Mind the Class Weight Bias: Weighted Maximum Mean Discrepancy for Unsupervised Domain Adaptation

Hongliang Yan[1], Yukang Ding[1], Peihua Li[2], Qilong Wang[2], Yong Xu[3], Wangmeng Zuo[1,*]
[1]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China
[2]School of Information and Communication Engineering, Dalian University of Technology, Dalian, China
[3]Bio-Computing Research Center, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen, China
yanhl@hit.edu.cn, dingyukang921@163.com, peihuali@dlut.edu.cn,
qlwang@mail.dlut.edu.cn, laterfall@hitsz.edu.cn, wmzuo@hit.edu.cn

Abstract

In domain adaptation, maximum mean discrepancy (MMD) has been widely adopted as a discrepancy metric between the distributions of source and target domains. However, existing MMD-based domain adaptation methods generally ignore the changes of class prior distributions, i.e., class weight bias across domains. This remains an open problem but ubiquitous for domain adaptation, which can be caused by changes in sample selection criteria and application scenarios. We show that MMD cannot account for class weight bias and results in degraded domain adaptation performance. To address this issue, a weighted MMD model is proposed in this paper. Specifically, we introduce class-specific auxiliary weights into the original MMD for exploiting the class prior probability on source and target domains, whose challenge lies in the fact that the class label in target domain is unavailable. To account for it, our proposed weighted MMD model is defined by introducing an auxiliary weight for each class in the source domain, and a classification EM algorithm is suggested by alternating between assigning the pseudo-labels, estimating auxiliary weights and updating model parameters. Extensive experiments demonstrate the superiority of our weighted MMD over conventional MMD for domain adaptation.

1. Introduction

Deep convolutional neural networks (CNNs) have achieved great success in various computer vision tasks such as image classification [21], object detection [12] and semantic segmentation [24]. Besides the inspiring progress in model and learning, the achievement of CNN is undoubtedly attributed to the availability of massive labeled datasets. For a CNN trained on large scale datasets [8], while the lower layers of features are safely transferable, the learned features gradually moves from general to specific along the network [40]. When the source and target tasks are significantly diverse, the CNN pretrained on the source task may not generalize well to the target task. Such scenario leads to an emerging topic to transfer the CNN from the source task to the target task with the enhanced and discriminative representation [2]. In this work, we study a special type of transfer learning task, *i.e.*, domain adaptation (DA) [30].

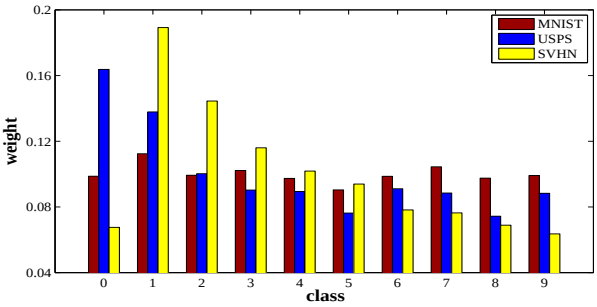

Figure 1. Class prior distributions of three domains for digit recognition. As is shown, class bias exists across domains. It is natural to see that the class weight of 0 and 1 are relatively high in postal service (*USPS*), and the class weight of 1 and 2 are relatively high in house numbers (*SVHN*).

One of the most fruitful lines for DA is MMD-based method [26, 29, 3, 41, 36]. Despite the great success achieved, existing ones generally ignore the changes of class prior distributions, dubbed by class weight bias. It is ubiquitous for domain adaptation and can be caused by changes in sample selection criteria [19] and application scenarios [28]. As shown in Fig. 1, the class prior distributions (*i.e.*, class weights) vary with domains for digit recognition. Moreover, a special case of class weight bias

*Corresponding author.

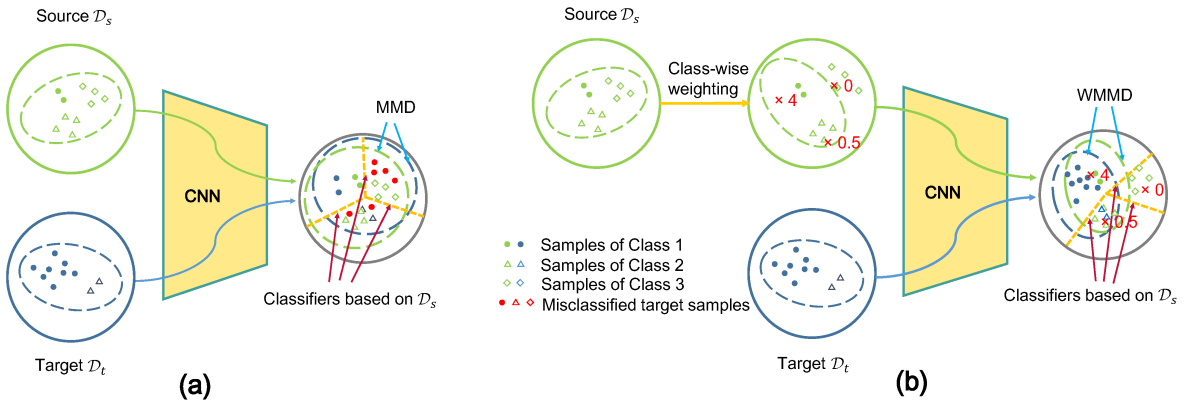

Figure 2. Results of minimizing MMD and WMMD regularizer under class weight bias are illustrated in (a) and (b), respectively. Minimizing MMD preserves the class weights in source domain and thus the target samples will be wrongly estimated, as indicated by yellow samples. On the contrary, the proposed weighted MMD removes the effect of class bias by first reweighting source data.

is the imbalanced cross-domain data problem [28] where several classes in source domain do not appear in target domain, as shown in Fig. 2. The Closest Common Space Learning (CCSL) method in [28] is suggested for imbalanced and multiple cross-domain visual classification. However, CCSL just combines conventional MMD with domain-dependent MMD without explicitly considering class weight bias.

For MMD-based methods, the ignorance of class weight bias can deteriorate the domain adaptation performance. In the case of class weight bias, the MMD can be minimized by either learning domain-invariant representation or preserving the class weights in source domain. As illustrated in Fig. 2 (a), it is unreasonable for domain adaptation to require that the class weights in target domain should keep the same as those in source domain. Our empirical experiments also reveal the limitation of MMD in coping with class weight bias (See Fig. 4).

In this paper, we propose a weighted MMD (WMMD) method to address the issue of class weight bias. As for DA, the challenge is that the class labels in target domain are unknown. So we first introduce class-specific auxiliary weights to reweight the source samples. In this way, the reweighted source data are expected to share the same class weights with target data. The auxiliary weights estimation and model parameters learning are jointly optimized by minimizing the objective function of weighted MMD. Different from MMD, the objective function based on our weighted MMD involves additional weight parameters, and we present a classification EM (CEM) scheme to estimate it. Inspired by the semi-supervised logistic regression in [1], we propose a weighted Domain Adapation Network (WDAN) by both incorporating the weighted MMD into CNN and taking into account the empirical loss on target samples. The CEM algorithm are developed for learning WDAN in three steps, *i.e.*, E-step, C-step, and M-step. In the E-step and the C-step, we calculate the class posterior probability, assign the pseudo-labels to the target samples, and estimate the auxiliary weight. In the M-step, model parameters are updated by minimizing the objective loss. Experimental results show our weighted MMD can learn better domain-invariant representation for domain adaptation. Moreover, the models based on weighted MMD also outperforms the MMD-based counterparts. In summary, the main contributions of this work are three-fold:

1. A weighted MMD model is proposed to alleviate the effect of class weight bias in domain adaptation. By taking class prior distributions into account, weighted MMD can provide a better metric for domain discrepancy.

2. Using unbiased estimate of multi-kernel MMD [15, 17], our proposed weighted MMD can be computed as mean embedding matching with linear time complexity and be incorporated into CNN for unsupervised domain adaptation. We further develop a CEM algorithm for training the weighted MMD model.

3. Experiments demonstrate that weighted MMD outperforms MMD for domain adaptation. The superiority of weighted MMD over MMD has been verified on various CNN architectures and different datasets.

In the remainder of this paper, we begin with a brief introduction to the preliminaries and related work in Section 2. In Section 3, by considering class weight bias, we propose weighted MMD on the basis of conventional

MMD. After that, in Section 4, we apply weighted MMD to unsupervised domain adaptation and present a model named WDAN. Extensive experimental results are given in Section 5 to verify the effectiveness of our proposed weighted MMD model and detailed empirical analysis to our proposed model is provided. Finally, we conclude this work in Section 6.

2. Preliminaries and Related Work

In this section, we first review MMD and its application in domain adaptation, and then survey several other methods used to measure domain discrepancy.

2.1. MMD and Its Application in Domain Adaptation

Domain adaptation aims at adapting the discriminative model learned on source domain into target domain. Depending on the accessibility of class labels for target samples during training, research lines can be grouped into three categories: supervised, semi-supervised, and unsupervised domain adaptation. In this paper, we focus on learning transferable CNN features for unsupervised domain adaptation (UDA), where the labels of all target samples are unknown during training. Compared with the other settings, UDA is more ubiquitous in real-world applications.

Due to the unavailability of labels in the target domain, one commonly used strategy of UDA is to learn domain invariant representation via minimizing the domain distribution discrepancy. Maximum Mean Discrepancy (MMD) is an effective non-parametric metric for comparing the distributions based on two sets of data [4]. Given two distributions s and t, by mapping the data to a reproducing kernel Hilbert space (RKHS) using function $\phi(\cdot)$, the MMD between s and t is defined as,

$$\mathrm{MMD}^2(s,t) = \sup_{\|\phi\|_{\mathcal{H}}\leq 1} \left\| E_{\mathbf{x}^s\sim s}\left[\phi(\mathbf{x}^s)\right] - E_{\mathbf{x}^t\sim t}\left[\phi(\mathbf{x}^t)\right] \right\|_{\mathcal{H}}^2, \tag{1}$$

where $E_{\mathbf{x}^s\sim s}[\cdot]$ denotes the expectation with regard to the distribution s, and $\|\phi\|_{\mathcal{H}} \leq 1$ defines a set of functions in the unit ball of a RKHS $\mathcal{H}$. Based on the statistical tests defined by MMD, we have $\mathrm{MMD}(s,t) = 0$ iff $s = t$. Denote by $\mathcal{D}_s = \{\mathbf{x}_i^s\}_{i=1}^M$ and $\mathcal{D}_t = \{\mathbf{x}_i^t\}_{i=1}^N$ two sets of samples drawn *i.i.d.* from the distributions s and t, respectively. An empirical estimate of MMD can be given by [16],

$$\mathrm{MMD}^2(\mathcal{D}_s,\mathcal{D}_t) = \left\| \frac{1}{M}\sum_{i=1}^{M}\phi(\mathbf{x}_i^s) - \frac{1}{N}\sum_{j=1}^{N}\phi(\mathbf{x}_j^t) \right\|_{\mathcal{H}}^2, \tag{2}$$

where $\phi(\cdot)$ denotes the feature map associated with the kernel map $k(\mathbf{x}^s,\mathbf{x}^t) = \langle\phi(\mathbf{x}^s),\phi(\mathbf{x}^t)\rangle$. $k(\mathbf{x}^s,\mathbf{x}^t)$ is usually defined as the convex combination of L basis kernels $k_l(\mathbf{x}^s,\mathbf{x}^t)$ [25],

$$k(\mathbf{x}^s,\mathbf{x}^t) = \sum_{l=1}^{L}\beta_l k_l(\mathbf{x}^s,\mathbf{x}^t), \text{s.t.}\beta_l \geq 0, \sum_{l=1}^{L}\beta_l = 1. \tag{3}$$

Most existing domain adaptation methods [26, 29, 3, 41, 36] are based on the MMD defined in Eqn. (2) and only linear kernel is adopted for simplicity. Because the formulation of MMD in Eqn. (2) is based on pairwise similarity and is computed in quadratic time complexity, it is prohibitively time-consuming and unsuitable for using mini-batch stochastic gradient descent (SGD) in CNN-based domain adaptation methods. Gretton *et al*. [16] further suggest an unbiased approximation to MMD_l with linear complexity. Without loss of generality, by assuming $M = N$, MMD_l can then be computed as,

$$\mathrm{MMD}_l^2(s,t) = \frac{2}{M}\sum_{i=1}^{M/2} h_l(\mathbf{z}_i), \tag{4}$$

where h_l is an operator defined on a quad-tuple $\mathbf{z}_i = (\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t, \mathbf{x}_{2j}^t)$,

$$\begin{aligned} h_l(\mathbf{z}_i) =& k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s) + k(\mathbf{x}_{2j-1}^t, \mathbf{x}_{2j}^t) \\ & - k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2j}^t) - k(\mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t). \end{aligned} \tag{5}$$

The approximation in Eqn. (4) takes a summation form and is suitable for gradient computation in a mini-batch manner. Based on the work in [16], Long *et al*. [25] propose deep adaptation networks and residual transfer networks for UDA by introducing MMD_l based adaptation layers into deep CNNs. However, the existing MMD-based UDA approaches all assume that the source and target data have the same class prior distributions, which does not always hold in real-world applications, as illustrated in Fig. 1. Our empirical experiments show that class weight bias can result in performance degradation for MMD-based UDA.

2.2. Metrics for Domain Discrepancy

Besides MMD, there are several other metrics for measuring domain discrepancy. Baktashmotlagh *et al*. [3] propose a distribution-matching embedding (DME) approach for UDA, where both MMD and the Hellinger distance are adopted to measure the discrepancy between the source and target distributions. Instead of embedding of distributions, discriminative methods such as domain classification [11] and domain confusion [35] have also been introduced to learn domain invariant representation. However, class weight bias is also not yet considered in these methods.

Several sample reweighting or selection methods [13, 19] are similar to our weighted MMD in spirit, and have been proposed to match the source and target distributions.

These methods aim to learn sample-specific weights or select appropriate source samples for target data. Different from them, our proposed weighted MMD alleviates class weight bias by assigning class-specific weights to source data.

3. Weighted Maximum Mean Discrepancy

In this section, we will introduce the proposed weighted MMD. Denote by $p_s(\mathbf{x}^s)$ and $p_t(\mathbf{x}^t)$ the probability density functions of the source data $\mathbf{x}^s$ and the target data $\mathbf{x}^t$, y^s and y^t be the class labels of $\mathbf{x}^s$ and $\mathbf{x}^t$, respectively. Actually, both $p_s(\mathbf{x}^s)$ and $p_t(\mathbf{x}^t)$ can be further represented as the mixtures of class conditional distributions,

$$\begin{aligned} p_u(\mathbf{x}^u) &= \sum_{c=1}^{C} p_u(y^u = c) p_u(\mathbf{x}^u | y^u = c) \\ &= \sum_{c=1}^{C} w_c^u p_u(\mathbf{x}^u | y^u = c), u \in \{s, t\}, \end{aligned} \tag{6}$$

where $w_c^s = p_s(y^s = c)$ and $w_c^t = p_t(y^t = c)$ denote the class prior probability (*i.e.*, class weights) of the source and target samples, respectively, and C denotes the number of classes.

Note that, the difference between the class conditional distributions $p_s(\mathbf{x}^s|y^s = c)$ and $p_t(\mathbf{x}^t|y^t = c)$ serves as a proper metric of domain discrepancy. However, due to the unavailability of class labels for target data in UDA, the MMD between $p_s(\mathbf{x}^s)$ and $p_t(\mathbf{x}^t)$ is usually adopted as a domain discrepancy metric. When $w_c^s = w_c^t$ (c = 1, 2, ..., C), we argue that it is a suitable alternative. Unfortunately, as shown in Fig. 1, the assumption $w_c^s = w_c^t$ generally does not hold. For this case, MMD cannot cope with class weight bias across domains. We propose to construct a reference source distribution $p_{s,\alpha}(\mathbf{x}^s)$ for comparing the discrepancy between the source and target domains. Specifically, we require that $p_{s,\alpha}(\mathbf{x}^s)$ has the same class weights with the target domain but owns the class conditional distributions in source domain. Let $\alpha_c = w_c^t / w_c^s$. In order to eliminate the effect of class weight bias, we define $p_{s,\alpha}(\mathbf{x}^s)$ as,

$$p_{s,\alpha}(\mathbf{x}^s) = \sum_{c=1}^{C} \alpha_c w_c^s p_s(\mathbf{x}^s | y^s = c). \tag{7}$$

Denote by $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^M$ the training set from source domain and $\mathcal{D}_t = \{\mathbf{x}_j^t\}_{j=1}^N$ the test set from the target domain. Given the class weights of the target samples, the empirical estimation of weighted MMD $p_{s,\alpha}(\mathbf{x}^s)$ and $p_t(\mathbf{x}^t)$ can be given by,

$$\mathrm{MMD}_w^2(\mathcal{D}_s, \mathcal{D}_t) = \left\| \frac{1}{\sum_{i=1}^M \alpha_{y_i^s}} \sum_{i=1}^{M} \alpha_{y_i^s} \phi(\mathbf{x}_i^s) - \frac{1}{N} \sum_{j=1}^{N} \phi(\mathbf{x}_j^t) \right\|_{\mathcal{H}}^2. \tag{8}$$

Assuming $M = N$, the linear time complexity approximation of weighted MMD can then be computed as,

$$\mathrm{MMD}_{l,w}^2(\mathcal{D}_s, \mathcal{D}_t) = \frac{2}{M} \sum_{i=1}^{M/2} h_{l,w}(\mathbf{z}_i), \tag{9}$$

where $h_{l,w}(\mathbf{z}_i)$ is an operator defined on a quad-tuple $\mathbf{z}_i = (\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t, \mathbf{x}_{2j}^t)$,

$$\begin{aligned} h_{l,w}(\mathbf{z}_i) = \alpha_{y_{2i-1}^s} \alpha_{y_{2i}^s} k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s) + k(\mathbf{x}_{2i-1}^t, \mathbf{x}_{2i}^t) \\ - \alpha_{y_{2i-1}^s} k(\mathbf{x}_{2i-1}^s, \mathbf{x}_{2j}^t) - \alpha_{y_{2i}^s} k(\mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t). \end{aligned} \tag{10}$$

4. Weighted Domain Adaptation Network

By far, we have introduced our weighted MMD for measuring domain discrepancy. But there are two remained issues to be addressed. On one hand, the proposed weighted MMD, similar to MMD, should be incorporated into some classifiers for domain adaptation. On the other hand, the class distribution on target domain is generally unknown during training. In this section, we propose a weighted domain adaptation network (WDAN) model, which is in essential an extension of the semi-supervised logistic regression [1] by adding the WMMD term and incorporating with CNN. Meanwhile, we employ the CEM [6] framework and show how we optimize the proposed WDAN without the label information of the target samples.

First, based on the research in [40, 25], the features gradually become task specific as the layers go toward the top one, resulting in increasing dataset bias for the higher layers of features. Therefore, to generalize CNN for domain adaptation, the weighted MMD-based regularizers are added to the higher layers of CNN. Second, the relationship between semi-supervised learning and domain adaptation has been studied in [33]. To further exploit the unlabelled data on target domain, we follow the semi-supervised CEM model in [1], leading to the following WDAN model,

$$\begin{aligned} \min_{\mathbf{W}, \{\hat{y}_j\}_{j=1}^N, \boldsymbol{\alpha}} & \frac{1}{M} \sum_{i=1}^{M} \ell(\mathbf{x}_i^s, y_i^s; \mathbf{W}) + \gamma \frac{1}{N} \sum_{j=1}^{N} \ell(\mathbf{x}_j^t, \hat{y}_j^t; \mathbf{W}) \\ & + \lambda \sum_{l=l_1}^{l_2} \mathrm{MMD}_{l,w}(\mathcal{D}_s^l, \mathcal{D}_t^l), \end{aligned} \tag{11}$$

where $\mathbf{W}$ denotes the model parameters to be learned, and $\{\hat{y}_j\}_{j=1}^N$ represent the assigned labels of target samples. λ and γ are two tradeoff parameters. $\mathcal{D}_s^l$ and $\mathcal{D}_t^l$ denote the l-th layer features of the source and target domains, respectively. w_c^s is estimated based on the source data $\mathcal{D}_s^l$, *i.e.*, $w_c^s = M_c/M$, where M_c is the number of samples of the c-th class. The first two terms of Eqn. (11) are the soft-max loss items on the source and target samples, respectively.

And the third term is the weighted MMD regularizers for the $l_1 \sim l_2$-th layers defined in Eqn. (8).

Next, we explain the optimization procedure of the proposed WDAN model. Following the CEM algorithm in [6], the WDAN model is optimized by alternating between: (i) E-step: estimating the class posterior probability of $\{\mathbf{x}_j^t\}_{j=1}^N$, (ii) C-step: assigning pseudo-labels $\{\hat{y}_j\}_{j=1}^N$ and estimating auxiliary weights α, (iii) M-step: updating the model parameters $\mathbf{W}$. Given the model parameters $\mathbf{W}$, for each $\mathbf{x}_j^t$, we first estimate the class posterior probability based on the output of softmax classifier. The pseudo-label to $\hat{y}_j$ is assigned to $\mathbf{x}_j^t$ based on the maximum posterior probability, and the auxiliary weights α are then estimated based on pseudo-labels. Given $\{\hat{y}_j\}_{j=1}^N$ and α, the conventional backpropagation algorithm is then deployed to update $\mathbf{W}$. In the following, we give more details on the E-step, C-step, and M-step.

E-step: Fixed $\mathbf{W}$, for each sample $\mathbf{x}_j^t$ from target domain, the CNN output to the cth class is represented as $g_c(\mathbf{x}_j^t; \mathbf{W})$. Here we simply define the class posterior probability $p(y_j^t = c|\mathbf{x}_j^t)$ as,

$$p(y_j^t = c|\mathbf{x}_j^t) = g_c(\mathbf{x}_j^t; \mathbf{W}). \tag{12}$$

C-step: With $p(y_j^t = c|\mathbf{x}_j^t)$, we assign pseudo-label $\hat{y}_j$ to $\mathbf{x}_j^t$ by,

$$\hat{y}_j = \arg\max_c p(y_j^t = c|\mathbf{x}_j^t). \tag{13}$$

Let $\mathbf{1}_c(\hat{y}_j)$ be an indicator function,

$$\mathbf{1}_c(\hat{y}_j) = \begin{cases} 1, \text{if } \hat{y}_j = c \\ 0, \text{otherwise.} \end{cases} \tag{14}$$

The class weight $\hat{w}_c^t$ can be estimated by $\hat{w}_c^t = \sum_j \mathbf{1}_c(\hat{y}_j) \Big/ N$, where N is the number of target samples. Then the auxiliary weight can be updated with $\alpha_c = \hat{w}_c^t / w_c^s$.

M-step: Fixed $\boldsymbol{\alpha}$, the subproblem on $\mathbf{W}$ can be formulated as,

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) = \frac{1}{M}\sum_{i=1}^{M} \ell(\mathbf{x}_i^s, y_i^s; \mathbf{W}) + \gamma \frac{1}{N}\sum_{j=1}^{N} \ell(\mathbf{x}_j^t, \hat{y}_j^t; \mathbf{W}) + \lambda \sum_{l=l_1}^{l_2} \mathrm{MMD}_{l,w}(\mathcal{D}_s^l, \mathcal{D}_t^l). \tag{15}$$

Since the gradients of the three terms in Eqn. (15) are computable, $\mathbf{W}$ can be updated with mini-batch SGD. Let $\mathbf{z}_i = (\mathbf{x}_{2i-1}^s, \mathbf{x}_{2i}^s, \mathbf{x}_{2j-1}^t, \mathbf{x}_{2j}^t)$ be a quad-tuple and $\mathbf{z}_i^l = (\mathbf{z}_{i,1}^l = \mathbf{f}_{2i-1}^{s,l}, \mathbf{z}_{i,2}^l = \mathbf{f}_{2i}^{s,l}, \mathbf{z}_{i,3}^l = \mathbf{f}_{2i-1}^{t,l}, \mathbf{z}_{i,4}^l = \mathbf{f}_{2i}^{t,l})$ be the l-th layer feature representation of $\mathbf{z}_i$. Given $\mathbf{z}_i$, the gradient with respect to the l-th layer parameter $\mathbf{W}^l$ can be written as,

$$\frac{\partial \mathcal{L}(\mathbf{W})}{\partial \mathbf{W}^l} = \frac{1}{2}\sum_{j=1}^{2} \frac{\partial \ell(\mathbf{z}_{i,j}, y_{i,j}; \mathbf{W})}{\partial \mathbf{z}_{i,j}^l} \frac{\partial \mathbf{z}_{i,j}^l}{\partial \mathbf{W}^l} + \frac{\gamma}{2}\sum_{j=3}^{4} \frac{\partial \ell(\mathbf{z}_{i,j}, \hat{y}_{i,j}; \mathbf{W})}{\partial \mathbf{z}_{i,j}^l} \frac{\partial \mathbf{z}_{i,j}^l}{\partial \mathbf{W}^l} + \lambda \sum_{k=1}^{4} \frac{\partial h_{l,w}(\mathbf{z}_i)}{\partial \mathbf{z}_{i,k}^l} \frac{\partial \mathbf{z}_{i,k}^l}{\partial \mathbf{W}^l}. \tag{16}$$

Taking $k = 1$ for example, $\frac{\partial h_{l,w}(\mathbf{z}_i^l)}{\partial \mathbf{z}_{i,1}^l}$ can be computed as,

$$\frac{\partial h_{l,w}(\mathbf{z}_i)}{\partial \mathbf{f}_{2i-1}^{s,l}} = \alpha_{y_{2i-1}^s} \alpha_{y_{2i}^s} \frac{\partial k(\mathbf{f}_{2i-1}^{s,l}, \mathbf{f}_{2i}^{s,l})}{\partial \mathbf{f}_{2i-1}^{s,l}} - \alpha_{y_{2i-1}^s} \frac{\partial k(\mathbf{f}_{2i-1}^{s,l}, \mathbf{f}_{2i}^{t,l})}{\partial \mathbf{f}_{2i-1}^{s,l}}. \tag{17}$$

Similarly, $\frac{\partial h_{l,w}(\mathbf{z}_i^l)}{\partial \mathbf{z}_{i,k}^l}$ can also be computed for other k values. Thus, the model parameters can be updated via backpropagaton with a mini-batch of quad-tuple. Moreover, following [25, 17], the multiple kernel parameters $\boldsymbol{\beta}$ can also be updated during training.

The algorithm described above actually is an extension of classification EM. The C-step in [6] only assigns pseudo-label to each unlabeled sample, while in this work we further estimate the auxiliary weights α with the pseudo-labels. As shown in [6], such a optimization procedure can converge to a stationary value. The experiment also empirically validate that our algorithm works well in estimating the auxiliary weights $\boldsymbol{\alpha}$.

5. Experiments

In this section, we first evaluate our proposed WDAN on four widely used benchmarks in UDA, *i.e.*, *Office-10+Caltech-10* [14], *Office31* [31], *ImageCLEF* [5] and *Digit Recognition*. Moreover, we also provide empirical analysis to our proposed WDAN model from three aspects, *i.e.*, hyper-parameter sensitivity, robustness to class weight bias, and feature visualization.

Following the common setting in UDA, we implement our WDAN model based on four widely used CNN architectures, *i.e.*, LeNet [22], AlexNet [21], GoogLeNet [34] and VGGnet-16 [32]. As suggested in [25], we train our method based on pre-trained AlexNet, VGGNet-16, or GoogLeNet on ImageNet, with the layers from conv1 to conv3 fixed for AlexNet and inception layers from inc1 to inc3 fixed for GoogLeNet. The WDAN (LeNet) is trained from the scratch (random initialization). In addition, the auxiliary weight is initialized with $\alpha_c = 1$ for each class. For l_1 and l_2, we follow the setting in [25].

Method	A→C	W→C	D→C	C→A	C→W	C→D	Avg.
AlexNet [21]	84.0±0.3	77.9±0.4	81.0±0.4	91.3±0.2	83.2±0.3	89.1±0.2	84.0
LapCNN (AlexNet) [38]	83.6±0.6	77.8±0.5	80.6±0.4	92.1±0.3	81.6±0.4	87.8±0.4	83.9
DDC (AlexNet) [36]	84.3±0.5	76.9±0.4	80.5±0.2	91.3±0.3	85.5±0.3	89.1±0.3	84.6
DAN (AlexNet) [25]	86.0±0.5	81.5±0.3	82.0±0.4	92.0±0.3	92.6±0.4	90.5±0.2	87.3
WDAN (AlexNet)	**86.9**±0.1	**84.1**±0.2	**83.9**±0.1	**93.1**±0.2	**93.6**±0.2	**93.4**±0.2	**89.2**
WDAN* (AlexNet)	87.1±0.2	85.1±0.3	85.2±0.2	93.2±0.1	93.5±0.3	94.5±0.2	89.8
GoogLeNet [34]	91.3±0.2	88.2±0.3	88.9±0.3	95.2±0.1	92.5±0.2	94.7±0.3	91.8
DDC (GoogLeNet) [36]	91.4±0.2	88.7±0.3	89.0±0.4	95.3±0.2	93.0±0.1	94.9±0.4	92.1
DAN (GoogLeNet) [25]	91.4±0.3	89.7±0.2	89.1±0.4	95.5±0.2	93.1±0.3	95.3±0.1	92.3
WDAN (GoogLeNet)	**92.2**±0.2	**91.0**±0.5	**89.8**±0.3	**95.5**±0.3	**95.4**±0.2	**95.5**±0.5	**93.2**
VGGnet-16 [32]	89.6±0.4	88.1±0.4	85.4±0.5	93.7±0.2	94.3±0.2	93.7±0.2	90.8
DAN (VGGnet-16) [25]	91.2±0.2	90.6±0.3	87.1±0.4	95.7±0.2	95.3±0.3	94.7±0.1	92.4
WDAN (VGGnet-16)	**91.4**±0.2	**91.0**±0.2	**89.0**±0.3	**95.7**±0.1	**95.8**±0.2	**95.9**±0.3	**93.1**

Table 1. Results (in %) of different methods based on *AlexNet*, *GoogleNet* and *VGGnet-16* on *Office-10+Caltech-10*. Note that the results of LapCNN, DDC and DAN are duplicated from [25]. $\star$ indicates that the ground truth class distributions in both source and target domain are used as prior.

Concretely, WMMD-based regularizers are added to the last three fully connected layers for AlexNet, the last inception and fully connected layers for GoogleNet, and the last fully connected layer for LeNet. All experiments are implemented by using Caffe Toolbox [20], and run on a PC equipped with a NVIDIA GTX 1080 GPU and 32G RAM. We set the batch size to 64 for all methods, and optimize the learning rate for each model independently. The tradeoff parameters λ and γ are optimized in sets $\{0,\ 0.03,\ 0.07,\ 0.1,\ 0.4,\ 0.7,\ 1.4,\ 1.7,\ 2\}$ and $\{0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5,\ 0.6,\ 0.7,\ 0.8,\ 0.9,\ 1.0\}$ by cross-validation, respectively. The source code of our WDAN is available at: https://github.com/yhldhit/WMMD-Caffe.

5.1. Comparison with State-of-the-arts

For UDA, we employ the standard protocols as [25, 23, 27], where all the samples in source and target domain are used for training. The averaged results over 10 trials on target domain set are reported for comparison.

5.1.1 Office-10+Caltech-10

Office-10+Caltech-10 [14] is widely used for domain adaptation, which picks up 10 classes shared in *Office-31* [31] and *Caltech-256* [18]. It consists of four domains where Amazon (A), Webcam (W) and DSLR (D) are from *Office-31*, and the another one is *Caltech-256* (C). On this dataset, we conduct experiments based on AlexNet, GoogLeNet and VGGnet-16, and exploit the same setting as [25] for performance comparison.

We compare our WDAN with several state-of-the-art methods as listed in Table 1, including its MMD counterpart DAN [25]. By AlexNet, GoogLeNet, and VGGnet-16 we indicate to fine-tune the pre-trained CNN models for special tasks. LapCNN [38] can be seen as a variant of CNN, which first shows deep structure learning can be improved by jointly learning an embedding with the unlabeled data, and then exploits the embedding as a regularizer. By embedding a single kernel MMD layer into CNN structure, DDC [36] develops a unified deep framework to jointly learn semantically meaningful feature and perform adaption cross domain.

Numerical results in Table 1 show that our weighted DAN achieves the best performance, independently of the employed CNN structure. Moreover, the WDAN is superior to DAN by 1.9%, 0.9% and 0.7%, respectively. We contribute this improvement to that our weighted MMD model can alleviate the effect of class weight bias. In addition, the superiority over other state-of-the-art methods demonstrate the effectiveness of the proposed WDAN. Finally, we exploit the ground truth class distributions in both source and target domains as prior of the proposed WDAN based on AlexNet, which is indicated as WDAN* (AlexNet) in Table 1. Although WDAN* can further improve the performance of WDAN, the smaller gap between them than one between weighted DAN and DAN validate the effectiveness of our proposed learning and estimation method.

5.1.2 ImageCLEF

ImageCLEF [5] is developed for the ImageCLEF domain adaptation task[1]. This dataset collects images from five widely used image benchmarks, including *Caltech256* [18], *Bing*, *PASCAL VOC2012* [9], *ImageNet2012* [7] and

[1] http://www.imageclef.org/2014/adaptation

Method	P→C	B→C	C→B	P→B	C→P	B→P	Avg.
GoogLeNet	91.0±0.5	92.4±0.3	61.2±0.4	**55.3**±0.3	61.2±0.2	58.1±0.3	69.9
DDC [36]	91.2±0.4	92.6±0.4	62.0±0.3	54.3±0.3	61.7±0.4	58.6±0.3	70.1
DAN	91.4±0.2	93.0±0.1	62.5±0.3	54.5±0.2	62.2±0.3	59.0±0.3	70.4
WDAN	**91.4**±0.1	**93.8**±0.2	**62.9**±0.3	55.2±0.3	**65.0**±0.2	**59.5**±0.3	**71.3**

Table 2. Results (in %) of different methods based on *GoogLeNet* on *ImageCLEF* dataset.

Method	M→S	S→M	M→U	U→M	Avg
LeNet	17.2±0.3	56.8±0.5	61.5±0.4	46.5±0.6	45.5
SA [10]	21.1±0.2	59.3±0.3	55.0±0.4	51.6±0.6	46.8
DAN	19.3±0.4	65.2±0.3	69.1±0.5	60.5±0.7	53.5
WDAN	**23.4**±0.2	**67.4**±0.4	**72.6**±0.3	**65.4**±0.4	**57.2**

Table 3. Results (in %) of different methods based on *LeNet* on *Digit Classification*.

Method	A→W	D→W	W→D	A→D	D→A	W→A	Avg.
AlexNet	60.4±0.5	94.0±0.3	92.2±0.3	58.6±0.6	46.0±0.6	49.0±0.5	66.7
DAN	66.0±0.5	94.3±0.3	95.2±0.3	63.2±0.4	50.0±0.5	51.1±0.5	70.0
WDAN	**66.8**±0.3	**95.9**±0.1	**98.7**±0.4	**64.5**±0.2	**53.8**±0.1	**52.7**±0.2	**72.1**

Table 4. Results (in %) of different methods based on *AlexNet* on *Office-31* dataset.

SUN [39]. This dataset is thought to be more difficult, since some domains contain low-quality images, making this benchmark a good compliance to the *Office-10+Caltech-10*, where the domain is more similar. Different from the original experimental setting, in this paper, we use a subset of *ImageCLEF*, which contains three datasets, *i.e.*, *Caltech256* (C), *Bing* (B) and *PASCAL VOC2012* (P). Meanwhile, we exploit all images in each subset rather than follow the standard protocol to sample the same number of images for each class. Such setting results in six domain adaptation tasks.

We compare WDAN with three related methods based on GoogLeNet, *i.e.*, GoogLeNet, DDC and DAN. We implement them by using the codes released by authors[2], and try our best to optimize them. The results of the competing methods are shown in Table 2, from which we can see that our proposed weighted DAN obtains the best performance in most of the cases, and achieves 1.4%, 1.2% and 0.9% gains over GoogLeNet, DDC and DAN on average, respectively. The above results show the proposed weighted MMD is helpful to improve the performance of domain adaptation task.

5.1.3 Digit Recognition

Furthermore, we conduct experiment on digit recognition, which is usually adopted to domain adaptation. In this paper, we only considering training images of three benchmarks, *i.e.*, *MNIST* (M), *SVHN* (S) and *USPS* (U) and conduct experiments on four tasks. As LeNet [22] is usually used for digit recognition, we implement our WDAN and the competing methods based on it. Among them, SA [10] proposes a subspace alignment method for domain adaptation, which aims at learning a feature mapping to align

[2] https://github.com/longmingsheng/mmd-caffe

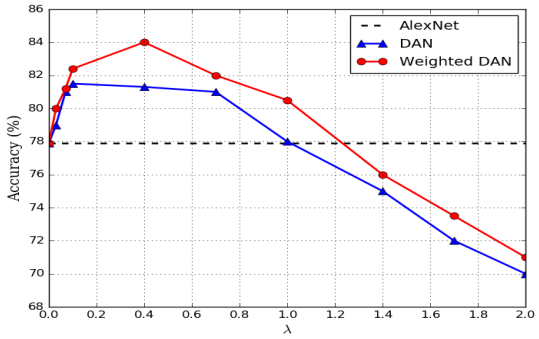

Figure 3. Performance (in %) of different methods *w.r.t.* λ.

source samples with target samples. For fair comparison, we implement SA by using the features from the fine-tuned LeNet. The results reported in Table 3 clearly show that our proposed WDAN achieves the best performance on all tasks, and outperforms LeNet, SA and DAN by 11.7%, 10.4% and 3.7% on average, respectively. The significant improvements over competing methods show the proposed weighted MMD model is effective and meaningful.

5.1.4 Office-31

Finally, experiments are further conducted to assess WDAN on datasets with more classes. We conduct experiments on a dataset with 31 classes (*i.e.* Office-31). There are three domains, *i.e.*, Amazon (A), Webcam (W) and DSLR (D), in Office-31. In this part, we consider all the six UDA tasks, and report the results using Alexnet. Table 4 shows the results of AlexNet, DAN, and WDAN. It can be seen that the proposed WMMD achieves better results than its MMD counterpart, indicating that WMMD also works well on dataset with more classes. To sum up, the promising performance of our weighted MMD model can be verified on various CNN architectures (*i.e.*, AlexNet, GoogLeNet and LeNet) and various datasets with different number of classes.

5.2. Empirical Analysis

In this subsection, we perform empirical analysis of the proposed WDAN from three aspects. Firstly, we evaluate the effect of hyper-parameter λ on our proposed WDAN model in Eqn. (11). Secondly, compared with its baselines, *i.e.*, Alexnet and DAN, we show our proposed WDAN is robust to class weight bias. Finally, we make a visualization of learned feature representations.

5.2.1 Effect of Parameter λ

The objective in Eqn. (11) of WDAN consists of three terms, *i.e.*, conventional empirical losses on the source

and target domains, and MMD-based regularizer. Generally speaking, the empirical risk term keeps the learned deep feature to be discriminative on source domain while the MMD-based regularizer encourages domain invariant feature representation. Both of this two aspects are of essential importance for domain adaptation. The parameter λ in the objective Eqn. (11) makes a tradeoff between this two parts, and could greatly impact the performance of domain adaptation. To have a closer look at this parameter, we evaluate our proposed WDAN based on AlexNet on the task W→C from *Office-10+Caltech-10* under various λ. As suggested above, λ belongs to the set $\{0.0, 0.03, 0.07, 0.1, 0.4, 0.7, 1, 1.4, 1.7, 2\}$. Meanwhile, we also compared our WDAN model with DAN under various λ. AlexNet is reported as baseline and corresponds to the case $\lambda = 0$. The results are illustrated in Fig. 3.

Obvious conclusions can be drawn from the results: (i) our proposed WDAN consistently outperforms the DAN, demonstrating that mining the class weight bias in MMD is meaningful and beneficial; (ii) WDAN and DAN achieve the best results at $\lambda = 0.4$ and $\lambda = 0.1$, and outperforms the baseline, *i.e.*, AlexNet, when $\lambda < 1.2$ and $\lambda < 1.0$, respectively, indicating that an appropriate balance is important and necessary.

5.2.2 Impact of Class Weight Bias

To further clarify the impact of class weight bias on MMD-based domain adaptation methods, we conduct experiments on a variant of the task V→C from *ImageCLEF* based on AlexNet. Specifically, we pick up two shared classes, *i.e.*, airplane and motorbike, in the source domain *PASCAL VOC2012* (V) and target domain *Caltech256* (C), which forms a two-class classification problem.

Then we fix the class weights as 0.5 for each class on source domain and train different methods with gradually changing the class distribution on target domain, which can be interpreted as different level of the class weight bias cross source and target domains. Fig. 4 show the results of WDAN, DAN and AlexNet under different levels of the class weighted bias. From it we can see that the class weight bias has great influence on performance of MMD-based domain adaptation methods. Moreover, the conventional MMD-based methods (*e.g.*, DAN) are limited in handling the class weight bias, as its results significantly degrade with increasing class weighted bias. In addition, our proposed WDAN is more robust to class weighted bias.

5.2.3 Feature Visualization

Following the work in [25], we visualize the features learned by WDAN and DAN on target domain in the D→C task from *Office-10+Caltech-10*. For feature visualization, we employ the t-SNE visualization method [37] whose source codes are provided[3]. The results of feature visualization for DAN and weighted DAN are illustrated in Fig. 5 (a) and Fig. 5 (b), respectively. As shown in the orange boxes of Fig. 5, features learned by the proposed WDAN can reserve more class discrepancy distance than ones learned by DAN. The underlying reason lies in the fact that WDAN, by considering a weighted MMD regularizer, does not minimize the class weight bias as DAN does, which also accounts for that weighted DAN can outperform DAN on a variety of unsupervised domain adaptation tasks.

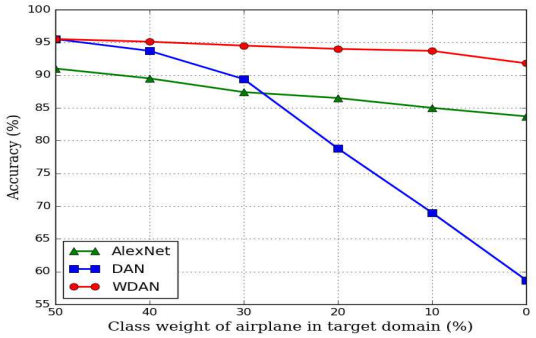

Figure 4. Performance (in %) of different methods *w.r.t.* class weight bias.

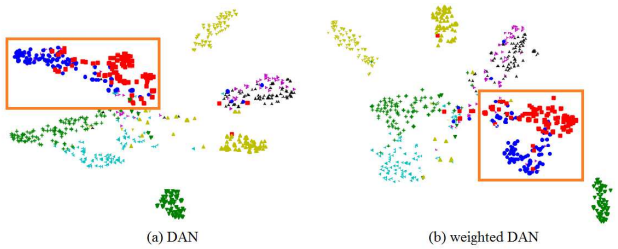

Figure 5. The t-SNE visualization of learned features of different methods.

6. Conclusion

In this paper, we focus on the uninvestigated issue of class weight bias in UDA, which has adverse effect on MMD-based domain adaptation methods. We first propose a novel weighted MMD to reduce the effect of class weight bias by constructing a reference source distribution based on target distribution. For UDA, we present a weighted DAN (WDAN) based on the proposed weighted MMD, and develop modified the CEM learning algorithm to jointly assign pseudo-labels, estimate the auxiliary weights and learn model parameters. Empirical results show that our proposed WDAN outperforms its MMD counterpart, *i.e.*, DAN, in various domain adaptation tasks. In future, there remains several issues to be investigated: (i) evaluation of weighted

[3] https://lvdmaaten.github.io/tsne/

MMD on non-CNN based UDA models, (ii) applications to other tasks (*e.g.*, image generation) based on measuring the discrepancy between distributions.

7. Acknowledgment

This work is supported in part by NSFC grant (61671182, 61471082, and 61370163). The authors also thank NVIDIA corporation for the donation of GTX 1080 GPU.

References

[1] M.-R. Amini and P. Gallinari. Semi-supervised logistic regression. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 390–394. IOS Press, 2002. 2, 4

[2] H. Azizpour, A. Sharif Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 36–45, 2015. 1

[3] M. Baktashmotlagh, M. T. Harandi, and M. Salzmann. Distribution-matching embedding for visual domain adaptation. *Journal of Machine Learning Research*, 2016. 1, 3

[4] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006. 3

[5] B. Caputo and N. Patricia. Overview of the imageclef 2014 domain adaptation task. In *ImageCLEF 2014: Overview and analysis of the results*, number EPFL-CONF-201812, 2014. 5, 6

[6] G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 14(3):315–332, 1992. 4, 5

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 6

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings the International Conference on Machine Learning*, pages 647–655, 2014. 1

[9] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 6

[10] B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2960–2967, 2013. 7

[11] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014. 3

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. 1

[13] B. Gong, K. Grauman, and F. Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *Proceedings the International Conference on Machine Learning*, pages 222–230, 2013. 3

[14] B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073, 2012. 5, 6

[15] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2006. 2

[16] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012. 3

[17] A. Gretton, D. Sejdinovic, H. Strathmann, S. Balakrishnan, M. Pontil, K. Fukumizu, and B. K. Sriperumbudur. Optimal kernel choice for large-scale two-sample tests. In *Advances in neural information processing systems*, pages 1205–1213, 2012. 2, 5

[18] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007. 6

[19] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, and A. J. Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2006. 1, 3

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014. 6

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 5, 6

[22] Y. Lecun, L. E. Bottou, Y. Bengio, and P. Haaner. Gradient-based learning applied to document recognition. 1998. 5, 7

[23] Y. Li, K. Swersky, and R. Zemel. Generative moment matching networks. In *Proceedings the International Conference on Machine Learning*, pages 1718–1727, 2015. 6

[24] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 1

[25] M. Long and J. Wang. Learning transferable features with deep adaptation networks. *CoRR, abs/1502.02791*, 1:2, 2015. 3, 4, 5, 6, 8

[26] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2200–2207, 2013. 1, 3

[27] M. Long, J. Wang, and M. I. Jordan. Unsupervised domain adaptation with residual transfer networks. *arXiv preprint arXiv:1602.04433*, 2016. 6

[28] T. Ming Harry Hsu, W. Yu Chen, C.-A. Hou, Y.-H. Hubert Tsai, Y.-R. Yeh, and Y.-C. Frank Wang. Unsupervised domain adaptation with imbalanced cross-domain data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4121–4129, 2015. 1, 2

[29] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2011. 1, 3

[30] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 1

[31] K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 5, 6

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5, 6

[33] A. Søgaard. Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies*, 6(2):1–103, 2013. 4

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 5, 6

[35] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015. 3

[36] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 1, 3, 6, 7

[37] L. Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 2014. 8

[38] J. Weston, F. Ratle, H. Mobahi, and R. Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012. 6

[39] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010. 7

[40] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014. 1, 4

[41] E. Zhong, W. Fan, J. Peng, K. Zhang, J. Ren, D. Turaga, and O. Verscheure. Cross domain distribution adaptation via kernel mapping. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1027–1036, 2009. 1, 3