Pattern Recognition 138 (2023) 109379

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Improving pseudo labels with intra-class similarity for unsupervised domain adaptation

Jie Wang [a,b], Xiao-Lei Zhang [a,b,*]

[a] School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, China
[b] Research & Development Institute of Northwestern Polytechnical University in Shenzhen, Shenzhen 518057, China

ARTICLE INFO

Article history:
Received 24 January 2022
Revised 10 November 2022
Accepted 1 February 2023
Available online 2 February 2023

Keywords:
Unsupervised domain adaptation
Intra-class similarity
Spanning trees
Pseudo labels

ABSTRACT

Unsupervised domain adaptation (UDA) transfers knowledge from a label-rich source domain to a different but related fully-unlabeled target domain. To address the problem of domain shift, more and more UDA methods adopt pseudo labels of the target samples to improve the generalization ability on the target domain. However, inaccurate pseudo labels of the target samples may yield suboptimal performance with error accumulation during the optimization process. Moreover, once the pseudo labels are generated, how to remedy the generated pseudo labels is far from explored. In this paper, we propose a novel approach to improve the accuracy of the pseudo labels in the target domain. It first generates coarse pseudo labels by a conventional UDA method. Then, it iteratively exploits the intra-class similarity of the target samples for improving the generated coarse pseudo labels, and aligns the source and target domains with the improved pseudo labels. The accuracy improvement of the pseudo labels is made by first deleting dissimilar samples, and then using spanning trees to eliminate the samples with the wrong pseudo labels in the intra-class samples. We have applied the proposed approach to several conventional UDA methods as an additional term. Experimental results demonstrate that the proposed method can boost the accuracy of the pseudo labels and further lead to more discriminative and domain invariant features than the conventional baselines.

1. Introduction

It is known that machine learning benefits from manually-labeled data. However, manual labeling is often time-consuming and laboring intensive. Moreover, it is even unlikely to obtain sufficient manual labels in some scenarios. How to address the insufficient labeling problem is a key task. One of the approaches to address the problem is domain adaptation, which aims to transfer knowledge from a label-rich source domain to a different but related target domain [1,2]. Based on whether the target domain is human labeled, domain adaptation can be divided into two categories [3], which are semi-supervised domain adaptation [4,5] and unsupervised domain adaptation [6,7]. In this paper, we focus on unsupervised domain adaptation (UDA) where the target domain does not have manual labels. It is not only challenging but also finds its applications in many real-world scenarios.

In the past decades, UDA has been widely studied [8–10]. The most common approach is to find a common subspace in which the data distributions of the source domain and target domain are similar. The first issue on learning the subspace is to define a suitable distribution divergence measurement between the two domains. A common measurement is the maximum mean discrepancy (MMD) [11–13].

By minimizing the distribution divergence as a regularizer, a common subspace could be found. For example, [14] learns a domain-invariant projection and meanwhile minimizes the marginal distribution divergences between the source domain and the target domain.

Recently, a new branch of the UDA research is to use the pseudo labels of the data in the target domain to align the distributions of the source and target domains [15–17], where the pseudo labels are usually obtained by a classifier trained on the source domain. For example, following [14], the work [18] further reduces the marginal distribution divergence and conditional distribution divergence between the source domain and the target domain iteratively with the pseudo labels of the target domain[16]. learns both domain invariant and class discriminative features with the

* Corresponding author.
E-mail addresses: wangjie2017@mail.nwpu.edu.cn (J. Wang), xiaolei.zhang@nwpu.edu.cn (X.-L. Zhang).

https://doi.org/10.1016/j.patcog.2023.109379
0031-3203/

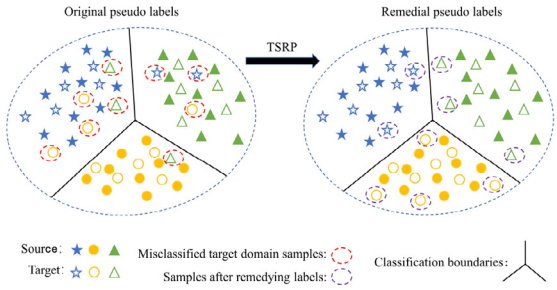


Fig. 1. Motivation of the proposed TSRP. Most related works focus on learning domain invariance features while ignoring the intra-class similarity between target samples. On the contrary, TSRP aims to explore the intra-class similarity between the samples in the target domain to remedy pseudo labels, which in turn leads to better domain invariance features.

pseudo labels. Li et al. [8] preserves the neighborhood relationship of samples and improves robustness against outliers by supervised locality preserving projection [19] with the pseudo labels. Wang et al. [11] proposes a discriminative MMD to mitigate the degradation of feature discriminability incurred by MMD.

Although the pseudo label generation approaches have made significant contribution to UDA, there are still two issues far from explored. First, the pseudo labels are mainly obtained by a good alignment between the source domain and the target domain, while the effect of the accuracy of the pseudo labels on performance is not studied deeply. When the pseudo labels are generated by a classifier trained on the source domain, which is the common way, there may be some incorrect pseudo labels in each optimization iteration as shown in Fig. 1. Due to the cumulation of the errors, the incorrect pseudo labels can greatly affect the final performance. Second, most of the methods mainly focus on mining the source domain to improve the accuracy of the pseudo labels in the target domain, however, to our knowledge, the intrinsic relationship between the samples in the target domain seems unexplored yet.

To address the aforementioned two issues, in this paper, we propose to mine the *target domain intra-class similarity to remedy the pseudo labels* (TSRP) in the target domain for improving the accuracy of the pseudo labels. A core idea of TSRP is to *use target similarity to pick pseudo labels with high confidence* (UTSP) by spanning trees [20]. Then, the selected highly-confident pseudo-labeled samples as well as the source data are used to train a strong classifier. The strong classifier is used to correct part of the the wrongly-labeled target samples that have low-confident pseudo labels. We call it the *remedial process of the pseudo labels*. Generally, our method can be integrated into any methods that generate the pseudo labels of the target domain by using the classifiers trained on the source domain.

Our contribution is summarized as follows:

- We propose TSRP to improve the accuracy of the pseudo labels in the target domain. TSRP iteratively exploits the intra-class similarity of the samples in the target domain for improving the generated coarse pseudo labels, and aligns the source and target domains with the improved pseudo labels.
- We propose UTSP to select highly-confident pseudo-labeled samples. UTSP first deletes dissimilar samples, and then uses spanning trees to eliminate the samples with the wrong pseudo labels in the intra-class samples.
- We extended four UDA algorithms [16,18,21,22] with TSRP, and evaluated the effectiveness of TSRP by comparing the UDA algorithms with their TSRP extensions. Experimental results on several benchmark datasets show that TSRP can be used as a term of the UDA methods for improving their generalization ability. Moreover, we have compared the proposed "DICD [16] +TSRP" algorithm with a number of representative UDA algorithms [14,23–29]. Experimental results show that the integrated method behaves better than the comparison methods.

The remainder of this paper is organized as follows. In Section 2, we review some related work. In Section 3, we propose TSRP to improve the accuracy of pseudo labels. The experimental results are reported in Section 4. Finally, we conclude this paper in Section 5.

2. Related work

Early works on UDA aim to align the marginal distributions of the source and target domains [24,30]. Due to the lack of labeled target samples, even though the marginal distributions are perfectly aligned, there is no guarantee that a good classification result will be produced since that the conditional distribution of the target domain may be misaligned with that of the source domain. To overcome this issue, many UDA methods resort to pseudo labels of the target domain [31–33]. If the pseudo labels of the target samples can be properly obtained, then supervised learning can be applied to train a good classifier. There are two strategies to generate the pseudo labels—hard labeling [16,18,34] and soft labeling [35]. Because the accuracy of the pseudo labels plays an important role to the quality of the classifier, we summarize some pseudo label generation and selection methods that focus on improving the accuracy of the pseudo labels as follows.

Pan et al. [36] proposes transferrable prototypical networks to learn an embedding space of two domains, and perform classification at both the class level and the sample level. Zheng and Yang [37] takes the prediction variance of two classifiers as an estimation of the uncertainty of pseudo labels, which induces the model to give more reliability to the samples with small prediction variances. Zou et al. [38] proposes two types of confidence regularization to eliminate overconfident pseudo labels. Zhang et al. [39] proposes to improve a label-propagation-based unsupervised domain adaptation algorithm via generating unlabeled virtual instances with high-confidence label predictions, named augmented anchors. Wang and Breckon [40] explores the structural information of the target domain by structured prediction, and combines the nearest class prototype and structured prediction to promote the accuracy of pseudo labels. Tian et al. [41] regards the samples of the same cluster in the target domain as a whole rather than individuals. It assigns pseudo labels to the target cluster by class centroid matching. Chen et al. [42] proposed an easy-to-hard strategy which divides target samples into three categories, namely easy samples, hard samples and incorrect-easy samples. It tends to generate pseudo labels for easy samples and tries to avoid hard samples. The easy-to-hard strategy may be biased to easy classes. To address this issue, a confidence-aware pseudo label selection strategy was proposed in [43]. It selects samples from each class independently by the probability of pseudo labels. In [42,43], they use the distances from the target samples to the centers of the source samples as the selection criterion to select highly confident pseudo labels. In [40,41], they iteratively generate-confident pseudo labels. However, they do not consider how to correct the falsely generated pseudo labels.

Different from the above methods, in this paper, we propose to correct the falsely generated pseudo labels by exploring the intra-class similarity in the target domain.

3. Method

In the following subsections, we give the formulation of the problem and our motivation, and describe the proposed method in detail.

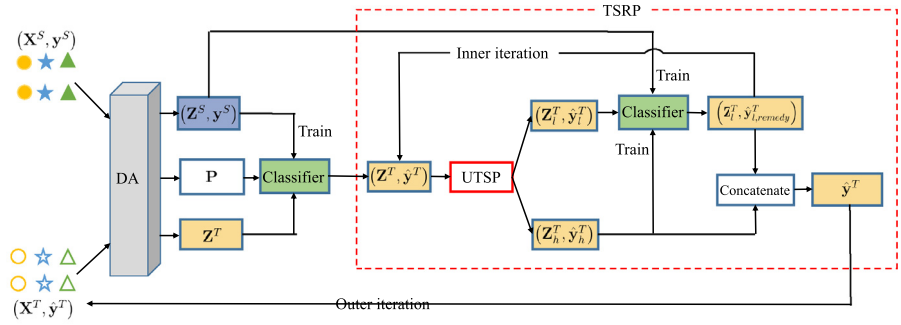


Fig. 2. Architecture of the proposed framework with TSRP. It conducts the outer iterations until convergence. Each loop of the outer iterations conducts the following three steps successively: First, a pseudo-label based unsupervised domain adaptation (UDA) method is used to learn a domain-invariant projection matrix $\mathbf{P}$ from both domains, i.e. $\{\mathbf{X}^S, \mathbf{y}^S\}$ and $\{\mathbf{X}^T, \hat{\mathbf{y}}^T\}$, which can obtain domain invariant features $\mathbf{Z}^S = \mathbf{P}\mathbf{X}^S$ and $\mathbf{Z}^T = \mathbf{P}\mathbf{X}^T$. Then, a weak classifier, which is trained on the source domain $\{\mathbf{Z}^S, \mathbf{y}^S\}$, is used to update the pseudo labels of the target samples $\hat{\mathbf{y}}^T$. Finally, the inner iterations are conducted until convergence, which produces the improved pseudo labels of the target domain $\hat{\mathbf{y}}^T$ for the next loop of the outer iterations. Each loop of the inner iterations conducts the following three steps successively: First, UTSP is proposed to partition the target samples into a set of samples with highly-confident pseudo labels, denoted as $\{\mathbf{Z}_h^T, \hat{\mathbf{y}}_h^T\}$, and the rest samples with low-confident pseudo labels $\{\mathbf{Z}_l^T, \hat{\mathbf{y}}_l^T\}$. Then, a strong classifier is trained with the source samples $\{\mathbf{Z}^S, \mathbf{y}^S\}$ and the target samples with the highly-confident pseudo labels $\{\mathbf{Z}_h^T, \hat{\mathbf{y}}_h^T\}$. Finally, the strong classifier is used to remedy/update the low-confident pseudo labels, denoted as $\{\mathbf{Z}_l^T, \hat{\mathbf{y}}_{l,\text{remedy}}^T\}$. The algorithm in the red dotted box is the proposed TSRP.. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

3.1. Framework

A UDA problem is formulated as follows. Given a manually labeled source domain $\mathcal{D}^S = \left\{\left(\mathbf{x}_i^S, y_i^S\right)\right\}_{i=1}^{n_s} = \left\{\mathbf{X}^S, \mathbf{y}^S\right\}$, and an unlabeled target domain $\mathcal{D}^T = \{\mathbf{x}_j^T\}_{j=1}^{n_t} = \left\{\mathbf{X}^T\right\}$, where $\mathbf{x}^S$ and $\mathbf{x}^T$ represent m-dimensional feature vectors of the samples in the source domain and target domain respectively, $\mathbf{y}^S$ is the manual label in the source domain, n_s and n_t are the number of the source and target samples respectively. The goal of UDA is to predict the labels of $\mathcal{D}^T$. In this paper, we focus on the problem that the source and target domains share the same object classes. Suppose there are C classes in both domains.

As summarized in Section 2, many UDA approaches focus on obtaining a good alignment of the source and target domains to generate pseudo labels, leaving the negative effect of the falsely generated pseudo labels unsolved. Because the inaccurate pseudo labels could result in catastrophic error accumulation during the learning process [40], intuitively, if we could increase the accuracy of the pseudo labels, then we may get a better alignment between the source domain and the target domain.

In this paper, we propose to steadily improve the accuracy of the pseudo labels in the target domain by iteratively exploiting the intra-class similarity between the target samples. An shown in Fig. 2. for each iteration, the proposed method runs the following two steps in sequence. First, it employs a traditional UDA method to generate the crude pseudo labels of the target samples for the current iteration given a set of improved pseudo labels from the previous iteration, see Section 3.2 for the details. Then, it uses TSRP to improve the accuracy of the pseudo labels, see Section 3.3 for the details where a key component named UTSP is presented in Section 3.4. The overall framework with TSRP module is summarized in Algorithm 1.

Algorithm 1: Proposed framework with TSRP module.

Input: Labeled source samples, $\left\{\mathbf{X}^S, \mathbf{y}^S\right\} = \left\{\left(\mathbf{x}_i^S, y_i^S\right)\right\}_{i=1}^{n_s}$;
Target samples, $\left\{\mathbf{X}^T\right\} = \{\mathbf{x}_j^T\}_{j=1}^{n_t}$; Trust parameter: ρ;
Output: Remedial pseudo labels of the target domain $\hat{\mathbf{y}}^T$;

```
while not converged do
    Get the projection matrix P by a UDA method with {X^S, y^S} and {X^T, ŷ^T};
    Z^S ← PX^S;
    Z^T ← PX^T;
    Train the classifier f(·) with {Z^S, y^S};
    ŷ^T ← f(Z^T);
    // TSRP start:
    while not converged do
        ({Z_h^T, ŷ_h^T}, {Z_l^T, ŷ_l^T}) ← UTSP({Z^T, ŷ^T});
        Train a strong classifier f_strong with (Z^S, y^S) and (Z_h^T, ŷ_h^T) ;
        Get the remedial pseudo labels y_{l,remedy}^T by (1);
        {Z^T, ŷ^T} ← {Z_l^T, ŷ_{l,remedy}^T};
    ŷ^T ← [ŷ_h^T, ŷ_{l,remedy}^T];
    // TSRP end;
```

3.2. UDA: Generating crude pseudo labels

An existing UDA algorithm is employed to learn a projection matrix $\mathbf{P}$ that maps the samples from both domains into a shared latent subspace. A requirement is that $\mathbf{P}$ should be learned in a supervised manner where the remedial pseudo labels of the target samples obtained from the previous iteration of the framework. $\hat{\mathbf{y}}^T = [\hat{y}_1^T, \ldots, \hat{y}_{n_t}^T]$ are treated as the labels. Various advanced UDA algorithms meet this requirement, such as those [16,18,21,22] employed in the experiments. Then, domain invariant features of the source and target domains are obtained by $\mathbf{z}_i^S = \mathbf{P}\mathbf{x}_i^S$ and $\mathbf{z}_j^T = \mathbf{P}\mathbf{x}_j^T$. Finally, a classifier $f(\cdot)$ is trained with $\left\{\mathbf{Z}^S, \mathbf{y}^S\right\}$ where $\mathbf{Z}^S = [\mathbf{z}_1^S, \ldots, \mathbf{z}_{n_s}^S]$. It is then used to classify $\mathbf{Z}^T = [\mathbf{z}_1^T, \ldots, \mathbf{z}_{n_t}^T]$ into C_y classes ($C_y \le C$). The predicted labels of the target samples are denoted as the *crude pseudo labels* $\hat{\mathbf{y}}^T = [\hat{y}_1^T, \ldots, \hat{y}_{n_t}^T]$.

3.3. TSRP: Using target domain intra-class similarity to remedy pseudo labels

There may always be some incorrect pseudo labels in $\hat{\mathbf{y}}^T$ in each iteration, especially, in the early training stage, therefore, when learning $\mathbf{P}$ with the incorrect pseudo labels, the final performance may not be good due to the cumulative errors in the iterative optimization process. If we could correct part of the incorrect pseudo labels in each iteration, then the performance might be improved steadily. To address this issue, a possible way is to pick the pseudo labels with low confidence, and conduct correction to them with a stronger classifier than the original classifier $f(\cdot)$.

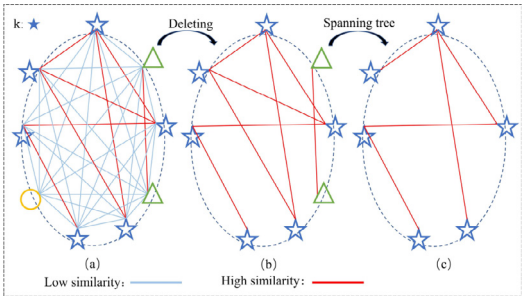


Fig. 3. Principle of UTSP. UTSP consists of two steps: (i) *Deleting*, which deletes the samples with low pairwise similarity scores as shown from Fig. (a) to Fig. (b), and (ii) *spanning tree*, which selects samples with highly-confident pseudo labels by spanning trees as shown from Fig. (b) to Fig. (c).

Motivated by the above analysis, TSRP aims to improve the accuracy of the crude pseudo labels by remedying the pseudo labels with low confidence. Specifically, TSRP first partitions the target samples into two sets, one with highly-confident pseudo labels, denoted as $\{\mathbf{Z}_h^T, \hat{\mathbf{y}}_h^T\}$ and the other one with low confident pseudo labels, denoted as $\{\mathbf{Z}_l^T, \hat{\mathbf{y}}_l^T\}$, by exploring the intra-class similarity in the target domain (see Section 3.4). Then, it trains a strong classifier $f_{\text{strong}}(\cdot)$ with $\{\mathbf{Z}_h^T, \hat{\mathbf{y}}_h^T\}$ and $\{\mathbf{Z}^S, \mathbf{y}^S\}$, and uses the classifier to predict the labels of $\mathbf{Z}_l^T$:

$$\hat{\mathbf{y}}_{l,\text{remedy}}^T = f_{\text{strong}}\left(\mathbf{Z}_l^T\right) \tag{1}$$

where $\hat{\mathbf{y}}_{l,\text{remedy}}^T$ is the remedial pseudo labels of the target samples with low confidence. Finally, we take the remedial pseudo labels and the pseudo labels with high confidence together as the target domain pseudo labels to train $\mathbf{P}$ in the next iteration.

3.4. UTSP: Using target intra-class similarity to pick pseudo labels with high confidence

UTSP aims to select the target samples whose pseudo labels are highly-confident. It consists of two steps—*deleting* and *spanning tree*. Its principle is illustrated in Fig. 3. We will present the details of the two steps in the following subsections with a summary in Algorithm 2.

Algorithm 2: UTSP.

Input: Domain-invariant features $\mathbf{z}^T$, its corresponding pseudo labels $\hat{\mathbf{y}}^T$ which contain C_y pseudo classes;
Output: Highly-confident samples $\left\{\mathbf{Z}_h^T, \hat{\mathbf{y}}_h^T\right\}$, low-confident samples $\left\{\mathbf{Z}_l^T, \hat{\mathbf{y}}_l^T\right\}$;

for $k = 1, \ldots, C_y$ **do**
- Calculate intra-class similarity matrix $\mathbf{S}^k$ by (2);
- Calculate δ by (4);
- Calculate adjacency matrix $\mathbf{M}_k$ by (5);
- Calculate diagonal matrix $\mathbf{D}^k$ by (6);
- Pick the root node of the spanning tree, i.e. $\mathbf{z}_m^{Tk}$, by (7);
- Get highly-confident and low-confident samples by the spanning tree;

3.4.1. Deleting

The *deleting* step aims to delete the samples with small similarity for each pseudo class. In the following, we present the *deleting* step for the kth pseudo class, $\forall k = 1, 2, \ldots, C_y$. It first calculates a pairwise intra-class similarity matrix $\mathbf{S}^k$ of the target samples by e.g. cosine similarity:

$$S_{i,j}^k = \begin{cases} 0, & i = j \\ \frac{\langle \mathbf{z}_i^{Tk}, \mathbf{z}_j^{Tk} \rangle}{\|\mathbf{z}_i^{Tk}\| \|\mathbf{z}_j^{Tk}\|}, & \text{otherwise} \end{cases}, \forall k = 1, 2, \ldots, C_y \tag{2}$$

where $S_{i,j}^k$ denotes the cosine similarity between the ith sample and jth sample of the kth pseudo class, $i, j \in \left\{1, 2, \ldots, n_t^k\right\}$ with n_t^k denoted as the number of samples in the kth pseudo class, $n_t = \sum_{k=1}^{C_y} n_t^k$, and $\mathbf{z}^{Tk}$ denotes a target sample belonging to the kth pseudo class. Because $\mathbf{S}^k$ is a symmetric matrix, we only keep the upper triangular matrix of $\mathbf{S}^k$, denoted as $\mathbf{S}_{\text{upper}}^k$. We sort the non-zero elements of $\mathbf{S}_{\text{upper}}^k$ in the ascending order:

$$S_{\text{rank}}^k = \left\{S_{\text{rank}(1)}^k, S_{\text{rank}(2)}^k, \ldots, S_{\text{rank}(n_p)}^k\right\} \tag{3}$$

where n_p is the number of non-zero elements in $\mathbf{S}_{\text{upper}}^k$.

In order to select highly-confident pseudo labels in each category, the *deleting* step sets a similarity threshold δ:

$$\delta = S_{\text{rank}(\lfloor \rho n_p \rfloor)}^k \tag{4}$$

where $\rho \in (0, 1)$ is a trust parameter. Then, we can obtain a mask matrix $\mathbf{M}^k$ as follows:

$$M_{ij}^k = \begin{cases} 1, & S_{i,j}^k \geq \delta \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

If we regard each sample $\mathbf{z}^{Tk}$ as a node, and take $\mathbf{M}^k$ as the adjacency matrix of the undirected graph that are composed of the n_t^k nodes, then we could observe commonly that the samples belonging to the same ground-truth category have high pairwise similarity scores in the pseudo class, on the contrary, the samples belonging to different ground-truth categories have low pairwise similarity scores. Here we regard the nodes with no neighbors as the samples with low-confident pseudo labels. We delete these nodes from the undirected graph. The process is illustrated in Fig. 3a to b.

3.4.2. Spanning tree

However, the samples of the kth pseudo class may be mixed with samples from multiple ground-truth categories that are geometrically very similar to each other. If we only select highly-confident pseudo labels by the threshold δ, some misclassified samples may be selected as highly-confident samples after the *deleting* step.

To further refine the samples selected by the *deleting* step, we explore an idea from *spanning forests* [20]. Specifically, we first get a diagonal matrix $\mathbf{D}^k$ by:

$$D_{ii}^k = \sum_j M_{ij}^k \tag{6}$$

The identity of the largest element of $\mathbf{D}^k$ can be computed as:

$$m = \arg\max_i D_{ii}^k \tag{7}$$

The node with the maximum degree, i.e. $\mathbf{z}_m^{Tk}$, represents the most confident sample in the k pseudo class, since that it has the maximum number of neighbors.

Then, we set $\mathbf{z}_m^{Tk}$ to the root node of a spanning tree and find its leaf nodes. We regard the samples in the same spanning tree as highly-confident samples of the kth pseudo class, and the samples that are not in the spanning tree as misclassified samples from other ground-truth categories. This process is illustrated in Fig. 3b to c.

By pooling the highly-confident samples throughout all pseudo classes, we finally get the highly-confident set $\{\mathbf{Z}_h^T, \hat{\mathbf{y}}_h^T\}$ and low-confident set $\{\mathbf{Z}_l^T, \hat{\mathbf{y}}_l^T\}$. Note that, in order to avoid a bad solution to

Table 1
Description of the visual cross-domain datasets in the experiments.

Dataset	Type	Samples	Classes	Features
CMU-PIE	Face	11,554	68	1024
MNIST	Digit	2000	10	256
USPS	Digit	1800	10	256
Office+Caltech-256	Object	2533	10	800/4096
COIL20	Object	1440	20	1024
Office-Home	Object	15,588	65	2048

class-imbalanced problems where a class with a small number of samples disappears after UTSP, we regard all samples of the pseudo class satisfying $n_t^k \leq 3$ as highly-confident samples.

4. Experiments

In this section, we evaluate the performance of the proposed methods. The source code of TSRP is available at https://github.com/02Bigboy/TSRP.

4.1. Datasets and cross-domain tasks

We used common cross-domain datasets [16], including CMU-PIE [44], MNIST [45], USPS [46], Office+Caltech256 [47], COIL20 [48], and Office-Home [49]. The statistics of the datasets are summarized in Table 1. The domain adaptation tasks on the datasets are described as follows.

CMU-PIE is a large face dataset. It consists of more than 40,000 face images from 68 individuals. The face images vary widely due to the variations of the illumination condition, poses, and expressions. In terms of different pose factors, we chose five subsets as in [16]: C05 (left pose), C07 (upward pose), C09 (downward pose), C27 (front pose) and C29 (right pose) to construct the cross-domain classification tasks. Following the experimental setting in [16], we randomly selected two subsets as the source domain and target domain respectively for each cross-domain task, which results in 20 cross-domain tasks in total, e.g. "C05 → C07", "C05 → C09", ..., "C29 → C27" in the form of "source→target".

MNIST-USPS consists of two classical hand written digit image datasets—USPS [46] and MNIST [45]. To speed up the experimental comparisons as that in [18], we randomly chose 1800 images from USPS and 2000 images from MNIST, and rescaled the images to a size of 16×16, which forms two cross-domain tasks, i.e. "MNIST → USPS" and "USPS → MNIST".

Office+Caltech dataset is one of the most commonly used datasets for unsupervised domain adaptation. It consists of four domains: Amazon (images downloaded from online merchants, abbreviated as A), Webcam (low-resolution images by a web camera, abbreviated as W), DSLR (high-resolution images by a digital SLR camera, abbreviated as D), and Caltech-256 (abbreviated as C). Here, ten common classes from all four domains were used, which are backpack, bike, calculator, headphone, computer-keyboard, laptop, computer-monitor, computer-mouse, coffee-mug, and video-projector respectively. There are 2533 images in total with 8 to 151 images per category per domain. We extracted two kinds of features, which are the 800-dim SURF [24] and 4096-dim DeCAF6 [50] respectively. Similar to [16], we obtained 12 cross-domain tasks for two kinds of features, e.g. "A → C", "A → D", ..., "W → C" and "W → D", by randomly choosing two domains from the data as the source and target respectively.

COIL20 dataset consists of 1440 grayscale images with 20 objects. Each object has 72 images of size 32×32 taken at pose intervals of 5 degrees rotating through 360 degrees. Like [16], we split the dataset into 2 subsets: COIL1 and COIL2. COIL1 includes all images at the directions of 0° to 85° and 180° to 265°. COIL2 contains the images of 90° to 175° and 270° to 355°. They follow different but related distributions since that COIL1 and COIL2 consist of the same objects with diverse shooting degrees. We randomly chose one as the source domain and the other as the target domain for two cross-domain tasks, e.g. "COIL1 → COIL2" and "COIL2 → COIL1".

Office-Home includes 65 object classes from four domains, i.e., Artistic images (A), Clipart (C), Product images (P) and Real-World images (R). It has 15,588 images. Any two out of the four domains can formulate a domain adaptation task, which results in 12 domain adaptation tasks in total.

Note that, the abbreviation "A" for the Office-Home dataset is different from that for Office+Caltech dataset. Because the abbreviation of a dataset is always aligned with the name of the dataset in the following paper, we think that this will not cause confusion, and bravely used the duplicated abbreviations.

4.2. Experimental settings

TSRP can be used as a term of many UDA algorithms. In order to verify the effectiveness of TSRP, we integrated it into 4 algorithms, namely 1-nearest neighbor (NN) [21], joint distribution adaptation (JDA) [18], balanced distribution adaptation (BDA) [22], and domain invariant and class discriminative feature learning (DICD) [16]. NN is a standard machine learning methods. JDA aligns both marginal distribution and conditional distribution of of the source and target domains. BDA aligns the marginal distribution and conditional distribution with different weights according to different tasks. DICD considers the class discrimination to learn both domain invariant and class discriminative features. We denote the extended methods of the above four UDA algorithms as NN+TSRP, JDA+TSRP, BDA+TSRP, and DICD+TSRP respectively.

Our TSRP approach consists of two hyper-parameters: The trust parameter ρ, and the number of inner iterations for TSRP IT. We set $IT = 3$ as a fixed parameter for all experiments. We set $\rho = 0.9$ for the datasets Office+Caltech-256 (SURF) and CMU-PIE, and $\rho = 0.85$ for the other datasets. The iteration number for JDA, BDA, DICD was set to $T = 10$. For a fair comparison, the parameters of the extended methods and the original methods were set the same. The classification accuracy on the target domain was used as the evaluation metric.

4.3. Experimental results

In this section, we compared the four UDA algorithms, i.e. NN, JDA, BDA, and DICD, with their TSRP extensions on the visual cross-domain tasks.

Table 2 shows the classification performance on the CMU-PIE dataset. From the table, we can see that, after incorporating TSRP, the performance of JDA, BDA, and DICD has been significantly improved. To be specific, DICD+TSRP, BDA+TSRP, and JDA+TSRP achieve 4.75%, 2.52%, 2.05% absolute improvement respectively over DICD, BDA, and JDA in terms of average accuracy. It is worthy noting that, compared to DICD, DICD+TSRP obtains the best results on all cross-domain tasks of PIE. BDA+TSRP achieves the best results in 17 tasks out of all 20 tasks, compared to BDA. JDA+TSRP achieves the best results in 15 tasks compared to JDA. In addition, we have also observed that the effect of NN+TSRP is worse than that of NN. It may be caused by that the original data of PIE in the source domain and target domain are quite different. Therefore, when NN is applied to the original data directly, the incorrectly generated pseudo labels may be the majority. Eventually, the majority is further enhanced after TSRP is applied.

Table 3 lists the classification accuracy of the comparison methods on the Office+Caltech-256 (SURF features) dataset. From the table, we see that, after combining with TSRP, all of the four UDA

Table 2
Average classification accuracy (%) of the comparison methods on the target domains of the CMU-PIE tasks.

Tasks/Methods	NN	NN+TSRP	JDA	JDA+TSRP	BDA	BDA+TSRP	DICD	DICD+TSRP
C05 → C07	26.09	23.51	58.81	**63.41**	58.81	**64.52**	72.99	**74.52**
C05 → C09	26.59	22.92	54.23	**56.92**	57.11	**58.70**	72.00	**76.16**
C05 → C27	30.67	27.31	84.50	81.98	84.50	82.31	92.22	**96.52**
C05 → C29	16.67	14.64	49.75	**54.47**	49.94	**57.41**	66.85	**71.32**
C07 → C05	24.49	24.10	57.62	**63.00**	57.77	**64.02**	69.93	**73.20**
C07 → C09	46.63	41.30	62.93	**62.93**	62.93	**63.60**	65.87	**67.83**
C07 → C27	54.07	52.42	75.82	75.52	76.06	**77.20**	85.25	**86.42**
C07 → C29	26.53	23.28	39.89	**46.69**	42.03	**46.69**	48.71	**56.43**
C09 → C05	21. 37	**21.76**	50.96	48.08	52.76	51.38	69.36	**70.14**
C09 → C07	41.01	34.56	57.95	**58.13**	57.95	**58.99**	65.44	**73.05**
C09 → C27	46.53	46.14	68.45	**69.45**	68.88	**70.68**	83.39	**94.26**
C09 → C29	26.23	23.65	39.95	**46.14**	42.65	**46.81**	61.40	**66.48**
C27 → C05	32.95	30.58	80.58	**81.54**	80.70	**81.99**	93.13	**94.72**
C27 → C07	62.68	59.30	82.63	82.01	83.18	**83.92**	90.12	**92.88**
C27 → C09	73.22	72.30	87.25	86.52	87.32	87.13	88.97	**90.26**
C27 → C29	37.19	33.58	54.66	**57.54**	55.64	**61.15**	75.61	**79.11**
C29 → C05	18.49	18.40	46.46	**56.66**	50.99	**57.47**	62.88	**73.68**
C29 → C07	24.19	21.42	42.05	**44.38**	45.92	**46.59**	57.03	**65.81**
C29 → C09	28.31	27.14	53.31	50.86	53.25	**55.76**	65.87	**70.10**
C29 → C27	31.24	**31.30**	57.01	**59.63**	57.28	**59.63**	74.77	**83.81**
Average accuracy	35.46	32.48	60.24	**62.29**	61.28	**63.80**	73.09	**77.84**
Average improvement		-2.98		**2.05**		**2.52**		**4.75**
Relative improvement		-3.49		**5.16**		**6.5**		**17.65**

Table 3
Classification accuracy (%) on the Office+Caltech-256 (surf features) tasks, where A = AMAZON, C = CALTECH, D = DSLR and W = WEBCAM.

Tasks/Methods	NN	NN+TSRP	JDA	JDA+TSRP	BDA	BDA+TSRP	DICD	DICD+TSRP
C → A (SURF)	23.70	23.49	44.78	**46.45**	46.14	**48.33**	47.29	**47.81**
C → W (SURF)	25.76	24.75	41.69	**46.10**	41.69	**47.46**	46.44	**50.85**
C → D (SURF)	25.48	24.84	45.22	**49.04**	47.13	**49.04**	49.68	**50.96**
A → C (SURF)	26.00	**26.63**	39.36	**39.63**	40.61	39.72	42.39	41.76
A → W (SURF)	29.83	**30.17**	37.97	**43.39**	40.00	39.72	45.08	**49.15**
A → D (SURF)	25.48	**26.75**	39.49	31.85	40.13	38.85	38.85	**42.04**
w → C (SURF)	19.86	18.25	31.17	**31.52**	32.06	**33.04**	33.57	32.95
w → A (SURF)	22.96	21.92	32.78	30.48	32.99	32.15	34.13	31.94
w → D (SURF)	59.24	**59.87**	89.17	**89.81**	89.17	**90.45**	89.81	**89.81**
D → C (SURF)	26.27	26.09	31.52	31.43	33.39	**33.57**	34.64	**37.04**
D → A (SURF)	28.50	**29.33**	33.09	32.78	33.72	**34.03**	34.45	**35.28**
D → W (SURF)	63.39	**65.08**	89.49	88.47	89.49	**90.51**	91.19	**91.19**
Average accuracy	31.37	**31.43**	46.31	**46.75**	47.21	**48.07**	48.96	**50.06**
Average improvement		**0.06**		**0.44**		**0.86**		**1.10**
Relative improvement		**0.09**		**0.82**		**1.63**		**2.16**

Table 4
Classification accuracy (%) on the MNIST+USPS tasks.

Tasks/Methods	NN	NN+TSRP	JDA	JDA+TSRP	BDA	BDA+TSRP	DICD	DICD+TSRP
USPS → MNIST	35.85	35.30	59.65	**62.40**	60.05	**62.40**	61.50	**67.55**
MNIST → USPS	64.44	**70.72**	67.28	**72.44**	69.89	**74.06**	73.28	**73.39**
Average accuracy	50.15	**53.01**	63.46	**67.42**	64.97	**68.23**	67.39	**70.47**
Average improvement		**2.86**		**3.96**		**3.26**		**3.08**
Relative improvement		**5.74**		**10.84**		**9.30**		**9.45**

Table 5
Classification accuracy (%) on the COIL20 tasks.

Tasks/Methods	NN	NN+TSRP	JDA	JDA+TSRP	BDA	BDA+TSRP	DICD	DICD+TSRP
COIL1 → COIL2	84.72	**88.75**	93.75	**95.14**	93.89	**96.53**	94.58	**95.69**
COIL2 → COIL1	83.33	83.19	92.64	**94.86**	93.33	**95.56**	93.47	**98.06**
Average accuracy	84.03	**85.97**	93.19	**95.00**	93.61	**96.04**	94.03	**96.88**
Average improvement		**1.94**		**1.81**		**2.43**		**2.85**
Relative improvement		**12.17**		**26.53**		**38.04**		**47.67**

algorithms have been improved. Specifically, among the 12 tasks, DICD+TSRP, BDA+TSRP, JDA+TSRP, and NN+TSRP outperforms their original counterparts in 9, 8, 7, and 6 tasks respectively.

Tables 4 and 5 list the classification accuracy on MNIST+USPS and COIL20 datasets. From the tables, we observe that JDA+TSRP, BDA+TSRP and DICD+TSRP outperform their original counterparts on all tasks. Particularly, DICD+TSRP achieves a relative improvement of 47.67% over DICD on COIL20. Table 6 shows the results on the Office+Caltech-256 (DECAF6 features) dataset.

We see from the table that the average accuracy of NN+TSRP, JDA+TSRP, BDA+TSRP and DICD+TSRP is 4.19%, 2.52%, 2.58%, 2.59% higher than that of NN, JDA, BDA and DICD respectively. Particu-

Table 6
Classification accuracy (%) on the Office+Caltech-256 (decaf6 features) tasks, where A = AMAZON, C = CALTECH, D = DSLR AND W = WEBCAM.

Tasks/Methods	NN	NN+TSRP	JDA	JDA+TSRP	BDA	BDA+TSRP	DICD	DICD+TSRP
C → A ($DeCAF_6$)	85.70	**88.31**	89.77	**92.38**	90.61	**92.38**	91.02	**92.38**
C → W ($DeCAF_6$)	66.10	**80.68**	83.73	**88.14**	84.07	**89.83**	92.20	**94.24**
C → D ($DeCAF_6$)	74.52	**83.44**	86.62	**90.45**	87.90	**90.45**	93.63	**94.90**
A → C ($DeCAF_6$)	70.35	**73.11**	82.28	**84.24**	83.17	**84.51**	86.02	**87.89**
A → W ($DeCAF_6$)	57.29	**62.37**	78.64	**87.80**	78.64	**88.47**	81.36	**89.49**
A → D ($DeCAF_6$)	64.97	**70.06**	80.25	**85.35**	84.71	**90.45**	83.44	**92.36**
W → C ($DeCAF_6$)	60.37	**60.37**	83.53	82.90	83.53	**83.53**	83.97	**87.09**
W → A ($DeCAF_6$)	62.53	**66.91**	90.19	**91.23**	90.50	**91.44**	89.67	**90.40**
W → D ($DeCAF_6$)	98.73	**98.73**	100.00	**100.00**	100.00	**100.00**	100.00	**100.00**
D → C ($DeCAF_6$)	52.09	49.24	85.13	**86.82**	85.22	**86.82**	86.11	**88.33**
D → A ($DeCAF_6$)	62.73	**64.61**	91.44	**92.59**	91.54	**92.69**	92.17	**93.63**
D → W ($DeCAF_6$)	89.15	**96.95**	98.98	**98.98**	98.98	**99.32**	98.98	**98.98**
Average accuracy	70.38	**74.57**	87.55	**90.07**	88.24	**90.82**	89.88	**92.47**
Average improvement		**4.19**		**2.52**		**2.58**		**2.59**
Relative improvement		**14.14**		**20.24**		**21.98**		**25.59**

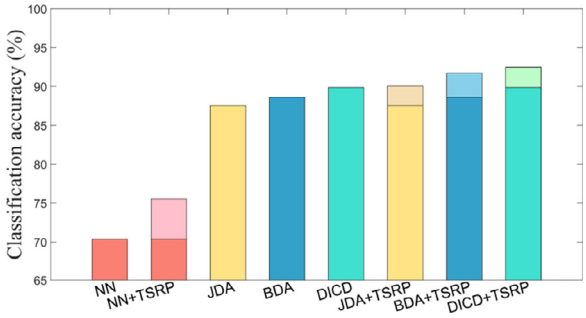


Fig. 4. Performance summary of the domain adaptation baselines and their TSRP extensions on Office+Caltech-256 in terms of average classification accuracy.

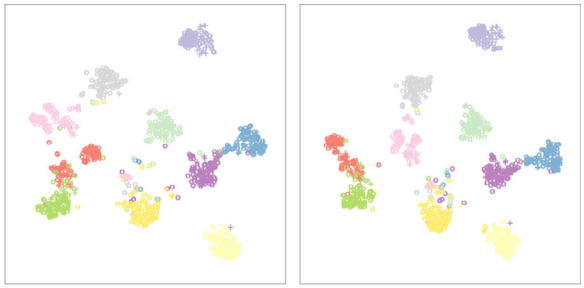


Fig. 5. Visualization of the features produced by the comparison methods on the "A → D (DeCAF6 features)" task of the Office+Caltech-256 data. Different colors represent different categories. The source samples are marked by the symbol "o". The target samples are marked by "+".

larly, BDA+TSRP and DICD+TSRP outperform their original counterparts on all tasks. Both JDA+TSRP and NN+TSRP outperform their counterparts in 11 out of 12 tasks. In order to better show the advantage of TSRP, we further draw the average accuracy of the comparison methods in Fig. 4 in the ascending order. From Fig. 4, we observe an interesting phenomenon that, although DICD considers more information than JDA and BDA, such as conditional distribution alignment and discriminative learning, JDA and BDA can still yield better result than DICD by generating more accurate pseudo labels. This phenomenon indicates that the accuracy of the pseudo labels has an important impact on performance. It also indicates that the accuracy of the pseudo labels can be improved by TSRP. The result also shows that TSRP can help learn better domain invariance features.

To demonstrate how the proposed method improves the alignment of the source and target domains directly, we visualize the representations produced by JDA and JDA+TSRP on the "A → D (DeCAF6 features)" task of the Office+Caltech-256 data in Fig. 5. From the figure, we see that TSRP+JDA has a better aligned conditional distribution than JDA.

4.4. Analysis

The results in Tables 2 to 6 have illustrated the effectiveness and generalization of TSRP, which in turn demonstrates that exploring the intra-class similarity of the target domain can remedy pseudo labels well. In this subsection, we conducted several analytical experiments to further verify the effectiveness of TSRP.

4.4.1. Effect of UTSP on performance

To demonstrate the effectiveness of UTSP in picking the pseudo labels with high confidence, we compared the performance of TSRP with the proposed UTSP and a variant of UTSP that does not use the spanning tree, denoted as UTSP_N. The result is shown in Fig. 6. From the figure, we can see that the TSRP without the spanning tree can still lead to a small improvement, while adding the spanning tree into TSRP can produce significantly better results. Because the main job of UTSP is to select highly confident pseudo labels, the improved performance not only supports the effectiveness of UTSP, but also reflects the effectiveness of the selected highly confident pseudo labels in promoting the alignment of the domains.

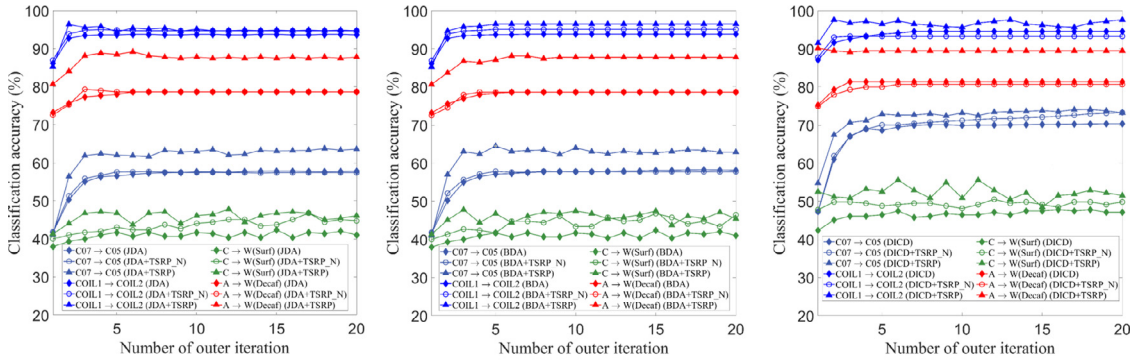
4.4.2. Effect of the highly confident pseudo labels on the classifier

To demonstrate the effectiveness of the selected highly confident pseudo labels in improving the performance of the classifier, we compared the classifier (denoted as "strong classifier") with the one that is trained with the source samples (denoted as "original classifier") only. The result is shown in Fig. 7. From the figure, we see that the performance of the strong classifier is much better than the original classifier. It indicates that TSRP can help learn good domain-invariant and discriminative features, and in turn, the features can help TSRP refine the pseudo labels, which is a co-promotion process.

From Fig. 7, we can also find that, no matter how many iterations are conducted, the baseline methods without TSRP are upper-bounded due to the low accuracy of the pseudo labels. On the other side, the proposed methods with TSRP can break through such limit.

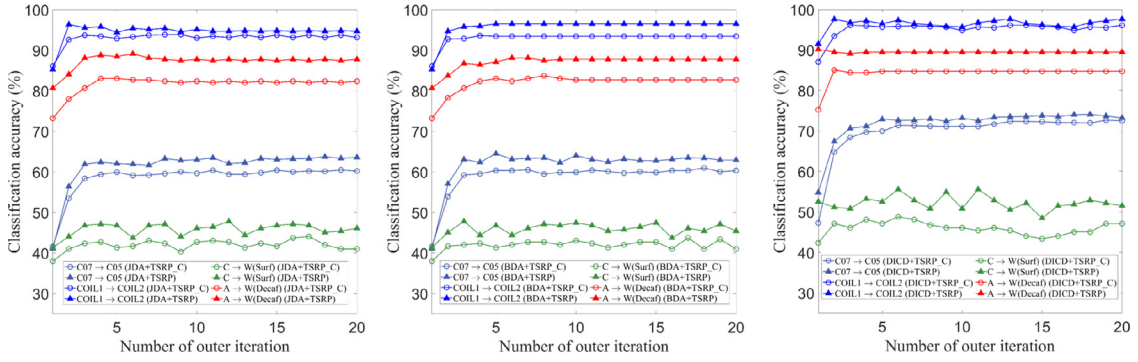
4.4.3. Effects of hyper-parameters on performance

Our approach consists of two hyper-parameters: The trust parameter ρ and the number of inner iterations IT. Here we take



(a) Validation with JDA (b) Validation with BDA (c) Validation with DICD

Fig. 6. Effect of UTSP on the performance of the Office+Caltech-256 datasets with respect to the optimization iterations. Different colors represent different domain adaptation tasks. The term "TSRP_N" denotes that UTSP does not contain the *spanning tree* step.



(a) Validation with JDA (b) Validation with BDA (c) Validation with DICD

Fig. 7. Effect of the strong classifier in TSRP on the performance of the Office+Caltech-256 datasets with respect to the optimization iterations. The term "TSRP_C" means that TSRP adopts the original classifier trained with the source data only, instead of the strong classifier.

DICD+TSRP as an example to study how the two hyperparameters affect the performance. The experiment was conducted on the tasks C27 → C29, C → W (SURF), W → C (DeCAF6), COIL2 → COIL1 and USPS → MNIST. The results are reported in Fig. 9. Specifically, Fig. 9a shows the effect of IT when ρ was set to 0.9 and IT was selected from $\{1, 2, \ldots, 10\}$. From the figure, we see obviously that our approach is not sensitive to IT. Figure 9b shows the effect of ρ when $IT = 4$ and ρ was selected from $\{0, 0.1, 0.2, \ldots, 1\}$. The result indicates that, although the the proposed method is relatively sensitive to the trust parameter ρ for each single task, it reaches the best performance on all tasks when $\rho \approx 0.8$.

4.5. Comparison with other domain adaptation methods

To further illustrate the effectiveness of TSRP, we compared DICD+TSRP with several conventional UDA methods, which are GFK [24], TCA [14], TJM [25], TSL [26], DTSL [27], CDML [28], RTML [29], and DICD [16], respectively. We also compared with a standard machine learning methods, i.e., PCA [23]. In order to make a fair comparison, the results of the comparison methods are from their public codes or the original papers. Table lists the comparison results on the Office+Caltech-256 (DECAF6 features) dataset. From the table, we can see that DICD+TSRP achieves the best performance in 10 out of 12 tasks, and ranks the second in the other 2 tasks. The results on the CMU-PIE and Office+Caltech-256 (surf features) datasets are listed in the supplementary material. The experimental conclusion is similar to that on the Office+Caltech-256 (DECAF6 features) dataset (Table 7).

We also compared DICD+TSRP with three recent deep-learning-based UDA methods, which are DAN [51], JAN [52], and DWT-MEC [53], respectively, on the Office-Home dataset. From the comparison results in Table 8, we see that the average classification accuracy of DICD+TSRP is 2.8% higher than that of DICD on this dataset, which is similar to the phenomena on the other datasets. DICD+TSRP also outperforms the deep-learning-based methods. For example, its average accuracy is 7.5% higher than JAN.

4.6. Comparison on transformed features

In Section, we only studied the situation that the domain adaptation algorithms use the original data or their deep features. In real-world scenarios, domain adaptation also faces scenarios that the data is transformed into other formats. Here we study such a problem with cycle generative adversarial networks (CycleGAN) [54].

Specifically, we first trained CycleGAN with both source and target samples, which could produce two generators. One generator maps the source domain S to the target domain T, which yields transformed samples of a source domain S_T that is similar to the target samples. The other generator maps the target domain T to the source domain S, which yields transformed samples of a target domain T_S that is similar to the source samples. Then, for each domain adaptation task, we could at least formulate three domain adaptation subtasks given the data representations produced by CycleGAN. For example, for the original task "$S \to T$", the three subtasks could be "$S \to T$", "$S_T \to T$", and "$S \to T_S$". For the original task of "$T \to S$", the three subtasks could be "$T \to S$" "$T_S \to S$", "$T \to S_T$" (Table 9).

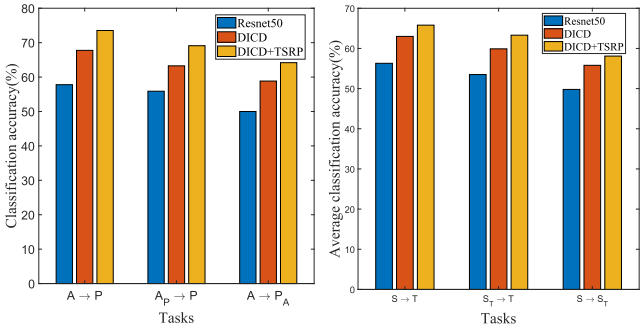
We conducted an evaluation on the Office-Home dataset. Given the four domains of Office-Home, we trained 6 CycleGAN models,

Table 7
Accuracy comparison of the proposed DICD+TSRP with representative UDA methods on the Office+Caltech-256 (Decaf6 features) dataset, where A = Amazon, C = Caltech, D = DSLR, and W = WEBCAM. The best result is marked in bold. The runner-up result is underlined.

Tasks/Methods	PCA	GFK	TCA	TJM	DTSL	CDML	RTML	DICD	DICD+TSRP
C → A(DeCAF$_6$)	88.10	87.79	89.46	89.46	91.54	86.24	90.62	91.02	**92.38**
C → W(DeCAF$_6$)	74.24	70.17	78.64	78.98	76.61	77.82	85.38	92.20	**94.24**
C → D(DeCAF$_6$)	83.44	88.54	81.53	85.35	87.90	83.74	89.32	93.63	**94.90**
A → C(DeCAF$_6$)	73.02	75.78	79.61	79.07	85.75	79.54	86.43	86.02	**87.89**
A → W(DeCAF$_6$)	57.63	76.95	73.22	76.95	73.56	76.27	80.26	81.36	**89.49**
A → D(DeCAF$_6$)	70.06	84.08	84.71	**85.35**	82.17	81.35	84.36	83.44	**92.36**
W → C(DeCAF$_6$)	63.58	75.07	78.09	76.49	72.75	77.64	83.13	83.97	**87.09**
W → A(DeCAF$_6$)	70.15	82.88	83.30	86.74	75.47	86.29	**91.37**	89.67	90.40
W → D(DeCAF$_6$)	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	98.42	**100.00**	**100.00**	**100.00**
D → C(DeCAF$_6$)	57.88	73.11	79.70	78.63	75.24	78.56	85.72	86.11	**88.33**
D → A(DeCAF$_6$)	68.16	85.18	88.52	89.77	84.97	89.47	91.86	92.17	**93.63**
D → W(DeCAF$_6$)	88.14	90.85	98.98	97.97	**99.32**	96.38	98.98	98.98	98.98
Average accuracy	74.53	82.53	84.65	85.40	83.77	84.31	88.95	89.88	**92.47**

Table 8
Classification accuracy (%) of the proposed DICD+TSRP with three deep-learning-based UDA methods on the target domains of the Office-Home dataset.

Tasks/Methods	DAN	JAN	DWT-MEC	DICD	DICD+TSRP
A → C	43.6	45.9	50.3	47.4	**50.4**
A → P	57.0	61.2	72.1	67.8	**73.5**
A → R	67.9	68.9	**77.0**	70.2	74.1
C → A	45.8	50.4	59.6	56.8	**60.5**
C → P	56.5	59.7	69.3	64.5	**70.1**
C → R	60.4	61.0	**70.2**	65.3	69.7
P → A	44.0	45.8	58.3	62.3	**63.4**
P → C	43.6	43.4	48.1	49.8	**50.2**
P → R	67.7	70.3	77.3	74.7	**77.5**
R → A	63.1	63.9	**69.3**	66.3	67.2
R → C	51.5	52.4	**53.6**	53.2	53.1
R → P	74.3	76.8	**82.0**	77.6	79.8
Average accuracy	56.3	58.3	65.6	63.0	**65.8**



(a) A representative case (b) Average results

Fig. 8. Performance of TSRP with transformed features on the Office-Home dataset, where the transformed features are generated by CycleGAN. (a) A case study on the three subtasks of the "A → P" domain adaptation task. (b) The average results over all 36 subtasks. The subscript represents the domain generated by CycleGan.

and further extracted deep features from both the original samples and the transformed samples by ResNet50. For each of the 12 domain adaptation tasks, we compared DICD+TSRP with the method without domain adaptation (denoted as ResNet50) and the domain adaptation method DICD on the aforementioned three subtasks. Fig. 8 reports the average results over all 36 subtasks, as well as a case study on the three subtasks of the "A → P" domain adaptation task. From the figure, we get a shared conclusion across the tasks: A domain adaptation algorithm with the proposed TSRP leads to higher accuracy than the algorithm without TSRP, regardless whether the features are extracted from the original samples or from the transformed samples.

Note that, although it is expected that CycleGAN may alleviate the domain shift problem, the transformed samples generated by

Table 9
Accuracy comparison (%) of the proposed DICD+TSRP with representative UDA methods on the CMU-PIE dataset.

Tasks/Methods	PCA	GFK	TCA	TSL	DTSL	CDML	RTML	DICD	DICD+TSRP
C05 → C07	24.80	26.15	40.76	44.08	65.87	53.22	60.12	72.99	**74.52**
C05 → C09	25.18	27.27	41.79	47.49	64.09	53.12	55.21	72.00	**76.16**
C05 → C27	29.26	31.15	59.63	62.78	82.03	80.12	85.19	92.22	**96.52**
C05 → C29	16.30	17.59	29.35	36.15	54.90	48.23	52.98	66.85	**71.32**
C07 → C05	24.22	25.24	41.81	46.28	45.04	52.39	58.13	69.93	**73.20**
C07 → C09	45.53	47.37	51.47	57.60	53.49	54.23	63.92	65.87	**67.83**
C07 → C27	53.35	54.25	64.73	71.43	71.43	68.36	76.16	85.25	**86.42**
C07 → C29	25.43	27.08	33.70	35.66	47.97	37.34	40.38	48.31	**56.43**
C09 → C05	20.95	21.82	34.69	36.94	52.49	43.54	53.12	69.36	**70.14**
C09 → C07	40.45	43.16	47.70	47.02	55.56	54.87	58.67	65.44	**73.05**
C09 → C27	46.14	46.41	56.23	59.45	77.50	62.76	69.81	83.39	**94.26**
C09 → C29	25.31	26.78	33.15	36.34	54.11	38.21	42.13	61.40	**66.48**
C27 → C05	31.96	34.24	55.64	63.66	81.54	75.12	81.12	93.13	**94.72**
C27 → C07	60.96	62.92	67.83	72.68	85.39	80.53	83.92	90.12	**92.88**
C27 → C09	72.18	73.35	75.86	83.52	82.23	83.72	89.51	88.97	**90.26**
C27 → C29	35.11	37.38	40.26	44.79	72.61	52.78	56.26	75.61	**79.11**
C29 → C05	18.85	20.35	26.98	33.28	52.19	27.34	29.11	62.88	**73.68**
C29 → C07	23.39	24.62	29.90	34.13	49.41	30.82	33.28	57.03	**65.81**
C29 → C09	27.21	28.49	29.90	36.58	58.45	36.34	39.85	65.87	**70.10**
C29 → C29	30.34	31.33	33.64	38.75	64.31	40.61	47.13	74.77	**77.84**
Average accuracy	33.85	35.35	44.75	49.43	63.53	53.68	58.80	73.09	**77.84**

Table 10
Accuracy comparison (%) of the proposed DICD+TSRP with representative UDA methods on the Office+Caltech-256 (surf features), MNIST+USPS, and COIL20 datasets.

Tasks/Methods	PCA	GFK	TCA	TJM	DTSL	CDML	RTML	DICD	DICD+TSRP
C → A (SURF)	36.95	41.02	38.20	46.76	**51.25**	47.82	49.26	47.29	47.81
C → W (SURF)	32.54	40.68	38.64	38.98	38.64	36.91	44.72	46.44	**50.85**
C → D (SURF)	38.22	38.85	41.40	44.59	47.13	43.93	47.56	49.68	**50.96**
A → C (SURF)	34.73	40.25	37.76	39.45	43.37	41.72	**43.68**	42.39	41.76
A → W (SURF)	35.59	38.98	37.63	42.03	36.61	38.25	44.32	45.08	**49.15**
A → D (SURF)	27.39	36.31	33.12	**45.22**	38.85	35.92	43.86	38.85	42.04
W → C (SURF)	26.36	30.72	29.30	30.19	29.83	31.14	**34.83**	33.57	32.95
W → A (SURF)	31.00	29.75	30.06	29.96	34.13	32.26	**35.28**	34.13	31.94
W → D (SURF)	77.07	80.89	87.26	89.17	82.80	84.84	**91.02**	89.81	89.81
D → C (SURF)	29.65	30.28	31.70	31.43	30.11	32.63	34.58	34.64	**37.04**
D → A (SURF)	32.05	32.05	32.15	32.78	32.05	29.87	33.26	34.45	**35.28**
D → W (SURF)	75.93	75.59	86.10	85.42	72.20	82.34	89.68	**91.19**	**91.19**
USPS → MNIST	44.95	46.45	51.05	52.25	55.50	52.25	61.82	61.50	**67.55**
MNIST → USPS	66.22	67.22	56.28	63.28	52.33	63.28	69.52	73.28	**73.39**
COIL1 → COIL2	84.72	72.50	88.47	91.53	88.61	88.93	91.23	94.58	**95.69**
COIL2 → COIL1	84.03	74.17	85.83	91.81	89.17	87.32	90.22	93.47	**98.06**
Average accuracy	47.34	48.48	50.31	53.43	51.41	51.84	56.55	56.90	**58.46**

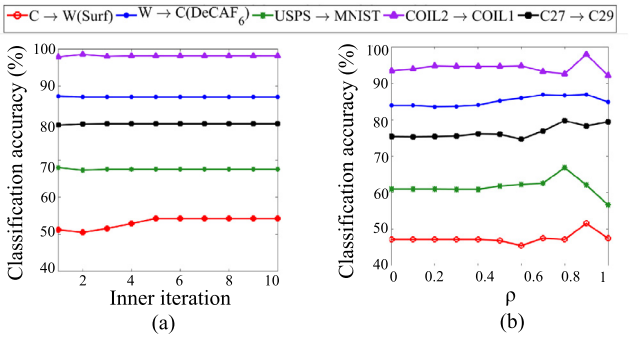


Fig. 9. Effect of the hyperparameters on five domain adaptation tasks. Different colors represent different domain adaptation tasks.

CycleGAN may not match the ResNet50 feature extractor, which accounts for the performance degradation of the comparison methods on the transformed data, compared with that on the original data (Table 10).

5. Conclusion

In this paper, we have proposed to use target domain intra-class similarity to remedy pseudo labels (TSRP) for improving the accuracy of the coarse pseudo labels that are generated from a conventional UDA method, which in turn improves the discriminant ability of the learned representation of the UDA. Specifically, TSRP first exploits the intra-class similarity and spanning trees to pick samples with high confident pseudo labels. Then, it trains a strong classifier with both the source samples and the target samples whose pseudo labels are highly-confident. Finally, it uses the strong classifier to remedy the pseudo labels of the target samples with low-confident pseudo labels. Experimental results on extensive visual cross-domain tasks have shown that applying TSRP to conventional UDA methods can improve the accuracy of the pseudo labels and further lead to more discriminative and domain invariant features than the conventional UDA baselines. In the future, we intend to extend the idea of this paper to deep domain adaptation.

Declaration of Competing Interest

We declare that there is no conflict of interests.

Data availability

The source code has been released at https://github.com/02Bigboy/TSRP.

Acknowledgments

This work was supported in part by National Science Foundation of China under Grant No. 62176211, in part by Project of the Science, Technology, and Innovation Commission of Shenzhen Municipality under grant No. JSGG20210802152546026 and JCYJ20210324143006016.

References

[1] S.J. Pan, Q. Yang, A survey on transfer learning, IEEE Trans. Knowl. Data Eng. 22 (10) (2010) 1345–1359, doi:10.1109/TKDE.2009.191.
[2] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, Q. He, A comprehensive survey on transfer learning, Proc. IEEE 109 (1) (2020) 43–76, doi:10.1109/JPROC.2020.3004555.
[3] J. Liang, D. Hu, J. Feng, Domain adaptation with auxiliary target domain-oriented classifier, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16632–16642.
[4] W. Wang, H. Wang, Z. Zhang, C. Zhang, Y. Gao, Semi-supervised domain adaptation via Fredholm integral based kernel methods, Pattern Recognit. 85 (2019) 185–197, doi:10.1016/j.patcog.2018.07.035.
[5] Y. Zhang, B. Deng, K. Jia, L. Zhang, Label propagation with augmented anchors: a simple semi-supervised learning baseline for unsupervised domain adaptation, in: European Conference on Computer Vision, Springer, 2020, pp. 781–797, doi:10.1007/978-3-030-58548-8_45.
[6] B. Fernando, A. Habrard, M. Sebban, T. Tuytelaars, Unsupervised visual domain adaptation using subspace alignment, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2013.
[7] W. Zellinger, T. Grubinger, E. Lughofer, T. Natschläger, S. Saminger-Platz, Central moment discrepancy (CMD) for domain-invariant representation learning, arXiv preprint arXiv:1702.08811(2017). 10.48550/arXiv.1702.08811
[8] J. Li, M. Jing, K. Lu, L. Zhu, H.T. Shen, Locality preserving joint transfer for domain adaptation, IEEE Trans. Image Process. 28 (12) (2019) 6103–6115, doi:10.1109/TIP.2019.2924174.
[9] Q. Wang, T.P. Breckon, Cross-domain structure preserving projection for heterogeneous domain adaptation, Pattern Recognit. 123 (2022) 108362, doi:10.1016/j.patcog.2021.108362.
[10] J. Li, K. Lu, Z. Huang, L. Zhu, H.T. Shen, Heterogeneous domain adaptation through progressive alignment, IEEE Trans. Neural Netw. Learn. Syst. 30 (5) (2019) 1381–1391, doi:10.1109/TNNLS.2018.2868854.
[11] W. Wang, H. Li, Z. Ding, F. Nie, J. Chen, X. Dong, Z. Wang, Rethinking maximum mean discrepancy for visual domain adaptation, IEEE Trans. Neural Netw. Learn. Syst. (2021), doi:10.1109/TNNLS.2021.3093468.
[12] Y. Zhu, F. Zhuang, J. Wang, G. Ke, J. Chen, J. Bian, H. Xiong, Q. He, Deep subdomain adaptation network for image classification, IEEE Trans. Neural Netw. Learn. Syst. 32 (4) (2020) 1713–1722, doi:10.1109/TNNLS.2020.2988928.
[13] N. Xiao, L. Zhang, Dynamic weighted learning for unsupervised domain adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 15242–15251.

[14] S.J. Pan, I.W. Tsang, J.T. Kwok, Q. Yang, Domain adaptation via transfer component analysis, IEEE Trans. Neural Networks 22 (2) (2011) 199–210, doi:10.1109/TNN.2010.2091281.
[15] Y.-W. Luo, C.-X. Ren, P. Ge, K.-K. Huang, Y.-F. Yu, Unsupervised domain adaptation via discriminative manifold embedding and alignment, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 5029–5036.
[16] S. Li, S. Song, G. Huang, Z. Ding, C. Wu, Domain invariant and class discriminative feature learning for visual domain adaptation, IEEE Trans. Image Process. 27 (9) (2018) 4260–4273, doi:10.1109/TIP.2018.2839528.
[17] Y. Chen, Y. Pan, Y. Wang, T. Yao, X. Tian, T. Mei, Transferrable contrastive learning for visual domain adaptation, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 3399–3408.
[18] M. Long, J. Wang, G. Ding, J. Sun, P.S. Yu, Transfer feature learning with joint distribution adaptation, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 2200–2207.
[19] X. He, P. Niyogi, Locality preserving projections, Adv. Neural Inf. Process. Syst. 16 (2003).
[20] C.J. Alpert, T.C. Hu, J.-H. Huang, A.B. Kahng, D. Karger, Prim-Dijkstra tradeoffs for improved performance-driven routing tree design, IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. 14 (7) (1995) 890–896, doi:10.1109/43.391737.
[21] K. Fukunaga, P.M. Narendra, A branch and bound algorithm for computing k-nearest neighbors, IEEE Trans. Comput. 100 (7) (1975) 750–753, doi:10.1109/T-C.1975.224297.
[22] J. Wang, Y. Chen, S. Hao, W. Feng, Z. Shen, Balanced distribution adaptation for transfer learning, in: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 1129–1134.
[23] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemom. Intell. Lab. Syst. 2 (1–3) (1987) 37–52.
[24] B. Gong, Y. Shi, F. Sha, K. Grauman, Geodesic flow kernel for unsupervised domain adaptation, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 2066–2073, doi:10.1109/CVPR.2012.6247911.
[25] M. Long, J. Wang, G. Ding, J. Sun, P.S. Yu, Transfer joint matching for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1410–1417.
[26] S. Si, D. Tao, B. Geng, Bregman divergence-based regularization for transfer subspace learning, IEEE Trans. Knowl. Data Eng. 22 (7) (2009) 929–942, doi:10.1109/TKDE.2009.126.
[27] Y. Xu, X. Fang, J. Wu, X. Li, D. Zhang, Discriminative transfer subspace learning via low-rank and sparse representation, IEEE Trans. Image Process. 25 (2) (2015) 850–863, doi:10.1109/TIP.2015.2510498.
[28] H. Wang, W. Wang, C. Zhang, F. Xu, Cross-domain metric learning based on information theory, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
[29] Z. Ding, Y. Fu, Robust transfer metric learning for image classification, IEEE Trans. Image Process. 26 (2) (2016) 660–670, doi:10.1109/TIP.2016.2631887.
[30] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: International Conference on Machine Learning, PMLR, 2015, pp. 1180–1189.
[31] X. Gu, J. Sun, Z. Xu, Spherical space domain adaptation with robust pseudo-label loss, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9101–9110.
[32] G. Kang, L. Jiang, Y. Yang, A.G. Hauptmann, Contrastive adaptation network for unsupervised domain adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4893–4902.
[33] Y. Zhang, Y. Zhang, Y. Wei, K. Bai, Y. Song, Q. Yang, Fisher deep domain adaptation, in: Proceedings of the 2020 SIAM International Conference on Data Mining, SIAM, 2020, pp. 469–477, doi:10.1137/1.9781611976236.53.
[34] J. Zhang, W. Li, P. Ogunbona, Joint geometrical and statistical alignment for visual domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1859–1867.
[35] Z. Pei, Z. Cao, M. Long, J. Wang, Multi-adversarial domain adaptation, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[36] Y. Pan, T. Yao, Y. Li, Y. Wang, C.-W. Ngo, T. Mei, Transferrable prototypical networks for unsupervised domain adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2239–2247.
[37] Z. Zheng, Y. Yang, Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation, Int. J. Comput. Vis. 129 (4) (2021) 1106–1120, doi:10.1007/s11263-020-01395-y.
[38] Y. Zou, Z. Yu, X. Liu, B. Kumar, J. Wang, Confidence regularized self-training, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 5982–5991.
[39] Y. Zhang, B. Deng, K. Jia, L. Zhang, Label propagation with augmented anchors: a simple semi-supervised learning baseline for unsupervised domain adaptation, in: European Conference on Computer Vision, Springer, 2020, pp. 781–797.
[40] Q. Wang, T. Breckon, Unsupervised domain adaptation via structured prediction based selective pseudo-labeling, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 6243–6250, doi:10.1609/aaai.v34i04.6091.
[41] L. Tian, Y. Tang, L. Hu, Z. Ren, W. Zhang, Domain adaptation by class centroid matching and local manifold self-learning, IEEE Trans. Image Process. 29 (2020) 9703–9718, doi:10.1109/TIP.2020.3031220.
[42] C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, J. Huang, Progressive feature alignment for unsupervised domain adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 627–636.
[43] Q. Wang, P. Bu, T.P. Breckon, Unifying unsupervised domain adaptation and zero-shot visual recognition, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.
[44] T. Sim, S. Baker, M. Bsat, The CMU pose, illumination, and expression (pie) database, in: Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition, IEEE, 2002, pp. 53–58.
[45] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324, doi:10.1109/5.726791.
[46] J.J. Hull, A database for handwritten text recognition research, IEEE Trans. Pattern Anal. Mach. Intell. 16 (5) (1994) 550–554, doi:10.1109/34.291440.
[47] G. Griffin, A. Holub, P. Perona, Caltech-256 object category dataset(2007).
[48] S.A. Nene, S.K. Nayar, H. Murase, et al., Columbia object image library (coil-100)(1996).
[49] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.
[50] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, DeCAF: a deep convolutional activation feature for generic visual recognition, in: International Conference on Machine Learning, PMLR, 2014, pp. 647–655.
[51] M. Long, Y. Cao, J. Wang, M. Jordan, Learning transferable features with deep adaptation networks, in: International Conference on Machine Learning, PMLR, 2015, pp. 97–105.
[52] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 2208–2217.
[53] S. Roy, A. Siarohin, E. Sangineto, S.R. Bulo, N. Sebe, E. Ricci, Unsupervised domain adaptation using feature-whitening and consensus loss, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9471–9480.
[54] J.-Y. Zhu, T. Park, P. Isola, A.A. Efros, Unpaired image-to-image translation using cycle-consistent adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2223–2232.

Jie Wang received the BS degree from Electronic and Communication Engineering, Northwestern Polytechnical University. He is currently pursuing the MS degree in Electronic and Communication Engineering at Northwestern Polytechnical University. His research interests are machine learning and data mining.

Xiao-Lei Zhang received the PhD degree in information and communication engineering from Tsinghua University, Beijing, China, in 2012. He is currently a Full Professor with the Center for Intelligent Acoustics and Immersive Communications, and the School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an, China. He was a Postdoctoral Researcher with Perception and Neurodynamics Laboratory, The Ohio State University, Columbus, OH, USA. His research interests include machine learning, statistical signal processing, and artificial intelligence.