Neural Processing Letters (2024) 56:162
https://doi.org/10.1007/s11063-024-11627-8

Dual Classifier Adaptation: Source-Free UDA via Adaptive Pseudo-Labels Learning

Yunyun Wang[1] · Qinghao Li[1] · Ziyi Hua[1]

Accepted: 16 April 2024

Abstract

Different from Unsupervised Domain Adaptation (UDA), Source-Free Unsupervised Domain Adaptation (SFUDA) transfers source knowledge to target domain without accessing the source data, using only the source model, has attracted much attention recently. One mainstream SFUDA method fine-tunes the source model by self-training to generate pseudo-labels of the target data. However, due to the significant differences between different domains, these target pseudo-labels often contain some noise, and it will inevitably degenerates the target performance. For this purpose, we propose an innovative SFUDA method with adaptive pseudo-labels learning named Dual Classifier Adaptation (DCA). In DCA, a dual classifier structure is introduced to adaptively learn target pseudo-labels by cooperation between source and target classifiers. Simultaneously, the minimax entropy is introduced for target learning, in order to adapt target data to source model, while capture the intrinsic cluster structure in target domain as well. After compared our proposed method DCA with a range of UDA and SFUDA methods, DCA achieves far ahead performance on several benchmark datasets.

Keywords Unsupervised domain adaptation · Pseudo-label learning · Source-free unsupervised domain adaptation · Dual classifier learning

1 Introduction

The application of deep neural networks(DNN) in many fields has made extraordinary progress, including image recognition [1], semantic segmentation [2], etc. However, the development of DNN has encountered a bottleneck, which is the need for huge labeled data as support in the training process, but obtaining such data is a time-consuming and costly task. If we can train a model on limited labeled data and successfully transfer this knowledge to unlabeled data, it will help solve the problem of scarcity of labeled data. In addition, another

✉ Yunyun Wang
wangyunyun@njupt.edu.cn

Qinghao Li
1021041212@njupt.edu.cn

Ziyi Hua
1221045508@njupt.edu.cn

1 School of Computer, Software and Cyberspace Security, Nanjing University of Posts and Telecommunications, Xianlin Street, Nanjing 210023, Jiangsu, China

Published online: 04 May 2024

challenge of DNN is their poor generalization ability across different data domains. To solve these problems, researchers have proposed unsupervised domain adaptation (UDA) methods, which aim to transfer knowledge from a source domain, rich in labels yet distinct, to a target domain where labels are scarce. Current UDA methods attempt to learn domain-invariant features [3–7], mainly by minimizing the distribution discrepancy between domains [4, 5, 8], or adopting adversarial learning strategies [6, 7, 9].

However, as the data privacy and security are gaining more and more attention, existing UDA methods need the source data as support during learning, thus could violate data privacy policies. Therefore, a more challenging but realistic task, namely source-free unsupervised domain adaptation (SFUDA), has recently attracted considerable attention. It differs from ordinary UDA tasks in that it just provides a trained source model for adaptation, in order to protect the privacy of the source domain. In addition, the source model consumes less storage space than the source data, which helps to save storage space cost for domain adaptation tasks and make data transmission more convenient.

Current SUFDA methods can predominantly fall into two categories: generation-based methods [10, 11] and self training-based methods [12–14]. Generative methods transfer knowledge in an adversarial manner by generating target-similar samples for source domain. Although generative methods have achieved some success, the data generation process usually consumes a lot of time and computing resources. Self-training methods directly fine-tune the source model by exploring the model outputs on target data. They commonly attempt to seek accurate pseudo-labels for target data, and have achieved SOTA performance in literature. For example, SHOT [12] introduces a pseudo-labeling strategy based on weighted K-means clustering over feature prototypes, and BMD [15] proposes a dynamic updating strategy for pseudo-labels during adaptation. However, due to the different distributions between the two domains, the prediction from source model for target data inevitably contains noise, that is, it will produces incorrect pseudo-labels, which may introduce negative transfer from source domain. At the same time, those methods commonly transfer source discriminative knowledge by aligning target features with the fixed source classifier, thus some intrinsic structure knowledge in target domain can be ignored.

In order to learn the target structure knowledge and alleviate the negative impact of noise samples, we propose an innovative approach for SFUDA, named Dual Classifier Adaptation (DCA) via adaptive pseudo-labels learning. Specifically, although the source classifier inherits the discriminative knowledge from source data, the target pseudo-labels by source classifier are still unreliable with distribution discrepancy across domains. As a result, we introduce a dual-classifier structure including an additional target classifier in DCA. The source classifier is freezed to keep the source knowledge, while the target classifier is updated in terms of target data. Both classifiers share the same feature extractor for knowledge transfer, which is also updated in learning. Finally, the discrepancy between the two classifiers is adopted as the weight to adjust the pseudo-labels generation. That is, we regard the pseudo-labels with smaller differences as more reliable ones, and weaken the negative impact caused by the unreliable pseudo-labels with larger differences. At the same time, we use minimax entropy to optimize both target classifier and feature extractor, i.e., the entropy is maximized over the target classifier while minimized on the feature extractor, in order to generate features that fit the source domain boundaries, while capture the intrinsic cluster structure in target domain. Finally, the mixup method was used to make the two classifiers learn from each other, thus promote each other as well.

The contributions of this paper are summarized as follows:

(1) A dual classifier structure is introduced for SFUDA in DCA, in order to adaptively learn target pseudo-labels by cooperation of two classifiers with source discriminative and target structure knowledge.
(2) The minimax entropy learning is introduced into the target classifier in DCA, in order to promote the features of the target data to better form a cluster structure near the target prototype.
(3) Empirical experiments on three adaptation tasks demonstrate the superiority and effectiveness of DCA.

2 Related Work

2.1 Unsupervised Domain Adaptation (UDA)

In recent years, UDA has attracted the attention of many researchers which aims to transfer knowledge from the labeled source domain to the unlabeled target domain. And the main UDA methods can be divided into three categories: The first is the difference-based method, which match higher moments of the distribution by using Maximum mean discrepancy (MMD) [16] or uses the feature set for clustering to improve the intra-class diversity and the accuracy of pseudo-labels, and then align the feature distributions between samples [17]. The second is the contrastion-based methods [18, 19], which bolster the overall class-level structure by reducing intra-class distances while simultaneously expanding inter-class distances, or use the memory bank [20, 21] to supply both class-level information from the source domain and instance-level information from the target domain, thus facilitating contrastive learning. The third is the adversarial-based methods [22, 23], which employ adversarial training to learn domain invariant features, effectively mitigating the discrepancies between the source and target domains. Although the above methods have also achieved some achievements, they need the source data as support during domain adaptation, which could lead to data privacy or security issues.

2.2 Source-Free Unsupervised Domain Adaptation(SFUDA)

The difference between SFUDA [24] and UDA is that in the process of domain adaptation, SFUDA does not have the support of source data, only the trained source model is available. Current SFUDA methods have been developed in many forms, which are defined by Li et al. [25] into two categories: data-based methods and model-based methods. In the data-based methods, the mainstream generation-based methods, such as MA [10], CPGA [11] and ASM [26] introduce a generative network to generate data similar to the source domain or target domain, but this method is inefficient and cannot guarantee the quality of the generated images. There are neighborhood clustering-based methods, such as G-SFDA [14] and NRC++ [27], to enforce consistency constraints by capturing the intrinsic cluster structure of the target data, but these methods only consider the consistency of the same cluster, and ignore the difference of different clusters. There are also intra-domain adversarial methods, such as BAIT [28], which introduces an additional bait classifier and uses the output entropy of the two classifiers for intra-domain adversarial alignment, but it ignores the certainty of the classifier output, which may lead to ambiguous output problems. In the model-based methods, feature prototype-based self-training methods have received more attention [12, 13, 29]. Among them, SHOT [12] introduces K-means algorithm to generate cluster centers, and uses the cluster centers as feature prototypes to assign pseudo-labels to the target data. However, since there are inevitably partial errors in the pseudo-labels, incorrect pseudo-labels can easily affect the adaptation process, which is also the problem we want to mainly solve. Therefore, we propose a novel method to optimize pseudo-labels with the dual classifier, and collaborate with minimax entropy to jointly train the feature extractor and target classifier, thus mitigating the adverse effects of noisy pseudo-labels.

2.3 Pseudo-Label Learning

Pseudo-label [30], first proposed by semi-supervised learning, have become popular in other related fields, such as domain adaptation (DA). The main approach is to make predictions on unlabeled data to obtain pseudo-labels and use these pseudo-labels to help train the target model, which is proven to be very effective. In the DA method, Zhang et al. [31, 32] directly use pseudo-labels as regularizers, and Zou et al. [33] design an ensemble framework that alternates between solving for target pseudo-labels and performing model training. In the UDA method, Pang et al. [34] proposed a concept based on probabilistic stability, assessing the prediction probabilities from both determinacy and stability perspectives to refine the reliability modeling of pseudo labels. Furthermore, DeepCluster [35] is a very advanced self-supervised pseudo-labeling method, generating pseudo-labels via k-means clustering and retraining the current model. However, pseudo-label learning has an inherent weakness that the pseudo-labels themselves contain noisy predictions. Although most pseudo-labels are accurate, it is inevitable that there will be false labels, and fine-tuning the model on noisy labels may affect subsequent training. Therefore, our method evaluates the reliability of generated pseudo-labels by leveraging the output differences between two classifiers and uses the output differences as a penalty term to penalize the unreliable pseudo-labels, thereby reducing the negative impact of unreliable pseudo-labels during the adaptation process.

3 Method

This paper is dedicated to solving the K-class image classification task and mainly follows the setting of SHOT [12] in terms of problem definition. We denote the labeled source domain data with n_s samples as $D_s = \left\{\left(x_s^i, y_s^i\right)\right\}_{i=1}^{n_s}$, where $x_s^i \in \mathcal{X}_s$, $y_s^i \in \mathcal{Y}_s$, and the unlabeled target domain data with n_t samples as $D_t = \{x_t^j\}_{j=1}^{n_t}$ where $x_t^j \in \mathcal{X}_t$. Our goal is to predict the label $\{y_t^j\}_{j=1}^{n_t}$ of the target domain data using the source model f_s, which is pre-trained on the source domain data. Different from the usual UDA methods, SFUDA's method setting is more challenging in that only the pre-trained source model f_s is accessed when adapting to the target data. In the following, we will successively divide into different chapters to detail the methods we applied in SFUDA.

3.1 Overall Scheme

Inspired by the fact that the difference in model output can be expressed as prediction variance [36], we explore ways to evaluate pseudo-label quality using the difference in output between two different classifiers and use them for domain alignment of SFUDA. As shown in Fig. 1, the proposed DCA method consists of two stages: source model pre-training and target model domain adaptation.

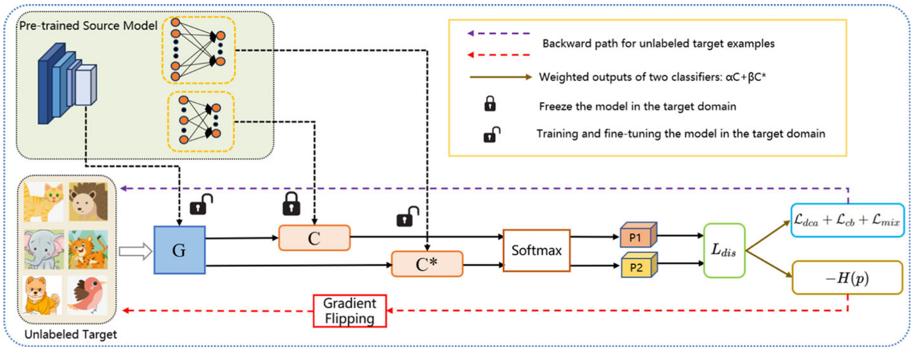

Pre-trained Source Model
Backward path for unlabeled target examples
Weighted outputs of two classifiers: αC+βC*
Freeze the model in the target domain
Training and fine-tuning the model in the target domain
G
C
C*
Softmax
P1
P2
$\mathcal{L}_{dis}$
$\mathcal{L}_{dca}+\mathcal{L}_{cb}+\mathcal{L}_{mix}$
$-H(p)$
Gradient Flipping
Unlabeled Target

Fig. 1 Overview of DCA. G denotes the feature extractor, C denotes the source classifier which has been frozen, and C* denotes the target classifier which continues training. In addition, P1, P2 represent the Softmax outputs of the two classifiers

In the stage one (Sect. 3.2), we follow the basic form of related SFUDA work [11, 12] and first we use the cross-entropy loss function $\mathcal{L}_{ce}$ to train the source data to generate the source model f_s.

In the stage two (Sects. 3.3 and 3.4), we initially freeze the source classifier f_s and continue to train the target classifier f_t. The primary goal is to fine-tune the source model to adapt to the target domain by learning target features. Specifically, we employ minimax entropy $H(p)$ cross training to jointly optimize the target classifier f_t and the feature extractor g, enabling f_t to better capture target domain features. Additionally, we introduce a novel pseudo label loss $\mathcal{L}_{dca}$, which leverages the output differences between the two classifiers to adaptively refine the pseudo labels of the target data, thereby achieving better classification alignment. Finally, to ensure the model outputs more balanced target features, we further apply class balance loss $\mathcal{L}_{cb}$ and mixup loss $\mathcal{L}_{mix}$.

3.2 Source Model Generation

In SFUDA, considering issues such as data security and privacy, we only have the source model available. Therefore, our first work is to use the source data to generate the source model. In related work, the basic framework of neural networks can be decomposed into three parts: feature extractor, bottleneck layer, and classifier head. However, in our work, we have introduced an additional target classifier that is directly connected to the feature extractor without passing through the bottleneck layer. In addition, the classifier is also composed of fully connected layers and has weight normalization. Our neural network framework is shown in Fig. 2. As for why an additional target classifier needs to be introduced, detailed explanations will be provided in Sect. 3.4.

We first use standard cross-entropy(CE) loss and label smoothing [37] techniques to train the source model to correctly classify the source data with labels(see Eq. 1):

$$\mathcal{L}_{ce} = -\mathbb{E}_{(x_s,y_s)} \sum_{k=1}^{K} q_k^{ls} \log \delta_k \left(f_s\left(x_s\right)\right) \tag{1}$$

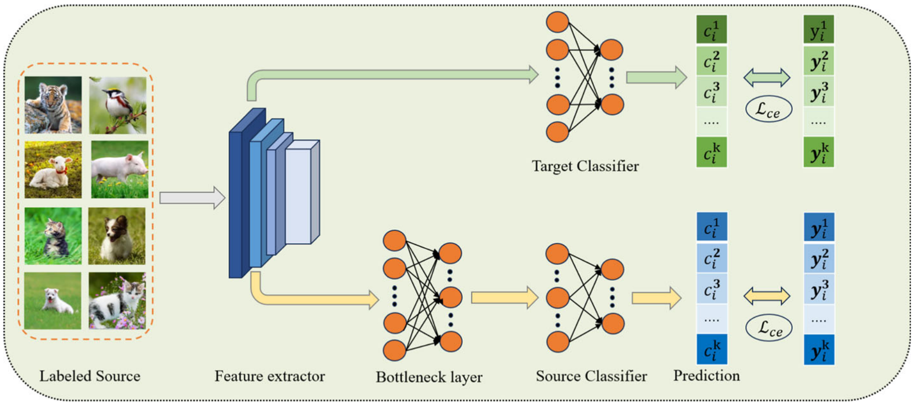


Fig. 2 Our neural network framework. First, the feature extractor extracts features from the labeled source data, and then the obtained features are fed into two classifiers. The difference is that the features go through the bottleneck layer for dimensionality reduction before entering the source classifier. In addition, we use cross-entropy loss $\mathcal{L}_{ce}$ to train the source data

where the $\delta_k(a) = \frac{\exp(a_k)}{\sum_i \exp(a_i)}$ is the k-th element of the softmax output, and q is the one-of-K encoding of y_s where q_k is '1' for the correct class and '0' for the rest, and $q_k^{ls} = (1-\alpha)q_k + \alpha/K$ is the smoothed label where α is set to 0.1.

During adaptation, unlike the SHOT [12] method, we only freeze the source classifier and continue to train target classifier and fine tune feature extractors. The purpose of the frozen source classifier is to retain the discriminative knowledge learned in the source domain so that it can be used as a source-like prototype in the feature space of the target domain. However, the benefit of training a target classifier is that the specific knowledge of the target domain can be further mined. Since the two classifiers share the feature extractor, we further optimize the feature extractor, so that the target classifier can learn more features far from the boundary of the source domain in the target domain.

3.3 Target Learning with Minimax Entropy

In the domain adaptation stage, we freeze the source classifier, so it only retains the knowledge of the source domain, the source classifier tends to classify the samples with similar sources in the target data. However, due to the difference between the distributions of the two domains, the source classifier is difficult to capture the complete target features. In order to further mine more information of the target domain, we need to train the target classifier to better identify the source dissimilar samples.

3.3.1 Training the Target Classifier by Maximizing Entropy

In UDA, we usually train our model so that it can well match the distribution of the target and source data. Nevertheless, it is difficult to estimate the distribution of source data by relying only on the source model. In this case, we can treat the weights of the classifier as the class prototypes and compute the entropy on the target data to represent the similarity between the target features and the class prototypes.

Since a uniform high-entropy output distribution indicates that the target sample is similar to the weight vector of the classifier, we first train the target classifier using entropy maximization. Maximum entropy can make the weight vector of the target classifier shift towards the direction of the target domain, so as to encourage the target classifier to learn part of the features of the target domain. Furthermore, we want the two classifiers not to deviate too far from each other either, to encourage both classifiers to still have consistent predictions for source similar features. Thus we use symmetric KL divergence to maintain partial consistency of the outputs of the two classifiers. The formula used is as follows:

$$\hat{\theta}_{C_2} = \underset{\theta_{C_2}}{\operatorname{argmin}} \mathcal{L}_{skl} - \lambda H \tag{2}$$

$$H = -\mathbb{E}_{(\mathbf{x},y)} \sum_{k=1}^{K} p_k \log p_k \tag{3}$$

$$\mathcal{L}_{skl}(C_2) = \frac{1}{2}\left(D_{kl}(C_1 \mid C_2) + D_{kl}(C_2 \mid C_1)\right) \tag{4}$$

where λ is a hyper-parameter that controls the minimax entropy training, D_{kl} is the KL divergence and p_k represents the softmax output of target classifier C_2.

3.3.2 Training the Feature Extractor by Minimizing Entropy

In addition to training the object classifier, we also consider training the feature extractor. The difference is that in this step we minimize the entropy of the unlabeled data instead of maximizing the entropy. The aim is to make the target samples better clustered around the class prototypes. The formula is as follows:

$$\hat{\theta}_G = \underset{\theta_G}{\operatorname{argmin}} \mathcal{L}_{cb} + \lambda H \tag{5}$$

where λ is a hyper-parameter that controls the minimax entropy training, our ultimate goal is to obtain discriminative features from unlabeled target data and prompt them to cluster around the class prototype of the classifier. In addition, to avoid assigning all uncertain features in the target domain to class prototypes [38], we adopt the class-balanced loss to optimize the feature extractor so that it can output more balanced target features. The formula for class-balanced loss is as follows:

$$\mathcal{L}_{cb} = \sum_{k=1}^{K} \hat{p}_k \log \hat{p}_k \tag{6}$$

where $\hat{p} = \mathbb{E}_{x_t \in \mathcal{X}_t}\left[\delta\left(f_t\left(x_t\right)\right)\right]$ is the Softmax output embedding based on the average of all target data.

3.3.3 Mixup Training for Target Data

Due to domain shift, it is difficult for a model trained in the source domain to directly match the data distribution in the target domain. Therefore, we consider using mixup [39] technology to expand the target domain data, and use mixed data to train our model. This further improves the robustness of our model to noisy labels as well as the generalization ability. Many Mixup training methods have been successfully applied to the UDA task [39–41]. Inspired by these methods, we first perform a mixup operation on the target data to generate new data:

$$\begin{aligned} \tilde{x}_{mix} &= \lambda x_t^1 + (1-\lambda) x_t^2 \\ \tilde{y}_{mix} &= \lambda \hat{y}_1 + (1-\lambda) \hat{y}_2 \end{aligned} \tag{7}$$

where $\hat{y}$ is the presudo label of x_t, $\lambda \in \mathrm{Beta}(\alpha, \alpha)$, and α is empirically set to 0.3. Then we adopt the KL divergence to constraint the mixup loss:

$$\mathcal{L}_{mix} = \mathrm{KL}\left(\tilde{y}_{mix} \| \delta\left(f_t\left(\tilde{x}_{mix}\right)\right)\right) \tag{8}$$

This loss function can expand the target data, simultaneously, we can use the output of two classifiers for Mixup training, so that the target model can obtain better generalization ability and robustness.

3.4 Rectifying Pseudo-Labels via Classifier Discrepancy

3.4.1 Self-supervised Pseudo-Labeling Strategy

Because of the lack of annotation on the target data, it is impractical to directly realize the alignment between domains by only relying on the features of the target output. So we use the same strategy as SHOT [12] to generate pseudo-labels: First, we attain the centroid for each class in the target domain:

$$c_k = \frac{\sum_{x_t \in \mathcal{X}_t} \delta_k\left(\hat{f}_t\left(x_t\right)\right) \hat{g}_t\left(x_t\right)}{\sum_{x_t \in \mathcal{X}_t} \delta_k\left(\hat{f}_t\left(x_t\right)\right)} \tag{9}$$

where $\hat{f}_t = \hat{g}_t \circ \hat{c}_t$ is the model output and $\delta_k(\hat{f}_t\left(x_t\right))$ denotes the softmax output of target data x_t belonging to the k-th class. Then by calculating the distance between the target x_t and the centroid, the closest centroid is selected to assign the pseudo label $\hat{y}_t$ to x_t as:

$$\hat{y}_t = \arg\min_k D_f\left(\hat{g}_t\left(x_t\right), c_k\right) \tag{10}$$

where $D_f(a, b)$ is the method used to measure the cosine distance. In addition, we update the centroid of each class by $c_k = \frac{\sum_{x_t \in \mathcal{X}_t} \mathbb{I}(\hat{y}_t = k)\hat{g}_t(x_t)}{\sum_{x_t \in \mathcal{X}_t} \mathbb{I}(\hat{y}_t = k)}$ and update pseudo-labels based on Eq. 10 in each epoch, where $\mathbb{I}(\cdot)$ is an indicator function.

Note that in this step, we utilize both classifiers to obtain the pseudo-labels on the target data simultaneously. Since the initialization weights of two classifiers are different and the target classifier is trained on the target data, the ability of the two classifiers to learn noisy labels is different.

3.4.2 Classifiers Discrepancy

Since the classification weights learned by different classifiers in the target domain are different, the decision boundaries produced by them in the target domain are also somewhat different. Therefore, even for the same target sample, there will be some differences in their output. This results in that for different classifier models, the pseudo-labels obtained by similar source samples or most clean samples are basically consistent, while inconsistent pseudo-labels are obtained by dissimilar source samples or noisy samples [42, 43]. This also shows that the output difference between the classifiers is the cause of the inconsistency of the pseudo-labels obtained by the two classifiers. In this paper, we argue that the output difference between classifiers can also be used to measure the stability of noisy labels.

We consider how to obtain two different classifiers. In A^{2}Net [44], the source classifier model is first initialized with two identical source and target classifiers. Then in order to distinguish the two classifiers, it adds some Gaussian noise to the initial weights of the target classifier. However, an obvious disadvantage of adding noise is that the accuracy of the target classifier will be greatly reduced, thus making the pseudo-labels obtained by it mostly wrong.

For this reason, we introduce an additional target classifier when training the source model, which is initialized differently from the source model because it is directly connected to the backbone layer without going through the bottleneck layer. Its benefit lies in that is that the trained target classifier is also able to correctly classify the source data. Because the initial weights learned by the two classifiers on the source data are different, the ability of the two classifiers to learn noisy labels on the target data will be different. Therefore, the two classifiers tend to output the same prediction for source similar samples, while the output for source dissimilar samples has some difference. Considering how to measure the output difference between classifiers, we use the KL distance as the metric:

$$D_{kl}(P \| Q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)} \tag{11}$$

where $p(x), q(x)$ is the softmax output of classifiers. We also considered using the L1-distance, but in our experiments, we found that the difference of classifiers measured based on KL divergence can better indicate the inconsistency of classifier outputs. So finally we used Eq. 11 to measure the difference between classifier outputs.
Discussion: There are three main reasons for the discrepancy between the source and target classifiers. Firstly, the main reason is that the two classifiers have different receptive fields. As shown in Fig. 2, since there is a bottleneck layer between the source classifier and the backbone layer, the source classifier can learn features from deeper layers. This results in different input activations between the two classifiers, resulting in prediction differences. Secondly, during target domain training, the source classifier is fixed while the target classifier continues training. Therefore, when a batch of the same target data is used as input, the outputs obtained by the two classifiers may have some differences. Third, in the network before the classifier layer, we applied the dropout function [45], which also leads to different predictions of the two classifiers during training.

3.4.3 Rectify Pseudo-Labels Method

Consider to the distribution difference between the source domain and the target domain, the pseudo-labels generated by self-supervision are not reliable, and the pseudo-labels containing noise may introduce negative transfer during the training process. For this reason, DCA is proposed to use classifier differences to dynamically adjust pseudo-labels. Our aim is to utilize reliable pseudo-labels for training, while weakening the negative impact brought by unreliable pseudo-labels in training. So how to identify the reliability of pseudo-labels?

As mentioned earlier, the output of the two classifiers for the target data is different. The most direct manifestation of this output difference is the difference in the ability of two classifiers to learn noisy labels for the same target data. For source similar samples or clean samples in the target domain, both classifiers tend to output the same pseudo-labels. On the contrary, they tend to output different pseudo-labels for different source samples or noisy samples. We believe that for the pseudo-labels output by two classifiers, the same pseudo-labels can be considered as the more reliable pseudo-labels, while the different pseudo-labels can be considered as less reliable. That is, we can use the output difference between classifiers to measure the reliability of pseudo-labels. When the output difference between classifiers is small, we encourage reliable pseudo-labels for training; However, when the output difference between classifiers is large, the obtained unreliable pseudo-labels may harm the transfer performance of the model, so we weaken the unreliable pseudo-labels for training. First, we present the CE loss function for pseudo-labels training:

$$\mathcal{L}_{\text{ce}} = \mathbb{E}_{(x_t, \hat{y}_t) \in \mathcal{X}_t \times \hat{\mathcal{Y}}_t} \sum_{k=1}^{K} \mathbb{I}_{[k=\hat{y}_t]} \log \delta_k \left(f_t \left(g_t \left(x_t \right) \right) \right) \tag{12}$$

Then we use KL distance(Eq. 11) to measure the output difference of the classifiers. And modify the Eq. 12 as shown in Eq. 13:

$$\mathcal{L}_{dca} = \mathbb{E} \left[\exp \left\{ -D_{kl} \right\} \mathcal{L}_{\text{ce}} + D_{kl} \right] \tag{13}$$

When the output difference of the classifier is small, it can be considered that the pseudo-labels obtained at this time is relatively accurate, and $\exp\{-D_{KL}\}$ is close to 1, and the formula is equivalent to the Eq. 12. On the contrary, when the output difference of the classifier is large, it can be considered that the pseudo-labels obtained at this time is relatively inaccurate, and then $\exp\{-D_{KL}\}$ will become smaller, so as to weaken the effect of CE loss. So as to prevent the output of the classifier from being too different, we add D_{KL} as a penalty term to play the role of regularization. Its benefit is that, in contrast to existing approaches [12], instead of considering aligning the entire distribution of the target and source domains, we consider selecting target samples with more consistent outputs from both classifiers to update the feature generator to partially align the distributions between the two domains. This can help us filter out part of the unreliable pseudo-labels, and then reduce the negative impact of noise pseudo-labels.

3.4.4 Overall Objective

The above descriptions introduce our proposed methods separately and illustrate how these methods can be applied to SFUDA. In addition, the training of our model mainly involves updating two modules: (the feature generator G and the target classifier C_2), with the overall objective as follows, where λ, η and β are trade-off parameters to balance losses:

$$\min_{C_2} \mathcal{L}_{dis} - \lambda H \tag{14}$$

$$\min_{G} \eta \mathcal{L}_{dca} + \beta \mathcal{L}_{mix} + \lambda (\mathcal{L}_{cb} + H) \tag{15}$$

To implement the adaptive adversarial operation, we optimize the two modules alternately in an iterative manner. Firstly, the source class classifier C_1 is frozen and the target classifier C_2 is trained by Eq. 14. Second, Eq. 15 is used to optimize the feature extractor. The whole domain adaptation process repeats the above two steps until we reach convergence or maximum epoch.

4 Experiment

4.1 Setup

In order to compare the innovativeness and superiority of DCA over other baseline methods, we evaluate DCA in mainstream unsupervised domain adaptation scenarios, mainly including the following three popular vision benchmarks. Our source code is available at https://github.com/MALA-NJUPT/DCA.

Office-31 [46] is a standard dataset commonly used in unsupervised domain adaptation (UDA) and soure-free unsupervised domain adaptation (SFUDA) research. The dataset contains A total of 4652 images of office objects divided into three domains(A,W,D), and each domain has 31 categories.
Office-Home [47] is also a medium-sized dataset commonly used in UDA and SFUDA research, which is relatively more challenging. It contains a total of 15,500 images, which are divided into four different domains(Ar,Cl,Pr,Rw), and each domain has 65 categories.
VisDA-C [48] is a large-scale vision dataset for UDA and SFUDA. This dataset consists of two main parts: a training set and a validation set. The training set contains around 280,000 synthetic images selected from the ImageNet dataset, which cover 12 main categories including aircraft, bicycles, buses, cars, horses, knives, motorcycles, people, plants, treadmachines, trucks, and monitors. The validation set contains around 55,400 real-world images, which are also classified into the 12 categories mentioned above.

4.1.1 Baseline Methods

We compared our method DCA with ordinary UDA methods that can access source data and SFUDA methods that have no source data, respectively. UDA methods including DANN [49], MCD [23], BNM [50], CDAN [51] and SRDC [52]. The recent SFUDA methods including SHOT [12], MA [10], CPGA [11], A^{2}Net [44], BMD [15], DMCD [53], BAIT [28], NRC++ [27], ASM [26].

The above comparison methods can be roughly divided into four categories:

(1) SHOT [12] and BMD [15] are both self-training methods, which use pseudo-labels to supervise the target data for training. SHOT [12] additionally uses information maximization to learn the target domain features, while BMD [15] designs an intra-class multi-center clustering strategy to generate prototypes to dynamically adjust the pseudo-labels.
(2) MA [10], CPGA [11] and ASM [26] are all based on generative methods, which use the generated data to train the model to achieve domain alignment. Different from MA [10], CPGA [11] additionally introduces contrastive learning while ASM [26] additionally uses pseudo-labels to maintain semantic consistency.
(3) A^{2}Net [44], DMCD [53] and BAIT [28] are all adversarial methods, and they all use multiple classifiers to achieve adversarial alignment in the domain.
(4) NRC++ [27] is based on domain clustering, which explores the inherent clustering structure of the target domain data to learn the potential structural information.

4.2 Implementation Details

4.2.1 Network Architecture

For all classification tasks, we use the pre-trained ResNet-50 [54] model as the backbone module for the Office dataset, and the pre-trained ResNet-101 model as the backbone module for the VisDA-C dataset. In the next step, referring to SHOT [12], we replace a normal fully connected (FC) layer with a bottleneck layer that can output 256-dimensional features and a fully connected classifier layer. However, we also directly connected a target classifier in the backbone layer without adding a bottleneck layer. Furthermore, we used a weight normalization layer in the classifier fully connected layer.

4.2.2 Network Hyper-Parameters

Our network is trained by backpropagation, and the learning rate of the newly added bottleneck layer as well as the classifier layer is 10 times that of the backbone network layer. Concretely, we adopt minibatch SGD with momentum 0.9 and weight decay $5e^{-4}$ and learning rate $\eta_0 = 1e^{-2}$ for the newly added layers except $\eta_0 = 1e^{-3}$ for VisDA-C. We further adopt the same learning rate scheduler $\eta = \eta_0 \cdot (1 + 10 \cdot p)^{-0.75}$ as CDAN [51], where p is the training progress changing from 0 to 1. In addition, we set the batch size to 64 for all experiments. During the training of the source model, the maximum number of epochs for Office, Office-Home and VisDA-C is empirically set as 20, 40 and 10, respectively. During the adaptation of the target domain, we update the pseudo-labels several times. Meanwhile, the maximum number of iterations is empirically set to 20, 15, 10. In addition, we set the following hyperparameters $\eta = 0.05$ for Office and $\eta = 0.3$ for VisDA-C, $\beta = 0.3$ for Office and $\beta = 1.0$ for VisDA-C, $\lambda = 0.1$ for Office and $\lambda = 0.5$ for VisDA-C.

4.3 Comparison Results

As shown in Tables 1, 2 and 3, we divide these tables into two parts, the upper part shows the UDA method and results, and the lower part shows the source-free method and results. In addition, to distinguish the two methods more clearly, we use $\times$ to denote the UDA method and $\surd$ to denote the SFUDA method in the Source-free column. Meanwhile, our proposed DCA method in this paper will be placed in the last row to compare with all methods. In order to better evaluate the superiority between different methods, we use the commonly used Accuracy(ACC) index to evaluate the classification effect of the model. ACC in the classification task refers to the accuracy of the model, which is the proportion of the number of samples correctly predicted by the model to the total number of samples.

As shown in Table 1, the proposed DCA method is far ahead of other baseline methods, in terms of average performance of six transfer tasks on the Office-31 dataset. At the same time, DCA achieves 1.9% improvement over the method SHOT [12] which also applies self-supervised pseudo-labels. In addition, DCA achieves the best results in the tasks A $\rightarrow$ D and W $\rightarrow$ D, indicating the superiority of our method.

As shown in Table 2, the proposed DCA method is far ahead of other baseline methods, in terms of average performance of 12 transfer tasks on the Office-Home dataset. At the same time, DCA achieves 1.1% improvement over the method SHOT [12] which also applies self-supervised pseudo-labels. In particular, our method DCA achieves excellent results on the effects of seven of these tasks, thus demonstrating the superiority of our method.

Table 1 Accuracies (%) on Office-31 dataset

Method	Source-free	A → D	A → W	D → A	D → W	W → A	W → D	Average
DANN [49]	×	79.7	82.0	68.2	96.9	67..4	99.1	82.2
MCD [23]	×	92.2	88.6	69.5	98.5	69.7	100.0	86.5
BNM [50]	×	90.3	91.5	70.9	98.5	71.6	100.0	87.1
CDAN [51]	×	92.9	94.1	71.0	98.6	69.3	100.0	87.7
SRDC [52]	×	95.8	95.7	76.7	99.2	77.1	100.0	90.8
SHOT [12]	√	94.0	90.1	74.7	98.4	74.3	99.9	88.6
MA [10]	√	92.7	93.7	75.3	98.5	**77.8**	99.8	89.6
CPGA [11]	√	94.4	94.1	76.0	98.4	76.6	99.8	89.9
A^2Net [44]	√	94.5	94.0	**76.7**	**99.2**	76.1	<u>100.0</u>	90.1
BMD [15]	√	95.6	93.0	75.6	97.5	75.0	99.8	89.4
DMCD [53]	√	94.1	93.5	<u>76.4</u>	98.8	76.4	100.0	89.9
BAIT [28]	√	92.0	94.6	74.6	98.1	75.2	100.0	89.1
NRC++ [27]	√	<u>95.9</u>	91.2	75.5	<u>99.1</u>	75.0	100.0	89.5
ASM [26]	√	95.1	**96.0**	75.3	98.7	<u>77.2</u>	100.0	<u>90.4</u>
DCA(ours)	√	**96.2**	<u>95.6</u>	76.0	98.7	76.6	**100.0**	**90.5**

The backbone network is ResNet-50. Source-free means that the domain adaptation process does not allow access to the source data, while only the source model is available. We use bold to mark the best result and underline to indicate the second highest result

As shown in Table 3, the proposed DCA achieves the second best result in terms of average accuracy, that is, per-class accuracy, on the more challenging VisDA-C dataset. In particular, our method DCA achieves 1.8% improvement compared to the baseline method SHOT [12] which also applies self-supervised pseudo-labels, so that proves the superiority of our proposed method.

In general, compared with a variety of UDA and SFUDA baseline methods, our method DCA achieves two optimal and one suboptimal performance on three public datasets.

The self-supervised pseudo-label strategy proposed by SHOT [12] is a simple and effective strategy, while due to the existence of domain shift, there are inevitably wrong labels in the obtained pseudo-labels, which seriously affects the performance of model classification. Different from BMD [15]'s strategy of using intra-class multi-center clustering to optimize pseudo-labels, we cleverly introduce an additional target classifier into DCA and use the difference between the two classifiers to adaptively optimize the pseudo-labels, thus alleviating the negative impact of some noisy pseudo-labels, supplemented by other strategies to optimize the model, thus significantly improving the classification performance of the model. However, the negative impact caused by noisy labels is inevitable, and we find that this negative impact is further amplified on large datasets. Therefore, how to better optimize the pseudo-labeling strategy is the direction of our future efforts.

Table 2 Accuracies (%) on Office-Home dataset (used ResNet-50)

Method	Source-free	Ar → Cl	Ar → Pr	Ar → Rw	Cl → Ar	Cl → Pr	Cl → Rw	Pr → Ar	Pr → Cl	Pr → Rw	Rw → Ar	Rw → Cl	Rw → Pr	Avg
DANN [49]	×	45.6	59.3	70.1	47.0	58.5	60.9	46.1	43.7	68.5	63.2	51.8	76.8	57.6
MCD [23]	×	48.9	68.3	74.6	61.3	67.6	68.8	57.0	47.1	75.1	69.1	52.2	79.6	64.1
BNM [50]	×	52.3	73.9	80.0	63.3	72.9	74.9	61.7	49.5	79.7	70.5	53.6	82.2	67.9
CDAN [51]	×	50.7	70.6	76.0	57.6	70.0	70.0	57.4	50.9	77.3	70.9	56.7	81.6	65.8
SRDC [52]	×	52.3	76.3	81.0	69.5	76.2	78.0	68.7	53.8	81.7	76.3	57.1	85.0	71.3
SHOT [12]	√	57.1	78.1	81.5	68.0	78.2	78.1	67.4	54.9	82.2	73.3	58.8	84.3	71.8
CPGA [11]	√	59.3	78.1	79.8	65.4	75.5	76.4	65.7	**58.0**	81.0	72.0	**64.4**	83.3	71.6
A^2Net [44]	√	58.4	79.0	82.4	67.5	79.3	78.9	68.0	56.2	82.9	74.1	60.5	85.0	72.8
BMD [15]	√	55.9	77.8	80.8	69.7	79.3	**79.9**	**69.6**	56.6	82.6	73.3	59.5	85.1	72.5
DMCD [53]	√	**59.4**	78.9	80.2	67.2	79.3	78.6	65.3	55.6	82.2	73.3	62.8	83.9	72.2
BAIT [28]	√	57.4	77.5	82.4	68.0	77.2	75.1	67.1	55.5	81.9	73.9	59.5	84.2	71.6
NRC++ [27]	√	57.8	**80.4**	81.6	69.0	**80.3**	79.5	65.6	57.0	**83.2**	72.3	59.6	**85.7**	72.5
ASM [26]	√	56.9	79.1	**82.9**	69.5	79.6	79.7	67.9	55.1	82.6	**74.7**	60.5	84.8	72.8
DCA(ours)	√	58.1	80.1	82.7	**69.7**	79.3	79.7	68.4	55.3	82.5	74.1	59.7	85.4	**72.9**

Table 3 Accuracies (%) on VisDA-C dataset (used ResNet-101)

Method	Source-free	Plane	Bicycle	Bus	Car	Horse	Knife	Motorcycle	Person	Plant	Skateboard	Train	Truck	Per-class
Resnet-101 [54]	×	55.1	53.3	61.9	59.1	80.6	17.9	79.7	31.2	81.0	26.5	73.5	8.5	52.4
DANN [49]	×	81.9	77.7	82.8	44.3	81.2	29.5	65.1	28.6	51.9	54.6	82..8	7.8	57.4
MCD [23]	×	87.0	60.9	83.7	64.0	88.9	79.6	84.7	76.9	88.6	40.3	83.0	25.8	71.9
CDAN [51]	×	85.2	66.9	83.0	50.8	84.2	74.9	88.1	74.5	83.4	76.0	81.9	38.0	73.9
SWD [55]	×	90.8	82.5	81.7	70.5	91.7	69.5	86.3	77.5	87.4	63.6	85.6	29.2	76.4
MCC [56]	×	88.7	80.3	80.5	71.5	90.1	93.2	85.0	71.6	89.4	73.8	85.0	36.9	78.8
MA [10]	√	94.8	73.4	68.8	**74.8**	93.1	95.4	88.6	**84.7**	89.1	84.7	83.5	48.1	81.6
SHOT [12]	√	94.3	88.5	80.1	57.3	93.1	94.9	80.7	80.3	91.5	89.1	86.3	58.2	82.9
CPGA [11]	√	**95.6**	**89.0**	75.4	64.9	91.7	**97.5**	**89.7**	83.8	**93.9**	**93.4**	87.7	**69.0**	**86.0**
A^{2}Net [44]	√	94.0	87.8	**85.6**	66.8	93.7	95.1	85.8	81.2	91.6	88.2	86.5	56.0	84.3
BAIT [28]	√	93.7	83.2	84.5	65.0	92.9	95.4	88.2	80.8	90.0	89.0	84.0	45.3	82.7
ASM [26]	√	95.2	87.8	79.7	60.3	94.1	94.8	85.0	81.1	91.9	89.9	87.3	61.6	84.1
DCA(ours)	√	95.1	88.5	82.7	62.9	**94.1**	96.4	83.9	81.4	90.4	89.8	**89.6**	61.7	84.7

Table 4 Influence of the hyper-parameters λ, η and β in terms of per-class accuracy (%) on VisDA-C

Hyper-parameters	λ					η					β				
	0.1	0.3	0.5	1.0	1.5	0.1	0.2	0.3	0.4	0.5	0.1	0.3	0.5	1.0	2.0
Accuracy (per-class)	83.4	83.9	**84.7**	84.3	83.8	84	84.3	**84.7**	84.3	84.1	83.9	84.2	84.4	**84.7**	84.5

Table 5 Accuracy (%) on Office-31 ablation study for DCA

Method	A → D	A → W	D → A	D → W	W → A	W → D	Average
Source-only model	82.5	78.8	62.7	94.7	64.0	98.6	80.2
$\mathcal{L}_{mme}$	90.3	90.7	65.0	97.2	65.9	99.5	84.8
$\mathcal{L}_{mme} + \mathcal{L}_{cb}$	93.0	92.6	68.5	98.3	69.6	99.9	87.0
$\mathcal{L}_{mme} + \mathcal{L}_{cb} + \mathcal{L}_{mix}$	93.8	93.0	70.1	98.3	70.8	99.9	87.7
$\mathcal{L}_{mme} + \mathcal{L}_{cb} + \mathcal{L}_{mix} + \mathcal{L}_{cs}$	95.6	94.6	75.3	98.4	75.7	100.0	89.9
$\mathcal{L}_{mme} + \mathcal{L}_{cb} + \mathcal{L}_{mix} + \mathcal{L}_{dca}$	96.2	95.6	76.0	98.7	76.6	100.0	**90.5**

4.4 Ablation Study

4.4.1 Influence of Hyper-Parameters

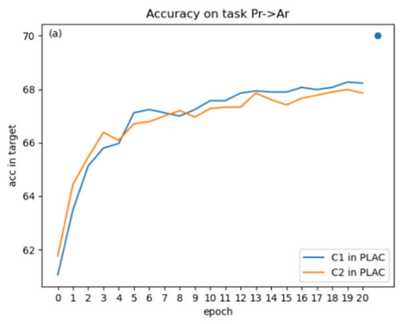
To evaluate whether our proposed method DCA is sensitive to hyperparameters, we investigate their influence on the VisDA-C dataset. As shown in Table 4, in our method DCA, our hyper-parameters λ, η, and β are insensitive.

4.4.2 Ablation Study for the Losses

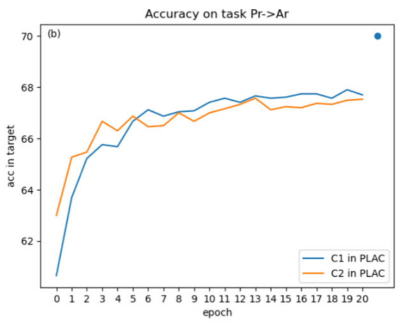
To investigate the effectiveness of our proposed pseudo-labeling method in our experiments, we compare it with baseline SHOT [12] on the Office dataset, which also applies the pseudo-labeling method. Meanwhile, we also conduct a series of ablation studies for other losses. We study in Table 5, the results clearly show that both pseudo-labeling strategies are effective, but our pseudo-labeling method is always much better than the method in SHOT [12].

4.4.3 Loss Weights Between Dual Classifier

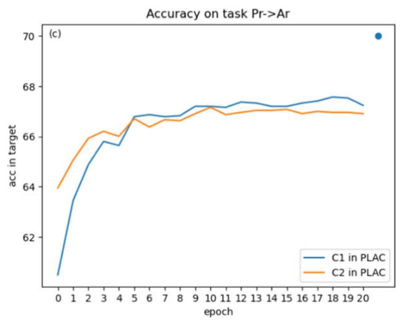
In the experiment, we tried three combinations of how to weight the $\mathcal{L}_{dca}$ loss of the two classifiers, namely [1.5, 0.5], [1.0, 1.0] and [0.5, 1.5]. We selected the task of Pr → Ar in the Office-home data set for ablation experiments. It can be seen from Fig. 3 that no matter which of the three combinations, the accuracy of the source classifier is finally better than that of the target classifier, and all our experimental results are also taken from the source classifier. In addition, it can be seen from the figure that when the combination of [1.5, 0.5] is selected, the source classifier will obtain the optimal effect. Therefore, in the experiment, we choose the weighted combination of [1.5, 0.5] for most tasks, and only when a few tasks fail to achieve the optimal effect, the other two combinations are considered (Fig. 3).



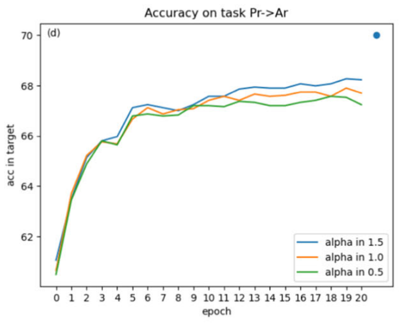
(a) α=1.5, β=0.5



(b) α=1.0, β=1.0



(c) α=0.5, β=1.5



(d) Only C1 accuracy in three situations

Fig. 3 Experimental results of two classifiers on the Pr $\rightarrow$ Ar task of the Office-Home dataset. Based on different L_{dca} loss weighting coefficients

4.4.4 Confusion Matrices

To more intuitively show the ability of DCA in correct classification, we show the classification prediction results of A $\rightarrow$ W task and W $\rightarrow$ A task on office-31 dataset using confusion matrix in Fig. 4. The results show that our DCA method can let the wrong classification significantly reduce, which verifies its effectiveness.

4.4.5 T-SNE Visualization

Figure 5 shows the t-SNE visualization of the target features obtained by the model on the A–W and W–A tasks on the Office-31 dataset. As shown in the figure, compared with the source model trained only on source data, the target feature clusters output by the model adapted by our proposed method DCA are more compact and clearer, demonstrating the excellent discriminative power of our model.

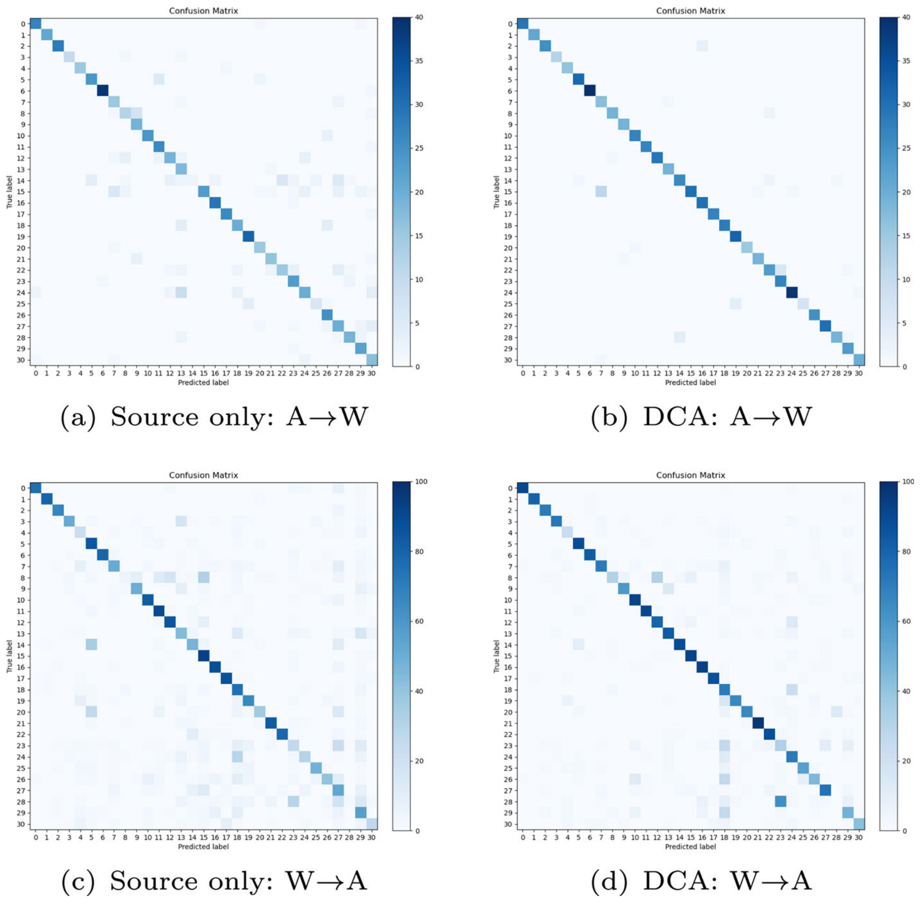


Fig. 4 Confusion matrix for A → W and W → A (Office-31) of the source model and DCA. The Y-axis shows the ground truth labels while the X-axis shows the predicted labels

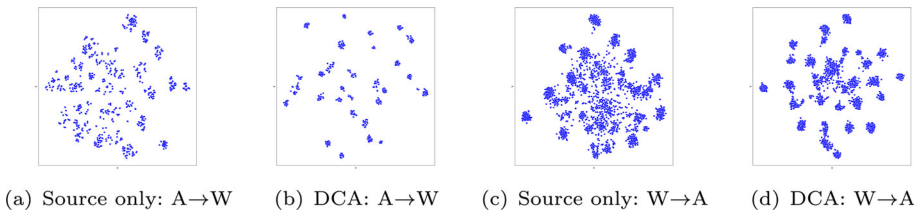


Fig. 5 T-SNE visualization for features from the target domain

5 Conclusion

As the privacy and security of data are more and more concerned, people began to consider how to perform domain adaptation without source data. In this paper, we investigate this challenging setting of SFUDA, and propose a dual-classifier adaptation method called DCA. In DCA, we introduce an additional target classifier to mine the specific information of the target domain, and cooperate with source classifier to fine-tune the feature extractor, so as to learn the knowledge of both domains. Firstly, the minimax entropy and mixup method are used to train the target classifier and feature extractor, and then the dual classifier is applied to dynamically adjust the pseudo-labeling method to align the features between the two domains. Finally, we use the class balance loss to further improve the robustness of the model. After our experiments, it is proved that DCA achieves far ahead results on multiple mainstream datasets. Furthermore, most existing SFUDA methods are researched under the ideal learning scenario of balanced classes in both source and target domains (closed-set), while real-world applications may involve open-set or general domain adaptation scenarios, which will be the focus of our future research directions.

Data availability The datasets that support the findings of this study are available: Office-31 [46], Office-Home [47], VisDA-C [48]. These data were derived from the following resources available in the public domain: https://faculty.cc.gatech.edu/~judy/domainadapt/, https://www.hemanthdv.org/officeHomeDataset.html, http://ai.bu.edu/visda-2017/.

Declarations

Competing interests We have no competing interests to disclose. Also there is no any funding support from any source to mention.

References

1. Long M, Cao Y, Wang J, Jordan M (2015) Learning transferable features with deep adaptation networks. In: International conference on machine learning. PMLR, pp 97–105
2. Zhang Y, David P, Gong B (2017) Curriculum domain adaptation for semantic segmentation of urban scenes. In: Proceedings of the IEEE international conference on computer vision, pp 2020–2030
3. Cicek S, Soatto S (2019) Unsupervised domain adaptation via regularized conditional alignment. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1416–1425
4. Long M, Cao Y, Wang J, Jordan M (2015) Learning transferable features with deep adaptation networks. In: International conference on machine learning. PMLR, pp 97–105
5. Long M, Zhu H, Wang J, Jordan MI (2016) Unsupervised domain adaptation with residual transfer networks. Adv Neural Inf Process Syst 29
6. Lu Z, Yang Y, Zhu X, Liu C, Song YZ, Xiang T (2020) Stochastic classifiers for unsupervised domain adaptation. In: Computer vision and pattern recognition
7. Zhang Y, Tang H, Jia K, Tan M (2019) Domain-symmetric networks for adversarial domain adaptation. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)
8. Long M, Cao Y, Cao Z, Wang J, Jordan MI (2018) Transferable representation learning with deep adaptation networks. IEEE Trans Pattern Anal Mach Intell 41(12):3071–3085
9. Tzeng E, Hoffman J, Saenko K, Darrell T (2017) Adversarial discriminative domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7167–7176
10. Li R, Jiao Q, Cao W, Wong H-S, Wu S (2020) Model adaptation: unsupervised domain adaptation without source data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9641–9650
11. Qiu Z, Zhang Y, Lin H, Niu S, Liu Y, Du Q, Tan M (2021) Source-free domain adaptation via avatar prototype generation and adaptation

12. Liang J, Hu D, Feng J (2020) Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In: International conference on machine learning. PMLR, pp 6028–6039
13. Liang J, Hu D, Wang Y, He R, Feng J (2021) Source data-absent unsupervised domain adaptation through hypothesis transfer and labeling transfer. IEEE Trans Pattern Anal Mach Intell 44(11):8602–8617
14. Yang S, Wang Y, Van De Weijer J, Herranz L, Jui S (2021) Generalized source-free domain adaptation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 8978–8987
15. Qu S, Chen G, Zhang J, Li Z, He W, Tao D (2022) Bmd: A general class-balanced multicentric dynamic prototype strategy for source-free domain adaptation. In: Computer vision–ECCV 2022: 17th European conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV. Springer, pp 165–182
16. Borgwardt KM, Gretton A, Rasch MJ, Kriegel H-P, Schölkopf B, Smola AJ (2006) Integrating structured biological data by kernel maximum mean discrepancy. Bioinformatics 22(14):49–57
17. Pang Z, Zhao L, Liu Q, Wang C (2022) Camera invariant feature learning for unsupervised person re-identification. IEEE Trans Multimed 25:6171–6182
18. Dai S, Cheng Y, Zhang Y, Gan Z, Liu J, Carin L (2020) Contrastively smoothed class alignment for unsupervised domain adaptation. In: Proceedings of the Asian conference on computer vision
19. Kang G, Jiang L, Yang Y, Hauptmann AG (2019) Contrastive adaptation network for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4893–4902
20. Ge Y, Zhu F, Chen D, Zhao R et al (2020) Self-paced contrastive learning with hybrid memory for domain adaptive object re-id. Adv Neural Inf Process Syst 33:11309–11321
21. Saito K, Kim D, Sclaroff S, Saenko K (2020) Universal domain adaptation through self supervision. Adv Neural Inf Process Syst 33:16282–16292
22. Ganin Y, Lempitsky V (2015) Unsupervised domain adaptation by backpropagation. In: International conference on machine learning. PMLR, pp 1180–1189
23. Saito K, Watanabe K, Ushiku Y, Harada T (2018) Maximum classifier discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3723–3732
24. Kim Y, Cho D, Panda P, Hong S (2020) Progressive domain adaptation from a source pre-trained model. arXiv:2007.01524
25. Li J, Yu Z, Du Z, Zhu L, Shen HT (2024) A comprehensive survey on source-free domain adaptation. IEEE Trans Pattern Anal Mach Intell
26. Jing M, Li J, Lu K, Zhu L, Shen HT (2024) Visually source-free domain adaptation via adversarial style matching. IEEE Trans Image Process
27. Yang S, Wang Y, Weijer J, Herranz L, Jui S, Yang J (2023) Trust your good friends: source-free domain adaptation by reciprocal neighborhood clustering. IEEE Trans Pattern Anal Mach Intell
28. Yang S, Wang Y, Herranz L, Jui S, Weijer J (2023) Casting a bait for offline and online source-free domain adaptation. Comput Vis Image Underst 234:103747
29. Ahmed SM, Raychaudhuri DS, Paul S, Oymak S, Roy-Chowdhury AK (2021) Unsupervised multi-source domain adaptation without access to source data. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10103–10112
30. Lee D-H et al (2013) Pseudo-label: the simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on challenges in representation learning, ICML, vol 3, p 896
31. Zhang W, Ouyang W, Li W, Xu D (2018) Collaborative and adversarial network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3801–3809
32. Choi J, Jeong M, Kim T, Kim C (2019) Pseudo-labeling curriculum for unsupervised domain adaptation. arXiv:1908.00262
33. Zou Y, Yu Z, Kumar B, Wang J (2018) Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In: Proceedings of the European conference on computer vision (ECCV), pp 289–305
34. Pang Z, Wang C, Wang J, Zhao L (2023) Reliability modeling and contrastive learning for unsupervised person re-identification. Knowl Based Syst 263:110263
35. Caron M, Bojanowski P, Joulin A, Douze M (2018) Deep clustering for unsupervised learning of visual features. In: Proceedings of the European conference on computer vision (ECCV), pp 132–149
36. Zheng Z, Yang Y (2021) Rectifying pseudo label learning via uncertainty estimation for domain adaptive semantic segmentation. Int J Comput Vis 129(4):1106–1120
37. Müller R, Kornblith S, Hinton GE (2019) When does label smoothing help? Adv Neural Inf Process Syst 32

38. Ghasedi Dizaji K, Herandi A, Deng C, Cai W, Huang H (2017) Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In: Proceedings of the IEEE international conference on computer vision, pp 5736–5745
39. Zhang H, Cisse M, Dauphin YN, Lopez-Paz D (2018) mixup: Beyond empirical risk minimization. In: International conference on learning representations
40. Berthelot D, Carlini N, Goodfellow I, Papernot N, Oliver A, Raffel CA (2019) Mixmatch: a holistic approach to semi-supervised learning. Adv Neural Inf Process Syst 32
41. Berthelot D, Carlini N, Cubuk ED, Kurakin A, Sohn K, Zhang H, Raffel C (2019) Remixmatch: semi-supervised learning with distribution alignment and augmentation anchoring. arXiv:1911.09785
42. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on computational learning theory, pp 92–100
43. Sindhwani V, Niyogi P, Belkin M (2005) A co-regularization approach to semi-supervised learning with multiple views. In: Proceedings of ICML workshop on learning with multiple views, vol 2005. Citeseer, pp 74–79
44. Xia H, Zhao H, Ding Z (2021) Adaptive adversarial network for source-free domain adaptation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9010–9019
45. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
46. Saenko K, Kulis B, Fritz M, Darrell T (2010) Adapting visual category models to new domains. In: Computer vision–ECCV 2010: 11th European conference on computer vision, Heraklion, Crete, Greece, September 5–11, 2010, Proceedings, Part IV 11. Springer, pp 213–226
47. Venkateswara H, Eusebio J, Chakraborty S, Panchanathan S (2017) Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5018–5027
48. Peng X, Usman B, Kaushik N, Hoffman J, Wang D, Saenko K (2017) Visda: The visual domain adaptation challenge. arXiv:1710.06924
49. Ganin Y, Ustinova E, Ajakan H, Germain P, Larochelle H, Laviolette F, Marchand M, Lempitsky V (2016) Domain-adversarial training of neural networks. J Mach Learn Res 17(59):1–35
50. Cui S, Wang S, Zhuo J, Li L, Huang Q, Tian Q (2020) Towards discriminability and diversity: batch nuclear-norm maximization under label insufficient situations. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE Computer Society, pp 3940–3949
51. Long M, Cao Z, Wang J, Jordan MI (2018) Conditional adversarial domain adaptation. Adv Neural Inf Process Syst 31
52. Tang H, Chen K, Jia K (2020) Unsupervised domain adaptation via structurally regularized deep clustering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8725–8735
53. Chu T, Liu Y, Deng J, Li W, Duan L (2022) Denoised maximum classifier discrepancy for source-free unsupervised domain adaptation. In: Proceedings of the AAAI conference on artificial intelligence, vol 36, pp 472–480
54. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
55. Lee C-Y, Batra T, Baig MH, Ulbricht D (2019) Sliced Wasserstein discrepancy for unsupervised domain adaptation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10285–10295
56. Jin Y, Wang X, Long M, Wang J (2020) Minimum class confusion for versatile domain adaptation. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16. Springer, pp 464–480