# Metric Learning for Domain Adversarial Network

**Haifeng Hu** *
School of Telecommunication and
Information Engineering,
Nanjing University of
Posts and Telecommunications,
Nanjing, 210023, China
Email: huhf@njupt.edu.cn

**Yan Yang**
**Yueming Yin**
School of Telecommunication and
Information Engineering,
Nanjing University of
Posts and Telecommunications,
Nanjing, 210023, China

**Jiansheng Wu**
School of Geographic and
Biological Information,
Nanjing University of
Posts and Telecommunications,
Nanjing, 210023, China

*Unsupervised domain adaptation based on domain adversarial training always suffers from easily-confused samples in the target domain due to the lack of supervised information. To solve this problem, we proposed a metric learning for domain adversarial network (ML-DAN) method, where the boundary threshold can be obtained to identify the easily-confused classes in the target domain by using the classification gap. Moreover, the margin between the positive and negative samples in the triplet loss is adjusted according to the boundary threshold during the metric learning, and an effective training batch mechanism is designed to accelerate the learning process. Finally, the domain alignment between the source and target domain is performed via domain adversarial training. Therefore, the easily-confused classes in the target domain can be separated indirectly to enhance the classification performance of the target domain. The experimental results on four standard benchmarks show that the ML-DAN method is superior to the state-of-the-art unsupervised domain adaptation methods in terms of the classification accuracy and negative transfer effect.*

## 1 Introduction

Many machine leaning and computer vision applications are the witnesses to the success of deep learning [1], including neural network approach [2] and deep reinforcement learning [3,4]. Especially, image classification based on deep learning has been applied widely due to its excellent performance. This promising performance depends on a large number of labeled data for supervised training. However, acquiring massive labeled data in the practical application is time-consuming and even infeasible [5–8]. Therefore, an effective way is to establish the deep learning model by using the labeled data in the source domain, and apply this model to the target domain without labeled data. However, there often exists the domain shift between the source domain and the target domain [9], and the deep learning model for the source domain always does not work well in the target domain. In this case, the domain adaptation method has been proposed to reduce the domain shift and enhance the generalization ability of the deep model.

The existing domain adaptation methods [10, 11] always aim to perform domain alignment between the source and target domain to alleviate the problem of domain shift. Specifically, deep learning model is trained by using the labeled data in the source domain, and domain adversarial training is performed to align the feature distribution of the source and target domain. Therefore, the classification performance in target domain is expected to be good enough. In fact, the existing methods based on adversarial training always cannot obtain the satisfying results. This is because the effect of alignment does not ensure the discriminability of the target domain, especially when the target data is relatively large compared with the source data. For example, the target domain samples are likely to be scattered on the classification boundary even after aligning operation, which can degrade the classification performance because of easily-confused classes in the target domain.

The target domain samples are likely to scatter on the classification boundary due to a lack of label information. Therefore, how to identify these overlapping classes in the target domain, called easily-confused classes, becomes the key to the improvement of classification performance. In-

*Address all correspondence related to ASME style format and figures to this author.

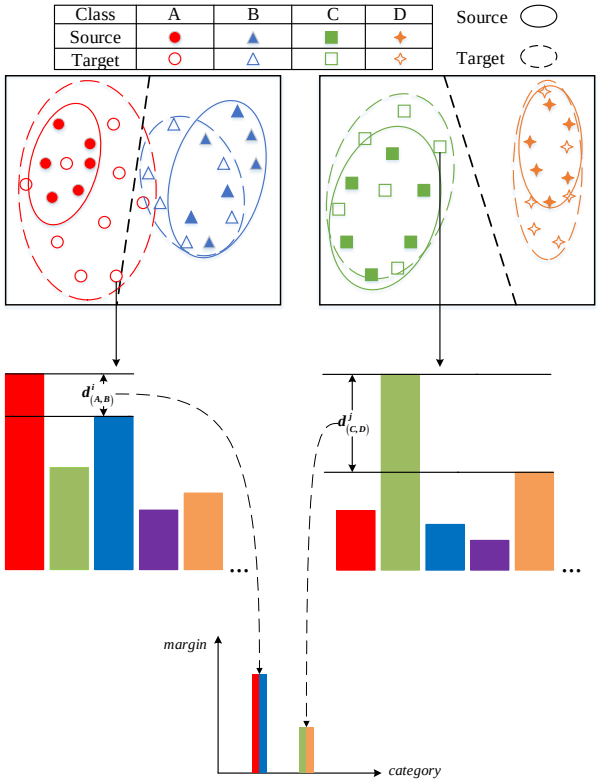


Fig. 1. An illustration of the margin for the easily-confused classes.

spired by [12], entropy minimization is used to force the classification boundary to pass through the low-density area of the target domain. Then for a given classifier, the entropies of these target samples scattering on the classification boundary are more likely to become relatively large compared with samples far away from the boundary. According to this rule, the output entropies of multi-binary classifiers can be used to identify the easily-confused classes in the target domain. Specifically, the high-entropy of a target sample means the small gap between the maximum classification value and the second-largest value of the multi-binary classifiers. This target sample with a small gap is in an uncertain state of classification. Therefore, we call this target sample a critical sample, and two classes corresponding to the maximum classification value and the second largest value are recognized as the easily-confused classes. The Fig. 1 provides an illustration of the margin for the easily-confused classes. From the above figure, we can observe that the easily-confused classes A and B are scattered around the decision boundary while the classes C and D are farther away from the boundary. The gap of the multi-binary classifiers for classes A and B is less than the gap for classes C and D. Therefore, the higher margin should be allocated to the easily-confused classes A and B to further separate them. In summary, according to the entropy theory, a relatively larger margin corresponds to a high entropy value for the multi-binary classifiers, which means there exist some easily-confused classes for this classifier. Hence, the margin based on the entropy theory is used to measure the degree to which the classification model matches the target domain. Specifically, the margin is utilized to identify the easily-confused classes in the target domain.

In order to alleviate the problem of easily-confused classes, domain adversarial training together with metric learning are applied to separate the easily-confused classes in the source domain, and to increase the margin between these classes in the target domain indirectly. As a result, the classification accuracy of the target domain can be improved significantly in the scenario of unsupervised domain adaptation. On this basis, the classification gap is defined as the difference between the maximum output of the multi-binary classifier and the second largest value, and we propose a metric learning for domain adversarial network (ML-DAN) where the classification gap and the associated boundary threshold are used to identify the easily-confused classes in the target domain. For each pair of easily-confused classes, an effective training batch mechanism is designed to dynamically adjust the margin between the positive pairs and negative pairs of the triplet loss in the source domain. Finally, the domain adversarial network performs the domain alignment to indirectly separate the easily-confused classes in the target domain. Therefore, the generalization ability of ML-DAN can be guaranteed.

The contribution of this paper can be listed as follows:

- In the scenario of unsupervised domain adaptation, the classification gap and associated boundary threshold for the multi-binary classifiers are proposed to identify the critical samples and easily-confused classes in the target domain.
- A metric learning for domain adversarial network (ML-DAN) is proposed to dynamically adjust the margin between the positive pairs and negative pairs of the triplet loss in the source domain according to the boundary threshold. Furthermore, an effective training batch mechanism is designed to accelerate the above metric learning process. Finally, the easily-confused classes in the target domain can be separated indirectly via domain adversarial training.
- The experimental results on four standard benchmarks show that the ML-DAN is superior to the state-of-the-art unsupervised domain adaptation methods in terms of classification accuracy and negative transfer effect, which verifies the effectiveness and advantage of the proposed method.

## 2 Related Work

### 2.1 Domain Adaptation

Domain adaptation mainly solves the problem of the domain shift between the source domain and the target domain. In recent years, the research on domain adaptation can be categorized as the discrepancy minimizing method and adversarial training method. One of the most representative methods for discrepancy method is to use maximum mean discrepancy (MMD) loss to map different domain distributions [13]. In [14], deep adaptation network (DAN) embeds hidden representations of all task-specific layers into a reproduc-

ing kernel Hilbert space where the mean embeddings of different domain distributions can be explicitly matched. Joint distribution adaptation (JDA) [15] aims to construct new feature representations by jointly adapting both the marginal and conditional distribution in a dimensionality reduction way. On this basis, a joint adaptation network (JAN) [16] method learns a transfer network by aligning the joint distributions of multiple domain-specific layers across domains based on a joint maximum mean discrepancy (JMMD) criterion.

The adversarial training method is to use the adversarial training to reduce domain shift. Recently, generative adversarial networks (GAN) [17] has received so much attention, and it is natural to apply the idea of adversarial learning to the domain adaptation problem. Domain-adversarial neural network (DANN) [18] introduces Gradient Reversal Layer (GRL) to enable the feature extractor and domain classifiers to conduct adversarial training for domain alignment. Adversarial discriminative domain adaptation (ADDA) [11] proposes a general framework for adversarial domain adaptation. In [19], multi-adversarial domain adaptation (MADA) captures multimode structures to enable the fine-grained alignment of different data distributions via multiple domain discriminators. Conditional domain adversarial network (CDAN) [20] proposes a new domain adaptation method, where two novel strategies, multilinear conditioning and entropy conditioning, are used to align the multimodal distributions. Saito et al. [21] attempted to align distributions of source and target domain by utilizing the task-specific decision boundaries.

All above-mentioned methods try to align the distribution between the source and target domain. However, there always exist the easily-confused classes in the target domain due to lack of the supervision information, which significantly limits the performance of unsupervised domain adaptation.

### 2.2 Metric Learning

Metric Learning [22] aims to map the data into the metric space and learn distance metrics to better measure the similarity between different objects. The most representative works are included as follows. Xing et al. [23] proposed a discriminative distance measure by minimizing the distance between similar pairs while maintaining the distance between dissimilar samples. Weinberger et al. [24] learned a Mahalanobis distance metric for the k-nearest neighbor (kNN) algorithm from the labeled samples. Davis et al. [25] proposed a Mahalanobis distance function by minimizing the relative entropy between multivariate Gaussians.

Some deep metric learning methods have been proposed to learn more discriminative metrics by exploiting the nonlinear capability of deep learning. For example, in [26], Duan et al. proposed a deep localized metric learning method to learn more fine-grained metrics for recognition tasks. Recently, some commonly-used losses have been studied in deep metric learning, such as lifted structure loss [27], triplet loss [28], triplet center loss [29], and quadruplet loss [30]. In particular, triplet loss [28] shows its promising performance in deep metric learning by enforcing a margin between positive and negative sample pairs during the training procedure. The triplet loss is to minimize the distances between the anchor and positive samples while maximizing the distances between the anchor and negative samples. After training by using the triplet loss, the distance in the embedding space can measure the semantic similarity degree between various samples. In ML-DAN, the triplet loss is applied to separate the samples of easily-confused classes to enhance the classification performance. Inspired by [28], in this paper, we dynamically adjust the margin according to the degree of the confusion between classes in the target domain, and employ a training batch mechanism to select the triplets that cause the triplet loss for accelerating the metric learning process.

## 3 Metric Learning in Unsupervised Domain Adaptation

### 3.1 Preliminaries

In the scenario of the unsupervised domain adaptation, labeled samples are drawn from the source domain and denoted by $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$, where $\boldsymbol{x}_i^s$ and $y_i^s$ are the $i$th source sample and its label, respectively, i.e., $\boldsymbol{x}_i^s \in \mathcal{X}_S$ and $y_i^s \in \mathcal{Y}_S$. $\mathcal{X}_S$ and $\mathcal{Y}_S$ represent the data space and label space of the source domain, respectively. Similarly, $n_t$ unlabeled samples are drawn from the target domain and denoted by $\mathcal{D}_t = \{(\boldsymbol{x}_i^t)\}_{i=1}^{n_t}$, where $\boldsymbol{x}_i^t$ is the $i$th target sample, i.e., $\boldsymbol{x}_i^t \in \mathcal{X}_T$. $\mathcal{X}_T$ represents the data space of the target domain, and $\mathcal{Y}_T$ represents the label space of the target domain, notice that $\mathcal{Y}_T$ is unknown during the training process. $S(\mathcal{X}_S, \mathcal{Y}_S)$ and $T(\mathcal{X}_T, \mathcal{Y}_T)$ denote the distribution of the source domain and the target domain, respectively. We assume that $S(\mathcal{X}_S, \mathcal{Y}_S) \neq T(\mathcal{X}_T, \mathcal{Y}_T)$, $\mathcal{Y}_S$=$\mathcal{Y}_T$ and $|\mathcal{Y}_S|$=$|\mathcal{Y}_T| = k$. Our goal is to simultaneously learn the feature extractor $F : (\mathcal{X}_S, \mathcal{X}_T) \to \mathbb{R}^n$, the label classifier $C_m : \mathbb{R}^n \to \mathbb{R}^k$, the multi-binary classifiers $C_b : \mathbb{R}^n \to \mathbb{R}^k$, and the domain classifier $D : \mathbb{R}^n \to \{0,1\}$ through training. For multi-binary classifiers, the $i$th binary-classifier corresponds to the $i$th class ($i \in \{1,...,k\}$), and for the domain classifier, the samples are predicted to be the source domain ($D(\boldsymbol{x}) = 1$) or the target domain ($D(\boldsymbol{x}) = 0$). So the classification model $H = C_m \circ F$ can accurately predict the labels of samples from the target domain.

### 3.2 Triplet loss for domain adversarial

The traditional domain adaptation method based on the domain adversarial cannot effectively address the problem of the wrongly classified target samples. In general, the domain alignment operation is a more coarse-grained approach for classifying the target samples due to lack of supervised information. Particularly, target samples scattered over the decision boundaries are easily misclassified. Therefore, in order to prevent these target samples from falling into the wrong category during the domain adversarial, we exploit the metric learning to push these target samples away from the corresponding boundary.

Metric learning is to project samples into the metric

space, such that the distances between samples of the same class are small, whereas the distances between samples of different classes remain relatively large. Therefore, metric learning is the effective way to separate the critical samples in the target domain from the boundaries. Among various losses in metric learning, the motivation of triplet loss is designed to enforce the margin between a pair of samples from the same class and a pair from the different classes, which can increase discriminability to each class in the target domain. Here we define triplet loss as follows:

$$\begin{aligned} E_G = \Big[ \big\| G\left(F\left(\boldsymbol{x}_i^s\right)\right) - G\left(F\left(\boldsymbol{x}_j^s\right)\right) \big\|^2 - \\ \left\| G\left(F\left(\boldsymbol{x}_i^s\right)\right) - G\left(F\left(\boldsymbol{x}_m^s\right)\right) \right\|^2 + \\ M_{(a,b)}^t \Big]_+, \\ \left(\boldsymbol{x}_i^s, y_i\right), \left(\boldsymbol{x}_j^s, y_j\right), \left(\boldsymbol{x}_m^s, y_m\right) \in \mathcal{D}_s, \\ y_i = y_j = y_a, y_m = y_b, \end{aligned} \tag{1}$$

where $F$ is the feature extractor, and $G$ is the metric generator. $\mathcal{D}_s$ denotes the source domain set. Source samples $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_j^s$ are drawn from the class $y_a$, and source sample $\boldsymbol{x}_k^s$ is drawn from the class $y_b$. Moreover, $M_{(a,b)}^t$ represents the margin between the class $y_a$ and $y_b$, which can be dynamically adjusted. $[q]_+ = q$ if $q \geq 0$; otherwise, $[q]_+ = 0$.

In the metric space, samples from the same class form the positive pairs, and samples from the different classes form the negative pairs. Minimizing the triplet loss in Eq. (1) is to enforce the margin between the positive pairs and the negative pairs, i.e., the distance between the negative pairs are expected to be at least $M_{(a,b)}^t$ larger than the distance between the positive pairs. Furthermore, $M_{(a,b)}^t$ can be adjusted according to the degree of confusion. Specifically, if the negative pairs come from the easily-confused classes in the target domain, $M_{(a,b)}^t$ needs to be set to a relatively large value. As a result, samples from the easily-confused classes can be separated by a larger margin.

### 3.3 Training batch mechanism based on margin

Each batch is composed of both labeled source samples and unlabeled target samples. The target samples in a batch are used to identify the easily-confused classes, and the source samples in the same batch are used to form the triplets in Eq. (1). During the metric learning process, for each triplet $\left(\boldsymbol{x}_i^s, \boldsymbol{x}_j^s, \boldsymbol{x}_k^s\right)$, if the corresponding triplet loss in Eq. (1) $E_G > 0$, then this triplet is an effective triplet to participant in the metric learning. Therefore, this triplet can accelerate the convergence speed of metric learning, and shrink the intra-class distance and extend the inter-class distance in metric space.

First, we need to identify the target samples in each batch close to the classification boundary, which are called critical samples. Obviously, the classes corresponding to these critical samples are recognized as easily-confused classes. Due to no label information available in the target domain, here we propose to identify the critical samples via the multi-binary classifiers.

In the supervised training, we train multi-binary classifiers for the classification of the source samples, where each class corresponds to a binary classifier. If a target sample is close to the classification boundary, the predicted values of the corresponding two or more binary classifiers will increase accordingly, i.e., the output entropy of the multi-binary classifiers will become relatively high. Specifically, a multi-binary classifiers $C_b^c|_{c=1}^k$ (i.e., $k$ is the number of classes) output predictions for a given target sample, and the gap between the maximum classification value and the second maximum value, called the classification gap, can be used to measure the degree of class confusion. The smaller the classification gap is, the more likely the target sample is the critical sample. Hence we define the maximum classification value and the second maximum value as follows:

$$y_a^j = \arg\max_{y_i} \hat{P}\left(y_i|\boldsymbol{x}_j^t\right), \boldsymbol{x}_j^t \in \mathcal{X}_T^{batch}, i \in \{1, ..., k\}, \tag{2}$$

$$y_b^j = \arg\max_{y_i \neq y_a^j} \hat{P}\left(y_i|\boldsymbol{x}_j^t\right), \boldsymbol{x}_j^t \in \mathcal{X}_T^{batch}, i \in \{1, ..., k\}, \tag{3}$$

where $\mathcal{X}_T^{batch}$ is the target sample set in a batch, $\hat{P}\left(y_i|\boldsymbol{x}_j^t\right)$ is the prediction value of the $i$th binary classifiers $C_b^i$ for $\boldsymbol{x}_j^t$, and $y_a^j$ and $y_b^j$ denote the class corresponding to the maximum and the second maximum prediction of the multi-binary classifiers, respectively. Notice that the output of the classifier for the target sample is called the pseudo label due to no label information during the training. We define the classification gap $d_{(a,b)}^j$ as:

$$d_{(a,b)}^j = \hat{P}\left(y_a^j|\boldsymbol{x}_j^t\right) - \hat{P}\left(y_b^j|\boldsymbol{x}_j^t\right), \boldsymbol{x}_j^t \in \mathcal{X}_T^{batch}. \tag{4}$$

On this basis, the training batch mechanism consists of the following steps:

1) $t$ denotes batch number counter, and $\mathcal{X}_T^{batch}$ denotes the target sample set in a batch. For each $\boldsymbol{x}_j^t \in \mathcal{X}_T^{batch}$, the classification gap $d_{(a,b)}^j$ is calculated according to Eq. (4), and $\mathcal{X}_T^{batch}$ is ranked by $d_{(a,b)}^j$ in ascending order. Then the top-$l$ target samples of $\mathcal{X}_T^{batch}$ form the critical sample set $\mathcal{X}_T^c$. For each $\boldsymbol{x}_j^t \in \mathcal{X}_T^c$, $y_a^j$ and $y_b^j$ correspond to a pair of easily-confused classes according to Eq. (2) and (3). Hence the easily-confused class set can be defined as $\mathcal{D}_y = \left\{\left(y_a^j, y_b^j\right)\right\}_{j=1}^l$.

2) Define the boundary threshold $\beta_{(a,b)}$ as

$$\beta_{(a,b)} = \alpha_0 + \mu \log\left(\frac{1}{d_{(a,b)}^j}\right), \forall\left(y_a^j, y_b^j\right) \in \mathcal{D}_y, \tag{5}$$

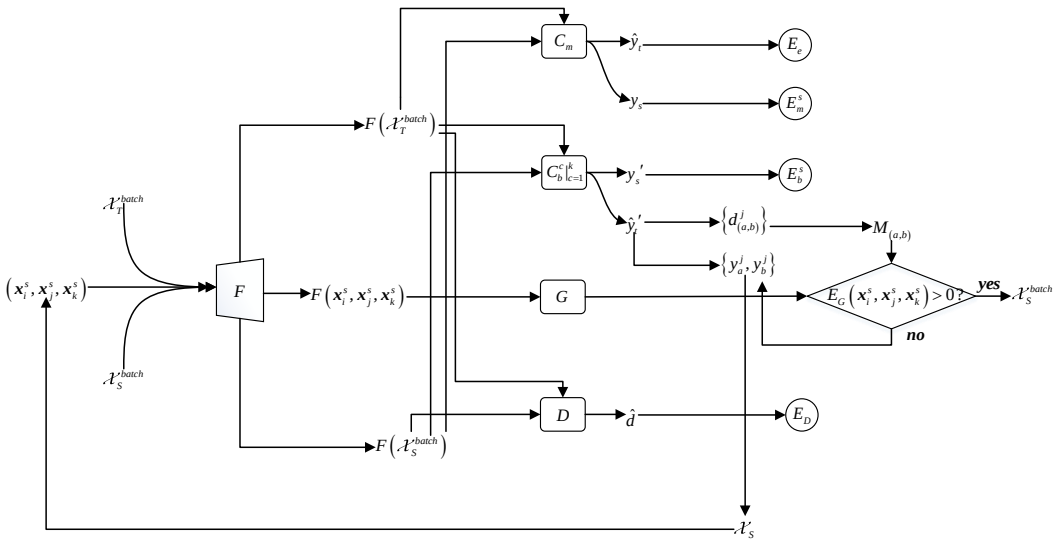


Fig. 2. Metric Learning for Domain Adversarial Network(ML-DAN).

where $\alpha_0$ is the initial value, $\mu$ is an adjustable coefficient, and $d_{(a,b)}^j$ is the classification gap of the easily-confused classes $y_a^j$ and $y_b^j$. The smaller $d_{(a,b)}^j$ corresponds to more confused classes in the feature space, which means the boundary threshold $\beta_{(a,b)}$ should be assigned a larger value to further separate $y_a^j$ and $y_b^j$.

3) Define the margin $M_{(a,b)}^t$ in the triple loss:

$$M_{(a,b)}^t = \begin{cases} \beta_{(a,b)}, & t = 1, \\ \frac{1}{t}\left((t-1)M_{(a,b)}^{t-1} + \beta_{(a,b)}\right), & t > 1, \end{cases} \tag{6}$$

where $t$ is the batch number during the training. According to the Equ. 5, the boundary threshold $\beta_{(a,b)}$ is calculated to measure the distance between the class a and b, an easily-confused class pair, by using target samples in one batch. In consideration of randomness, it is not guaranteed that $\beta_{(a,b)}$ can calculate an accurate and stable distance between an easily-confused class pair within one batch process. So we design to use the margin to express the more accurate easily-confused class distance in an iterative manner. The margin $M_{(a,b)}^t$ denotes the average of the boundary threshold $\beta_{(a,b)}$ over all $t$ batches, which can further improve the estimation accuracy of the classification gap of the easily-confused classes $y_a^j$ and $y_b^j$.

4) When $M_{(a,b)}^t$ is given, for each $\left(y_a^j, y_b^j\right) \in \mathcal{D}_y$, we extract the triplet $\left(\left(\boldsymbol{x}_i^s, y_a^j\right), \left(\boldsymbol{x}_j^s, y_a^j\right), \left(\boldsymbol{x}_k^s, y_b^j\right)\right) \in \mathcal{D}_s$ to meet the requirement of $E_G > 0$ in Eq. (1), and this triplet is included in $\mathcal{X}_S^{batch}$, i.e., the source sample set in this batch. Note that if no samples can meet this requirement, samples are randomly selected from $\mathcal{D}_s$ to form $\mathcal{X}_S^{batch}$.

In each batch during the training, the critical samples and the easily-confused classes are identified via the target samples, and the margin is calculated to measure the confusion degree of classes in the target domain. Then in the same batch, source samples are selected to ensure that the triplet constraint in Eq. (1) is violated. Therefore, in each batch, there are source samples that can contribute to the triplet loss, which can accelerate the metric learning. As a result, easily-confused classes in the source domain are effectively separated by a large margin. On this basis, after the alignment operation, the easily-confused classes in the target domain can also be separated indirectly.

### 3.4 Metric learning for domain adversarial network

Fig. 2 shows the framework of our proposed metric learning for domain adversarial network (ML-DAN), which mainly consists of five parts, including feature extractor $F$, multi-classifier $C_m$, multi-binary classifiers $C_b^c|_{c=1}^k$, metric generator $G$, and domain classifier $D$.

First, in each batch, the target sample set $\mathcal{X}_T^{batch}$, randomly extracted from the target domain, is inputted into the feature extractor $F$ and the multi-classifier $C_m$ to minimize the entropy loss as follows,

$$E_e = -\frac{1}{\left|\mathcal{X}_T^{batch}\right|} \sum_{\boldsymbol{x} \in \mathcal{X}_T^{batch}} H\left(C_m\left(F\left(\boldsymbol{x}\right)\right)\right), \tag{7}$$

where $H(\cdot)$ is the entropy function. Minimizing the entropy is to reduce the uncertainty of the target domain and force the decision boundary to pass through the low-density area for better clustering effect.

**Algorithm 1** Metric Learning for Domain Adversarial Network

**Input:**

labeled source samples: $\mathcal{D}_s = \{(\boldsymbol{x}_i^s, y_i^s)\}_{i=1}^{n_s}$;
unlabeled source samples: $\mathcal{D}_t = \{(\boldsymbol{x}_i^t)\}_{i=1}^{n_t}$;
$F : \mathcal{X} \to \mathbb{R}^n$: feature extractor parameterized by $\theta_f$;
$C_m : \mathbb{R}^n \to \mathbb{R}^k$: multi-classifier parameterized by $\theta_m$;
$C_b^c|_{c=1}^k : \mathbb{R}^n \to \mathbb{R}^k$: multi-binary classifiers parameterized by $\theta_b$;
$D : \mathbb{R}^n \to \{0,1\}$: domain classifier parameterized by $\theta_d$;
$G : \mathbb{R}^n \to \mathbb{R}^m$: metric generator parameterized by $\theta_g$;
$T$ : number of iterations;

load parameters pre-trained from the ImageNet to $\theta_f$;
set $t = 0$;
**while** $t < T$ **do**
  **for** each batch $\mathcal{X}_T^{batch} \subset \mathcal{D}_t$ **do**
    calculate $E_e$ for $\mathcal{X}_T^{batch}$ by Eq. (7);
    find $l$ pairs of easily-confused classes $\mathcal{D}_y = \left\{\left(y_a^j, y_b^j\right)\right\}_{j=1}^{l}$ by Eq. (4);
    **for** $j = 1$; $j \le l$; $j{+}{+}$ **do**
      $\boldsymbol{x}_i^s$ and $\boldsymbol{x}_j^s$ are selected from class $y_a^j$ in $\mathcal{D}_s$;
      $\boldsymbol{x}_k^s$ is selected from class $y_b^j$ in $\mathcal{D}_s$;
      if the triplet $\left(\left(\boldsymbol{x}_i^s, y_a^j\right), \left(\boldsymbol{x}_j^s, y_a^j\right), \left(\boldsymbol{x}_k^s, y_b^j\right)\right) \in \mathcal{D}_s$ meets the requirement $E_G > 0$ in Eq. (1), then $\left\{\boldsymbol{x}_i^s, \boldsymbol{x}_j^s, \boldsymbol{x}_k^s\right\} \subset \mathcal{X}_S^{batch}$;
    **end for**
    randomly select samples from $\mathcal{D}_s$ to form the remaining samples in $\mathcal{X}_S^{batch}$;
    calculate $E_m^s$ for $\mathcal{X}_S^{batch}$ by Eq. (8);
    calculate $E_b^s$ for $\mathcal{X}_S^{batch}$ by Eq. (9);
    calculate $E_D$ for $\left(\mathcal{X}_S^{batch}, \mathcal{X}_T^{batch}\right)$ by Eq. (10);
    update $\theta_f, \theta_m, \theta_b, \theta_d$ and $\theta_g$ by optimizing the Eq. (11) using stochastic gradient descent;
  **end for**
  $t = t + 1$;
**end while**

**Output:** model parameters $\theta_f, \theta_m, \theta_b, \theta_d$ and $\theta_g$.

Second, in the same batch, the source sample set $\mathcal{X}_S^{batch}$ is inputted into the feature extractor $F$ , and extracted features will feed into the multi-classifier $C_m$ and the multi-binary classifiers $C_b^c|_{c=1}^k$, respectively. The loss for these two classifiers can be defined as

$$E_m^s = \frac{1}{\left|\mathcal{X}_S^{batch}\right|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_S^{batch}} L_y\left(C_m\left(F\left(\boldsymbol{x}_i\right)\right), y_i\right), \tag{8}$$

$$E_b^s = \sum_{c=1}^{k} \frac{1}{\left|\mathcal{X}_S^{batch}\right|} \sum_{\boldsymbol{x}_i \in \mathcal{X}_S^{batch}} L_b\left(C_b^c\left(F\left(\boldsymbol{x}_i\right)\right), I\left(y_i, c\right)\right), \tag{9}$$

where $y_i$ is the true label of the source sample $\boldsymbol{x}_i$, $L_y$ and $L_b$ are the cross-entropy loss and binary cross-entropy loss, respectively. $I(y_i, c)$=1 if $y_i$=$c$; otherwise, $I(y_i, c)$=0.

Finally, the extracted features of $\mathcal{X}_T^{batch}$ and $\mathcal{X}_S^{batch}$ are forwarded into the domain classifier $D$ alternately. the output of $D : \mathbb{R}^n \to \{0,1\}$ discriminates between the source features and the target features, while the feature extractor $F$ tries to fool $D$. Through the above adversarial training, the distribution between the source domain and the target domain is aligned for generating the domain-invariant features. The loss for adversarial domain discriminator is defined as

$$\begin{aligned} E_D = &- \frac{1}{\left|\mathcal{X}_S^{batch}\right|} \sum_{x_i \in \mathcal{X}_S^{batch}} \log D\left(F\left(x_i\right)\right) - \\ &\frac{1}{\left|\mathcal{X}_T^{batch}\right|} \sum_{x_j \in \mathcal{X}_T^{batch}} \log\left(1 - D\left(F\left(x_j\right)\right)\right). \end{aligned} \tag{10}$$

The whole training can be expressed as

$$\max_{D} \min_{F, C_m, C_b^c|_{c=1}^k} E_m^s + E_b^s + \zeta E_G + \eta E_e - E_D, \tag{11}$$

where $\zeta$ and $\eta$ are hyper-parameters to trade off the loss. In summary, the detailed procedure of the ML-DAN is listed in the algorithm description.

## 4 Experiments

### 4.1 Datasets and implementation details

**Office-31** is the most commonly-used benchmark dataset in the field of the domain adaptation. The dataset consists of three domains: Amazon (A), Webcam (W) and Dslr (D), and each domain has 31 classes. There are 4110 images in total and the images in Amazon are from the amazon.com, and the images in Dslr and Webcam are taken by digital SLR camera and web camera, respectively. To enable an unbiased evaluation, we will evaluate all the methods on 6 transfer tasks: A→W, D→W, A→D, W→D, D→A and W→A.

**Office-Home** dataset consists of four domains: Art (AR), Clipart (CI), Product (PR), and Real-World (RW). The dataset contains 15,500 images, and each domain is composed of the images from 65 classes. A total of 12 transfer tasks are evaluated on these four domains.

**ImageCLEF-DA** is a benchmark dataset for the task of the image annotation in the domain adaptation. The dataset contains 3 domains: Caltrch-256 (C), ImageNet ILSVRC 2012 (I), and Pascal VOC 2012 (P). 12 common categories are shared by the above 3 domains, and 600 images compose each domain. We evaluate all the methods on 6 transfer tasks: I→P, P→I, I→C, C→I, C→P and P→C.

**Visda2017** is a fairly large dataset in the field of visual domain adaptation, which provides two different domains: Synthetic, 3D renderings synthesized from different angles and different lighting conditions, and Real, real world images collected from MSCOCO. There are 280,000 images in this dataset, and 12 categories in each domain.

All experiments are implemented using the Pytorch framework. We compare ML-DAN with the most representative methods for the unsupervised domain adaptation. These methods include Deep Adaptation Network (DAN) [14], Reverse Gradient (RevGrad) [31], Domain Adversarial Neural Network (DANN) [18], Joint Adaptation Network (JAN) [16], Adversarial Discriminative Domain Adaptation (ADDA) [11], Multi-Adversarial Domain Adaptation (MADA) [19], Maximum Classifier Discrepancy (MCD) [21] and Conditional Domain Adversarial Network (CDAN) [20]. The performances of the above methods are evaluated in the same environment for fairness. For each experiment, we take the average of the results over the 3 repetitions.

For ML-DAN, the feature extractor uses ResNet50 pretrained on ImageNet [31], whose output is 1000-dimensional feature vectors. The two-layer neural networks are designed for multi-binary classifiers, multi-classifiers and metric generator. We use the mini-batch SGD with the momentum 0.9. In all experiments, batch size is set to 32, and the learning rate is set to 0.01. Moreover, we set $\eta$ in Eq. (11) to 0.1 and the initial value $\alpha_0$ of the margin to 5.

### 4.2 Parameters Sensitivity

In this experiment, there exist $l$ critical samples in the target batch according to the definition of the easily-confused class set $\mathcal{D}_y = \left\{ \left( y_a^j, y_b^j \right) \right\}_{j=1}^{l}$. we first discuss the impact of $l$ on the tasks: D→A and W→A in the Office-31 dataset, and the experimental results are shown in Table 1. We can observe that in both tasks, the accuracy of ML-DAN fluctuates slightly with the different value of $l$, and when $l$ is set to 2, ML-DAN can achieve best average accuracy. Therefore, in the following experiments, we set $l$ to 2.

Then, we discuss the impact of $\zeta$, the weight coefficient of the triplet loss, in Eq. (11) on the tasks: D→A and W→A in the Office-31 dataset. The experimental results in Table 2 show that the accuracy does not vary significantly, and when the weight coefficient $\zeta$ is 0.08, the best average accuracy can be achieved. Therefore, in the following experiments, we set $\zeta$ to 0.08.

In addition, the introduction of the margin in Eq. (6) is an important part of the proposed ML-DAN. Different from the existing methods using the fixed value as the margin, ML-DAN dynamically adjusts the margin between the positive pairs and negative pairs of the triplet loss in the source domain according to the boundary threshold. That is the larger the boundary threshold is, the more difficult it is to separate corresponding target classes in the feature space. Hence, the larger margin in the triplet loss function is introduced to increase the separation distance of corresponding source classes in the metric space. In D→A and W→A tasks, the final experimental results in Table 3 also show that the dynamic adjusting margin can achieve the best average accuracy, which verifies the effectiveness of the proposed ML-DAN.

Table 1. Accuracy (%) of D→A and W→A tasks with different $l$.

| $l$ | D→A | W→A | Avg |
|---|---|---|---|
| 1 | 71.35 | 71.12 | 71.24 |
| 2 | **72.18** | 71.19 | **71.69** |
| 3 | 71.81 | 71 | 71.41 |
| 4 | 71.72 | **71.26** | 71.5 |
| 5 | 70.94 | 71.07 | 71.01 |

Table 2. Accuracy (%) of D→A and W→A tasks with different weight $\zeta$.

| $\zeta$ | D→A | W→A | Avg |
|---|---|---|---|
| 0.01 | 69.98 | 69.15 | 69.57 |
| 0.05 | 70.79 | 69.77 | 70.28 |
| 0.08 | **72.18** | **71.19** | **71.69** |
| 0.1 | 72.03 | 69.96 | 71 |
| 0.2 | 70.95 | 71 | 70.98 |
| 0.5 | 70.49 | 69.74 | 70.12 |

Table 3. Accuracy (%) of D→A and W→A tasks with different weight margin.

| margin | D→A | W→A | Avg |
|---|---|---|---|
| 0.1 | 71.36 | 70.78 | 71.07 |
| 0.2 | 70.75 | 71.18 | 70.97 |
| 0.4 | 69.96 | 70.81 | 70.39 |
| 0.6 | 71.15 | 70.77 | 70.96 |
| 0.8 | 70.88 | **71.23** | 71.01 |
| 1 | 71.21 | 69.95 | 70.58 |
| dynamic adjusting margin | **72.18** | 71.19 | **71.69** |

### 4.3 Performance Comparison

Table 4 - 7 show the classification accuracy of ML-DAN and other comparable methods on Office-31, Office-home, ImageCLEF-DA and Visda2017, respectively. The results show that ML-DAN outperforms the other methods by a large margin on these four datasets, and except that CDAN achieves higher accuracy on the I→C task of ImageCLEF-DA, ML-DAN yields significant improvements in both the specific task accuracy and the average accuracy under the scenario of unsupervised domain adaption.

From Table 4 to Table 7, the advantage of ML-DAN lies in the following: the effective way to identify the easily-confused class set in the target domain, the introduction of triplet loss based on the dynamic adjusting margin, and an efficient training batch mechanism for accelerating the metric training process. Finally, the combination of the domain adversarial training and metric learning can enhance the performance of the proposed ML-DAN. The above experiments also verify the superiority and generalization of the ML-DAN method.

Figure 3 shows the negative transfer of DANN, CDAN, and ML-DAN when compared with ResNet, and the experiments are conducted on the D→A task of Office-31. DANN and CDAN suffer from the negative transfer in most classes. Meanwhile, except for ML-DAN has the negative gain in the first class, ML-DAN obtains the positive gain in other classes. Therefore, under the conditions of the domain shift, ML-DAN can improve the classification accuracy on most transfer tasks.

### 4.4 Ablation Experiments

In this subsection, the different loss function combinations will be evaluated in term of the classification accuracy of ML-DAN on the office31 dataset. The experimental results are shown in Table 8.

(1) $E_m^s$ vs. $E_m^s - E_D$ vs. $E_m^s + \zeta E_G$. Compared with $E_m^s$, the target accuracy improved by $E_m^s - E_D$ on small-to-large transfer tasks is not as high as that by using $E_m^s + \zeta E_G$. This is due to the fact that insufficient samples in the source domain result in less original joint distribution information. Though the domain alignment is performed between the source and target domain, the better clustering effect of the target samples is hard to be achieved. While the triplet loss in the metric space can effectively separate the target classes to further enhance the target accuracy.

(2) $E_m^s + \zeta E_G - E_D$ vs. $E_m^s + \zeta E_G$ vs. $E_m^s - E_D$. Compared with $E_m^s + \zeta E_G$ and $E_m^s - E_D$, $E_m^s + \zeta E_G - E_D$ corresponds to the combination of the triplet loss and domain alignment loss, the performance of which is relatively better than that of the above two terms. This further validates that the domain alignment combined with the dynamical margin in the triplet loss can more effectively separate the easily-confused classes in the target domain.

(3) $E_m^s + \eta E_e - E_D$ vs. $E_m^s + \zeta E_G + \eta E_e - E_D$ vs. $E_m^s + E_b^s + \zeta E_G + \eta E_e - E_D$. The use of target entropy loss is to reduce the uncertainty of the target domain for better clustering effect, and the multi-binary classifiers are designed to calculate the classification gap for identifying the critical samples and the corresponding easily-confused classes. The combination of all these losses can guarantee the superior performance of the proposed ML-DAN method.

From the ablation experiments, we can also obtain the conclusion that the domain adversarial training plus triplet loss are the major factor improving the performance for relatively smaller source domain samples. It is due to the fact that samller source samples can provide limited supervised information, and the scattering of target domain will become more significant. Therefore, the domain adversarial training plus triplet loss will play a critical role in the improvement in classification performance.

### 4.5 Performance Demonstration

In this subsection, we will validate the effectiveness of ML-DAN by comparing the cosine distances among the positive pairs and negative pairs in the target domain. Specifically, each positive and negative pair are composed of an anchor, a positive sample and a negative sample from the easily-confused classes during the training procedure. As we know, cosine distance is a measure of difference between two vectors in the feature space, and a larger cosine distance means the further separation of two features. From Table 9, in DA, the negative pair distance, i.e. the distance between the anchor and the nearest negative sample, is less than or comparable to the positive pair distance, i.e. the distance between the anchor and the farthest positive sample. This is why the DA method suffers from the easily-confused classes in the target domain. In contrast, the proposed ML-DAN can improve the classification performance by identifying and separating the easily-confused classes. In Table 9, ML-DAN effectively increases the negative pair distances while reducing the positive pair distances, as a result, the positive pair distances are always significantly less than the negative pair distance. Hence, the content of Table 9 verifies the advantage of ML-DAN again.

Table 4. Accuracy (%) on Office-31 for unsupervised domain adaptation (ResNet-50).

| Method | A→W | D→W | W→D | A→D | D→A | W→A | Avg |
|---|---|---|---|---|---|---|---|
| Resnet50 | 68.4 | 96.7 | 99.3 | 68.9 | 62.5 | 62.7 | 76.4 |
| DAN | 80.5 | 97.1 | 99.6 | 78.6 | 63.6 | 62.8 | 80.4 |
| DANN | 82 | 96.9 | 99.1 | 79.7 | 68.2 | 67.4 | 82.2 |
| ADDA | 86.2 | 96.2 | 98.4 | 77.8 | 69.5 | 68.9 | 82.8 |
| JAN | 85.4 | 97.4 | 99.8 | 84.7 | 68.6 | 70 | 84.3 |
| MADA | 90 | 97.4 | 99.6 | 87.8 | 70.3 | 66.4 | 85.2 |
| MCD | 89.6 | 98.5 | **100** | 91.3 | 69.6 | 70.8 | 86.6 |
| CDAN | 93.1 | 98.2 | **100** | 89.8 | 70.1 | 68 | 86.6 |
| **ML-DAN** | **93.64** | **99.05** | **100** | **94.56** | **72.18** | **71.19** | **88.44** |

Table 5. Accuracy (%) on Office-Home 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Resnet50 | 42.5 | 50 | 58 | 37.4 | 41.9 | 46.2 | 38.5 | 42.4 | 60.4 | 53.9 | 41.2 | 59.9 | 47.7 |
| DAN | 43.6 | 57 | 67.9 | 45.8 | 56.5 | 60.4 | 44 | 43.6 | 67.7 | 63.1 | 51.5 | 74.3 | 56.3 |
| DANN | 45.6 | 59.3 | 70.1 | 47 | 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 | 45.8 | 43.4 | 70.3 | 63.9 | 52.4 | 76.8 | 58.3 |
| CDAN | 49 | 69.3 | 74.5 | 54.4 | 66 | 68.4 | 55.6 | 48.3 | 75.9 | 68.4 | 55.4 | 80.5 | 63.8 |
| **ML-DAN** | **56.41** | **73.17** | **76.71** | **61.23** | **70.27** | **71.58** | **60.55** | **53.54** | **78.59** | **69.63** | **61.4** | **83.08** | **68.01** |

Table 6. Accuracy (%) on ImageCLEF-DA for unsupervised domain adaptation (ResNet-50).

| Method | I→P | P→I | I→C | C→I | C→P | P→C | Avg |
|---|---|---|---|---|---|---|---|
| Resnet50 | 74.8 | 83.9 | 91.5 | 78 | 65.5 | 91.2 | 80.8 |
| DAN | 74.5 | 82.2 | 92.8 | 86.3 | 69.2 | 89.8 | 82.5 |
| DANN | 75 | 86 | 96.2 | 87 | 74.3 | 91.5 | 85 |
| JAN | 76.8 | 88 | 94.7 | 89.5 | 74.2 | 91.7 | 85.8 |
| MADA | 75 | 87.9 | 96 | 88.8 | 75.2 | 92.2 | 85.9 |
| CDAN | 76.7 | 90.6 | **97** | 90.5 | 74.5 | 93.5 | 87.1 |
| **ML-DAN** | **79.17** | **92** | 96 | **91.33** | **76.67** | **94.33** | **88.25** |

### 4.6 Robustness Analysis

The effective way to evaluate the robustness of various methods is to measure the anti-noise ability. According to [32],virtual adversarial training (VAT) is employed to calculate the gradient of loss function to generate the noisy target input samples, which can test the classification performance of the corresponding methods. Furthermore, the scaling weight $I_n$ is used to control the degree of noise, and higher $I_n$ means more noisy input data. From Table 10, we can observe that all methods degrade the performance with the increase of $I_n$. However, ML-DAN still outperforms the other methods in all cases, and has smaller drop in performance.

## 5 Conclusion

In this paper, we proposed a Metric Learning for Domain Adversarial Network (ML-DAN), which can outperform the existing methods for unsupervised domain adaptation by improving the classification accuracy of easily-

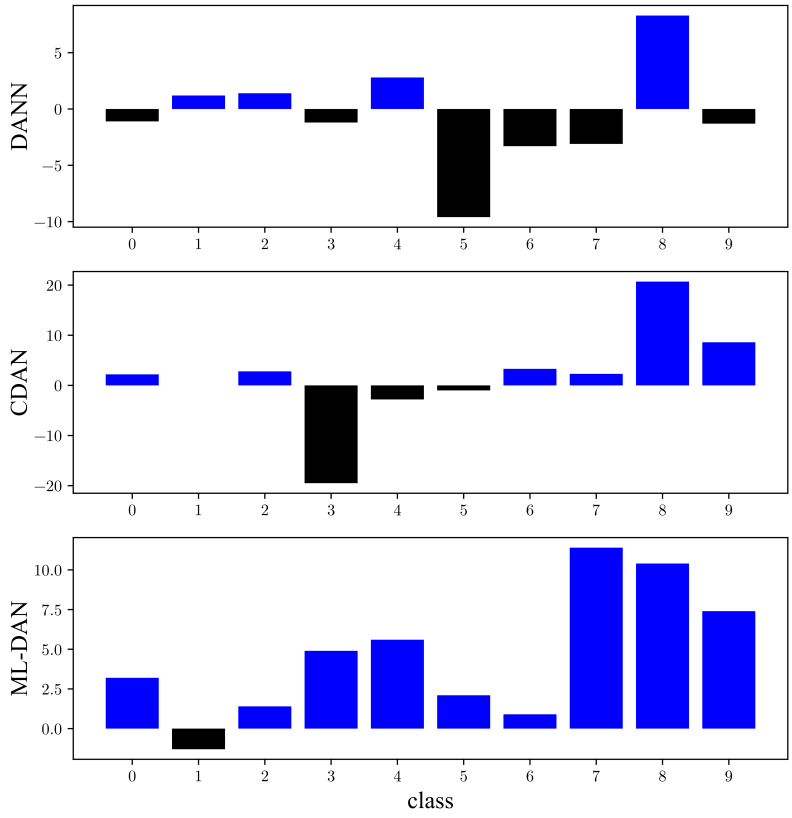


Fig. 3. The negative transfer influence on the D→A task of Office-31.

Table 7. Accuracy (%) on Visda2017 for unsupervised domain adaptation (ResNet-50).

| Method | Synthetic→Real |
|---|---|
| Resnet50 | 52.4 |
| RevGrad | 57.4 |
| DAN | 61.1 |
| MCD | 71.9 |
| CDAN | 73.7 |
| **ML-DAN** | **75.27** |

confused classes of the target domain. Specifically, ML-DAN can provide the performance guarantee by identifying easily-confused classes, dynamically adjusting the margin in the triplet loss, and designing the training batch mechanism. Finally, the domain alignment between the source and target domain is performed via domain adversarial training to indirectly separate the easily-confused classes. Extensive experimental results demonstrate that ML-DAN can provide a significant improvement in classification performance in the scenario of UDA when compared with the existing methods.

## References


[1] Lecun, Y., Bengio, Y., and Hinton, G., 2015. "Deep learning". *Nature,* **521**(7553), p. 436.

[2] Xu, C., Wang, K., Sun, Y., Guo, S., and Zomaya, A., 2020. "Redundancy avoidance for big data in data centers: a conventional neural network approach". *IEEE Transactions on Network Science and Engineering,* **7**(1), pp. 104–114.

[3] Lu, H., He, X., Du, M., Ruan, X., Sun, Y., and Wang, K., 2020. "Edge qoe: computation offloading with deep reinforcement learning for the internet of things". *IEEE Internet of Things Journal,* **7**(10), pp. 9255–9265.

[4] He, X., Wang, K., Huang, H., Wang, Y., and Guo, S., 2020. "Green resource allocation based on deep reinforcement learning in content-centric IoT". *IEEE Transactions on Emerging Topics in Computing,* **8**(3), pp. 781–796.

[5] Madadi, Y., Seydi, V., Nasrollahi, K., Hosseini, R., and Moeslund, T. B., 2020. "Deep visual unsupervised do-

Table 8. Ablation experiments on Office-31 for unsupervised domain adaptation (ResNet-50).

| Loss function combinations | A→W | D→W | W→D | A→D | D→A | W→A | Avg |
|---|---|---|---|---|---|---|---|
| $E_m^s$ | 79.1 | 96 | 99.3 | 73.7 | 62.1 | 60.2 | 78.4 |
| $E_m^s - E_D$ | 86.8 | 98.3 | **100** | 82.1 | 63 | 61.9 | 82 |
| $E_m^s + \zeta E_G$ | 82 | 98.3 | 99.9 | 84.7 | 67.1 | 65.3 | 82.9 |
| $E_m^s + \zeta E_G - E_D$ | 88.6 | 96.2 | **100** | 88.1 | 69.5 | 68.9 | 85.2 |
| $E_m^s + \eta E_e - E_D$ | 91.2 | 98.2 | 99.8 | 92.5 | 69.6 | 69 | 86.7 |
| $E_m^s + \zeta E_G + \eta E_e - E_D$ | 92.7 | 98.8 | **100** | 94 | 71.3 | 70.6 | 87.9 |
| $E_m^s + E_b^s + \zeta E_G + \eta E_e - E_D$ | **93.64** | **99.05** | **100** | **94.56** | **72.18** | **71.19** | **88.44** |

Table 9. Comparison of cosine distance between easily-confused classes on four datasets.

| Dataset | Method | Anchor | The farthest positive sample | The nearest negative sample |
|---|---|---|---|---|
| Office31 D→A | DA |  Label: desktop computer |  Label: desktop computer Cosine distance:1.007 |  Label: monitor Cosine distance:0.614 |
| | ML-DAN |  Label: laptop computer |  Label: laptop computer Cosine distance:0.473 |  Label: phone Cosine distance:0.824 |
| OfficeHome (Ar→Cl) | DA |  Label: radio |  Label: radio Cosine distance:0.758 |  Label: oven Cosine distance:0.957 |
| | ML-DAN |  Label: calculator |  Label: calculator Cosine distance:0.309 |  Label: keyboard Cosine distance:0.847 |
| ImageCLEF (P→I) | DA |  Label: bike |  Label: bike Cosine distance:0.734 |  Label: dog Cosine distance:0.955 |
| | ML-DAN |  Label: bird |  Label: bird Cosine distance:0.635 |  Label: bird Cosine distance:0.913 |
| VisDA-2017 (Syn→Real) | DA |  Label: train |  Label: train Cosine distance:0.961 |  Label: truck Cosine distance:0.830 |
| | ML-DAN |  Label: person |  Label: person Cosine distance:0.451 |  Label: person Cosine distance:0.871 |

Table 10. Accuracy (%) on ResNet-50 with noisy data.

| Method | $I_n$ | D→A | W→A | Ar→Cl | Pr→Cl | Avg |
|---|---|---|---|---|---|---|
| Resnet50 | $I_n$=0 | 62.5 | 62.7 | 42.5 | 42.4 | 52.5 |
| | $I_n$=3.5 | 38.9 | 41.3 | 31.1 | 29.2 | 35.1 |
| | $I_n$=5.0 | 37.0 | 38.7 | 31.8 | 28.2 | 33.9 |
| DA | $I_n$=0 | 58.6 | 62.1 | 45.6 | 43.7 | 56.2 |
| | $I_n$=3.5 | 40.5 | 42.9 | 32.0 | 33.0 | 37.1 |
| | $I_n$=5.0 | 36.5 | 39.4 | 30.3 | 31.6 | 35.2 |
| CDAN | $I_n$=0 | 70.1 | 68.0 | 49.0 | 48.3 | 58.9 |
| | $I_n$=3.5 | 33.7 | 34.4 | 23.0 | 21.8 | 28.2 |
| | $I_n$=5.0 | 33.8 | 22.8 | 22.3 | 17.8 | 24.2 |
| ML-DAN | $I_n$=0 | **72.2** | **71.2** | **56.4** | **53.5** | **63.3** |
| | $I_n$=3.5 | **59.7** | **57.3** | **47.5** | **45.9** | **52.6** |
| | $I_n$=5.0 | **55.9** | **53.6** | **47.3** | **43.4** | **50.1** |


main adaptation for classification tasks: a survey". *IET Image Processing*.

[6] You, K., Long, M., Cao, Z., Wang, J., and Jordan, M. I., 2019. "Universal domain adaptation". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2720–2729.

[7] Zhuang, F., Qi, Z., Duan, K., andYongchun Zhu, D. X., Zhu, H., Xiong, H., and He, Q., 2020. "A comprehensive survey on transfer learning". *Proceedings of the IEEE,* **109**(1), pp. 43–76.

[8] Venkateswara, H., Chakraborty, S., and Panchanathan, S., 2017. "Deep-learning systems for domain adaptation in computer vision: Learning transferable feature representations". *IEEE Signal Processing Magazine,* **34**(6), pp. 117–129.

[9] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W., 2010. "A theory of learning from different domains". *Machine learning,* **79**(1), pp. 151–175.

[10] Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., and Marchand, M., 2014. "Domain-adversarial neural networks". *arXiv preprint arXiv:1412.4446*.

[11] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T.,

2017. “Adversarial discriminative domain adaptation”. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7167–7176.

[12] Long, M., Zhu, H., Wang, J., and Jordan, M. I., 2016. “Unsupervised domain adaptation with residual transfer networks”. *arXiv preprint arXiv:1602.04433*.

[13] Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N., 2008. Covariate shift and local learning by distribution matching. MIT Press.

[14] Long, M., Cao, Y., Wang, J., and Jordan, M., 2015. “Learning transferable features with deep adaptation networks”. In International conference on machine learning, PMLR, pp. 97–105.

[15] Long, M., Wang, J., Ding, G., Sun, J., and Yu, P. S., 2013. “Transfer feature learning with joint distribution adaptation”. In Proceedings of the IEEE international conference on computer vision, pp. 2200–2207.

[16] Long, M., Zhu, H., Wang, J., and Jordan, M. I., 2017. “Deep transfer learning with joint adaptation networks”. In International conference on machine learning, PMLR, pp. 2208–2217.

[17] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., 2014. “Generative adversarial networks”. *arXiv preprint arXiv:1406.2661*.

[18] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V., 2016. “Domain-adversarial training of neural networks”. *Journal of machine learning research,* **17**(1), pp. 2096–2030.

[19] Pei, Z., Cao, Z., Long, M., and Wang, J., 2018. “Multi-adversarial domain adaptation”. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. **32** of *1*.

[20] Long, M., Cao, Z., Wang, J., and Jordan, M. I., 2017. “Conditional adversarial domain adaptation”. *arXiv preprint arXiv:1705.10667*.

[21] Saito, K., Watanabe, K., Ushiku, Y., and Harada, T., 2018. “Maximum classifier discrepancy for unsupervised domain adaptation”. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3723–3732.

[22] Yang, L., and Jin, R., 2006. “Distance metric learning: A comprehensive survey”. *Michigan State Universiy,* **2**(2), p. 4.

[23] Xing, E. P., Ng, A. Y., Jordan, M. I., and Russell, S., 2002. “Distance metric learning with application to clustering with side-information”. In Proceedings of the 2003 Advances in Neural Information Processing Systems, pp. 521–528.

[24] Weinberger, K. Q., and Saul, L. K., 2009. “Distance metric learning for large margin nearest neighbor classification”. *Journal of Machine Learning Research,* **10**(1), pp. 207–244.

[25] Davis, J. V., Kulis, B., Jain, P., Sra, S., and Dhillon, I. S., 2007. “Information-theoretic metric learning”. In Proceedings of the 24th international conference on Machine learning, pp. 209–216.

[26] Hadsell, R., Chopra, S., and Lecun, Y., 2006. “Dimensionality reduction by learning an invariant mapping”. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06).

[27] Song, H. O., Yu, X., Jegelka, S., and Savarese, S., 2016. “Deep metric learning via lifted structured feature embedding”. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[28] Schroff, F., Kalenichenko, D., and Philbin, J., 2015. “Facenet: A unified embedding for face recognition and clustering”. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 815–823.

[29] He, X., Zhou, Y., Zhou, Z., Bai, S., and Bai, X., 2018. “Triplet-center loss for multi-view 3d object retrieval”. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1945–1954.

[30] Chen, W., Chen, X., Zhang, J., and Huang, K., 2017. “Beyond triplet loss: a deep quadruplet network for person re-identification”. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 403–412.

[31] Ganin, Y., and Lempitsky, V., 2015. “Unsupervised domain adaptation by backpropagation”. In International conference on machine learning, PMLR, pp. 1180–1189.

[32] Miyato, T., Maeda, M., Koyama, M., and Ishii, S., 2019. “Virtual adversarial training: A regularization method for supervised and semi-supervised learning”. *IEEE Trans. Pattern Anal. Mach. Intell.,* **41**(8), pp. 1979–1993.