Front.Comput.Sci.
DOI

Supplementary File

Dynamic Bias Alignment and Discrimination Enhancement for Unsupervised Domain Adaptation

Qing Tian (✉)[1,2], Hong Yang [1], Yanan Zhu [1], Heyang Sun [1], Heng Xu [1], Yi Chu [1]

1 School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China
2 Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science and Technology, Nanjing 210044, China

Abstract Unsupervised Domain Adaptation (UDA) aims to explore the knowledge of labeled source domain to help training the model of unlabeled target domain. By now, while most existing UDA approaches typically learn domain-invariant representations by directly matching the distributions across the domains, they pay less attention on respecting the cross-domain similarity and discrimination exploration. To address these issues, this article designs a kind of UDA with Dynamic Bias Alignment and Discrimination Enhancement (UDA-DBADE). Specifically, in UDA-DBADE we define a dynamic balance factor by the ratio of the normalized cross-domain discrepancy over the discrimination, which decreases gradually in the process of UDA-DBADE. Afterwards, we construct domain alignment with adversarial learning as well as distinguishable representations through advancing the discrepancy of multiple classifiers, and dynamically balance them with the defined dynamic factor. In this way, a larger weight is originally assigned on the domain alignment and then gradually on the discrimination enhancement in the learning process of UDA-DBADE. In addition, we further construct a bias matrix to characterize the discrimination alignment between the source and target domain samples. Finally, extensive experiments demonstrate that UDA-DBADE has an excellent performance.

Keywords Unsupervised domain adaptation, bias alignment, bias matrix, discrimination enhancement, cross-domain alignment

Received month dd, yyyy; accepted month dd, yyyy

E-mail: tianqing@nuist.edu.cn

1 Introduction

In recent years, machine learning models especially deep networks have achieved wide success, as in image classification [1] and semantic segmentation [2]. Nevertheless, these methods typically follow the assumption that both the training and test data are collected from the same distributions. In real applications, it frequently does not hold because of the distribution discrepancy between the domains, i.e. domain shift. To address this issue, the paradigm of Domain Adaptation (DA) [3–5] was proposed to match the domains. On the other hand, the methods aforementioned typically reply on large numbers of labeled data; however, labeling data is time-consuming and laborious, or even difficult to obtain. To further address such challenges, unsupervised DA (UDA) was raised. The UDA methodology exploits source domain knowledge to handle unlabeled target tasks.

The modeling strategies of existing UDA works can be mainly divided into two categories. On the one hand, it seeks to match the source and target domains by reducing their distribution discrepancy [6–9]. On the other hand, it performs domain adaptation by learning domain-invariant representations encouraged by adversarial domain discriminator [3, 4, 10], conditional domain discriminator [5] or task classifier [11, 12]. The former usually uses the momentum distance [13, 14], or second-order correlation [8, 15, 16], between the source and target domains to align their distributions. The latter learns domain-invariant feature representations to achieve UDA via the generative adversarial network [17] or domain-adversarial training. Although most of these UDA methods with adversarial learning [4, 10, 18–20] have achieved promising results on UDA tasks, they still suffer from the following limitations. Firstly, they have not considered the quantity-imbalance issue between the domains, since the domain with more samples affects more on the process of UDA. Even worse, this issue tends to result in an undesirably biased UDA model. Thirdly, these methods usually directly align the source and target domains without preserving the class diversity, which may lead to excessive domain alignment. Finally, such methods like [21–23] usually assume that the model trained on source domain data generalize well on the target domain tasks and consequently only align the cross-domain marginal distributions but ignore the data classes bias across domains, which easily leads to misclassification. As shown in Fig. 1(b), even though the marginal distributions of the source and target domains are aligned well, the target samples are still misclassified seriously by the source classifier. For example, Xiao et al. [24] proposed a dynamic weighting strategy to balance domain alignment and class discrimination, but ignored the class-specific structure within data labels during the alignment process, resulting in noisy labels near the classifier boundary.

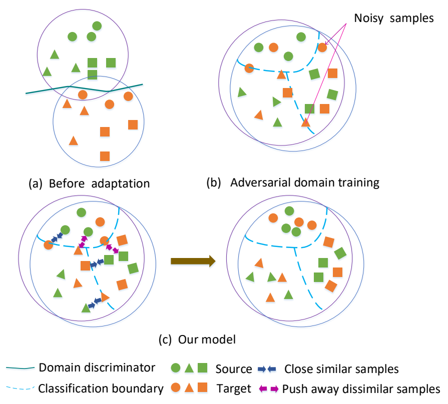


Fig. 1: Comparison of existing and the proposed DA methods. (a) The domain adaptation to the previous source and target domains; (b) Most of the existing domain adaptation methods directly align the Marginal distribution, which often leads to noise near the classification boundary; (c) UDA-DBADE firstly uses the balance factor ω to control the degree of domain alignment to prevent negative migration caused by excessive domain alignment; Secondly, the sample similarity loss is calculated using sample bias matrix S to narrow the similar samples and push the dissimilar samples.

In order to address the issues aforementioned, in this article we propose a kind of UDA model via Dynamic Bias Alignment and Discrimination Enhancement (UDA-DBADE). Specifically, we firstly construct a dynamic balance factor by the ratio of the normalized cross-domain discrepancy over the inter/intra-class discrimination, whose value decreases gradually with iterated process of UDA-DBADE. Then, with the balance factor, we dynamically regulate the adversarial-domain alignment and distinguishable representations. As a result, UDA-DBADE pays more attention on domain alignment and then gradually more on the discrimination enhancement in the learning process of domain adaptation. In addition, we design a bias matrix to characterize the discrimination alignment between the source and target domains. In summary, our main contributions are fourfold as follows:

- Proposing a novel kind of Unsupervised Domain Adaptation (UDA) model via Dynamic Bias Alignment and Discrimination Enhancement (UDA-DBADE), which jointly achieves the goal of domain alignment and discrimination enhancement.
- In UDA-DBADE, a dynamic balance factor is constructed by the ratio of the normalized cross-domain discrepancy over the target domain inter/intra-class discrimination, which encourages the model pay more attention on domain alignment and then gradually more on the discrimination enhancement in the learning process of domain adaptation.
- A bias matrix is designed to characterize the discrimination alignment between the source and target domain samples to further regularize the performance of UDA-DBADE.
- Extensive comparison and ablation experiments validate the effectiveness and superiority of the proposed method.

The rest of this article is organized as follows. In Section 2, we give a brief overview of the related work. Section 3 introduces the method in details. Then, experiments and analyses are performed in Section 4. Finally, conclusions and future directions are given in Section 5.

2 RELATED WORK

In this section we briefly review some representative UDA approaches mostly related to our work.

UDA with discrepancy measurement These approaches mainly match the source and target domains though reducing their distribution discrepancy. Representative measures include maximum mean difference (MMD) [7, 25], correlation alignment (CORAL) [8] and central moment difference (CMD) [14], etc. In articles [7] and [25], the distribution divergence between the source and target domains were measured with variants of MMD, such as multi-core MMD (MK-MMD) and joint maximum mean difference (JMMD). The authors of [26] designed a weighted MMD by assigning class-specific weights into the MMD measure. In D-CORAL [16], the CORAL was improved by incorporating the correlations between the active layers of the deep networks. Moreover, the central moment difference (CMD) [14] was also used to UDA by matching higher-order central moments across domain distributions.

UDA through domain-adversarial learning This kind of UDA is inspired by Generative Adversarial Network (GAN) [17], which uses adversarial training to learn domain-invariant representations. Along this line, the methods as Domain Adversarial training of Neural Networks (DANN) [3], Adversarial Discriminative Domain Adaptation (ADDA) [4] and Conditional Domain Adversarial Network (CDAN) [5] adopted a domain discriminator to distinguish the divergence among domain representations. Moreover, Wasserstein Distance Guided Representation Learning (WDGRL) [27] and Re-weighted Adversarial Adaptation Network (RAAN) [28] predicted the distribution distance between the source and target domain samples via a domain critical network with adversarial learning. Maximum Classifier Discrepancy (MCD) [11] and Sliced Wasserstein discrepancy (SWD) [12] performed domain alignment through building task-specific classifiers as domain discriminators to train domain-invariant representations. Recently, Domain-symmetric Networks (SymNets) [29] was modeled with an improved adversarial learning objective with a two-layer domain obfuscation structure. In addition, Transferable Adversarial Training (TAT) [30] was modeled to reduce the cross-domain gap by performing UDA with the generated transferable samples, as well as the reverse-trained depth classifier to make consistent predictions on the transferable samples.

UDA with metric learning To facilitate the alignment across domain samples, the methodology of distance metric has been introduced in UDA. Most of related works were modeled with metric loss on the samples [31–34] or proxies [35–38] to learn class distinction boundary, in which the key issue is how to characterize both the intra- and inter-class differences. In the article [39], the authors employ a memory mechanism and develop two types of nonparametric classifiers that assign pseudo-labels to target samples using only target data. Different from [39], we use the source domain data, then follow the K-nearest neighbors algorithm and employ a ratio test to assign the target sample pseudo-labels. In articles [40] and [41], the authors use soft-max contrast loss and noise contrast loss to characterize intra- and inter-class differences, respectively. We use the useful sample pair relation of pair domain adaptation classification to construct the sample pair similarity loss as processing multiple positive and negative sample pair information at one time. Although domain adaptation algorithms based on metric learning have been proposed by many previous studies, the principle of metric learning is rarely considered to improve conventional domain adaptation problems. Previous related work either required triplet losses with complex sampling strategies or did not use sample-level similarity relationships. In this article, we calculate the sample pair similarity loss from the sample level, which makes the close similarity more compact and the dissimilar samples more discrete.

3 PROPOSED METHODOLOGY

In this section, we describe the details of our approach, and the overall architecture of our model is shown in Fig. 2. The symbols used in this article are defined in Table 1. First, in order to prevent the deviation of the trained model due to the large difference in the number of samples in the two domains, we weight the samples in the source domain and target domain. Secondly, we calculate the equilibrium factor ω according to the degree of domain alignment and class differentiability and use it to adjust the domain alignment and class difference loss to prevent excessive domain alignment. Finally, we construct the sample pair bias matrix to calculate the sample similarity loss and optimize the sample similarity loss to make the intra-class more compact and the inter-class more discrete. Consider the classification of image X in class C problems. For UDA, we are typically given a source domain $X_s = \left\{(\boldsymbol{x}_s^i, \boldsymbol{y}_s^i)\right\}_{i=1}^{N_s}$ with N_s labeled examples and a target domain $X_t = \left\{\boldsymbol{x}_t^j\right\}_{j=1}^{N_t}$ with N_t unlabeled examples.

3.1 Weight Adaptation

Sample Weighting: In the process of model training, when the sample number difference between the two domains is too large, it will lead to a deviation in model training, and the model will be biased toward the domain with a large sample number. In order to avoid such problems, we intuitively weight the samples before they are input into the model to prevent the model deviation caused by the unbalanced number of samples. For each domain, the weight of the sample should be inversely proportional to its total sample size in both domains. Specifically, we weight the samples of each domain as follows:

$$\bar{\boldsymbol{x}}_s^i = \alpha(\frac{N_s + N_t}{N_s})\boldsymbol{x}_s^i, i = 1, 2, ..., N_s \tag{1}$$

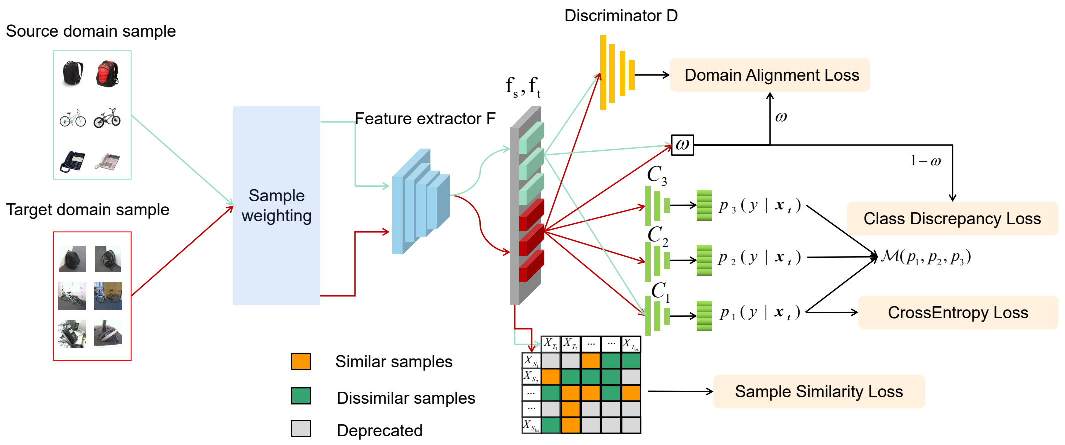


Fig. 2: The overall architecture of the UDA-DBADE method in this article. Our network architecture is divided into three modules: feature extractor (F), domain discriminator (D), the classifier (C_1, C_2, C_3), and related parameters ϕ_f, ϕ_d, ($\phi_{c1}, \phi_{c2}, \phi_{c3}$). The balance factor ω acts on the loss of domain alignment, and $(1-\omega_t)$ acts on the loss of class discrepancy to balance the degree of domain alignment. Additionally, we use source domain features and target domain features to construct a sample pair bias matrix $\boldsymbol{S}$, which preserves the similarity between samples. Then, we use the value of bias matrix $\boldsymbol{S}$ to calculate the sample similarity loss.

$$\bar{x}_t^j = \alpha(\frac{N_t + N_s}{N_t})x_t^j, j = 1, 2, ..., N_t \tag{2}$$

where $\alpha \in (0, 1]$ is the hyperparameter controlling the degree of sample weighting.

Table 1: Definition of variables and symbols.

Notation	Meaning
x_s^i, x_t^j	The source/target domain samples
y_i^s	The label of the i-th source domain sample
$\bar{x}_s^i, \bar{x}_t^j$	The weighted source/target domain samples
N_s	The number of samples from the source domain
N_t	The number of samples from the target domain
B_S, B_T	The source and target domain batch sample collection
F	The model feature extractor
D	The model discriminator
C_i	The i-th model classifier
C	The class number of the source or target domain
$\boldsymbol{W}$	The projection matrix
$\boldsymbol{d}$	The dimension after dimensionality reduction
$\boldsymbol{S}$	The sample bias matrix
t	The current iteration number
T	The maximum iterations

Domain Alignment and Class Discriminability Weighting: During the optimization process of domain alignment loss and class discrepancy loss, excessive domain alignment or class discrimination is easy to occur, leading to the occurrence of negative transfer. In order to avoid this situation and make the domain alignment and class differentiability be optimized together, we calculate the domain alignment degree and class discrepancy during each iteration and get the balance factor ω of the current iteration. ω is used as the weight of domain alignment loss and class discrepancy to control the degree of domain alignment. We use Maximum Mean Discrepancy (MMD) and Linear Discriminant Analysis (LDA) [42] to calculate the degree of domain alignment and class differentiability of the current network model. As one of the widely used distance measures for domain adaptation, MMD can express the difference in cross-domain distribution between the source domain and target domain after mapping:

$$MMD(X_S, X_T) = \|\frac{1}{N_s}\sum_{i=1}^{N_s} F(\bar{x}_s^i) - \frac{1}{N_t}\sum_{j=1}^{N_t} F(\bar{x}_t^j)\|_H^2 \tag{3}$$

In addition, the definition of linear class discriminator $LDA(\boldsymbol{W})$ based on LDA is as follows:

$$\arg\max_{\boldsymbol{W}} LDA(\boldsymbol{W}) = \frac{tr(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S_b}\boldsymbol{W})}{tr(\boldsymbol{W}^{\mathrm{T}}\boldsymbol{S_w}\boldsymbol{W})} \tag{4}$$

where $\boldsymbol{W} \in R^{n\times d}$ is the projection matrix, $\boldsymbol{d}$ is the dimension projected into low-dimensional space, $\boldsymbol{S_b}$ is the interclass divergence matrix, and $\boldsymbol{S_w}$ is the intra-class divergence matrix. By maximizing the inter-class divergence matrix and minimizing the intra-class divergence matrix, the larger $LDA(\boldsymbol{W})$ value is obtained. A larger $LDA(\boldsymbol{W})$ value represents the smallest difference within a class and the largest difference between classes, that is, the class is more distinguishable. We normalize the original values obtained from Eq. (3) and Eq. (4) by using the min-max standardization method. In order to balance the complexities of $MMD(X_S, X_T)$ and $LDA(\boldsymbol{W})$, we normalize them respectively in Eq. (5) and Eq. (6):

$$MMD(X_S,X_T)_t^* = \frac{MMD(X_S,X_T)_t - MMD(X_S,X_T)_{\min}}{MMD(X_S,X_T)_{\max} - MMD(X_S,X_T)_t + \delta} \tag{5}$$

where δ is an infinitesimal value (e.g., 1e-3) to guarantee the denominator not equal to zero, $t \in [1, T]$ indicates current iteration number. $MMD(X_S,X_T)_t$ represents the domain alignment degree at current tth iteration. $MMD(X_S,X_T)_{\min}$ and $MMD(X_S,X_T)_{\max}$ respectively indicate the minimal value and maximal value of $MMD(X_S,X_T)$ in previous iterations of the model training process, and are updated at each iteration.

$$LDA(\boldsymbol{W})_t^* = \frac{LDA(\boldsymbol{W})_t - LDA(\boldsymbol{W})_{\min}}{LDA(\boldsymbol{W})_{\max} - LDA(\boldsymbol{W})_t + \delta} \tag{6}$$

where $LDA(\boldsymbol{W})_t$ represents the class discrepancy at current tth iteration. $LDA(\boldsymbol{W})_{\min}$ and $LDA(\boldsymbol{W})_{\max}$ respectively indicate the minimal value and maximal value of $LDA(\boldsymbol{W})$ in previous iterations of the model training process, and are also updated at each iteration. We can easily draw the conclusion from Eq. (5) and Eq. (6) that, $MMD(X_S,X_T)_t^* \in [0,1]$ and $LDA(\boldsymbol{W})_t^* \in [0,1]$.

For the sake of dynamically balancing between domain alignment and cross-domain discrimination, with the normalized $MMD(X_S,X_T)_t^*$ and $LDA(\boldsymbol{W})_t^*$, we design the balancing factor ω_t for the tth iteration as follows:

$$\omega_t = \frac{MMD(X_S,X_T)_t^*}{MMD(X_S,X_T)_t^* + (1 - LDA(\boldsymbol{W})_t^*)} \tag{7}$$

The smaller the value of $MMD(X_S,X_T)_t^*$, the better the alignment of the current domain, and the larger the value of $LDA(\boldsymbol{W})_t^*$, the stronger the distinguishability of the current class. When the degree of domain alignment is far worse than the degree of class discrimination, the $MMD(X_S,X_T)_t^*$ approaches 1, the $(1 - LDA(\boldsymbol{W})_t^*)$ approaches 0, and the ω_t approaches 1. When the degree of domain alignment is far better than the class discriminability, the $MMD(X_S,X_T)_t^*$ approaches 0, the $(1 - LDA(\boldsymbol{W})_t^*)$ approaches 1, and the ω_t approaches 0. Then, ω_t gradually converges to the value of 0.5 with increased iteration epochs.

3.2 Domain Alignment and Class Discrepancy

Adversarial learning has been widely used in domain adaptation tasks to learn domain invariant representation. In adversity-learning, weighted samples $\bar{\boldsymbol{x}}_s^i$ and $\bar{\boldsymbol{x}}_t^j$ are used as the input of feature extractor F to obtain domain-invariant feature representation. By training model network, parameter ϕ_f of feature extractor F and parameter ϕ_d of domain discriminator D are updated to optimize the domain alignment loss in the following formula:

$$\begin{aligned}\min_{\phi_f}\max_{\phi_d} \mathcal{L}_{dom}(\phi_f,\phi_d) &= \tfrac{1}{N_s}\sum_{i=1}^{N_s}\log[D(F(\bar{\boldsymbol{x}}_s^i))] \\ &+ \tfrac{1}{N_t}\sum_{j=1}^{N_t}\log[1 - D(F(\bar{\boldsymbol{x}}_t^j))]\end{aligned} \tag{8}$$

The domain alignment task can be achieved by optimizing Eq. (8). However, optimizing domain alignment loss does not guarantee class distinguishability. In order to get the feature representations which have good discriminability, we are inspired by MCD [11] to maximize the discrepancies between the classifiers, which benefits for generating more discriminative features. Therefore, the classification discrepancy measure is defined as Eq. (9):

$$\begin{aligned}\mathcal{M}(p_1,p_2,p_3) &= \tfrac{1}{C}\sum_{k=1}^{C}\|p_1^k - p_2^k\|_1 + \tfrac{1}{C}\sum_{k=1}^{C}\|p_1^k - p_3^k\|_1 \\ &+ \tfrac{1}{C}\sum_{k=1}^{C}\|p_2^k - p_3^k\|_1\end{aligned} \tag{9}$$

where the classifiers C_1, C_2 and C_3 are obtained through pre-training on the source domain. In addition, p_1, p_2 and p_3 denote the probability labels predicted by the classifiers C_1, C_2 and C_3, respectively. The superscript K represents categories, for example, p_1^k, p_2^k and p_3^k represent probability outputs of class k. In order to obtain class features with large discrepancy, we optimize the loss of class discrepancy as follows:

$$\min_{\phi_f,\phi_{c1}}\max_{\phi_{c2},\phi_{c3}} \mathcal{L}_{cl}(\phi_f,\phi_{c1},\phi_{c2},\phi_{c3}) = E_{\bar{\boldsymbol{x}}_t^j \sim X_T}[\mathcal{M}(p_1,p_2,p_3)] \tag{10}$$

First, we train the feature extractor F by fixing C_2 and C_3 to minimize feature discrepancy. Then, we fix F and C_1 to maximize the discrepancy between classifiers C_2 and C_3 in the target domain. ϕ_{c1}, ϕ_{c2} and ϕ_{c3} are parameters of classifiers C_1, C_2 and C_3, respectively. It is worth noting that different from MCD [11], we add a main classifier C_1, whose decision hyperplane is between C_2 and C_3, to make the distance between the classified samples and the decision boundary larger.

It can be seen from Eq. (7) that the larger the value of ω_t, the worse the degree of domain alignment, and the larger the value of $(1-\omega_t)$, the worse the class difference. With this observation, we take ω_t as the weight of the domain alignment loss, and $(1 - \omega_t)$ as the weight of the class difference loss. The weighted model loss is as follows:

$$\min_{\phi_f,\phi_{c1}}\max_{\phi_d,\phi_{c2},\phi_{c3}} \omega_t\mathcal{L}_{dom}(\phi_f,\phi_d) + (1-\omega_t)\mathcal{L}_{cl}(\phi_f,\phi_{c1},\phi_{c2},\phi_{c3}) \tag{11}$$

When the degree of domain alignment is less than the class distinguishability, we increase the weight of domain alignment loss. In contrast, when the class distinguishability is less than the domain alignment degree, we increase the weight of class distinguishability. With the iteration of training, we use ω_t to adjust the domain alignment and class discrepancy loss, and this weight enables the model to maintain the consistency of domain alignment and class differentiability, effectively avoiding negative migration.

3.3 Sample Similarity Loss

To constrain alignment at the class level, we explore the bias relationships between source and target sample pairs for each batch at the sample level and use them in calculating sample similarity losses. However the target domain samples are unlabeled, if the classifier trained by source domain data is used to label the target domain with pseudo-labels, the sample bias relationship we get is wrong due to the influence of label noise. Therefore, we use the *KNN* classifier to assign pseudo-labels to the target domain samples. First of all, for each target domain sample, we take the first K source domain samples closest to it as pseudo-label samples. Secondly, the pseudo-label samples are labeled and voted, and the results are regarded as pseudo-label in the target domain. Finally, the label information is used to fill the sample bias matrix as follows: $S_{ij} = 1, if \;\; \boldsymbol{y}_s^i = \hat{\boldsymbol{y}}_t^j;\; S_{ij} = -1,\; otherwise$. The pseudo-label of the target sample obtained from the *KNN* algorithm also has noise sample. Therefore, after constructing sample bias matrix $\boldsymbol{S}$, we filter out pseudo-labels that may be noise. We use the rejection confidence measure based on the neighborhood similarity test commonly used in *KNN* to filter noise labels. B_S and B_T respectively represent the sample set of the current batch in the source and target domain. Define N_{j}^p to represent the sample set of similar source domain near the target sample $\bar{\boldsymbol{x}}_t^j \in B_T$, which is obtained by $N_j^p = \{\bar{\boldsymbol{x}}_s^i \in B_S | \boldsymbol{y}_s^i = \hat{\boldsymbol{y}}_t^j\}$. Similarly, N_{j}^n is defined to represent the dissimilar source domain sample set near the target domain sample $\bar{\boldsymbol{x}}_t^j \in B_T$,which is obtained by $N_j^n = \{\bar{\boldsymbol{x}}_s^i \in B_S | \boldsymbol{y}_s^i \neq \hat{\boldsymbol{y}}_t^j\}$. We calculate the ratio of similar set to dissimilar set to serve as the consistency score Ω_j of pseudo-label prediction of sample $\bar{\boldsymbol{x}}_t^j$ in the target domain. The definition is as follows:

$$\Omega_j = \frac{\sum_{\bar{\boldsymbol{x}}_s^i \in N_j^p} d(\boldsymbol{f}_s^i, \boldsymbol{f}_t^j)}{\sum_{\bar{\boldsymbol{x}}_s^i \in N_j^n} d(\boldsymbol{f}_s^i, \boldsymbol{f}_t^j)} \tag{12}$$

where $\boldsymbol{f}_s^i$ and $\boldsymbol{f}_t^j$ respectively represent the output features of the samples in the source domain and target domain calculated using feature extractor F, and $d(.,.)$ is the similarity score between features. After sorting Ω_j from large to small, the confidence factor μ is used to select the sorted confidence samples, and the bias matrix value of the remaining target samples is set as $S_{ij} = 0$. For example, if our batch size is 64 and confidence factor $\mu = 0.75$, the first 48 target domain samples are taken as confidence samples in order of consistency score predicted by pseudo-label. When we randomly sample batches from the source and target domains, it can happen that some classes cannot be selected in the source domain, which is problematic. For example, some target samples might not have a corresponding true source sample, leading to incorrect pseudo-labels. To address this issue, we perform class-balanced sampling for the mini-batch B_S on the source domain, and extract the same representations for all classes of the source domain. For the target domain, the instances are sampled randomly since they are unlabeled. In this way, the sample information with noise labels will not be involved in the calculation of sample similarity loss $\mathcal{L}_S$, to prevent the influence of noise labels on model training.

For each source domain sample $\bar{\boldsymbol{x}}_s^i$, we divide the same batch of target domain samples into relevant sample set $B_T^{S_i^+} = \{\bar{\boldsymbol{x}}_t^j \in B_T | S_{ij} = 1\}$ and unrelated sample set $B_T^{S_i^-} = \{\bar{\boldsymbol{x}}_t^j \in B_T | S_{ij} = -1\}$. Using the above two sets, we optimize Eq. (13) below to make source domain sample $\bar{\boldsymbol{x}}_s^i$ more compact with related samples and more separated from those irrelated:

$$\mathcal{L}_S^i = -\log \frac{\sum\limits_{\bar{\boldsymbol{x}}_t^j \in B_T^{S_i^+}} e^{d(\boldsymbol{f}_s^i, \boldsymbol{f}_t^j)}}{\sum\limits_{\bar{\boldsymbol{x}}_t^j \in B_T^{S_i^+}} e^{d(\boldsymbol{f}_s^i, \boldsymbol{f}_t^j)} + \sum\limits_{\bar{\boldsymbol{x}}_t^j \in B_T^{S_i^-}} e^{d(\boldsymbol{f}_s^i, \boldsymbol{f}_t^j)}} \tag{13}$$

The overall similarity loss of the current batch of source domain samples is defined as follows:

$$\mathcal{L}_S = \frac{1}{|B_S|} \sum_{\bar{\boldsymbol{x}}_s^i \in B_S} \mathcal{L}_S^i \tag{14}$$

We use normalized inverse Euclidean distance [43] as the similarity measure, which is defined as follows:

$$d(\boldsymbol{f}_s^i, \boldsymbol{f}_t^j) = \frac{1}{1 + \|\boldsymbol{f}_s^i - \boldsymbol{f}_t^j\|^2} \tag{15}$$

If $S_{ij} = 1$, it means that $\bar{\boldsymbol{x}}_s^i$ and $\bar{\boldsymbol{x}}_t^j$ are similar sample pairs, and the similarity degree obtained by Eq. (15). Similarly, when $S_{ij} = -1$,it means that $\bar{\boldsymbol{x}}_s^i$ and $\bar{\boldsymbol{x}}_t^j$ dissimilar sample pairs, and the similarity degree value is close to 0. Next, we introduce how to construct sample bias matrix $\boldsymbol{S}$.

3.4 The Overall Objective

In order to transfer the source knowledge to supervise target model training, we also need to incorporate the source domain classification in the UDA process. As a result, taking into account the source domain classification, domain alignment, class discrepancy and sample similarity aforementioned, we can naturally design the overall objective function of UDA with Dynamic Bias Alignment and Discrimination Enhancement (UDA-DBADE) as follows:

$$\mathcal{L}_{\text{total}} = \min_{\phi_f,\phi_{c1}} \max_{\phi_d,\phi_{c2},\phi_{c3}} \mathcal{L}_{\text{sup}} + \mathcal{L}_S + \omega_t \mathcal{L}_{dom}(\phi_f,\phi_d) + (1-\omega_t)\mathcal{L}_{cl}(\phi_f,\phi_{c1},\phi_{c2},\phi_{c3}) \tag{16}$$

where

$$\mathcal{L}_{\text{sup}} = \frac{1}{N_s}\sum_{i=1}^{N_s} \mathcal{L}_{ce}(F(\tilde{x}_s^i), y_s^i) \tag{17}$$

denotes the classification loss on the source domain. As shown in Eq. (16), it mainly contains four parts: supervised classification loss $\mathcal{L}_{\text{sup}}$ of the source domain, similar relationship loss $\mathcal{L}_S$ between samples, domain alignment loss $\mathcal{L}_{dom}$, and class difference loss $\mathcal{L}_{cl}$. For the sake of clarification, we summarize the complete steps of UDA-DBADE in Algorithm 1.

Remark It should be noted that there are significant differences between our model and the work [24] in the following aspects:

Firstly, the work of Xiao et al. [24] mainly focuses on domain adaptation through dynamic weighted learning, in which the more refined sample-level discriminative alignment was not modeled in its subsequent domain adaptation (see Eq. (14) in [24]). In contrast, we have specifically designed a sample bias matrix in which the target instances are distinguished into similar, dissimilar and irrelevant to the source samples in terms of their inverse Euclidean distance. Then, we accordingly built a minimized sample similarity loss to guide the intra-class scatter more compact and the inter-class as pushed away as possible, which is not considered at all in [24]. For more details, please refer to Section 3.3 of this article, as well as the second term (i.e. $\mathcal{L}_S$) of our overall objective in Eq. (16).

Secondly, the work of Xiao et al. [24] has not specially explored the effect of the involved pseudo-label noise on its model performance. By comparison, we have definitely constructed a sample bias matrix and assigned pseudo-labels to target instances through the *KNN* algorithm and filled the sample bias matrix with the generated label information. More importantly, we designed a rejection confidence measurement to filter out those label noises from the pseudo-labels. In this way, the quality of the pseudo-labels in our model is greatly improved, which consequently benefits the subsequent domain adaptation.

Algorithm 1 The UDA-DBADE Algorithm

```
Input: $X_S = \{x_s^i\}_{i=1}^{N_s}$, $X_T = \{x_t^j\}_{j=1}^{N_t}$, $Y_S = \{y_s^i\}_{i=1}^{N_s}$, iteration number $T$.
Output: The classifier $C_1, C_2, C_3$.
 1: Initialize $iter = 0$;
 2: Sample weighting by Eq. (1) and Eq. (2);
 3: repeat
 4:     Compute balance factor $\omega_t$ by Eq. (7);
 5:     Compute domain alignment loss $\mathcal{L}_{dom}$ by Eq. (8);
 6:     Compute class divergency loss $\mathcal{L}_{cl}$ by Eq. (10);
 7:     Compute across entropy loss $\mathcal{L}_{\text{sup}}$ by Eq. (17);
 8:     Compute bias matrix S by Eq. (8);
 9:     Compute sample similarity loss $\mathcal{L}_S$ by Eq. (13);
10:     Compute total loss $\mathcal{L}_{\text{total}}$ by Eq. (16);
11:     Update $F$ by Eq. (8), Eq. (10) and Eq. (13);
12:     Update $D$ by Eq. (8);
13:     Update $C_1, C_2, C_3$ by Eq. (10) and Eq. (17);
14:     $iter = iter + 1$;
15: until $iter > T$.
```

4 EXPERIMENTS

In this section, we conduct several experiments to evaluate the validity of the proposed method. First, we introduce four UDA datasets: Digits, Office-31, Imageclef-DA, and Office-Home, along with their experimental settings. Then, we compare the proposed method with existing methods. Finally, we perform ablation experiments to verify the validity of each part of the model.

4.1 Datasets

Digital data sets[1]. We construct domain adaptive tasks among MNIST [44], USPS, and SVHN [45] three digital datasets. Both MNIST (M) and USPS (U) datasets are handwritten numeric datasets from 0~9. SVHN (S) is a data set of real images in Google Street View images. We perform domain adaptation experiments on M → U, U → M, and S → M tasks.

Office-31[2] [46] This dataset consists of three different domains, including Amazon (A), Webcam (W), and DSLR (D), each with 31 classes. We conduct experiments on all six domain adaptation tasks, namely, A → W, D → W, W → D, A → D, D → A, and W → A.

ImageCLEF-DA[3] [47] This dataset consists of three domains, including Caltech256 (C), ImageNet ILSVRC 2012 (I), and Pascal VOC 2012 (P), each with 12 classes.

1) http://yann.lecun.com/exdb/mnist/
2) http://ai.bu.edu/adaptation.html
3) https://www.imageclef.org/2014/adaptation

Office-Home[4]. This dataset is a large benchmark dataset containing around 15,500 images divided into 65 classes. The dataset comprises four domains: Artistic (Ar), Clip Art (Cl), Product (Pr) and Real-World (Rw).

Table 2: Accuracy (%) on the Digital data sets for unsupervised domain adaptation.

Method	M → U	U → M	S → M	Avg
ResNet-50	76.7±0.2	63.4±0.1	67.1±0.3	69.1
DAN	80.3±0.2	77.8±0.4	73.5±0.2	77.2
DANN	90.8±0.4	93.9±0.1	83.1±0.2	89.2
CDAN	93.9±0.2	96.9±0.2	88.5±0.3	93.1
CyCADA	95.6±0.1	96.5±0.2	90.4±0.1	94.2
CAT	90.6±2.3	80.9±3.1	**98.1**±1.3	89.9
SimNet	95.6±0.2	96.4±0.2	-	-
MCD	94.2±0.7	94.1±0.3	96.2±0.4	94.8
TPN	92.1±0.6	94.1±0.8	93.0±0.5	93.1
LWC	95.6±0.3	97.1±0.5	97.1±0.4	96.6
ETD	**96.4**±0.5	96.3±0.7	97.9±0.4	96.9
CGDM	96.0±0.2	97.0±0.1	97.6±0.5	96.8
Ours	96.3±0.6	**97.5**±0.4	98.0±0.3	**97.3**

4.2 Implementation Details

We compare the proposed method with several state-of-the-art domain adaptation methods: DAN [7], LDC [48], DANN [3], JAN [25], ADDA [4], GoGAN [49], CyCADA [50], CDAN [5], MCD [11], TAT [30], CAT [51], SimNet [52], TPN [53], SAFN [54], LWC [55], ETD [56], CGDM [57], CSDA [58]. According to the standard protocol of UDA, all labeled source domain samples and unlabeled target domain samples participate in the training phase. For the domain adaptation task on the handwritten digit set, we follow the protocol in MCD [11]. We use 2K images from MNIST and 1.8K images on USPS to perform domain adaptation tasks between MNIST and USPS, and use the entire training set to perform domain adaptation between SVHN and MNIST. During the experiment, to train our model, we use ADAM whose weight attenuation of the learning rate is 0.0005 to optimize the network weight parameters. The learning rate is set as 0.0002, the sample batch size is set as 128, and the number of training iterations is set as 200. The classification accuracy of the target domain is adopted as the evaluation standard of the experiment. For image data sets such as Office-31, the original data sets are programmed on PyTorch, and the original features of the dataset are extracted by ResNet [17] network pre-trained on ImageNet [59]. The classifier network of the model in this article is set to be a two-layer network, and the domain discriminator is also composed of two-layer networks including ReLU and Dropout (0.5). We use small batches of SGD with a lot size of 32, a learning rate of 0.001, and a momentum of 0.9.

[4] https://www.hemanthdv.org/officeHomeDataset.html

4.3 Experimental Results

In this section, we conduct extensive experiments to evaluate our model, and all comparative method results are taken from relevant literature. Experimental results on three data sets are shown in Table 2-5. Our method is superior to many previous methods in different data sets.

We present three-domain adaptation scenarios on handwritten numeral sets, and Table 2 reports the experimental results. The domain adaptation results of our method between MNIST and USPS reach 96.3% and 97.5%, respectively, and its classification accuracy is better than in previous work. The proposed method focuses on preventing excessive domain alignment and constructing a sample bias matrix by introducing metric learning to calculate sample similarity loss to make the classification boundary clearer and prevent negative migration.

We show the results of the six preadaptation tasks on the Office-31 dataset and their averages in Table 3. We observe that the proposed method achieves the best results on two tasks with an average accuracy of 86.9% , which is superior to the previous comparison method. The accuracy of the model is 100% in W→D and D→W tasks. From the observation of classification accuracy, it can be seen that the proposed method can effectively balance the degree of domain alignment and class differentiation to prevent excessive domain alignment and smooth the classification boundary by using similarity loss among samples, thus improving the performance of the classifier in the target domain. For D→A and W→A tasks with large domain displacement and difficult domain adaptation, our model can still achieve 71.2% and 71% classification accuracy, which is better than 71.0% and 67.8% of the Enhanced Transport Distance (ETD) [56].

In Table 4, we show the results of six preadaptation tasks on the ImageCLEF-DA dataset and their average values. When training is stopped after 200 iterations, it is shown in Table 4 that our method is superior to the previous work and achieve the best average accuracy (89.8%).

The evaluation results on Office-Home are reported in Table 5. It can be observed that the average accuracy of the proposed method UDA-DBADE achieves 70.4%, which is higher than 70.3% of GSDA. More importantly, UDA-DBADE achieves significant improvement on Pr→Ar and Rw→Ar tasks. It demonstrates the advantage of this method, especially when deal with transferring from a complicated scenario to a simple scenario. Moreover, when encounter-

Table 3: Accuracy (%) on Office-31 dataset for unsupervised domain adaptation (ResNet-50).

Method	A → W	D → W	W → D	A → D	D → A	W → A	Avg
ResNet-50	68.4±0.2	96.7±0.1	99.3±0.1	68.9±0.2	62.5±0.2	60.7±0.2	76.1
DAN	80.5±0.4	97.1±0.2	99.6±0.1	78.6±0.2	63.6±0.3	62.8±0.2	80.4
LDC	78.6±0.5	98.7±0.1	**100.0**±.0	79.1±0.3	61.9±0.3	59.6±0.5	79.7
DANN	82.0±0.4	96.9±0.2	99.1±0.1	79.7±0.4	68.2±0.4	67.4±0.5	82.2
ADDA	86.28±0.5	96.28±0.3	98.48±0.3	77.88±0.3	69.58±0.4	68.9±0.5	82.9
CDAN	**93.1**±0.1	98.0±0.1	**100.0**±.0	89.8±0.2	70.1±0.3	68.0±0.3	86.6
CAT	91.1±0.2	98.6±0.6	99.6±0.1	**90.6**±1.0	70.4±0.7	66.5±0.4	86.1
SimNet	88.6±0.5	98.2±0.2	99.7±0.2	85.3±0.3	**73.4**±0.8	**71.6**±0.6	86.2
SAFN	88.8±0.4	98.4±0.0	99.8±0.0	87.7±1.3	69.8±0.4	69.7±0.2	85.7
ETD	92.1±0.5	**100.0**±.0	**100.0**±.0	88.0±0.3	71.0±0.4	67.8±0.2	86.2
CGDM	90.1±0.2	99.3±0.1	**100.0**±.0	89.0±0.3	70.0±0.2	70.8±0.2	86.5
Ours	88.5±0.2	**100.0**±.0	**100.0**±.0	89.2±0.4	71.2±0.3	71.0±0.1	**86.9**

Table 4: Accuracy (%) on ImageCLEF-DA dataset for unsupervised domain adaptation (ResNet-50).

Method	I → P	P → I	I → C	C → I	C → P	P → C	Avg
ResNet-50	74.8±0.3	83.9±0.1	91.5±0.3	78.0±0.2	65.5±0.3	91.2±0.3	80.7
DAN	74.5±0.4	82.2±0.2	92.8±0.2	86.3±0.4	69.2±0.4	89.8±0.4	82.5
DANN	75.0±0.6	86.0±0.3	96.2±0.4	87.0±0.5	74.3±0.5	91.5±0.6	85.0
JAN	76.8±0.4	88.0±0.2	94.7±0.2	89.5±0.3	74.2±0.3	91.7±0.3	85.5
CDAN	76.7±0.3	90.6±0.2	97.0±0.3	90.5±0.3	74.5±0.2	93.5±0.3	87.1
CAT	76.7±0.2	89.0±0.7	94.5±0.4	89.8±0.3	74.0±0.2	93.7±1.0	86.3
SAFN	78.0±0.4	91.7±0.5	96.2±0.1	91.1±0.3	77.0±0.5	94.7±0.3	88.1
TAT	78.8±0.2	92.0±0.2	97.5±0.3	92.0±0.3	78.2±0.4	94.7±0.4	88.9
ETD	81.0±0.6	91.7±0.4	**97.9**±0.3	**93.3**±0.2	**79.5**±0.5	95.0±0.3	89.7
CGDM	78.7±0.2	93.3±0.1	97.5±0.2	92.7±0.2	79.2±0.1	95.7±0.2	89.5
Ours	**81.2**±0.2	**93.6**±0.1	97.5±0.3	92.4±0.4	77.6±0.2	**96.8**±0.3	**89.8**

Table 5: Accuracy (%) on Office-Home dataset for unsupervised domain adaptation (ResNet-50).

Method	Ar→Cl	Ar→Pr	Ar→Rw	Cl→Ar	Cl→Pr	Cl→Rw	Pr→Ar	Pr→Cl	Pr→Rw	Rw→Ar	Rw→Cl	Rw→Pr	Avg
ResNet-50	34.9	50.0	58.0	37.4	41.9	46.2	38.5	31.2	60.4	53.9	41.2	59.9	46.1
DAN	43.6	57.0	67.9	45.8	56.5	60.4	44.0	43.6	67.7	63.1	51.5	74.3	56.3
DANN	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
JAN	45.9	61.2	68.9	50.4	59.7	61.0	45.8	43.4	70.3	63.9	52.4	76.8	58.3
CDAN	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
TAT	51.6	69.5	75.4	59.4	69.5	68.6	59.5	50.5	76.8	70.9	56.6	81.6	65.8
GSDA	**61.3**	**76.1**	79.4	**65.4**	73.3	74.3	65.0	53.2	80.0	72.2	**60.6**	83.1	70.3
Ours	57.0	74.7	**79.8**	64.6	**74.1**	**74.6**	**65.2**	**55.1**	**81.0**	**74.6**	59.7	**84.3**	**70.4**

ing a large domain discrepancy, UDA-DBADE still achieves promising results on complex transfer tasks such as Ar→Rw, Cl→Rw and Pr→Rw, which further demonstrates its efficiency.

4.4 Experimental Analysis

In this section, we further analyze the advantages and disadvantages of the model from the convergence, parameter sensitivity, feature visualization, and ablation experiments of the proposed method.

Convergence Analysis

Since the objective of UDA-DBADE is optimized in iterative manner, we evaluate the classification accuracy with iterations. Specifically, Fig. 3(a) shows the experimental results of Digits domain adaptation task M→U, ImageCLEF domain adaptation task P→C, and office-31 domain adaptation tasks W→A and A→D, respectively. It can be seen that the accuracy ascends gradually and comes to stable with about 90 epochs.

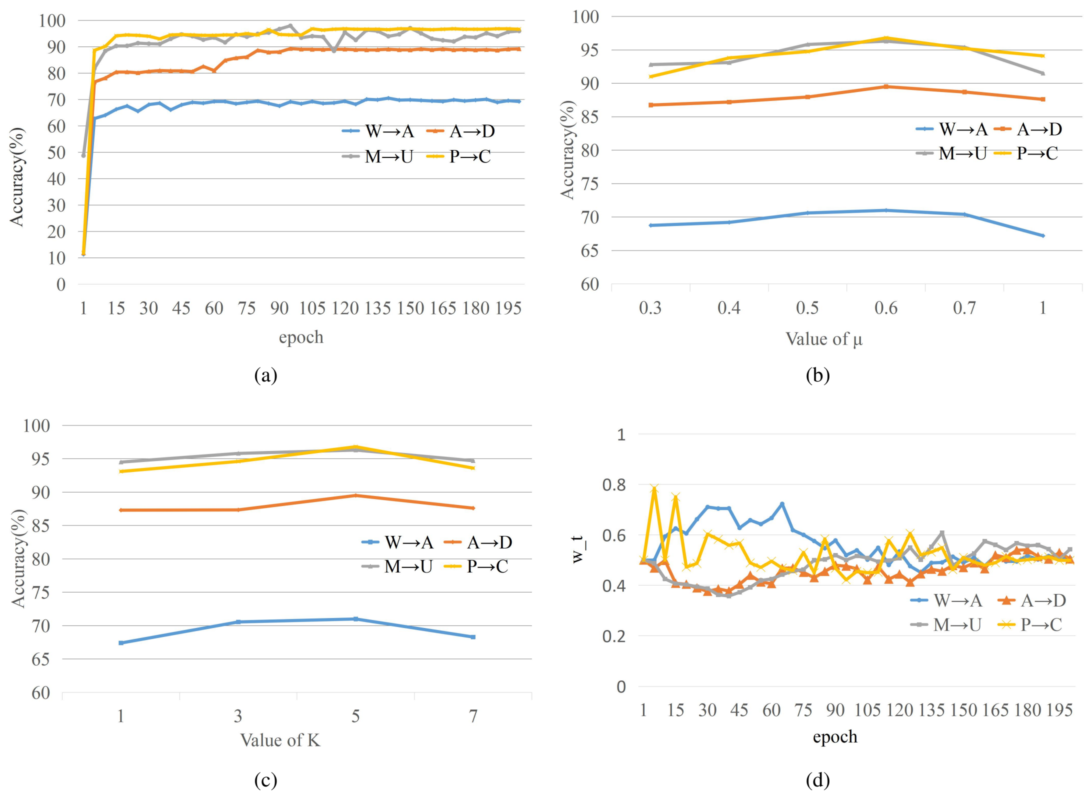


Fig. 3: The model convergence and parameter sensitivity. (a) Model convergence analysis of UDA-DBADE on domain adaptation tasks M→U, W→A, A→D, and P→C. (b)-(c) Sensitivity analysis of UDA-DBADE to parameters μ and K of domain adaptation tasks M→U, W→A, A→D, and P→C. (d) Iterative change analysis of UDA-DBADE to parameters ω_t of domain adaptation tasks M→U, W→A, A→D, and P→C.

Parameter Sensitivity Analysis

It can be concluded from Eq. (12) that a high μ value will lead to the selection of many noisy pseudo-label samples, thus leading to the deviation of model classification; By comparison, a low μ value will even filter out some confident samples that may be positive. In order to evaluate the influence of the confidence factor μ, we adjust its value and use it to predict the threshold of the consistency score Ω_j for target domain sample pseudo-labels. As shown in Fig. 3(b), sensitivity evaluations in terms of the confidence factor μ are conducted on the Digits domain adaptation task M→U, ImageCLEF domain adaptation task P→C, and office-31 domain adaptation tasks W→A and A→D, respectively. we observe that model reaches the optimum when $\mu = 0.75$, which is equivalent to accepting two-thirds of the pseudo-label prediction.

We use the K-nearest neighbor algorithm to assign pseudo-labels to the target samples, but the value of K is closely related to the accuracy of the pseudo-labels of the target samples. Larger values of K lead to assigning wrong pseudo-labels to target samples, while lower values of K miss the correct pseudo-labels of target samples. In order to evaluate the influence of the K, we adjust its value and use it to select pseudo-labels for target samples. As shown in Fig. 3(c), sensitivity evaluations in terms of the confidence factor K are conducted on the Digits domain adaptation task M→U, ImageCLEF domain adaptation task P→C, and office-31 domain adaptation tasks W→A and A→D, respectively. we observe that model reaches the optimum when $K = 5$. It is not difficult to understand that $K > 1$ is beneficial to the pseudo-label of the target sample, because it helps to deal with the noise prediction of the classifier boundary.

For the balance factor ω_t, its value changing rule in the model training process is shown in Fig 3(d). We observe that at the beginning, the value shakes seriously within the range of 0.4 to 0.8, and then gradually converges to the value of 0.5 with increased iteration epochs. It endorses the theoretical analysis below Eq. (7).

4.5 Feature Visualization

We use t-SNE to visualize the features learned by ResNet-50, DANN, and the model UDA-DBADE in this article on the

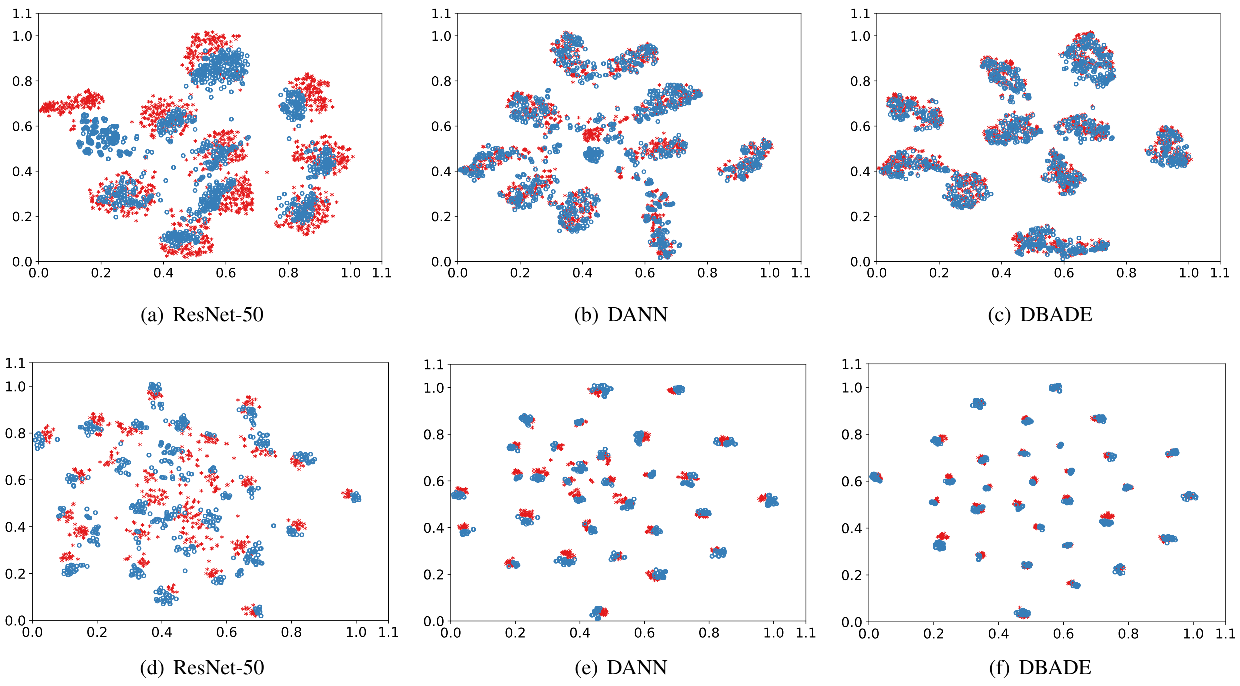


Fig. 4: t-SNE on the tasks of M→U and W→A, where red circles and blue circles denote the samples of source and target domains, respectively.

Digits domain adaptation task M→U, ImageCLEF domain adaptation task P→C, and office-31 domain adaptation tasks W→A and A→D, respectively, and the results are shown in Fig. 4. It can be seen from the Fig. 4 that the feature distribution of RESNET-50 is disordered, and the source and target domain are not aligned. DANN can alleviate this problem to some extent, but there are still big differences between the two domains. UDA-DBADE achieves the best adaptation results with clear class boundaries.

4.6 Ablation Studies

To evaluate the contribution of different modules to the model in this article, we conduct ablation experiment, and the experimental results are shown in Table 6-7. We select the Office-31 domain adaptation task W→A and Digits domain adaptation task M→U for the ablation experiments. We observe that sample weight, balance factor ω_t, and sample similarity loss L_S all play a key role in promoting the performance of the model.

5 CONCLUSION

This article proposed a kind of UDA through Dynamic Bias Alignment and Discrimination Enhancement (UDA-DBADE). Specifically, in UDA-DBADE we define a dynamic balance factor by the ratio of the normalized cross-domain discrepancy over the discrimination. Afterwards, we construct domain alignment with adversarial learning as well as distinguishable representations through advancing the discrepancy of multiple classifiers, and dynamically balance them with the defined dynamic factor. Finally, we further construct a bias matrix to characterize the discrimination alignment between the source and target domain samples. Our experiments on multiple UDA data sets clearly showed that UDA-DBADE is superior to the most advanced methods.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62176128, the Open Projects Program of State Key Laboratory for Novel Software Technology of Nanjing University under Grant KFKT2022B06, the Fundamental Research Funds for the Central Universities No. NJ2022028, the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund, as well as the Qing Lan Project.

Table 6: The accuracy (%) of ablation experiments on the domain adaptation task W→A.

Sample weighting	Balancing factor ω	Sample similarity loss $\boldsymbol{S}$	Accuracy(%)
			67.9
√			68.3
√	√		69.5
√	√	√	71.0

Table 7: The accuracy (%) of ablation experiments on the domain adaptation task M→U.

Sample weighting	Balancing factor ω	Sample similarity loss $\boldsymbol{S}$	Accuracy(%)
			93.6
√			94.1
√	√		95.8
√	√	√	96.3

References

[1] Ding Y, Feng J, Chong Y, Pan S, Sun X. Adaptive sampling toward a dynamic graph convolutional network for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing, 2021

[2] Xu H, Yang M, Deng L, Qian Y, Wang C. Neutral cross-entropy loss based unsupervised domain adaptation for semantic segmentation. IEEE Transactions on Image Processing, 2021, 30: 4516–4525

[3] Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V. Domain-adversarial training of neural networks. The Journal of Machine Learning Research, 2016, 17(1): 2096–2030

[4] Tzeng E, Hoffman J, Saenko K, Darrell T. Adversarial discriminative domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, 7167–7176

[5] Long M, Cao Z, Wang J, Jordan M I. Conditional adversarial domain adaptation. arXiv preprint arXiv:1705.10667, 2017

[6] Geng B, Tao D, Xu C. Daml: Domain adaptation metric learning. IEEE Transactions on Image Processing, 2011, 20(10): 2980–2989

[7] Long M, Cao Y, Wang J, Jordan M. Learning transferable features with deep adaptation networks. In: International Conference on Machine Learning. 2015, 97–105

[8] Sun B, Feng J, Saenko K. Return of frustratingly easy domain adaptation. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2016

[9] Qing TIAN S P T MH. S. Self-adaptive label filtering learning for unsupervised domain adaptation. Frontiers of Computer Science doi=https://doi.org/10.1007/s11704-022-1283-6

[10] Ganin Y, Lempitsky V. Unsupervised domain adaptation by backpropagation. In: International Conference on Machine Learning. 2015, 1180–1189

[11] Saito K, Watanabe K, Ushiku Y, Harada T. Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 3723–3732

[12] Lee C Y, Batra T, Baig M H, Ulbricht D. Sliced wasserstein discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 10285–10295

[13] Peng X, Bai Q, Xia X, Huang Z, Saenko K, Wang B. Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, 1406–1415

[14] Zellinger W, Grubinger T, Lughofer E, Natschläger T, Saminger-Platz S. Central moment discrepancy (cmd) for domain-invariant representation learning. arXiv preprint arXiv:1702.08811, 2017

[15] Peng X, Saenko K. Synthetic to real adaptation with generative correlation alignment networks. In: 2018

IEEE Winter Conference on Applications of Computer Vision (WACV). 2018, 1982–1991

[16] Sun B, Saenko K. Deep coral: Correlation alignment for deep domain adaptation. In: European Conference on Computer Vision. 2016, 443–450

[17] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. Advances in Neural Information Processing Systems, 2014, 27

[18] Bousmalis K, Silberman N, Dohan D, Erhan D, Krishnan D. Unsupervised pixel-level domain adaptation with generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, 3722–3731

[19] Sener O, Song H O, Saxena A, Savarese S. Learning transferrable representations for unsupervised domain adaptation. In: Advances in Neural Information Processing Systems. 2016, 2110–2118

[20] Bousmalis K, Trigeorgis G, Silberman N, Krishnan D, Erhan D. Domain separation networks. Advances in Neural Information Processing Systems, 2016, 29: 343–351

[21] Kang G, Jiang L, Yang Y, Hauptmann A G. Contrastive adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 4893–4902

[22] Xie S, Zheng Z, Chen L, Chen C. Learning semantic representations for unsupervised domain adaptation. In: International Conference on Machine Learning. 2018, 5423–5432

[23] Pei Z, Cao Z, Long M, Wang J. Multi-adversarial domain adaptation. In: Thirty-second AAAI Conference on Artificial Intelligence. 2018

[24] Xiao N, Zhang L. Dynamic weighted learning for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, 15242–15251

[25] Long M, Zhu H, Wang J, Jordan M I. Deep transfer learning with joint adaptation networks. In: International Conference on Machine Learning. 2017, 2208–2217

[26] Yan H, Ding Y, Li P, Wang Q, Xu Y, Zuo W. Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017, 2272–2281

[27] Shen J, Qu Y, Zhang W, Yu Y. Wasserstein distance guided representation learning for domain adaptation. In: Thirty-Second AAAI Conference on Artificial Intelligence. 2018

[28] Chen Q, Liu Y, Wang Z, Wassell I, Chetty K. Reweighted adversarial adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 7976–7985

[29] Zhang Y, Tang H, Jia K, Tan M. Domain-symmetric networks for adversarial domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 5031–5040

[30] Liu H, Long M, Wang J, Jordan M. Transferable adversarial training: A general approach to adapting deep classifiers. In: International Conference on Machine Learning. 2019, 4013–4022

[31] Schroff F, Kalenichenko D, Philbin J. Facenet: A unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 815–823

[32] Oh Song H, Xiang Y, Jegelka S, Savarese S. Deep metric learning via lifted structured feature embedding. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, 4004–4012

[33] Sohn K. Improved deep metric learning with multi-class n-pair loss objective. In: Advances in Neural Information Processing Systems. 2016, 1857–1865

[34] Wang X, Han X, Huang W, Dong D, Scott M R. Multi-similarity loss with general pair weighting for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 5022–5030

[35] Aziere N, Todorovic S. Ensemble deep manifold similarity learning using hard proxies. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 7299–7307

[36] Kim S, Kim D, Cho M, Kwak S. Proxy anchor loss for deep metric learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 3238–3247

[37] Qian Q, Shang L, Sun B, Hu J, Li H, Jin R. Softtriple loss: Deep metric learning without triplet sampling. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, 6450–6458

[38] Movshovitz-Attias Y, Toshev A, Leung T K, Ioffe S, Singh S. No fuss distance metric learning using proxies. In: Proceedings of the IEEE International Conference on Computer Vision. 2017, 360–368

[39] Liang J, Hu D, Feng J. Domain adaptation with auxiliary target domain-oriented classifier. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, 16632–16642

[40] Oord A v d, Li Y, Vinyals O. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018

[41] Mnih A, Kavukcuoglu K. Learning word embeddings efficiently with noise-contrastive estimation. In: Advances in Neural Information Processing Systems. 2013, 2265–2273

[42] Dorfer M, Kelz R, Widmer G. Deep linear discriminant analysis. arXiv preprint arXiv:1511.04707, 2015

[43] Maaten V. d L, Hinton G. Visualizing data using t-sne. Journal of Machine Learning Research, 2008, 9(11)

[44] LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278–2324

[45] Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng A Y. Reading digits in natural images with unsupervised feature learning. 2011

[46] Saenko K, Kulis B, Fritz M, Darrell T. Adapting visual category models to new domains. In: European Conference on Computer Vision. 2010, 213–226

[47] Long M, Zhu H, Wang J, Jordan M I. Deep transfer learning with joint adaptation networks. In: International Conference on Machine Learning. 2017, 2208–2217

[48] Li S, Song S j, Wu C. Layer-wise domain correction for unsupervised domain adaptation. Frontiers of Information Technology & Electronic Engineering, 2018, 19(1): 91–103

[49] Liu M Y, Tuzel O. Coupled generative adversarial networks. Advances in Neural Information Rrocessing Systems, 2016, 29: 469–477

[50] Hoffman J, Tzeng E, Park T, Zhu J Y, Isola P, Saenko K, Efros A, Darrell T. Cycada: Cycle-consistent adversarial domain adaptation. In: International Conference on Machine Learning. 2018, 1989–1998

[51] Deng Z, Luo Y, Zhu J. Cluster alignment with a teacher for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, 9944–9953

[52] Pinheiro P O. Unsupervised domain adaptation with similarity learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018, 8004–8013

[53] Pan Y, Yao T, Li Y, Wang Y, Ngo C W, Mei T. Transferrable prototypical networks for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 2239–2247

[54] Xu R, Li G, Yang J, Lin L. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019, 1426–1435

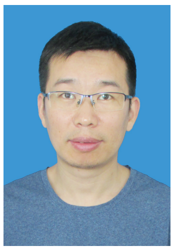
[55] Ye S, Wu K, Zhou M, Yang Y, Tan S H, Xu K, Song J, Bao C, Ma K. Light-weight calibrator: a separable component for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 13736–13745

[56] Li M, Zhai Y M, Luo Y W, Ge P F, Ren C X. Enhanced transport distance for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 13936–13944

[57] Du Z, Li J, Su H, Zhu L, Lu K. Cross-domain gradient discrepancy minimization for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021, 3937–3946

[58] Hu L, Kan M, Shan S, Chen X. Unsupervised domain adaptation with hierarchical gradient synchronization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 4043–4052

[59] Deng J, Dong W, Socher R, Li L J, Li K, Fei-Fei L. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009, 248–255



Qing Tian received his Ph.D. degree in computer science from Nanjing University of Aeronautics and Astronautics, China, in 2016. He is an Associate Professor in School of Computer and Software, Nanjing University of Information Science and Technology, China. He was an Academic Visitor at the University of Manchester, UK, from 2018 to 2019. He is the recipient of the Best Scientific Paper Award of ICPR, the Excellent Doctoral Dissertation Award of Jiangsu Province of China, etc. He has served as PC member for prestigious conferences, such as AAAI, IJCAI, PRICAI, ICPR, and reviewer for international journals and conferences, such as IEEE TPAMI, IEEE TNNLS, IEEE TCYB, IEEE TIFS, AAAI, IJCAI, ICDM, CVPR. His research interests include machine learning and pattern recognition.

Hong Yang received his B.S. degree in Computer Science and Technology from WUZHOU University in 2020, China. He is currently pursuing his master degree at Nanjing University of Information Science and Technology. His research interests include machine learning and pattern recognition.

Yanan Zhu received her B.S. degree in software engineering from BOHAI University in 2020, China. She is currently pursuing his master degree at Nanjing University of Information Science and Technology. Her research interests include machine learning and pattern recognition.

Heyang Sun received his B.S. degree in computer science from Nanjing University of Information Science and Technology (NUIST) in 2019, China. He is currently pursuing his master degree at the NUIST. His research interests include machine learning and pattern recognition.

Heng Xu received his B.S. degree in computer science from Nanjing University of Information Science and Technology (NUIST) in 2020, China. He is currently pursuing his master degree at the NUIST. His research interests include machine learning and pattern recognition.

Yi Chu received his B.S. degree in software engineering from Nanjing University of Information Science and Technology (NUIST) in 2020, China. He is currently pursuing his master degree at the NUIST. His research interests include machine learning and pattern recognition.