Exploring Uncertainty in Pseudo-label Guided Unsupervised Domain Adaptation

Jian Liang[1,2,3], Ran He[1,3,4], Zhenan Sun[1,3,4] and Tieniu Tan[1,3,4]

[1] *Center for Research on Intelligent Perception and Computing, National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences (CASIA)*
[2] *Department of ECE, National University of Singapore (NUS), Singapore*
[3] *University of Chinese Academy of Sciences (UCAS)*
[4] *Center for Excellence in Brain Science and Intelligence Technology, CAS*
liangjian92@gmail.com, {rhe,znsun,tnt}@nlpr.ia.ac.cn

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 [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 [2]. According to the availability of labeled target instances, domain adaptation methods can be divided into two main categories, unsupervised one [3, 4] and semi-supervised one [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) [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 [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 [14], which is rather hard to meet. In contrast, feature transformation methods [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) [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 [15, 16], JDA [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 [20, 21] or non-linear feature representations learning of deep methods [22]. Extensive efforts have been devoted to leveraging MMD to explicitly minimize the distances of the empirical expectations of different distributions

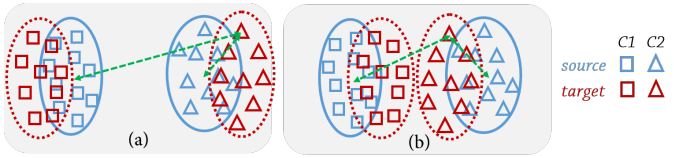


Figure 1: Toy example of the instance-to-center margins. (a) is more preferred than (b) since two target classes are more separable. (Green arrow: instance-to-center distance, best viewed in color.)

(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. Empirical 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,

$$\mathrm{MMD}^2[\mathcal{F}, p, q] = \sup_{\|f\|_{\mathcal{H}} \leq 1} \|\mathbb{E}_x[f(x)] - \mathbb{E}_y[f(y)]\|^2_{\mathcal{H}},$$

where the notations $E_x[f(x)] = E_{x\sim p}[f(x)]$ and $E_y[f(y)] = E_{y\sim q}[f(y)]$ 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], $\mathrm{MMD}^2[\mathcal{F}, p, q] = 0$ 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 $\mathbb{E}_x[f(x)]$ can be rewritten as $\mathbb{E}_x[f(x)] = \langle \mu[p], f \rangle$, where $\mu[p] = \mathbb{E}_{x\sim p}[\phi(x)]$ is the expectation of $\phi(x)$. Given observations $X := \{x_1, \cdots, x_m\}$ and $Y := \{y_1, \cdots, y_n\}$, drawn i.i.d. from the distributions p and q respectively, an empirical estimate of squared MMD [17] is further given

$$\widehat{\mathrm{MMD}}^2[\mathcal{F}, X, Y] = \|\frac{1}{m}\sum_{i=1}^{m}\phi(x_i) - \frac{1}{n}\sum_{j=1}^{n}\phi(y_j)\|^2_{\mathcal{H}}. \quad (1)$$

2.2. Feature Transformation based UDA Methods

We introduce several similar works [19, 15, 20, 16, 21, 28, 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 source within-class 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 minimum center shift strategy for weakly supervised source domain. DGA-DA [29] extends JDA by putting forward a repulsive force term, which drags different classes across domains far away from each other. Besides, JGSA [20] designs two geometrically close projections for the source and target domains, respectively. Besides the coupled projections learning, it also accounts for the source discriminative information preservation as well as total variance maximization.

Apart from data-centric transformation methods, subspace centric transformation methods [3, 7] directly align two domains rather than discover a new common space and recent studies [30, 31] in the literature exploit the optimal transport theory. Benefiting from the powerful representations of deep neural networks, recent deep DA methods [11, 32, 33, 34, 35] have achieved great successes. Among them, [11, 32] try to minimize the domain discrepancy while [33, 34] explore the adversarial objective to encourage domain confusion. Researchers further address some novel DA problems, e.g., DA with supervision from source classifiers [36], open-set DA [37]. Regardless of shallow and deep DA methods, the majority of them utilize MMD to measure the domain difference, however, none of them account for the weights of target instances, resulting into the degraded class means in the target domain.

Generally, our method is built on DICE [21] by adding one local repulsive term and considering the probabilistic divergences and variances as well as a curriculum learning framework. Specifically, the cross-class repulsive term differs from that of DGA-DA in considering the locality (pushing far away the most indistinguishable class rather than all other classes). To the best of our knowledge, our work is among the first attempts to incorporate the uncertainties of pseudo target labels to domain adaptation.

2.3. Semi-supervised Learning

In reality, UDA can be considered as a vanilla semi-supervised learning problem if the difference across domains vanishes. Thus, some popular techniques, e.g., self-training [38] and co-training [39], adopted in semi-supervised learning can be naturally applied to UDA. Typically, at each iteration, self-training (also called self-teaching) adds the most confident unlabeled points together with their predicted labels to the training set, and re-trains the classifier to obtain the new confidences until there are no unlabeled data left. Following this idea, [40] 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 $p_s(x_s)$ and $p_t(x_t)$ the probability density functions of the source data and target data respectively, then each column of $X_s \in \mathbb{R}^{d\times n_s}$ represents one sample drawn from $p_s(x_s)$, each column of $X_t \in \mathbb{R}^{d\times n_t}$ represents one sample drawn from $p_t(x_t)$, where d is the feature length of the data instance, n_s 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., $\mathcal{D}_s = \{(x_i, y_i)\}_{i=1}^{n_s}$, and for the target domain $\mathcal{D}_t = \{(x_j, y_j)\}_{j=1}^{n_t}$, labels are unavailable during the training phrase. Note that the label space $\mathcal{Y}_s = \mathcal{Y}_t$ is assumed and each domain contains C identical classes, when the feature space $\mathcal{X}_s = \mathcal{X}_t$ is also assumed, but $p_s(x_s) \neq p_t(x_t)$, which is named dataset shift. Typically, a k-dimensional space is sought to align different domains in itself, an explicit feature mapping matrix $W \in \mathbb{R}^{d\times k}$ is introduced for simplicity.

Through 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 dented 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 $\|v\|_2$. $\mathbf{1}_n$ represents a full one vector of length n, and I_n denotes an identity matrix. $A = \mathrm{diag}(b)$ is a diagonal matrix with its diagonal entry as $a_{i,i} = b_i$.

3.2. Objective Function

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 ($p_s(x_s)$ and $p_t(x_t)$) and C conditional distributions ($p(x_s|y_s = c)$ and $p(x_t|y_t = c)$, where $c \in [1, C]$) across domains. Since these distribution differences are non-trivial, we follow [19, 15] to adopt the empirical squared MMD to measure them as

$$\|\frac{1}{n_s}\sum_{i=1}^{n_s} W^T x_i - \frac{1}{n_t}\sum_{j=n_s+1}^{n} W^T x_j\|_2^2 + \frac{1}{C}\sum_{c=1}^{C}\|W^T e_c^s - W^T e_c^t\|_2^2, \quad (2)$$

where a total data matrix $X = [X_s, X_t] \in \mathbb{R}^{d\times n}$ is defined for convenience, and $n = n_s + n_t$. Besides, $y = [y_1, \cdots, y_{n_s}, \cdots, y_n]$ is a class indicator variant, $y_i \in \{1, \cdots, C\}$ is an integer variable, and n_s^c and n_t^c are the numbers of source samples and target instances belonging to class c respectively. e_c^s and e_c^t respectively denote the source and target centers of the c-th class, i.e., $e_c^s = \sum_{i=1}^{n_s} x_i \mathbf{1}(y_i, c)/n_s^c$, $e_c^t = \sum_{j=n_s+1}^{n} x_j \mathbf{1}(y_j, c)/n_t^c$. The binary indicator function $\mathbf{1}(a, b)$ 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,

$$\sum_{j=n_s+1}^{n} \|W^T x_j - W^T e_{y_j}^s\|_2^2 - \|W^T x_j - W^T e_{z_j}^s\|_2^2, \quad (3)$$

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,

$$\sum_{i=1}^{n_s}\left\|W^T x_i - W^T e_{y_i}^s\right\|_2^2 + \sum_{j=n_s+1}^{n}\left\|W^T x_j - W^T e_{y_i}^t\right\|_2^2,$$

$$\text{and} \quad \sum_{i=1}^{n}\left\|W^T x_i - \frac{1}{n}W^T\sum_{q=1}^{n} x_q\right\|_2^2. \quad (4)$$

To this end, we illustrate all four objective functions in Eq. (2) ~ Eq. (4) 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 $p(x_i|y)$, we find that

$$p(x_u|y_u = c) = \frac{p(y_u = c|x_u)p(x_u)}{p(y_u = c)},\ u \in \{s, t\},$$
$$\text{where}\ \ p(y_u = c) = \sum\nolimits_{x_u} p(y_u = c|x_u)p(x_u)$$

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 $p(x_s|y_s = c)$ is calculated as $e_c^s = \sum_{i=1}^{n_s} x\mathbf{1}(y_i, c)/n_s^c$.

3.3. Confidence-weight Strategy

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

$$e_c^t = \sum\nolimits_{j=n_s+1}^{n} x_j \mathbf{s}(y_j, c),$$
$$\text{where}\ \mathbf{s}(y_j, c) = p(y_j = c|x_j)/\sum\nolimits_j p(y_j = c|x_j), \quad (5)$$

where $p(y_j = c|x_j)$ measures the posterior probability of target instance x_j belonging to the c-th class, and $\mathbf{s}(y_j, c)$ 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 $p(y_j = c|x) = \sum_q \mathbf{1}(y_q, c)/K$, 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

$$y_j = \arg\max_{1\le c\le C}\ p(y = c|x_j),\ z_j = \arg\max_{1\le c\le C, c\neq y_j}\ p(y = c|x_j).$$

3.4. Reformulation and Optimization

Following [19, 15], we first provide an equal and simple formulation for the optimization problem in Eq. (2) as

$tr(W^T X M_0 X^T W) + \frac{1}{C}\sum_c tr(W^T X M_c X^T W)$ respectively, where $M_0, M_c \in \mathbb{R}^{n\times n}$ are expressed as

$$(M_0)_{ij} = \begin{cases} \frac{1}{n_s n_s}, & x_i, x_j \in \mathcal{D}_s \\ \frac{1}{n_t n_t}, & x_i, x_j \in \mathcal{D}_t \\ \frac{-1}{n_s n_t}, & \text{otherwise.} \end{cases}, (M_c)_{ij} = \begin{cases} \frac{1}{n_s^c n_s^c}, & x_i, x_j \in \mathcal{D}_s^c \\ \mathbf{s}(y_i, c)\cdot\mathbf{s}(y_j, c), & x_i, x_j \in \mathcal{D}_t \\ \frac{-\mathbf{s}(y_i, c)}{n_s^c}, & x_i \in \mathcal{D}_s^c, x_j \in \mathcal{D}_t \\ \frac{-\mathbf{s}(y_j, c)}{n_s^c}, & x_j \in \mathcal{D}_s^c, x_i \in \mathcal{D}_t \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Here $\mathcal{D}_s^c$ represents the set of source samples belonging to class c, and $\mathbf{s}(y_j, c)$ indicates the normalized weight of target instance x_j to class c. For convenience, we introduce four auxiliary matrices $Y_s \in \mathbb{R}^{n_s\times C}$, $\hat{Y}_t \in \mathbb{R}^{n_t\times C}$, $Z_t \in \mathbb{R}^{n_t\times C}$, all of them are one-hot encoding matrices of corresponding variables y, z, like

$$(\hat{Y}_t)_{ij} = \begin{cases} 1, & y_{n_s+i} = j \\ 0, & \text{otherwise.} \end{cases}, (Z_t)_{ij} = \begin{cases} 1, & z_{n_s+i} = j \\ 0, & \text{otherwise.} \end{cases}, \tag{7}$$

and $S_t \in \mathbb{R}^{n_t\times C}$ being the target posterior probability matrix, $(S_t)_{i,j} = p(y_{n_s+i} = j | x_{n_s+i})$. Subsequently, Eq. (3) can be rewritten as

$$\|W^T X[Y_s(Y_s^T Y_s)^{-1}\hat{Y}_t^T; -I_{n_t}]\|_F^2 - \|W^T X[Y_s(Y_s^T Y_s)^{-1}Z_t^T; -I_{n_t}]\|_F^2. \tag{8}$$

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

$$M_w = \begin{bmatrix} I_{n_s} - Y_s(Y_s^T Y_s)^{-1}Y_s^T & \mathbf{0} \\ \mathbf{0} & I_{n_t} - S_t(S_t^T S_t)^{-1}S_t^T \end{bmatrix}.$$

Note that, inspired by reverse prediction [46], the target within-class variance is alternated by introducing a latent center variable $E_t \in \mathbb{R}^{d\times C}$ which minimizes $\|E_t Y_t^T - X_t\|_F^2$ 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,

$$\min_W \Omega(W) = tr(W^T X \widetilde{M} X^T W) + \lambda\|W\|_F^2, \ s.t.\ W^T X H X^T W = I_k, \tag{9}$$

where $H = I_n - \mathbf{1}_n\mathbf{1}_n^T \in \mathbb{R}^{n\times n}$ is the data centering matrix, $\widetilde{M} = M_0 + \frac{1}{C}\sum_c M_c + \alpha M_i + \beta M_w$, $M_i = [Y_s(Y_s^T Y_s)^{-1}\hat{Y}_t^T; -I_{n_t}][Y_s(Y_s^T Y_s)^{-1}\hat{Y}_t^T; -I_{n_t}]^T - [Y_s(Y_s^T Y_s)^{-1}Z_t^T; -I_{n_t}][Y_s(Y_s^T Y_s)^{-1}Z_t^T; -I_{n_t}]^T$, α, β are two trade-off parameters, and λ is a hyper parameter (regularization parameter) to guarantee the matrix being nonsingular. Using Lagrange multiplier $\Psi = \text{diag}(\psi_1, \cdots, \psi_k) \in \mathbb{R}^{k\times k}$, we can obtain the Lagrange function for problem (9)

$$\mathcal{L}(W, \Psi) = tr\left(W^T\left(X\widetilde{M}X^T + \lambda I_k\right)W\right) - tr\left(\Psi\left(W^T X H X^T W - I_k\right)\right). \tag{10}$$

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,

$$\left(X\widetilde{M}X^T + \lambda I_k\right)W = \Psi \cdot X H X^T W. \tag{11}$$

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 $\alpha = 1$, $\beta = 0.01$, # maximum iteration $T = 15$; neighborhood size K, subspace dimensionality k, tradeoff parameters λ, initial proportion p_i, proportion increasing rate θ ($\theta \leq 0.1$).

Output: Projection matrix W; embedding $R \in \mathbf{R}^{k\times n}$; classifier f.

```
Initialize {M_c}_{c=1}^C, M_i, M_w as 0 and construct M_o in Eq. (6);
while not converge and iter ≤ T do
    Solve the generalized eigen-decomposition problem in Eq. (11) for W.
    Obtain the k-dimensional embedding R = W^T[X_s, X_t].
    Train a K-NN classifier on source instances {(r_i, y_i)}_{i=1}^{n_s}.
    Predict the targets' posterior probability p(y_j = c|z_j).
    Update matrices Ŷ_t, Z_t, and arrange target instances.
    Pick (p_i * n_t) 'easy' target instances to constitute new X_t.
    Update p_i = min(p_i * (1 + θ), 1).
    Update the matrices {M_c}_{c=1}^C, M_i, M_w.
end while
Re-train one 1-NN (or SVM) classifier f on {(r_i, y_i)}_{i=1}^{n_s}.
```

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]. Detailedly, 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

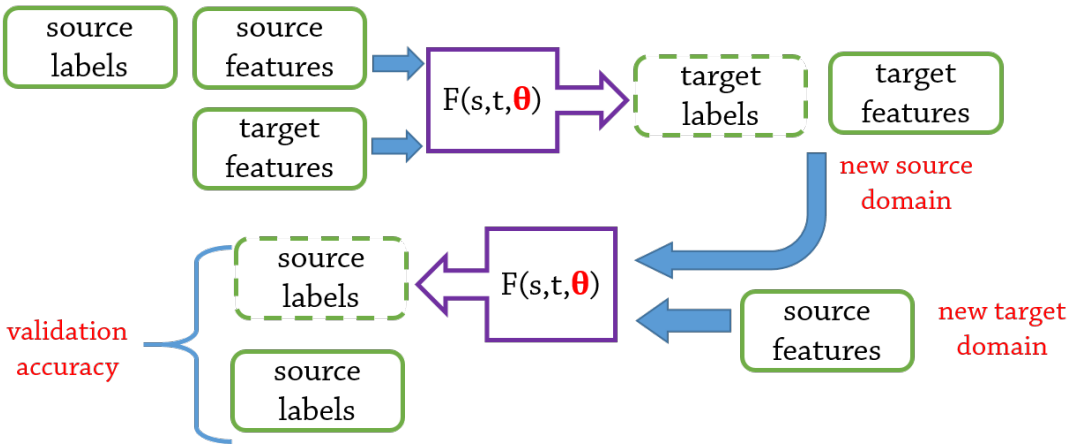


Figure 2: Toy example of reverse validation for UDA. $F(s,t,\theta)$ denotes the model where s is the labeled source domain, t is the unlabeled target domain, and θ represents model parameter, best viewed in color.

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 $O(n_s n_t d)$. 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 32×32.

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 term 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[1] 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-FC_7 features fine-tuned on the source domain in [7] and ResNet50 features[2]. The $DeCAF_6$ [20] and VGG-$FC_{6,7}$ [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

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvm/
[2] https://twanvl.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 α, β are fixed leaving only three parameters λ, k, K and hyper-parameters p_i, θ tunable. Note that, we discard the progressive strategy and carry out reverse validation in Fig. 2 to obtain optimal parameters for λ, 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. Specially, PACET performs the best for small source to large target like D→A and W→A.

Table 1: Results on the Office31 dataset with AlexNet-FC$_7$ and ResNet50 features using the full protocol [32]. The best (in bold red), the second best (in bold blue). [‡ Yes/ no in the column 'feature' denotes with/ without feature learning, respectively.]

model	feature‡	method	A→D	A→W	D→A	D→W	W→A	W→D	Avg.
AlexNet	no	1-NN	59.4	57.5	47.2	96.1	44.8	99.0	67.3
		SVM*	59.2	57.9	48.8	95.2	46.5	98.8	67.7
		GFK-pls [10]	58.2	59.4	45.9	95.6	43.8	98.6	66.9
		SA [3]	61.0	59.5	46.9	95.1	46.6	98.2	67.9
		JDA [19]	66.5	68.8	56.3	97.7	53.5	99.6	73.7
		CORAL* [7]	60.4	57.0	47.6	96.2	46.3	99.0	67.8
		JGSA [20]	67.5	62.3	55.6	**98.1**	52.0	99.8	72.5
		ILS [8]	62.9	63.9	50.0	97.2	48.8	99.4	70.4
		ATI* [37]	**70.3**	68.7	55.3	95.0	56.9	98.7	74.2
		LDADA* [51]	**70.3**	68.8	55.9	94.2	53.2	98.5	73.5
		DICE [21]	66.7	**71.4**	**56.5**	96.9	**58.6**	**99.8**	**75.0**
		PACET (ours)	**69.1**	**71.7**	**62.3**	**97.4**	**59.2**	**100.0**	**76.6**
	yes	RevGrad [53]	72.3	73.0	53.4	96.4	51.2	99.2	74.3
		DRCN [54]	66.8	68.7	56.0	96.4	54.9	99.0	73.6
		RTN-res [52]	71.0	73.3	50.5	**96.8**	51.0	**99.6**	73.7
		JAN-A [32]	**72.8**	**75.2**	**57.5**	**96.6**	**56.3**	**99.6**	**76.3**
		I2I-Adapt [55]	71.1	**75.3**	50.1	96.5	52.1	99.6	74.1
	no	PACET*(ours)	70.3	72.2	**61.8**	96.0	**59.0**	**99.4**	**76.5**
ResNet50	yes	RevGrad [53]	79.7	82.0	68.2	96.9	67.4	99.1	82.2
		RTN-res [52]	77.5	84.5	66.2	96.8	64.8	99.4	81.6
		JAN-A [32]	85.1	86.0	69.2	96.7	70.7	99.7	84.6
		SimNet [56]	85.3	88.6	**73.4**	**98.2**	**71.8**	99.7	86.2
		iCAN [57]	**90.1**	**92.5**	72.1	**98.2**	69.9	**100.0**	**87.2**
	no	SVM*	77.5	73.8	61.6	96.4	63.2	**99.8**	78.7
		PACET*(ours)	**90.8**	**89.1**	**73.5**	**97.6**	**73.6**	**99.8**	**87.4**

Once equipped with ResNet50 features, PACET clearly and consistently beats JAN-A for all six cases, which demonstrate the potential 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 verifies 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 this dataset in Tables 2~3. 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 term 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-FC$_6$ 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. Observing the results in Table 4, PACET

Table 2: Results on the Office-Caltech dataset with SURF (above) and DeCAF$_6$ features using the full protocol [9, 52]. To the best of our knowledge, RTN-res [52] is a deep DA method that achieves the highest accuracy.

Source / Target	A / C	A / D	A / W	C / A	C / D	C / W	D / A	D / C	D / W	W / A	W / C	W / D	Avg.
1-NN	34.2	35.7	31.2	36.0	38.2	29.2	28.3	29.6	83.7	31.6	28.8	84.7	40.9
SVM*	42.7	39.5	39.0	53.0	42.0	45.1	33.7	30.4	82.0	36.2	34.4	86.6	47.1
GFK-pls [10]	43.6	44.6	45.1	51.6	43.3	44.1	39.1	31.7	85.4	31.8	34.3	87.9	48.5
SA* [3]	44.3	36.3	38.3	**54.8**	45.2	44.4	39.4	34.3	85.1	36.3	33.2	83.4	47.9
JDA [19]	39.4	39.5	38.0	44.8	45.2	41.7	33.1	31.5	89.5	32.8	31.2	89.2	46.3
CORAL* [7]	45.1	39.5	44.4	54.3	36.3	38.6	37.7	33.8	84.7	35.9	33.7	86.6	47.6
JGSA [20]	41.5	47.1	45.8	51.5	45.9	45.4	38.0	29.9	**91.9**	39.9	33.2	90.5	50.0
ILS [8]	40.0	40.1	39.0	48.5	45.9	41.4	41.2	34.6	85.8	37.6	31.2	86.0	47.6
LDADA* [51]	36.7	40.4	46.0	54.6	47.0	**60.5**	**42.0**	26.5	72.1	**41.6**	32.6	70.5	47.5
DICE [21]	42.7	**49.7**	52.2	50.2	51.0	48.1	41.1	33.7	84.1	37.5	37.8	87.3	51.3
KWC* [58]	**46.0**	47.8	41.7	53.2	49.7	47.8	**41.7**	**39.4**	85.8	39.0	36.4	82.8	50.9
PACET (ours)	44.3	**50.3**	**53.2**	52.2	**52.2**	51.5	40.8	34.5	**91.5**	40.8	**39.0**	**92.4**	**53.6**
PACET*(ours)	**45.9**	**49.7**	**52.5**	**57.5**	**54.8**	**53.9**	40.3	**35.7**	87.8	**41.8**	**40.1**	**91.1**	**54.2**
SVM*	85.0	87.9	79.0	91.4	89.8	80.0	87.1	78.8	98.6	75.7	72.0	**99.4**	85.4
LSC* [59]	87.9	**94.9**	88.8	**94.3**	95.3	91.2	92.4	86.2	98.6	**93.3**	**88.0**	**100.0**	**92.6**
JDOT* [30]	85.2	87.9	84.8	91.5	89.8	88.8	88.1	84.3	96.6	90.7	82.6	98.1	89.0
JGSA [20]	84.9	88.5	81.0	91.4	93.6	86.8	92.0	86.2	**99.7**	90.7	85.0	**100.0**	90.0
ATI* [37]	86.5	**92.8**	88.7	93.8	89.6	93.6	**93.4**	85.9	98.9	**93.6**	86.3	**100.0**	91.9
LDADA* [51]	**88.0**	91.5	**93.2**	92.2	**96.1**	90.9	**92.7**	87.0	94.2	92.3	**87.9**	97.5	92.0
DICE* [21]	87.6	89.8	88.5	93.0	**96.2**	**95.3**	92.5	**88.3**	**99.0**	91.8	86.9	**100.0**	92.4
KWC* [58]	87.3	87.3	84.8	93.5	91.1	86.4	89.8	85.2	97.3	89.0	83.2	**99.4**	89.5
PACET*(ours)	**88.3**	**94.9**	**92.2**	**93.6**	**96.2**	**95.6**	**92.7**	**89.3**	**99.0**	92.9	86.9	**100.0**	**93.5**
RTN-res [52]	88.1	95.5	95.2	93.7	94.2	96.9	93.8	84.6	99.2	92.5	86.6	100.0	93.4

Table 3: Results on the Office-Caltech dataset with VGG-FC$_{6,7}$ (above/ below) via sampling protocol [10]. Some results are reproduced from [26].

Source	A	A	A	C	C	C	D	D	D	W	W	W	Avg.
Target	C	D	W	A	D	W	A	C	W	A	C	D	
1-NN	70.1	52.3	60.9	81.9	55.6	65.9	57.0	48.0	86.7	66.4	60.2	91.3	66.4
SVM*	74.2	51.7	63.1	86.7	61.5	74.8	58.7	55.5	91.8	73.3	68.2	94.2	71.1
GFK-pls [10]	77.7	63.5	74.1	86.2	66.5	76.5	69.9	64.0	92.4	81.1	73.5	96.6	76.8
SA [3]	77.1	64.9	76.0	83.9	66.2	76.0	69.0	62.3	90.5	80.2	71.9	94.2	76.0
CORAL* [7]	79.0	67.1	74.8	89.4	67.6	77.6	75.8	64.7	94.6	82.3	75.9	96.0	78.7
JGSA [20]	79.9	71.7	82.6	90.2	76.4	84.5	82.7	73.5	**95.4**	**91.6**	79.0	96.3	83.6
ILS [8]	78.9	72.5	82.4	87.6	73.0	84.4	79.2	66.5	94.2	87.2	79.9	89.3	81.3
PUnDA* [26]	82.3	**76.2**	82.7	90.3	76.2	88.3	83.1	69.2	93.4	86.9	82.6	89.8	83.4
LDADA* [51]	81.6	69.5	81.1	90.2	70.3	83.9	78.0	60.6	89.3	90.2	82.0	89.2	80.5
DICE [21]	83.0	66.4	75.9	**91.9**	67.4	83.7	84.4	78.6	94.8	90.3	80.7	93.8	82.6
PACET (ours)	**85.1**	72.7	**85.6**	90.7	**79.7**	**89.7**	**89.7**	**80.6**	**94.8**	90.7	**84.0**	**97.0**	**86.7**
PACET*(ours)	**87.8**	**73.4**	**86.7**	**92.4**	**81.0**	**92.0**	**90.8**	**83.7**	94.7	**91.5**	**86.5**	**97.6**	**88.2**
1-NN	72.6	50.8	64.0	82.6	54.9	65.3	61.2	52.8	88.2	67.8	64.2	88.8	67.8
SVM*	76.2	51.8	68.0	86.7	61.3	74.8	58.7	56.0	91.2	74.6	70.6	93.0	71.9
GFK-pls [10]	76.6	57.6	74.0	84.1	63.4	73.6	67.5	62.9	91.9	76.0	69.5	92.9	74.2
SA [3]	76.2	60.7	75.0	82.6	63.2	73.6	66.0	59.4	89.5	76.4	69.0	94.0	73.8
CORAL* [7]	78.6	61.3	71.8	88.6	63.8	76.0	71.2	63.0	93.5	82.0	73.7	94.6	76.5
JGSA [20]	81.1	72.3	81.4	88.3	72.3	82.5	78.9	72.3	93.6	**89.8**	79.8	**95.8**	82.3
ILS [8]	78.4	71.3	80.9	87.1	67.1	80.1	76.5	66.2	91.8	86.7	76.3	88.2	79.2
PUnDA* [26]	81.0	**75.8**	81.4	91.1	70.8	83.8	80.4	69.1	92.0	85.7	80.1	90.1	81.7
LDADA* [51]	82.5	71.4	86.3	91.2	68.8	85.7	74.5	58.5	88.7	87.8	79.8	86.9	80.2
DICE [21]	83.7	62.9	79.3	**91.7**	63.8	84.3	82.3	76.4	**94.2**	89.4	**82.1**	91.0	81.7
PACET (ours)	**83.9**	73.9	**85.1**	89.8	**79.2**	**88.1**	**83.1**	**76.8**	93.8	88.8	81.5	94.0	**84.8**
PACET*(ours)	**86.9**	**75.4**	**86.8**	**92.3**	**80.9**	**91.5**	**85.8**	**81.1**	**94.3**	**90.9**	**85.6**	**94.7**	**87.2**

Table 4: Results on the PIE dataset. Some results are reproduced from [21].

Source	P1	P1	P1	P2	P2	P2	P3	P3	P3	P4	P4	P4	Avg.
Target	P2	P3	P4	P1	P3	P4	P1	P2	P4	P1	P2	P3	
1-NN	26.1	26.6	16.7	24.5	46.6	26.5	21.4	41.0	26.2	18.5	24.2	28.3	27.2
SVM*	30.9	33.9	23.8	31.8	41.0	28.8	32.3	39.7	37.7	29.4	33.1	40.6	33.6
GFK-pls [10]	29.4	28.3	21.8	30.0	45.5	27.7	26.9	41.4	31.7	31.4	28.2	34.4	31.4
SA* [3]	32.8	34.5	22.5	27.7	37.3	27.1	29.1	37.0	30.5	34.5	30.9	31.9	31.3
JDA [19]	73.1	69.3	55.1	73.8	**74.9**	61.5	69.0	74.5	60.4	59.6	67.5	69.5	67.3
CORAL* [7]	31.8	31.9	19.9	26.6	35.0	25.9	25.1	36.5	26.0	32.0	30.4	32.6	29.5
RTML [60]	60.1	55.2	53.0	58.1	63.9	40.4	53.1	58.7	42.1	29.1	33.3	39.9	48.9
OTGL [31]	59.4	58.7	48.4	61.9	64.4	52.7	57.9	64.7	52.8	45.7	51.3	52.6	55.9
JGSA [20]	62.2	60.0	45.1	68.2	64.9	52.3	62.9	60.3	51.2	53.5	57.5	54.3	57.7
ILS [8]	37.8	35.2	21.8	40.4	31.8	25.5	29.1	31.9	18.0	25.7	21.4	31.7	29.2
DGA-DA* [29]	76.4	72.5	60.8	77.0	**77.5**	63.6	**80.8**	72.2	64.5	67.7	65.4	71.6	70.8
DICD [61]	73.0	72.0	**66.9**	69.9	65.9	48.7	69.4	65.4	61.4	62.9	57.0	65.9	64.9
DICE [21]	**84.1**	77.9	**66.5**	81.3	74.0	68.8	78.8	76.7	**70.8**	73.8	**71.2**	74.1	74.8
PACET (ours)	**82.2**	**80.8**	64.5	**82.9**	73.5	**72.4**	79.7	**79.3**	**70.2**	**76.2**	69.2	**79.2**	**75.8**
PACET*(ours)	80.9	**79.0**	65.9	**83.7**	71.9	**73.5**	**80.0**	**78.1**	69.7	**77.6**	**70.5**	**80.5**	**75.9**

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 $\lambda = 0.01, k = 100, K = 5$. All the empirical comparisons results in Tables 1~4 indicate that the proposed method always achieves better recognition accuracies 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[3] to compare subspaces 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 source classes (red color) 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 source class area.

4.5. Ablation Study, Parameter Sensitivity and Convergence Analysis

We perform an *ablation study* to investigate the effectiveness of two proposed techniques (i.e., confidence-weight strategy and progressive learning scheme) to addressing 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. Their average accuracies on several datasets are shown in Table 5.

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 relations are more accurate. In addition, ACET performs better than AT in all the tasks, while PAT is worse than AT for PIE and Office-Caltech with SURF features, that may be because such 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 DeCAF$_6$ features, PACET is even inferior to PAT because 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.

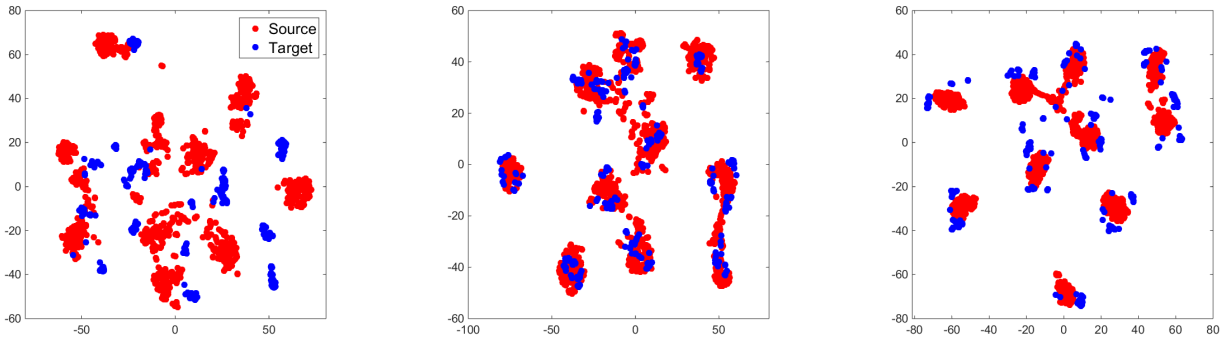
Table 5: The average accuracies (%) of different methods (ablation study).

Dataset	Office31	Office-Caltech			PIE
Feature	AlexNet	SURF	DeCAF$_6$	VGG-FC$_6$	
PACET	**76.61**	**53.57**	**92.28**	**86.70**	**75.83**
ACET (w/o Progressive learning)	**75.85**	**53.27**	91.50	**85.91**	**74.86**
PAT (w/o Confidence wEight)	74.77	52.18	**92.37**	85.85	73.58
AT (w/o both)	74.46	52.44	90.89	83.77	73.80

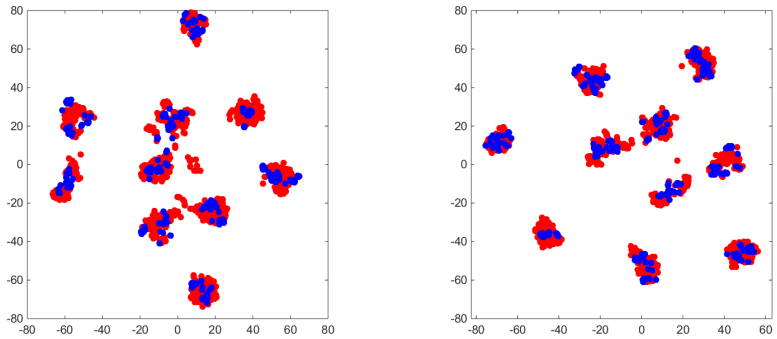
Table 6: Average accuracy (%) of PACET on the Office-Caltech (DeCAF$_6$) by using different parameters p_i (initial proportion) and θ (increasing rate).

		θ									
		0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.10
p_i	0.65	88.92	90.39	**91.61**	91.15	**91.59**	**91.63**	**92.40**	**92.36**	**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**	**92.33**	**92.32**	**92.13**	**91.83**	**91.92**	**91.89**	**91.87**
	0.80	**91.91**	**92.06**	**92.00**	**91.98**	**92.09**	**91.89**	**91.55**	**91.88**	**91.81**	**91.80**
	0.85	**92.30**	**92.22**	**92.06**	**91.64**	**91.89**	**91.81**	**92.01**	**91.93**	**91.89**	**91.75**
	0.90	**91.95**	**91.77**	**91.81**	**91.61**	**91.70**	**91.67**	**91.69**	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

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

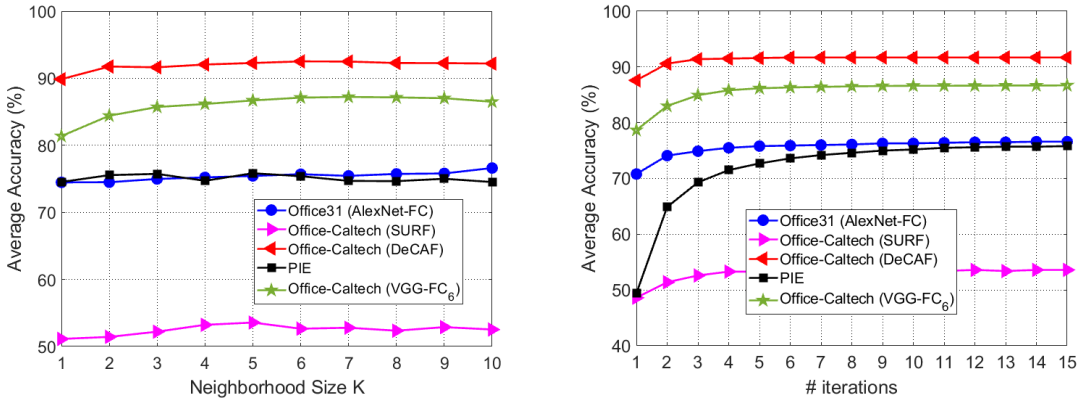


(a) No adaptation (b) JDA [19] (c) JGSA [20]



(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 ($DeCAF_6$) 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, θ that are involved into the progressive learning scheme in Algorithm. 1. Specifically, we set p_i in the range of $\{0.65, 0.70, \cdots, 0.90, 0.95\}$ and θ in the range of $\{0.01, 0.02, \cdots, 0.10\}$, 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 θ 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

[1] G. Csurka, Domain adaptation for visual applications: A comprehensive survey, arXiv preprint arXiv:1702.05374, 2017.

[2] V. M. Patel, R. Gopalan, R. Li, R. Chellappa, Visual domain adaptation: A survey of recent advances, IEEE Signal Processing Magazine 32 (3) (2015) 53–69.

[3] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Unsupervised visual domain adaptation using subspace alignment, in: IEEE Conference on Computer Vision, 2013, pp. 2960–2967.

[4] B. Yang, A. J. Ma, P. C. Yuen, Learning domain-shared group-sparse representation for unsupervised domain adaptation, Pattern Recognition 81 (2018) 615–632.

[5] W. Wang, H. Wang, Z. Zhang, C. Zhang, Y. Gao, Semi-supervised domain adaptation via fredholm integral based kernel methods, Pattern Recognition 85 (2019) 185–197.

[6] L. A. Pereira, R. da Silva Torres, Semi-supervised transfer subspace for domain adaptation, Pattern Recognition 75 (2018) 235–249.

[7] B. Sun, J. Feng, K. Saenko, Return of frustratingly easy domain adaptation., in: AAAI Conference on Artificial Intelligence, 2016, pp. 2058–2065.

[8] S. Herath, M. Harandi, F. Porikli, Learning an invariant hilbert space for domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3845–3854.

[9] B. Gong, K. Grauman, F. Sha, Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation, in: International Conference on Machine Learning, 2013, pp. 222–230.

[10] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2066–2073.

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

[12] J. Wei, J. Liang, R. He, J. Yang, Learning discriminative geodesic flow kernel for unsupervised domain adaptation, in: International Conference on Multimedia and Expo, 2018, pp. 1–6.

[13] J. Huang, A. Gretton, K. M. Borgwardt, B. Schölkopf, A. J. Smola, Correcting sample selection bias by unlabeled data, in: Advances in Neural Information Processing Systems, 2007, pp. 601–608.

[14] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE TKDE 22 (10) (2010) 1345–1359.

[15] S. J. Pan, I. W. Tsang, J. T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE TNN 22 (2) (2011) 199–210.

[16] M. Baktashmotlagh, M. T. Harandi, B. C. Lovell, M. Salzmann, Unsupervised domain adaptation by domain invariant projection, in: IEEE Conference on Computer Vision, 2013, pp. 769–776.

[17] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, A. J. Smola, A kernel method for the two-sample-problem, in: Advances in Neural Information Processing Systems, 2007, pp. 513–520.

[18] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, JMLR 13 (Mar) (2012) 723–773.

[19] M. Long, J. Wang, G. Ding, J. Sun, P. S. Yu, Transfer feature learning with joint distribution adaptation, in: IEEE Conference on Computer Vision, 2013, pp. 2200–2207.

[20] J. Zhang, W. Li, P. Ogunbona, Joint geometrical and statistical alignment for visual domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1859–1867.

[21] J. Liang, R. He, Z. Sun, T. Tan, Aggregating randomized clustering-promoting invariant projections for domain adaptation, IEEE TPAMI 41 (5) (2019) 1027–1042.

[22] S. Xie, Z. Zheng, L. Chen, C. Chen, Learning semantic representations for unsupervised domain adaptation, in: International Conference on Machine Learning, 2018, pp. 5419–5428.

[23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE TPAMI 32 (9) (2010) 1627–1645.

[24] S. Motiian, M. Piccirilli, D. A. Adjeroh, G. Doretto, Unified deep supervised domain adaptation and generalization, in: ICCV, 2017, pp. 5715–5725.

[25] M. P. Kumar, B. Packer, D. Koller, Self-paced learning for latent variable models, in: Advances in Neural Information Processing Systems, 2010, pp. 1189–1197.

[26] B. Gholami, O. Rudovic, V. Pavlovic, Punda: Probabilistic unsupervised domain adaptation for knowledge transfer across visual categories, in: IEEE Conference on Computer Vision, 2017, pp. 3601–3610.

[27] C. Cortes, M. Mohri, M. Riley, A. Rostamizadeh, Sample selection bias correction theory, in: International Conferences on Algorithmic Learning Theory, 2008, pp. 38–53.

[28] J. Liang, R. He, Z. Sun, T. Tan, Distant supervised centroid shift: A simple and efficient approach to visual domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2975–2984.

[29] L. Luo, L. Chen, S. Hu, Y. Lu, X. Wang, Discriminative and geometry aware unsupervised domain adaptation, arXiv preprint arXiv:1712.10042,

2017.
[30] N. Courty, R. Flamary, A. Habrard, A. Rakotomamonjy, Joint distribution optimal transportation for domain adaptation, in: Advances in Neural Information Processing Systems, 2017, pp. 3733–3742.
[31] N. Courty, R. Flamary, D. Tuia, A. Rakotomamonjy, Optimal transport for domain adaptation, IEEE TPAMI 39 (9) (2017) 1853–1865.
[32] M. Long, H. Zhu, J. Wang, M. I. Jordan, Deep transfer learning with joint adaptation networks, in: International Conference on Machine Learning, 2017, pp. 2208–2217.
[33] E. Tzeng, J. Hoffman, T. Darrell, K. Saenko, Adversarial discriminative domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7167–7176.
[34] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: Maximizing for domain invariance, arXiv preprint arXiv:1412.3474, 2014.
[35] Y. Li, N. Wang, J. Shi, X. Hou, J. Liu, Adaptive batch normalization for practical domain adaptation, Pattern Recognition 80 (2018) 109–117.
[36] B. Chidlovskii, S. Clinchant, G. Csurka, Domain adaptation in the absence of source domain data., in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2016, pp. 451–460.
[37] P. Panareda Busto, J. Gall, Open set domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 754–763.
[38] X. J. Zhu, Semi-supervised learning literature survey, Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (2005).
[39] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Annual Conference on Learning Theory, 1998, pp. 92–100.
[40] L. Bruzzone, M. Marconcini, Domain adaptation problems: A dasvm classification technique and a circular validation strategy, IEEE TPAMI 32 (5) (2010) 770–787.
[41] Y. Chen, G. Wang, S. Dong, Learning with progressive transductive support vector machine, Pattern Recognition Letters 24 (12) (2003) 1845–1855.
[42] H. S. Bhatt, R. Singh, M. Vatsa, N. K. Ratha, Improving cross-resolution face matching using ensemble-based co-transfer learning, IEEE TIP 23 (12) (2014) 5654–5669.
[43] F. Ma, D. Meng, Q. Xie, Z. Li, X. Dong, Self-paced co-training, in: International Conference on Machine Learning, 2017, pp. 2275–2284.
[44] K. Tang, V. Ramanathan, L. Fei-Fei, D. Koller, Shifting weights: Adapting object detectors from image to video, in: Advances in Neural Information Processing Systems, 2012, pp. 638–646.
[45] X. He, Y. Zhou, Z. Zhou, S. Bai, X. Bai, Triplet-center loss for multi-view 3d object retrieval, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1945–1954.
[46] L. Xu, M. White, D. Schuurmans, Optimal reverse prediction: a unified perspective on supervised, unsupervised and semi-supervised learning, in: International Conference on Machine Learning, 2009, pp. 1137–1144.
[47] H. Wang, C. Ding, H. Huang, Multi-label linear discriminant analysis, in: European Conference on Computer Vision, 2010, pp. 126–139.
[48] T. Tommasi, T. Tuytelaars, A testbed for cross-dataset analysis, in: European Conference on Computer Vision, 2014, pp. 18–31.
[49] E. Zhong, W. Fan, Q. Yang, O. Verscheure, J. Ren, Cross validation framework to choose amongst models and datasets for transfer learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2010, pp. 547–562.
[50] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: European Conference on Computer Vision, 2010, pp. 213–226.
[51] H. Lu, C. Shen, Z. Cao, Y. Xiao, A. van den Hengel, An embarrassingly simple approach to visual domain adaptation, IEEE TIP 27 (7) (2018) 3403–3417.
[52] M. Long, H. Zhu, J. Wang, M. I. Jordan, Unsupervised domain adaptation with residual transfer networks, in: Advances in Neural Information Processing Systems, 2016, pp. 136–144.
[53] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: International Conference on Machine Learning, 2015, pp. 1180–1189.
[54] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, W. Li, Deep reconstruction-classification networks for unsupervised domain adaptation, in: European Conference on Computer Vision, 2016, pp. 597–613.
[55] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, K. Kim, Image to image translation for domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4500–4509.
[56] P. O. Pinheiro, Unsupervised domain adaptation with similarity learning, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8004–8013.
[57] W. Zhang, W. Ouyang, W. Li, D. Xu, Collaborative and adversarial network for unsupervised domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3801–3809.
[58] Z. Zhang, M. Wang, Y. Huang, A. Nehorai, Aligning infinite-dimensional covariance matrices in reproducing kernel hilbert spaces for domain adaptation, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3437–3445.
[59] C.-A. Hou, Y.-H. H. Tsai, Y.-R. Yeh, Y.-C. F. Wang, Unsupervised domain adaptation with label and structural consistency, IEEE TIP 25 (12) (2016) 5552–5562.
[60] Z. Ding, Y. Fu, Robust transfer metric learning for image classification, IEEE TIP 26 (2) (2017) 660–670.
[61] S. Li, S. Song, G. Huang, Z. Ding, C. Wu, Domain invariant and class discriminative feature learning for visual domain adaptation, IEEE TIP 27 (9) (2018) 4260–4273.