Exploring Uncertainty in Pseudo-label Guided Unsupervised Domain Adaptation

Jian Liang\textsuperscript{1,2,3}, Ran He\textsuperscript{1,3,4}, Zhenan Sun\textsuperscript{1,3,4} and Tieniu Tan\textsuperscript{1,3,4}

\textsuperscript{1} Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences (CASIA)
\textsuperscript{2} Department of ECE, National University of Singapore (NUS), Singapore
\textsuperscript{3} University of Chinese Academy of Sciences (UCAS)
\textsuperscript{4} Center for Excellence in Brain Science and Intelligence Technology, CAS

Abstract

Due to the unavailability of labeled target data, most existing unsupervised domain adaptation (UDA) methods alternately classify the unlabeled target samples and discover a low-dimensional subspace by mitigating the cross-domain distribution discrepancy. During the pseudo-label guided subspace discovery step, however, the posterior probabilities (uncertainties) from the previous target label estimation step are totally ignored, which may promote the error accumulation and degrade the adaptation performance. To address this issue, we propose to progressively increase the number of target training samples and incorporate the uncertainties to accurately characterize both cross-domain distribution discrepancy and other intra-domain relations. Specifically, we exploit maximum mean discrepancy (MMD) and within-class variance minimization for these relations, yet, these terms merely focus on the global class structure while ignoring the local structure. Then, a triplet-wise instance-to-center margin is further maximized to push apart target instances and source class centers of different classes and bring closer them of the same class. Generally, an EM-style algorithm is developed by alternating between inferring uncertainties, progressively selecting certain training target samples, and seeking the optimal feature transformation to bridge two domains. Extensive experiments on three popular visual domain adaptation datasets demonstrate that our method significantly outperforms recent state-of-the-art approaches.

Keywords: Unsupervised domain adaptation, Pseudo labeling, Feature transformation, Progressive learning, Transfer learning

1. Introduction

Researchers always assume that the training and testing data are drawn from the same distribution for simplicity in the field of pattern recognition and machine learning. Nevertheless, the assumption does not always hold in real-world applications, and the performance at testing time can be significantly degraded \cite{1}. For instance, a classifier trained on the annotated near-infrared human faces (the so-called source domain) may fail to recognize the face images under the visible light (target domain). To tackle this issue, a naive strategy would be to collect a certain number of labeled target data, yet it is time-consuming and expensive. Benefiting from cheap and massive publicly available datasets, researchers come up with another emerging strategy, domain adaptation (DA). DA tries to leverage domain shift characteristics from labeled data in an auxiliary and related domain for learning with new unlabeled or partially labeled data \cite{2}. According to the availability of labeled target instances, domain adaptation methods can be divided into two main categories, unsupervised one \cite{3, 4} and semi-supervised one \cite{5, 6}. The common practice of discriminative training is not generally feasible for unsupervised cases, however, making it especially challenging to describe the cross-domain relation. In this paper, we focus on this challenging but practical unsupervised domain adaptation problem. Numerous approaches have been proposed in the last decade to address unsupervised domain adaptation (UDA) \cite{7, 8, 9, 10, 11, 12}. To relate two different feature spaces together, one natural solution is instance re-weighting, i.e., adjusting the weights of source instances to better match the target data distribution \cite{13}. However, this solution works well only when certain parts of source instances can be reused for learning with the target domain, and the conditional distributions are not identical \cite{14}, which is rather hard to meet. In contrast, feature transformation methods \cite{15, 16} are more flexible where one common subspace is sought to make two domains distribute similarly to each other. The majority of existing feature transformation methods incorporate the empirical maximum mean discrepancy (MMD) \cite{17, 18} as a regularizer on the feature mapping function to mitigate various distribution differences across domains. Apart from the marginal distribution difference exploited in \cite{15, 16}, JDA \cite{19} further considers the conditional distribution difference across domains, which greatly improves the recognition accuracy.

Due to the lack of labeled target instances, an iterative pseudo-label guided scheme is always adopted by UDA methods to estimate the pseudo target labels, which further guide the feature transformation \cite{20, 21} or non-linear feature representations learning of deep methods \cite{22}. Extensive efforts have been devoted to leveraging MMD to explicitly minimize the distances of the empirical expectations of different distributions
Figure 1: Toy example of the instance-to-center margins. (a) is more preferred than (b) since two target classes are more separable. (Green arrow: instance-to-center distance, best viewed in color.)

(i.e., the domain mean and the class means), i.e., the distribution discrepancy is measured by several pairwise distances. Looking at Fig. 1, we find that (a) and (b) share nearly the same empirical MMD values, however, we expect the margin between each one target instance to its corresponding source center and other source centers to be maximized like (a). Obviously, maximizing this triplet-wise instance-to-center margin will enforce more separable target classes in turn. Actually, we focus on local separability by exploring hard negative mining [23] where the closest source center from different classes instead is picked as a negative sample within the triplet.

Different from supervised DA methods like [24] where the target labels are accurately provided, the pseudo target labels appearing during the iteration process are often not accurate, which can deteriorate the following feature transformation optimization problem for UDA. For example, if the initialized domain-invariant projection is not good enough, some hard target instances lying on the border of multiple classes would be misclassified, which severely affects the feature projection step in turn. Nevertheless, none of the previous DA methods have ever considered the confidences of the pseudo labeled target data.

To address these issues above, we put forward a novel method called Progressive leArning with Confidence-wEighted Targets (PACET) for UDA. We follow [21] and exploit the within-domain intra-class compactness and cross-domain distribution difference to infer the domain-invariant projection, thus two domains onto the learned subspace are expected to be semantically aligned and discriminative. As mentioned above, the discriminative ability is further enhanced by considering a local repulsive force term in Fig. 1. Then we train a standard classifier with projected source instances and exploit the conditional posterior probability as confidence of each uncertain target to each class rather than treat these target samples equally. Furthermore, these target confidences are seamlessly incorporated into both cross-domain and with-domain objectives for better adaptation. Intuitively, previous class centers are calculated by weighting different target instances, where the hard samples contribute less to updating the target class centers, which is considered somewhat to misclassified instances. Additionally, inspired by self-paced learning [25] where training samples are automatically included from ‘easy’ to ‘hard’, we further arrange the target instances in order of the formerly learned confidences and progressively pick them with the increase of the number of iterations, which is expected to alleviate the error accumulation. Finally, an EM-style scheme is exploited to alternately learn feature transformation, estimate the pseudo target labels as well as their class confidence scores (i.e., class posterior probability) and select certain training target instances. Experimental experiments reveal that this EM-style scheme quickly converges in several iterations (See Fig. 4(b)). The main contributions of this paper are summarized as follows:

- We introduce the problem of addressing the uncertainties of target pseudo labels, which is important yet understudied in the domain adaptation area.
- We propose a novel approach that progressively includes more target samples into training and incorporates previously estimated class confidence scores to characterize both the within- and cross-domain relations. Especially, we provide a more accurate conditional distribution discrepancy than that of [19, 21].
- To fully exploit the discriminative cross-domain structures, we compensate joint distribution adaptation by designing a new local triplet-wise instance-to-center margin for better separability.
- Experimental results demonstrate the superiority of our method over recent state-of-the-art approaches. Particularly, on the challenging Office-Caltech dataset with VGG features show that our method advances the best reported average accuracies [26] from 83.4% to 88.2% and 81.7% to 87.2%, respectively.

2. Related Work

Here we provide a brief review of MMD, then discuss some closely related feature transformation based UDA methods.

2.1. Maximum Mean Discrepancy

MMD [17] is primarily proposed to test whether distributions $p$ and $q$ are different on the basis of samples drawn from each of them, by finding a smooth function that behaves distinctively on the points drawn from $p$ and $q$. Specifically, the squared MMD in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ is formulated as follows.

$$
\text{MMD}^2[\mathcal{F}, p, q] = \sup_{\|f\|_{\mathcal{H}} \leq 1} \|E_p[f(x)] - E_q[f(y)]\|_{\mathcal{H}},
$$

where the notations $E_p[f(x)] = E_{x \sim p}[f(x)]$ and $E_q[f(y)] = E_{y \sim q}[f(y)]$ denote the expectations w.r.t. $p$ and $q$, and $\|f\|_{\mathcal{H}} \leq 1$ represents a set of functions in the unit ball of $\mathcal{H}$. As proven in [18, 27], MMD$^2[\mathcal{F}, p, q] = 0$ iff $p = q$.

In a RKHS, function evaluations can be written as $f(x) = \langle \phi(x), f \rangle$, and $k(\cdot, \cdot) = \langle \phi(\cdot), \phi(\cdot) \rangle$ is the universal kernel associated with this mapping function. Then the expectation $E_p[f(x)]$ can be rewritten as $E_p[f(x)] = \langle \mu [p], f \rangle$, where $\mu[p] = \mathbb{E}_{x \sim p} [\phi(x)]$ is the expectation of $\phi(x)$. Given observations $X := \{x_1, \ldots, x_m\}$ and $Y := \{y_1, \ldots, y_n\}$, drawn i.i.d. from the distributions $p$ and $q$ respectively, an empirical estimate of squared MMD [17] is further given

$$
\overline{\text{MMD}}^2[\mathcal{F}, X, Y] = \|\frac{1}{m} \sum_{i=1}^{m} \phi(x_i) - \frac{1}{n} \sum_{j=1}^{n} \phi(y_j)\|_{\mathcal{H}}^2. \quad (1)
$$
2.2. Feature Transformation based UDA Methods

We introduce several works [19, 15, 20, 16, 21, 29] in spirit to our work and discuss the differences between them. Being one of the pioneering feature transformation based DA methods, TCA [15] embeds both the source and target data into a shared low-dimensional space, and attempts to jointly preserve the variance and reduce the marginal distribution difference across domains. DIP [16] exploits manifold learning and Gaussian kernel for MMD, and accounts the source discriminative information by minimizing the within-source variance. Besides previous marginal distributions, JDA [19] first exploits the conditional distributions across domains in the common space together to better characterize the cross-domain difference. Once pseudo target labels are obtained, JDA would pull closer not only the overall domain means but also the same class means from different domains, which is more desirable. DICE [21] proposes a simple domain-irrelevant class clustering objective which has been proven to include both JDA and DIP as special cases, while MCS [28] develops a related minimization objective which has been proven to include both JDA and DICE [21].

DGA-DA [29] extends JDA by putting forward a repulsive center shift strategy for weakly supervised source DIP as special cases, while MCS [28] develops a related minimizing objective which has been proven to include both JDA and DICE [21] proposes a simple domain-irrelevant class clustering objective which has been proven to include both JDA and DIP as special cases, while MCS [28] develops a related minimizing objective which has been proven to include both JDA and DICE [21].

MCS [28] exploits a strategy inspired by [41] in which the unlabeled samples with the maximum and minimum values of the decision function are moved into the training set, and designs an effective solution for domain adaptation.

On the contrary, co-training always assumes that each example is described using two different feature sets that provide different, complementary information about the instance. [42] develops a co-transfer learning framework which seamlessly combines the co-training and transfer learning paradigms for cross-resolution face matching. Inspired by self-paced learning, [43] improves co-training with a “draw with replacement” learning module to remove false labeled instances in the initial training rounds. Similarly, [44] proposes a self-paced adaptation to shift object detection model from images to videos by learning labeled source samples and target data with pseudo-labels in an easy-to-hard way.

Actually, our method exploits a progressive learning paradigm that behaves like self-training and co-training, however, we neither move the most confident labeled target data to the source domain nor address a new semi-supervised DA problem by adding previous pseudo-labeled target data to the target domain.

3. Proposed Approach

3.1. Notations and Definitions

Denote by \( p_s(x_i) \) and \( p_t(x_i) \) the probability density functions of the source data and target data respectively, then each column of \( X_s \in \mathbb{R}^{d \times n} \) represents one sample drawn from \( p_s(x_i) \), each column of \( X_t \in \mathbb{R}^{d \times m} \) represents one sample drawn from \( p_t(x_i) \), where \( d \) is the feature length of the data instance, \( n_i \) and \( n_t \) indicate the number of samples in the source and target domain respectively. Regarding UDA, the labels of the whole source domain are available, i.e., \( D_t = \{ (x_i, y_i) \}_{i=1}^{n_t} \), and for the target domain \( D_s = \{ (x_i, y_i) \}_{i=1}^{n_s} \), labels are unavailable during the training phrase. Note that the label space \( Y_s = Y_t \) is assumed and each domain contains \( C \) identical classes, when the feature space \( X_i = X_s \) is also assumed, but \( p_s(x_i) \neq p_t(x_i) \), which is named dataset shift. Typically, a \( k \)-dimensional space is sought to align different domains in itself, an explicit feature mapping matrix \( W \in \mathbb{R}^{k \times d} \) is introduced for simplicity.

Throughout the paper, all the matrices are written as uppercase, and lowercase letters for vectors. For a matrix \( M \), its \( (i, j) \)-th element is denoted by \( m_{ij} \), and the transpose of its \( i \)-th row is denoted by \( m_i \). The transpose and trace of \( M \) are denoted by \( M^T \) and \( tr(M) \) respectively, the Frobenius norm of matrix \( M \) is denoted by \( \|M\|_F \), and the \( L_2 \)-norm of a vector \( v \) is denoted by \( \|v\|_2 \). \( I_n \) represents a full one vector of length \( n \), and \( I_n \) denotes an identity matrix. \( A = \text{diag}(b) \) is a diagonal matrix with its diagonal entry as \( a_{ii} = b_i \).
The proposed method PACET tackles unsupervised domain adaptation by mitigating the joint distributions across domains and preserving the discriminative information for both domains at the same time. The detailed objective function of each component in PACET is discussed as follows.

**Joint Distribution Discrepancy.** As suggested in previous works [19, 21], we first aim to reduce the distances of the marginal distributions \(p(x_i|y_i)\) and \(p(x_i)\) and \(p(y_i)\) across domains. Since these distribution differences are nontrivial, we follow [19, 15] to adopt the empirical squared MMD to measure them as

\[
\frac{1}{n} \sum_{i=1}^{n} W^T x_i - \frac{1}{n} \sum_{i=1}^{n} W^T x_i + C \sum_{c=1}^{C} \|W^T e_i^c - W^T e_i^c\|^2_2, \tag{2}
\]

where a total data matrix \(X = [X_s, X_t] \in \mathbb{P}^{n \times m}\) is defined for convenience, and \(n = n_s + n_t\). Besides, \(y = [y_1, \cdots, y_n, \cdots, y_n]\) is a class indicator variant, \(y_i \in \{1, \cdots, C\}\) is an integer vector and \(n_s, n_t, n_c\) are the numbers of source samples and target instances belonging to class \(c\) respectively. \(e_i^c\) and \(e_i^c\) respectively denote the source and target centers of the \(c\)-th class, i.e.,

\[
e_i^c = \sum_{j=1}^{n_c} n_s x_j 1(y_j = c)/n_c, \quad e_i^c = \sum_{j=1}^{n_c} n_t x_j 1(y_j = c)/n_c.
\]

The binary indicator function \(1(a, b)\) outputs 1 when \(a = b\) is true, 0 otherwise. That is to say, the overall means and class-wise means of the \(k\)-dimensional embedding of the source and target data are required to be as close as possible.

**Cross-Domain Local Structure.** Since the MMD based cross-domain discrepancy measures the distances between sets, we further introduce the aforementioned triplet-wise instance-to-center margins below to consider the cross-domain local structure,

\[
\sum_{c=1}^{C} \|W^T e_i^c - W^T e_i^c\|^2_2, \tag{3}
\]

where \(z_j\) denotes the closest negative class of target instance \(x_j\), which will be explained below. Intuitively, minimizing this margin will enforce two groups of different classes apart from each other locally. Note that, even though both consider the triplet-center loss, this term here is quite different from that of [45]. We use a simple \(l_2\) loss instead of the hinge loss in [45], which helps find a closed-form solution. Besides, the centers come from one identical modality for object retrieval in [45] while two centers in Eq. 3 come from different domains.

**Intra-class Variance and Total Variance.** Inspired by linear discriminant analysis (LDA), the within-domain intra-class variances are minimized to preserve the discriminative information while maximizing the total variance of both source and target instances. Obviously, such operations make the inter-class variance larger, where separate source/target classes are expected. Both variances are naturally defined below,

\[
\sum_{i=1}^{n_s} \|W^T x_i - W^T e_i^c\|^2_2 + \sum_{j=1}^{n_t} \|W^T x_j - W^T e_i^c\|^2_2,
\]

and

\[
\frac{1}{n} \sum_{i=1}^{n} W^T x_i - \frac{1}{n} \sum_{i=1}^{n} W^T x_i + \frac{1}{n} \sum_{j=1}^{n} \|W^T x_j - W^T e_i^c\|^2_2. \tag{4}
\]

To this end, we illustrate all four objective functions in Eq. 2 ~ Eq. 3 for domain-invariant projection learning. Note that all of them need to know the pseudo labels of target instances \([y_{n_s+1}, \cdots, y_n]\). In the following, we explain how to estimate them and \(z_j, j \in [n_s + 1, n]\) with previous learned projection \(W\). Actually, the pseudo target labels can be directly estimated from the source classifier, however, we expect more information from these conditional posterior probabilities. As explained in [19], it is quite hard to directly model \(p(y|x)\), thus we resort to explore the sufficient statistics of class-conditional distributions \(p(x|y)\) instead. Carefully examining the complete form of the conditional probabilities \(p(x|y)\), we find that

\[
p(x_i|y_u = c) = \frac{p(y_u = c|x_i)p(x_i)}{p(y_u = c)}, \quad u \in \{s, t\},
\]

are the class prior probabilities of being \(c\) in the source and target domains, respectively. We should mention that, all the source instances are labeled beforehand, hence both the conditional probability and the prior probability are easily obtained. Subsequently, the expectation of conditional probability \(p(x_i|y = c)\) is calculated as \(e_i^c = \sum_{j=1}^{n_c} x_j I(y_j = c)/n_c\).

**3.2. Objective Function**

Once the intra-class and total variance are calculated, we adopt the similar statistics of class-conditional distributions to fit the model. The detailed objective function of Eq. 2 is

\[
p(x_i|y_u = c) = \frac{p(y_u = c|x_i)p(x_i)}{p(y_u = c)}, \quad u \in \{s, t\},
\]

where \(p(y_u = c) = \sum_{j=1}^{n_c} p(y_u = c|x_j)p(x_j)\) are the class prior probabilities of being \(c\) in the source and target domains, respectively. We should mention that, all the source instances are labeled beforehand, hence both the conditional probability and the prior probability are easily obtained. Subsequently, the expectation of conditional probability \(p(x_i|y = c)\) is calculated as \(e_i^c = \sum_{j=1}^{n_c} x_j I(y_j = c)/n_c\).

**3.3. Confidence-weight Strategy**

To address these probabilities in the target domain, we further leverage the posterior probabilities \(p(y|x_i)\) to accurately estimate \(C\) target class means, i.e.,

\[
e_i^c = \sum_{j=1}^{n_c} x_j s(y_j, c),
\]

where \(s(y_j, c) = p(y_j = c|x_j)/\sum_{j} p(y_j = c|x_j)\),

while \(p(y_j = c|x_j)\) measures the posterior probability of target instance \(x_j\) belonging to the \(c\)-th class, and \(s(y_j, c)\) denotes the normalized weight of target instance \(x_j\) to class \(c\). These probability values can be easily obtained by logistic regression or other classifiers built on the annotated projected source instances. In this paper, we adopt the \(K\)-nearest neighbor classifier for simplicity, which owns only one parameter \(K\). The detailed posterior probability of projected instance \(x\) is described as \(p(y_j = c|x) = \sum_{q} I(q_j, c)/K\), where \(q\) denotes the indexes of \(K\) nearest neighbors of \(x\) in the previous projected source domain.

Last but not least, we need to estimate the pseudo label \(y_j\) and the closest negative class \(z_j\) for each target instance \(x_j\) via the posterior probability

\[
y_j = \arg\max_{1 \leq c \leq C} p(y = c|x_j), \quad z_j = \arg\max_{1 \leq c \leq C, c \neq y_j} p(y = c|x_j).
\]

**3.4. Reformulation and Optimization**

Following [19, 13], we first provide an equal and simple formulation for the optimization problem in Eq. 2, as
\( tr(W^T X M_0 X^T W) + \frac{1}{2} \sum tr(W^T X M_c X^T W) \) respectively, where \( M_0, M_c \in \mathbb{R}^{n \times n} \) are expressed as

\[
(M_0)_{ij} = \begin{cases} 
\frac{1}{n_p n_t}, & x_i, x_j \in D_t \\
\frac{1}{n_p n_t}, & x_i, x_j \in D_t \\
\frac{1}{n_p n_t}, & \text{otherwise.}
\end{cases}
\]

Here \( D_t \) represents the set of source samples belonging to class \( c \), and \( s(y, c) \) indicates the normalized weight of target instance \( x_j \) to class \( c \). For convenience, we introduce four auxiliary matrices \( Y_t \in \mathbb{R}^{n \times n}, Y_s \in \mathbb{R}^{n \times n}, Z_t \in \mathbb{R}^{n \times c}, \) all of them are one-hot encoding matrices of corresponding variables \( y, z, \) like \( (Y_t)_{ij} = \begin{cases} 
1, & y_{n, i} = j \\
0, & \text{otherwise.}
\end{cases} \)

\[ (Z_t)_{ij} = \begin{cases} 
1, & z_{n, i} = j \\
0, & \text{otherwise.}
\end{cases} \]

and \( S_t \in \mathbb{R}^{n \times c} \) being the target posterior probability matrix, \( (S_t)_{ij} = p(y_{n, i} = j | x_i) \). Subsequently, Eq. (3) can be rewritten as

\[
\|W^T X Y_t Y_t^{-1} Y_t^T - I_n\|_F^2 \leq \|W^T X Y_t Y_t^{-1} Z_t^T - I_n\|_F^2.
\]

Likewise, Eq. (4) is easily reformulated as \( tr(W^T X M_0 X^T W) \) by defining

\[
M_w = \begin{bmatrix} I_n - Y_t (Y_t^T Y_t)^{-1} Y_t^T & 0 \\
0 & I_n - S_t (S_t^T S_t)^{-1} S_t^T 
\end{bmatrix}.
\]

Note that, inspired by reverse prediction [49], the target within-class variance is alternated by introducing a latent center variable \( E_t \in \mathbb{R}^{n \times c} \) which minimizes \( \|E_t Y_t^T - X_t\|^2_F \) as much as possible. In this way, the larger posterior probability, the closer distance between target instance to its associating center will be enforced like [47]. Finally, we arrive at one equivalent but simplified optimization problem for PACET in the following,

\[
\min_{W} \Omega(W) = tr(W^T X M X^T W) + \lambda \|W\|_F^2, \ s.t. \ W^T X H X^T W = I_k.
\]

(9)

where \( H = I_n - 1_n 1_n^T \in \mathbb{R}^{n \times n} \) is the data centering matrix, \( M = M_0 + \frac{1}{2} \sum M_c + \alpha M_0 + \beta M_0 \), \( M_c = [Y_c Y_c^T]^{-1} Y_c^T; -I_n], [Y_c Y_c^T]^{-1} Y_c^T; -I_n], [Y_t Y_t^T]^{-1} Z_t^T; -I_n], [Y_t Y_t^T]^{-1} Z_t^T; -I_n], \alpha, \beta \) are two trade-off parameters, and \( \lambda \) is a hyper parameter (regularization parameter) to guarantee the matrix being nonsingular. Using Lagrange multiplier \( \Psi = diag(\phi_1, \cdots, \phi_k) \in \mathbb{R}^{k \times k} \), we can obtain the Lagrange function for problem (9)

\[
\mathcal{L}(W, \Psi) = tr(W^T X M X^T W + \lambda I_k) - tr(\Psi (W^T X H X^T W - I_k)).
\]

(10)

To seek the minimization of this objective, we set the partial derivative of \( \mathcal{L} \) w.r.t. \( W \) to zero, and obtain the following generalized eigenvalue problem

\[
(X M X^T + \lambda I_k) W = \Psi \cdot X H X^T W.
\]

(11)

Thus, the optimal state arrives when \( W \) corresponds to the \( k \) smallest eigenvectors of the above eigen-decomposition. Luckily, this solution is closed-form.

**Progressive learning scheme** Inspired by self-training [40, 48] and self-paced learning [25], we also develop a progressive framework that incrementally includes more target instances with the increase of iterations. Concretely, we rank the target instances from ‘easy’ to ‘hard’ via the maximum posterior probability, and pick certain numbers of ‘easy’ instances into the training process. Eventually, all the target instances are considered.

Once the projection \( W \) is learned, we immediately obtain a \( k \)-dimensional space, then we re-train a classifier (KNN in the experiment) using all the labeled source instances and predict the posterior probability (i.e., label proportions) of a target instance belonging to different classes. Finally, these three steps including feature projection learning, posterior probability inference, target instance selection are alternately carried out to constitute an EM-style algorithm, the pseudo-code is summarized in Algorithm 1.

**Algorithm 1 Progressive leArning with Confidence-wEighted Targets for Unsupervised Domain Adaptation**

**Input:** Source data \( \{X_s, Y_s\} \) and target data \( X_t \); trade-off parameters \( a = 1, \beta = 0.01, \# \text{maximum iteration} T = 15; \) neighborhood size \( K, \) subspace dimensionality \( k, \) tradeoff parameters \( \lambda, \) initial proportion \( p, \) proportion increasing rate \( \theta (\theta \leq 0.1) \).

**Output:** Projection matrix \( W; \) embedding \( R \in \mathbb{R}^{k \times n}; \) classifier \( f \).

1. Initialize \( \{M_c\}_{c=1}^C, M_0, M_k \) as 0 and construct \( M_o \) in Eq. (6);
2. while not converge and iter \( \leq T \) do
3. Solve the generalized eigen-decomposition problem in Eq. (11) for \( W \).
4. Obtain the \( k \)-dimensional embedding \( R = W^T X, X_t \).
5. Train a K-NN classifier on source instances \( \{(r_t, y_t)\}_{t=1}^{n_t} \).
6. Predict the targets’ posterior probability \( p(y_t = c | z_t) \).
7. Update matrices \( Y_t, Z_t, \) and arrange target instances.
8. Pick \( (p, * n_t) \) ‘easy’ target instances to constitute new \( X_t \).
9. Update \( p_t = min(p_t, * (1 + \theta, 1)) \).
10. Update the matrices \( \{M_c\}_{c=1}^C, M_t, M_k \).
11. end while
12. Re-train one 1-NN (or SVM) classifier \( f \) on \( \{(r_t, y_t)\}_{t=1}^{n_t} \).

3.5. Model Selection

Traditional cross-validation does not work for unsupervised domain adaptation since no labeled data in the target domain are available. In this paper, we adopt a reverse validation strategy that behaves similar as [49]. Detaieldly, given a group of parameters, we can train multiple models together with their predictions on unlabeled target data. Then we reverse the source domain and target domain, i.e., we consider the target data and their predictions as new source domain, and infer the labels of the originally labeled source data (new target domain) via the
same parameters, thus we can obtain the validation accuracy to determine the optimal parameters. The detailed flowchart of reverse validation is shown in Fig. 2.

3.6. Complexity Analysis

We now discuss the computational complexity of the proposed method in Algorithm 1. The projection step consists of constructing some matrices and solving the eigen-decomposition problem, respectively occupies $O(nC^2 + n^2d + nd^2)$ and $O(kd^2)$. On the other hand, the prediction step occupies $O(n^2n^t d)$. The ranking and picking step occupies $O(n^t \log n^t)$. Since $C \ll d$ and $k \ll n$, the overall complexity of PACET is $O(Tnd^2 + Tn^2d)$.

4. Experiments

We extensively compare our method with recent state-of-the-art unsupervised domain adaptation approaches on several real-world visual benchmark datasets.

4.1. Datasets

Office31 [50] contains 4,110 images of 31 categories of objects that provides three different domains, i.e., Amazon (A), DSLR (D), and Webcam (W). The Amazon subset contains product images on the web, while the DSLR and Webcam subsets capture objects with different quality-level cameras.

Office-Caltech [10] extends the Office31 dataset to include an auxiliary object dataset Caltech-256, and extracts 10 common categories shared by them.

PIE includes massive facial images of 68 people with various pose, illumination, and expression changes. We follow the setting in [31] instead of [19] by selecting 4 poses captured by cameras in 4 different positions, i.e., P1 (C05, left), P2 (C07, upward), P3 (C09, downward), and P4 (C29, right). All these facial images are normalized to the size $32 \times 32$.

4.2. Experiment Setting

For each dataset and each kind of features, we need to project them onto the unit ball (i.e., $l_2$ normalization) at first. Note that $A \rightarrow B$ indicates that $A$ is the source and $B$ is the target domain. Besides, we evaluate different methods in terms of the cross-domain instance-level accuracy (%).

Baseline methods. 1-NN and SVM are two the basic methods that are trained on the raw source data and infer the labels of target instances. We compare PACET with GFK [10], SA [3], JDA [19], CORAL [7], JGSA [20], Invariant Latent Space (ILS) [8], LDADA [51], DICE [21] and ATI [37], etc. For fair comparisons, a linear SVM classifier is also built by LIBLINEAR in the last step to further verify the effectiveness of learned embedding, dubbed PACET*. Note that, all these non-1-NN based methods are marked with * to distinguish them from 1-NN based methods. Once incorporated with deep features, PACET* is also compared with deep DA methods (e.g., Residual Transfer Networks (RTN) [52] and Joint Adaptation Networks (JAN) [32]).

Features & Protocols. For all datasets, we exploit the standard features kindly provided by previous works. For Office31, we use the AlexNet-FC7 features fine-tuned on the source domain in [7] and ResNet50 features. The DeCAF6 [20] and VGG-FC6,7 [8] as well as the shallow z-scored SURF features [50] are adopted for Office-Caltech. Besides, we adopt the 1,024 dimensional raw-pixel features provided in [19] for PIE. For each dataset, all the labeled source instances are utilized for training unless otherwise specified. For the Office-Caltech dataset, another popular protocol named ‘sampling protocol’ [7, 8, 10] is also adopted. Under this protocol, 20 instances per class for A, C, W and 8 instances per class for D are randomly selected for the source domain.

Implementation Details. Almost all the methods mentioned above have hyper-parameters, yet, it is impossible to conduct a standard cross-validation with no target labels being unavailable. Following [19], we evaluate all methods whose codes are provided.

https://www.csie.ntu.edu.tw/~cjlin/libsvm/
https://tvamvl.nl/research/domain-adaptation-2017/
kindly provided by empirically searching the parameter space for the optimal parameter settings, and report the best results of each method. For other methods, we directly adopt their reported results in the original papers that are assumed to be their best performance. For our method PACET, two free parameters \( \alpha, \beta \) are fixed leaving only three parameters \( \lambda, k, \) and hyper-parameters \( p_i, \theta \) tunable. Note that, we discard the progressive strategy and carry out reverse validation in Fig. 5 to obtain optimal parameters for \( \lambda, k, \) where \( \lambda \in [0.001, 0.01, 0.05, 0.1], \) and \( k \) ranges from \( C \) to \( 2C. \) The detailed values and the parameter sensitivity study are provided in the following section.

### 4.3. Comparison Results

#### Results on Office31

Table 1 tabulates the cross-domain recognition results of different UDA methods on the Office31 dataset. Among the counterparts with the AlexNet architecture, DICE and JAN-A are the state-of-the-art approaches for shallow and deep models, respectively. PACET ranks the highest and second highest in all six tasks, and it consistently outperforms DICE. Compared to deep DA methods, PACET is clearly superior and achieves similar accuracies to JAN-A. Specifically, PACET performs the best for small source to large target like D→A and W→A.

<table>
<thead>
<tr>
<th>Model</th>
<th>Feature</th>
<th>method</th>
<th>A→D</th>
<th>A→W</th>
<th>D→A</th>
<th>D→W</th>
<th>W→A</th>
<th>W→D</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>no</td>
<td>1-NN</td>
<td>59.4</td>
<td>57.5</td>
<td>47.2</td>
<td>96.1</td>
<td>44.8</td>
<td>99.0</td>
<td>67.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SVM†</td>
<td>59.2</td>
<td>57.9</td>
<td>48.8</td>
<td>95.2</td>
<td>46.5</td>
<td>98.8</td>
<td>67.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GFK-pls [10]</td>
<td>58.2</td>
<td>59.4</td>
<td>54.9</td>
<td>95.6</td>
<td>43.8</td>
<td>98.6</td>
<td>66.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SA [9]</td>
<td>61.0</td>
<td>59.5</td>
<td>46.9</td>
<td>95.1</td>
<td>46.6</td>
<td>98.2</td>
<td>67.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JDA [9]</td>
<td>66.5</td>
<td>68.8</td>
<td>56.7</td>
<td>97.7</td>
<td>53.5</td>
<td>99.6</td>
<td>73.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CORAL [7]</td>
<td>60.4</td>
<td>57.0</td>
<td>47.6</td>
<td>96.2</td>
<td>46.3</td>
<td>99.0</td>
<td>67.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JSAA [10]</td>
<td>67.5</td>
<td>62.3</td>
<td>55.6</td>
<td>98.1</td>
<td>52.0</td>
<td>99.8</td>
<td>72.5</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>ILS [8]</td>
<td>62.9</td>
<td>63.9</td>
<td>50.9</td>
<td>97.2</td>
<td>48.8</td>
<td>99.4</td>
<td>70.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ATY [52]</td>
<td>70.3</td>
<td>68.4</td>
<td>55.9</td>
<td>94.2</td>
<td>53.2</td>
<td>98.5</td>
<td>73.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LDADA [51]</td>
<td>66.7</td>
<td>71.4</td>
<td>56.5</td>
<td>96.9</td>
<td>58.8</td>
<td>99.8</td>
<td>75.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PACET (ours)</td>
<td>69.1</td>
<td>67.1</td>
<td>62.3</td>
<td>97.4</td>
<td>59.2</td>
<td>100.0</td>
<td>76.6</td>
</tr>
</tbody>
</table>

#### Results on PIE

Table 2 shows the results on the Office-Caltech benchmark dataset using the full protocol [52]. To the best of our knowledge, RTN-res [52] is a deep DA method that achieves the highest accuracy.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>A</th>
<th>A</th>
<th>A</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>D</th>
<th>D</th>
<th>W</th>
<th>W</th>
<th>W</th>
<th>W</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-NN</td>
<td>34.2</td>
<td>35.7</td>
<td>31.2</td>
<td>36.0</td>
<td>38.2</td>
<td>39.2</td>
<td>28.3</td>
<td>93.6</td>
<td>83.7</td>
<td>31.6</td>
<td>28.8</td>
<td>84.7</td>
<td>40.9</td>
<td></td>
</tr>
<tr>
<td>SVM†</td>
<td>42.7</td>
<td>39.5</td>
<td>39.0</td>
<td>53.0</td>
<td>42.0</td>
<td>45.1</td>
<td>33.7</td>
<td>30.4</td>
<td>30.2</td>
<td>36.2</td>
<td>34.4</td>
<td>86.6</td>
<td>47.1</td>
<td></td>
</tr>
<tr>
<td>GFK-pls [10]</td>
<td>43.6</td>
<td>44.6</td>
<td>45.1</td>
<td>51.6</td>
<td>43.3</td>
<td>44.1</td>
<td>39.1</td>
<td>31.7</td>
<td>85.4</td>
<td>31.8</td>
<td>34.3</td>
<td>87.9</td>
<td>48.5</td>
<td></td>
</tr>
<tr>
<td>SA [9]</td>
<td>44.3</td>
<td>36.3</td>
<td>38.3</td>
<td>54.8</td>
<td>45.2</td>
<td>44.2</td>
<td>39.4</td>
<td>34.3</td>
<td>85.3</td>
<td>36.3</td>
<td>33.2</td>
<td>84.3</td>
<td>47.9</td>
<td></td>
</tr>
<tr>
<td>JDA [9]</td>
<td>39.4</td>
<td>39.5</td>
<td>38.0</td>
<td>44.8</td>
<td>45.2</td>
<td>41.7</td>
<td>33.1</td>
<td>31.5</td>
<td>89.5</td>
<td>32.8</td>
<td>31.2</td>
<td>89.2</td>
<td>46.3</td>
<td></td>
</tr>
<tr>
<td>CORAL [4]</td>
<td>41.5</td>
<td>39.4</td>
<td>45.4</td>
<td>53.6</td>
<td>38.6</td>
<td>37.7</td>
<td>33.8</td>
<td>34.7</td>
<td>53.9</td>
<td>33.7</td>
<td>86.6</td>
<td>47.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JSAA [10]</td>
<td>41.5</td>
<td>47.1</td>
<td>45.8</td>
<td>51.5</td>
<td>45.9</td>
<td>45.4</td>
<td>38.0</td>
<td>29.9</td>
<td>91.1</td>
<td>39.9</td>
<td>33.2</td>
<td>90.5</td>
<td>50.0</td>
<td></td>
</tr>
<tr>
<td>ILS [8]</td>
<td>40.0</td>
<td>40.1</td>
<td>39.0</td>
<td>48.5</td>
<td>45.9</td>
<td>41.4</td>
<td>41.2</td>
<td>34.6</td>
<td>85.6</td>
<td>37.6</td>
<td>31.2</td>
<td>86.0</td>
<td>47.6</td>
<td></td>
</tr>
<tr>
<td>LDADA [50]</td>
<td>36.7</td>
<td>40.4</td>
<td>46.0</td>
<td>44.7</td>
<td>40.5</td>
<td>40.2</td>
<td>26.5</td>
<td>72.1</td>
<td>41.6</td>
<td>32.6</td>
<td>70.5</td>
<td>47.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PACET (ours)</td>
<td>42.7</td>
<td>49.7</td>
<td>52.2</td>
<td>50.2</td>
<td>51.0</td>
<td>48.1</td>
<td>41.1</td>
<td>33.7</td>
<td>84.1</td>
<td>37.5</td>
<td>37.8</td>
<td>87.3</td>
<td>51.3</td>
<td></td>
</tr>
</tbody>
</table>

Once equipped with ResNet50 features, PACET clearly and consistently beats JAN-A for all six cases, which demonstrates the capacity of our method. Compared to two recent state-of-the-art approaches SimNet and iCAN, PACET still achieves the best average accuracy. For different kinds of features, PACET† respectively outperforms the baseline SVM† by 13.0% and 11.1%, which again validates the effectiveness of our method. Regarding the parameters, we set \( \lambda = 0.1, k = 31, K = 10 \) according to the reverse validation.

Results on Office-Caltech. We investigate two protocols and four kinds of features in Table 2. Regarding the SURF features with full protocol in Table 2, PACET advances the state-of-the-art shallow approach DICE from 51.3 to 53.6 in terms of the average accuracy. Concretely, PACET† and PACET respectively rank the best or the second best results in 10 and 6 of 12 tasks, which significantly outperform LDADA†, DICE, and KWC†. Obviously, deep features strongly enhance the shallow models like JGSA, LDADA†, DICE† and PACET† in terms of the average accuracy. PACET† still obtains the highest average accuracy (93.5) when two previous highest reported methods are LSC† (92.6) and DICE† (92.4). Carefully accounting the underlined values, we find that RTN-res merely wins PACET† in 6 out of 12 cases, which further supports that our shallow PACET† method is still very competitive with RTN-res [52] that achieves the best performance on this dataset. For the free parameters, we set \( \lambda = 0.1, k = 20, K = 5 \).

On the other hand, we compare PACET with two state-of-the-art methods, ILS and PUnDA under the standard sampling protocol [10]. For almost all methods except 1-NN and SVM, VGG-FC6 features are more favorable, that is maybe because deeper layer provides discriminative features that are not suitable for adaptation. It can be easily seen that PACET and PACET† always significantly surpass other methods including JGSA, ILS, PUnDA, LDADA and DICE for both features. The improvements of PACET† over best reported results in [26] w.r.t. the average accuracy are 5.8% and 6.7%, respectively. Once changed to the sampling protocol, the improvement of PACET grows much larger, thus, we can draw a conclusion that our method performs much better especially when the source is relatively small. For the free parameters, we set \( \lambda = 0.1, K = 15, K = 5 \).

### Results on PIE

Table 2: Results on the Office-Caltech dataset with SURF (above) and DeCAF6 features using the full protocol [52]. To the best of our knowledge, RTN-res [52] is a deep DA method that achieves the highest accuracy.
and PACET* are two best-performing methods w.r.t. the average accuracy, which also rank within top two in both 8 out of 12 tasks, respectively. Among the counterparts, DICE [21] performs the best, which outperforms the second best DGA-DA by 5.6% w.r.t. the average accuracy. In addition, we find that GFK-pls, SA*, CORAL*, and ILS perform much worse in this dataset, which may indicate they are not suitable for raw-pixel features. For the free parameters, we set $\alpha = 0.01$, $k = 100$, $K = 5$. All the empirical comparisons results in Tables 1-4 indicate that the proposed method always achieves better recognition accuracy than its counterparts.

### 4.4. Subspace Visualization with t-SNE

In addition to the quantitative comparison results, we exploit one of the most popular visualization tool t-SNE [29] to compare subspace learned by different methods qualitatively. Here we choose three public methods JDA [19], JGSA [20] and DICE [21] for comparison. Observing Fig. 3, carefully, we can discover that the sources class (red colors) are always separable, however, PACET is the best performing method that pursues both inter-class separability and within-class compactness in the target domain. Besides, the target instances always lie in the same class area.

### 4.5. Ablation Study to investigate the effects of two proposed techniques (i.e., confidence-weight strategy and progressive learning scheme) to address uncertainties within target instances. Particularly, we introduce three different variants of PACET, i.e., ACET, PAT, and AT. Among them, ACET and PAT are the special cases when PACET neglects the progressive learning scheme and the confidence-weight strategy, respectively. AT is a baseline method where both two techniques are discarded. Some results are reproduced from [26].

Obviously, PACET is consistently superior to ACET that ignores the progressive learning strategy, which indicates the importance of such a self-learning technique. Besides, PACET outperforms PAT that neglects the confidence-weight strategy via posterior probability in 4 out of 5 tasks, which highlights that the proposed weighted measures are more accurate. In addition, PAT performs better than AT in all tasks, while PAT is worse than AT for PIE and ODD (below).

We can observe that some shallow features are not powerful enough to distinguish ‘easy’ and ‘hard’ samples. However, such drawback can be mitigated to some degree via using the confidence-weight strategy. While for DeCAF6 features, PACET is even inferior to PAT for the deep features here are quite discriminative, estimating the uncertainties via K-NN may be relatively inaccurate. Generally, both techniques are effective, and the confidence-weight strategy is more important than the progressive learning scheme.

<table>
<thead>
<tr>
<th>Dataset Feature</th>
<th>Office31 AlexNet</th>
<th>Office-Caltech SURF DeCAF6 VGG-Fc6</th>
<th>PIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PACET</td>
<td>76.61</td>
<td>53.57</td>
<td>92.28</td>
</tr>
<tr>
<td>ACET (w/o Progressive learning)</td>
<td>75.85</td>
<td>53.27</td>
<td>91.50</td>
</tr>
<tr>
<td>PAT (w/o Confidence wEight)</td>
<td>74.76</td>
<td>52.18</td>
<td>92.37</td>
</tr>
<tr>
<td>AT (w/o both)</td>
<td>74.21</td>
<td>52.44</td>
<td>90.89</td>
</tr>
</tbody>
</table>

Table 6: Average accuracy (%) of PACET on the Office-Caltech (DeCAF6) by using different parameters $\alpha$ (initial proportion) and $\beta$ (increasing rate).

$$
\begin{array}{cccccc}
\hline
\hline
\theta & 0.01 & 0.02 & 0.03 & 0.04 & 0.05 & 0.06 & 0.07 & 0.08 & 0.09 & 0.10 \\
\hline
0.65 & 88.92 & 90.39 & 91.61 & 91.15 & 91.59 & 91.63 & 92.40 & 92.40 & 92.36 & 92.54 \\
0.70 & 89.67 & 91.41 & 92.41 & 92.10 & 92.16 & 92.46 & 92.40 & 92.52 & 92.42 & 92.28 \\
0.75 & 91.52 & 92.44 & 92.25 & 92.3 & 92.3 & 92.13 & 91.83 & 91.92 & 91.89 & 91.87 \\
0.80 & 91.91 & 92.06 & 92.00 & 91.98 & 92.07 & 91.89 & 91.95 & 91.98 & 91.81 & 91.80 \\
0.85 & 92.36 & 92.22 & 92.06 & 91.64 & 91.89 & 91.91 & 92.01 & 92.00 & 92.07 & 91.75 \\
0.90 & 91.95 & 91.77 & 91.81 & 91.61 & 91.70 & 91.67 & 91.68 & 91.40 & 91.46 & 91.47 \\
0.95 & 91.64 & 91.44 & 91.44 & 91.44 & 91.43 & 91.50 & 91.50 & 91.50 & 91.50 & 91.50 \\
\hline
\hline
\end{array}
$$

The empirical comparisons results in Tables 1-4 are reproduced from [26].
(a) No adaptation
(b) JDA [19]
(c) JGSA [20]
(d) DICE [21]
(e) PACET (ours)

Figure 3: The t-SNE visualizations of baseline DA methods and our PACET on A→W on the Office-Caltech (DeCAF) dataset. (Best viewed in color.)

(a) accuracy w.r.t. parameter \( K \)
(b) accuracy w.r.t. # iterations

Figure 4: Average accuracy (%) on five datasets w.r.t. neighborhood size \( K \) and the number of iterations.

Then we investigate the sensitivity of parameters \( p_i, \theta \) that are involved into the progressive learning scheme in Algorithm. 1. Specifically, we set \( p_i \) in the range of \([0.65, 0.70, \ldots, 0.90, 0.95]\) and \( \theta \) in the range of \([0.01, 0.02, \ldots, 0.10]\), and other parameters are kept the same for fair comparisons. Carefully looking at Table 6, we observe that the average accuracies are relatively stable when setting \( p_i \) between 0.65 and 0.70 and \( \theta \) between 0.07 and 0.10. Besides, this table again validates the effectiveness of the developed progressive learning scheme, because the majority of accuracies outperforms that of ACET (w/o progressive learning) (i.e., 91.50). Actually, \( p_i = 0.7, \theta = 0.1 \) is a default choice in this paper.

In addition, as shown in Fig. 4, we exploit five various datasets to verify the sensitivity of parameter \( K \) and the experimental convergence analysis. It is easily observed that the accuracies tend to grow up and become steady when the neighbor size \( K \) becomes larger in Fig. 4(a) in most cases. Actually, \( K = 5 \) always achieves promising performance, making it being a default choice. Besides, we show the 1-NN accuracies of PACET within the iterative process in Fig. 4(b). As expected, the average recognition accuracies on all four datasets grow up.
quickly and gradually tend to stable after about 8 iterations.

5. Conclusion

In this paper, we have proposed a novel approach named Progressive leArning with Confidence-wEighted Targets (PACET) for unsupervised domain adaptation. Particularly, PACET well tackled the uncertainty in pseudo labels of the target domain from two aspects, progressive target instance selection and incorporating the learned class confidence scores to characterize both the within- and cross-domain relations. To increase the cross-domain inter-class separability, we further develop a new triplet-wise instance-to-center margin to compensate traditional pairwise empirical MMD. Further, an EM-style algorithm has been naturally developed to alternately learn feature transformation, estimate the target class confidence scores and select certain training target instances progressively. Extensive empirical results on three benchmark datasets have validated the superiority of PACET to recent state-of-the-art UDA methods. Particularly, PACET achieves the state-of-the-art accuracies and surpasses the recently reported best results on the most popular Office31 dataset.

Generally speaking, our method first considers the uncertainty in domain discrepancy for pseudo-label guided unsupervised domain adaptation, which can provide some useful insights to pseudo-label guided transfer learning methods. There are still, however, some aspects to be improved. Our method is somewhat heuristic and lacks some theoretical justification. It also highly relies on the closed set assumption where two domains share the same label space. Hence, in the future, we would like to discover some theoretical insights behind our method, and extend it from the closed set setting to some challenging settings like open-set and partial domain adaptation. Besides, we also plan to extend it by incorporating the deep learning technique to learn the domain-invariant feature representations.

Acknowledgement

The authors would like to greatly thank the editors and the reviewers for their valuable comments and advices. This work was funded by State Key Development Program (Grant No. 2016YFB1001001), and National Natural Science Foundation of China (Grant Nos. 61622310, 61473289).

References
