average accuracy rate of three test results. The best-performing methods for each source-to-target combination are highlighted in bold. The "Source-only" category indicates that the classifier is directly trained using the source domain data without domain adaptation, and then tested using the target domain data. The "Target-supervised" category shows that the classifier is directly trained using the target domain data and tested using the target data. Typically, the accuracies of "source-only" and "target supervised" serve as the lower and upper bounds for domain adaptation accuracy, but there's no guarantee that the accuracy will fall within this range.

As can be seen from Table 2, the proposed method outperforms other methods in testing most digital dataset pairs except S → M, and achieves the highest average accuracy. It is worth noting that SVHN images have obvious color changes and noise. Compared with other digital imaging datasets, the USPS is a smaller digital dataset with smaller images. Hence, the test results for the combinations of USPS and SVHN are not very informative. In view of these results, the datasets M → S, S → U, and U → S are not used in the digital dataset experiment. As can be seen from Table 3, the proposed method outperforms other methods in testing two of the three digital dataset combinations and achieves the highest average accuracy.

Table 2. Accuracies (%) of several approaches on some digit datasets.

source→target methods	M→U	U→M	S→M	Average
Source-only	69.6	82.2	67.1	73.0
ADDA [17]	90.1	89.4	76.0	85.2
ADR [48]	93.1	93.2	95.0	93.8
CDAN [49]	98.0	95.6	89.2	94.3
CyCADA [50]	96.5	95.6	90.4	94.2
SWD [51]	97.1	98.1	98.9	98.0
SHOT [36]	97.8	97.6	**99.0**	98.1
ours	**98.0**	**98.2**	98.8	**98.3**
target-supervised	99.4	98.1	99.4	98.9

Table 3. Accuracies (%) of several approaches on the Office-31dataset.

source→target methods	A→D	A→W	D→A	D→W	W→A	W→D	Average
Source-only	68.9	68.4	62.5	96.7	60.7	99.3	76.1
Wang et al. [33]	90.76	88.93	75.43	98.49	75.15	99.80	88.06
DAN [18]	78.6	80.5	63.6	97.1	62.8	99.6	80.4
DANN[16]	79.7	82.0	68.2	96.9	67.4	99.1	82.2
ADDA [17]	77.8	86.2	69.5	96.2	68.9	98.4	82.9
MADA [52]	87.8	90.0	70.3	97.4	66.4	99.6	85.2
SHOT [36]	93.9	90.1	75.3	98.7	75.0	**99.9**	88.8
CAN [14]	95.0	94.5	**78.0**	99.1	**77.0**	99.8	90.6
MDGE [23]	90.6	89.4	69.5	98.9	68.4	99.8	86.1
ours	**96.3**	**94.9**	77.9	**99.5**	76.5	99.6	**90.8**
target-supervised	98.0	98.7	86.0	98.7	86.0	98.0	94.3

5. Discussion and conclusions

In this paper, we tackled the domain adaptation problem by using a deep network architecture with a DCWMMD as a loss function. The MMD used is based on embedding distribution metrics in the reproducing kernel Hilbert space. This not only leverages the kernel trick to enhance computational efficiency but also conforms to the original MMD definition. Marginal MMD helps align the data distributions regardless of class alignment. To alleviate this limitation, CWMMD was introduced to align data distributions of the same class from the two domains. However, this adjustment may lead to a reduction in feature discriminativeness. By deconstructing CWMMD into variance minus intra-class distance, an adjustable weight parameter for the intra-class distance term was introduced, providing flexibility to preserve feature discriminability. The experimental results show that our proposed method improves upon the approach proposed by Wang et al. [33]. In terms of the error function, we not only applied the LS technique to the cross entropy for the training of the source domain, but we also added the entropy of the predicted label for the target samples to enhance the overall training performance. The proposed architecture was evaluated using two datasets, the digital dataset and Office-31 dataset. The results demonstrate competitive accuracy rates for domain adaptation when compared to other methods.

In the future, we will continue to improve the performance of training process in the system, such as applying data augmentation to increase the diversity of data, using high-confidence data from the target domain to provide pseudo-labels for supervised post-processing training, etc. Last but not least, we also want to apply our work to other domain adaptation tasks, such as face recognition, object recognition, and image-to-image translation.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the National Science and Technology Council, Taiwan, R.O.C. under the grant NSTC 112-2221-E-032-041.

Conflict of interest

The authors declare no conflict of interest.

References

1. K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, In: *Proceedings of conference on computer vision and pattern recognition (CVPR)*, 2016, 770–778. https://doi.org/10.1109/CVPR.2016.90
2. S. Ren, K. He, R. Girshick, J. Sun, Faster R-cnn: Towards real-time object detection with region proposal networks, *IEEE Trans. Pattern Anal. Machine Intel.*, **39** (2017), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
3. K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, In: *2017 IEEE International conference on computer vision (ICCV)*, 2017, 2980–2988. https://doi.org/10.1109/ICCV.2017.322
4. S. J. Pan, Q. Yang, A survey on transfer learning, *IEEE Trans. Knowl. Data Eng.*, **22** (2010), 1345–1359. https://doi.org/10.1109/TKDE.2009.191
5. J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, B. Schölkopf, Correcting sample selection bias by unlabeled data, In: *Advances in neural information processing systems*, The MIT Press, 2007. https://doi.org/10.7551/mitpress/7503.003.0080
6. S. Li, S. Song, G. Huang, Prediction reweighting for domain adaptation, *IEEE Trans. Neural Netw. Learn. Syst.*, **28** (2017), 1682–169. https://doi.org/10.1109/TNNLS.2016.2538282
7. M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, M. Salzmann, Domain adaptation on the statistical manifold, In: *2014 IEEE conference on computer vision and pattern recognition*, 2014, 2481–2488. https://doi.org/10.1109/CVPR.2014.318
8. M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, Transfer feature learning with joint distribution adaptation, In: *2013 IEEE international conference on computer vision*, 2013, 2200–2207. https://doi.org/10.1109/ICCV.2013.274
9. M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, Transfer joint matching for unsupervised domain adaptation, In: *2014 IEEE conference on computer vision and pattern recognition*, 2014, 1410–1417. https://doi.org/10.1109/CVPR.2014.183
10. M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, M. Salzmann, Unsupervised domain adaptation by domain invariant projection, In: *2013 IEEE international conference on computer vision*, 2013, 769–776. https://doi.org/10.1109/ICCV.2013.100
11. S. J. Pan, J. T. Kwok, Q. Yang, Transfer learning via dimensionality reduction, In: *Proceedings of the AAAI conference on artificial intelligence*, **23** (2008), 677–682.
12. M. Long, J. Wang, G. Ding, S. J. Pan, P. S. Yu, Adaptation regularization: A general framework for transfer learning, *IEEE Trans. Knowl. Data Eng.*, **26** (2014), 1076–1089. https://doi.org/10.1109/TKDE.2013.111

13. L. Bruzzone, M. Marconcini, Domain adaptation problems: A DASVM classification technique and a circular validation strategy, *IEEE Trans. Pattern Anal. Machine Intell.*, **32** (2010), 770–787. https://doi.org/10.1109/TPAMI.2009.57
14. W. Zhang, W. Ouyang, W. Li, D. Xu, Collaborative and adversarial network for unsupervised domain adaptation, In: *2018 IEEE/CVF conference on computer vision and pattern recognition*, 2018. https://doi.org/10.1109/CVPR.2018.00400
15. K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, D. Krishnan, Unsupervised pixel-level domain adaptation with generative adversarial networks, In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, 95–104. https://doi.org/10.1109/CVPR.2017.18
16. Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, et al., Domain adversarial training of neural networks, *J. Machine Learn. Res.*, **17** (2016), 1–35.
17. E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, 2962–2971. https://doi.org/10.1109/CVPR.2017.316
18. M. Long, Y. Cao, J. Wang, M. I. Jordan, Learning transferable features with deep adaptation networks, In: *Proceedings of the 32nd international conference on international conference on machine learning*, **37** (2015), 97–105.
19. M. Long, H. Zhu, J. Wang, M. I. Jordan, Unsupervised domain adaptation with residual transfer networks, In: *Proceedings of the 30th international conference on neural information processing systems*, 2016, 136–144. https://dl.acm.org/doi/10.5555/3157096.3157112
20. B. Sun and K. Saenko, Deep coral: Correlation alignment for deep domain adaptation, In: *European conference on computer vision*, 2016, 443–450. https://doi.org/10.1007/978-3-319-49409-8_35
21. M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, W. Li, Deep reconstruction-classification networks for unsupervised domain adaptation, In: *European conference on computer vision*, 2016, 597–613. https://doi.org/10.1007/978-3-319-46493-0_36
22. S. Khan, M. Asim, S. Khan, A. Musyafa, Q. Wu, Unsupervised domain adaptation using fuzzy rules and stochastic hierarchical convolutional neural networks, *Comput. Elect. Eng.*, **105** (2023), 108547. https://doi.org/10.1016/j.compeleceng.2022.108547
23. S. Khan, Y. Guo, Y. Ye, C. Li, Q. Wu, Mini-batch dynamic geometric embedding for unsupervised domain adaptation, *Neural Process. Lett.*, **55** (2023), 2063–2080. https://doi.org/10.1007/s11063-023-11167-7
24. L. Zhang, W. Zuo, D. Zhang, LSDT: Latent sparse domain transfer learning for visual adaptation, *IEEE Trans. Image Process.*, **25** (2016), 1177–1191. https://doi.org/10.1109/TIP.2016.2516952
25. Y. Chen, W. Li, C. Sakaridis, D. Dai, L. V. Gool, Domain adaptive faster R-CNN for object detection in the wild, In: *2018 IEEE/CVF conference on computer vision and pattern recognition*, 2018, 3339–3348. https://doi.org/10.1109/CVPR.2018.00352
26. K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, D. Krishnan, Unsupervised pixel-level domain adaptation with generative adversarial networks, In: *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, 2017, 95–104. https://doi.org/10.1109/CVPR.2017.18
27. H. Xu, J. Zheng, A. Alavi, R. Chellappa, Cross-domain visual recognition via domain adaptive dictionary learning, *arXiv:1804.04687*, 2018. https://doi.org/10.48550/arXiv.1804.04687
28. A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Scholkopf, A. Smola, A kernel two-sample test, *J. Machine Learn. Res.*, **13** (2012), 723–773. https://doi.org/10.5555/2188385.2188410

29. S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, *IEEE Trans. Neural Netw.*, **22** (2011), 199–210. https://doi.org/10.1109/TNN.2010.2091281
30. K. M. Borgwardt, A. Gretton, M. J. Rasch, H. P. Kriegel, B. Scholkopf, A. J. Smola, Integrating structured biological data by kernel maximum mean discrepancy, *Bioinformatics*, **22** (2006), e49–e57. https://doi.org/10.1093/bioinformatics/btl242
31. S. Si, D. Tao, B. Geng, Bregman divergence-based regularization for transfer subspace learning, *IEEE Trans. Knowl. Data Eng.*, **22** (2010), 929–942. https://doi.org/10.1109/TKDE.2009.126
32. J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, J. Wortman, Learning bounds for domain adaptation, In: *Advances in neural information processing systems*, **20** (2007), 129–136.
33. W. Wang, H. Li, Z. Ding, Z. Wang, Rethink maximum mean discrepancy for domain adaptation, *arXiv:2007.00689*, 2020. https://doi.org/10.48550/arXiv.2007.00689
34. L. Devroye, G. Lugosi, Combinatorial methods in density estimation, In: *Combinatorial methods in density estimation*, New York: Springer, 2001. https://doi.org/10.1007/978-1-4613-0125-7
35. Y. Baraud, L. Birgé, Rho-estimators revisited: General theory and applications, *Ann. Statist.*, **46** (2018), 3767–3804. https://doi.org/10.1214/17-AOS1675
36. J. Liang, D. Hu, J. Feng, Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation, In: *Proceedings of the 37th international conference on machine learning*, **119** (2020), 6028–6039.
37. L. Song, A. Gretton, D. Bickson, Y. Low, C. Guestrin, Kernel belief propagation, In: *Proceedings of the 14th international conference on artificial intelligence and statistics*, **15** (2011), 707–715.
38. M. Park, W. Jitkrittum, D. Sejdinovic, K2-ABC: Approximate bayesian computation with kernel embeddings, In: *Proceedings of the 19th international conference on artificial intelligence and statistics*, **51** (2015), 398–407.
39. W. Jitkrittum, W. Xu, Z. Szabo, K. Fukumizu, A. Gretton, A linear-time kernel goodness-of-fit test, In: *Advances in neural information processing systems*, 2017, 262–271.
40. Y. Li, K. Swersky, R. S. Zemel, Generative moment matching networks, *arXiv:1502.02761*, 2015. https://doi.org/10.48550/arXiv.1502.02761
41. S. Zhao, J. Song, S. Ermon, Infovae: Information maximizing variational autoencoders, *arXiv:1706.02262*, 2018. https://doi.org/10.48550/arXiv.1706.02262
42. R. Müller, S. Kornblith, G. Hinton, When does label smoothing help? In: *33rd Conference on neural information processing systems*, 2019.
43. Y. Grandvalet, Y. Bengio, Semi-supervised learning by entropy minimization, In: *Advances in neural information processing systems*, **17** (2004), 529–536.
44. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE*, **86** (1998), 2278–2324. https://doi.org/10.1109/5.726791
45. J. J. Hull, A database for handwritten text recognition research, *IEEE Trans. Pattern Anal. Machine Intell.*, **16** (1994), 550–55. https://doi.org/10.1109/34.291440
46. Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Ng, Reading digits in natural images with unsupervised feature learning, *Proc. Int. Conf. Neural Inf. Process. Syst. Workshops*, 2011.
47. K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, In: *Lecture notes in computer science*, Berlin: Springer, **6314** (2010), 213–226. https://doi.org/10.1007/978-3-642-15561-1_16
48. K. Saito, Y. Ushiku, T. Harada, K. Saenko, Adversarial dropout regularization, *arXiv:1711.01575*, 2018. https://doi.org/10.48550/arXiv.1711.01575

49. M. Long, Z. Cao, J. Wang, M. I. Jordan, Conditional adversarial domain adaptation, In: *32nd Conference on neural information processing systems*, 2018, 1647–1657.
50. J. Hoffman, E. Tzeng, T. Park, J. Y. Zhu, P. Isola, K. Saenko, et al., Cycada: Cycle-consistent adversarial domain adaptation, In: *Proceedings of the 35th international conference on machine learning*, 2018, 1989–1998.
51. C. Y. Lee, T. Batra, M. H. Baig, D. Ulbricht, Sliced wasserstein discrepancy for unsupervised domain adaptation, In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, 2019, 10285–10295.
52. Z. Pei, Z. Cao, M. Long, J. Wang, Multi-adversarial domain adaptation, In: *Thirty-second AAAI conference on artificial intelligence*, **32** (2018). https://doi.org/10.1609/aaai.v32i1.11767

AIMS Press