Applied Intelligence (2023) 53:18277–18293
https://doi.org/10.1007/s10489-022-04414-2

Center transfer for supervised domain adaptation

Xiuyu Huang[1,2] · Nan Zhou[3] · Jian Huang[4] · Huaidong Zhang[5] · Witold Pedrycz[2,6,7,8] · Kup-Sze Choi[1]

Accepted: 15 December 2022 / Published online: 26 January 2023

Abstract
Domain adaptation (DA) is a popular strategy for pattern recognition and classification tasks. It leverages a large amount of data from the source domain to help train the model applied in the target domain. Supervised domain adaptation (SDA) approaches are desirable when only few labeled samples from the target domain are available. They can be easily adopted in many real-world applications where data collection is expensive. In this study, we propose a new supervision signal, namely center transfer loss (CTL), to efficiently align features under the SDA setting in the deep learning (DL) field. Unlike most previous SDA methods that rely on pairing up training samples, the proposed loss is trainable only using one-stream input based on the mini-batch strategy. The CTL exhibits two main functionalities in training to increase the performance of DL models, i.e., domain alignment and increasing the feature's discriminative power. The hyper-parameter to balance these two functionalities is waived in CTL, which is the second improvement from the previous approaches. Extensive experiments completed on well-known public datasets show that the proposed method performs better than recent state-of-the-art approaches.

Keywords Supervised domain adaptation · Deep learning · Center transfer loss · Transfer learning

1 Introduction

Deep learning (DL) methods have been successfully applied to various areas, such as computer vision [1], brain-computer interface [2], and medical diagnosis [3]. The outstanding performance of DL approaches benefits from numerous training data. However, it is sometimes tough to acquire sufficient data to train a DL model for a specific task at hand, since data recording and label annotation are costly and labor-intensive. One of the popular choices to address such an issue is domain adaptation (DA) [4]. Its main idea is to use available large-scale datasets in the source domain ($\mathcal{D}_s$) to assist the model training in the target domain ($\mathcal{D}_t$), where the training data is scarce.

✉ Xiuyu Huang
xiuyu.huang@connect.polyu.hk

✉ Witold Pedrycz
wpedrycz@ualberta.ca

✉ Kup-Sze Choi
thomasks.choi@polyu.edu.hk

Extended author information available on the last page of the article.

According to [5, 6], DA can be either unsupervised, semi-supervised, or supervised, determined by the availability of the labeled data in $\mathcal{D}_t$. Unsupervised domain adaptation (UDA) [7] only carries unlabeled target data. In a semi-supervised DA (SSDA) scheme [8], both a small amount of labeled and a considerable amount of unlabelled data are accessible. Alternatively, supervised domain adaptation (SDA) [9, 10] supposes that all available target samples are annotated, although the number is small. Sophisticated SDA approaches can usually outperform UDA and SSDA ones when the amount of available data in $\mathcal{D}_t$ is small [6]. Making annotations on a small dataset is likely to be practical and does not require too much effort. Therefore, SDA methods are more appealing if only very few samples from $\mathcal{D}_t$ are accessible. They have been performed in many applications such as cross-subject EEG emotion classification [11], CT scan-based Covid-19 diagnosis [12], emotion detection from the speech [13], and radar-based human activity recognition [14]. These applications only allow recording a small set of data from the target domain, as a massive data collection is either extremely expensive or impossible. Therefore, they require a suitable method to make use of such a small number of samples in $\mathcal{D}_t$ to generate a reliable model applied to the target domain. SDA also has another name, i.e., few-shot domain adaptation [15], which more directly expresses the scenario of using few samples from $\mathcal{D}_t$ in DA problems.

The typical way to implement an SDA approach in the DL community for a classification task is to learn a deep transformation that draws same-class samples close together, regardless from $\mathcal{D}_s$ or $\mathcal{D}_t$. A popular strategy to perform such a mapping is to operate a two-stream network, i.e., siamese network [16] or correlated network [17]. The training process of these networks typically starts with either a sample-based [6, 10, 15, 18, 19] or a batch-based [9, 17] pair-wise input. However, such a pairing mechanism leads to a quadratic increase of the sample size from the original dataset and unavoidably results in redundancy, slow convergence, and unstable performance [20]. For example, one standard protocol [19, 21] of MNIST → USPS domain adaptation task is to use 2000 labeled and 70 labeled samples from MNIST and USPS datasets, respectively, to train a model that applies to the classification task in USPS. Several recent state-of-the-arts (SOTA) methods [6, 10, 19] utilize a siamese network that trains the model with 56000 (2000*70*σ; σ is the ratio to control the redundancy and is equal to 0.4 in these studies.) pairs of samples. It is impractical to train a network when the sample size of either source or target dataset further enlarges.

This study proposes a simple but efficient loss function, namely center transfer loss (CTL), to address the above-mentioned issues and increase the discriminative power of deep learning features. Specifically, we learn a center (a vector with the same dimensionality as that of the feature) for features of each class in each domain and update these centers in training. In addition, we minimize the distance between the features and their corresponding class centers in the opposite domain rather than the same domain. In other words, we minimize the distance between features of $\mathcal{D}_s$ and class centers of $\mathcal{D}_t$, as well as the distance between features of $\mathcal{D}_t$ and class centers of $\mathcal{D}_s$. For example, the features of class 1 in $\mathcal{D}_s$ are pushed to the feature center of class 1 in $\mathcal{D}_t$, and the features of class 1 in $\mathcal{D}_t$ are pushed to the feature center of class 1 in $\mathcal{D}_s$. Deep neural networks (DNNs) are trained by the joint supervision of softmax loss and CTL. Intuitively, the softmax loss ensures that features of the different classes stay apart. CTL pushes samples to the class center of the opposite domain. The same-class samples between different domains will eventually align by constant center update and distance minimization. More interestingly, CTL achieves a feature alignment at the beginning of training, see Fig. 1(b). In the later stage, when the distribution of features between domains is sufficiently aligned, CTL alternatively acts as another function to decrease the intra-class variation of the features and increases their discriminative power, Fig. 1(c). We do not require to set hyper-parameters for controlling the shift between early and latter training stages.

In sum, the proposed method has **two major contributions** in comparison with previous approaches.

1. It is very convenient to employ CTL in DNNs. Our DL models are trainable by the mini-batch strategy in a single-stream setting without running two-stream architectures. The issue of redundancy, slow convergence, and unstable performance can be significantly avoided.
2. The learning features can achieve both domain alignment and intra-class variation minimization by using the proposed CTL in the model training. Although several previous SDA approaches (e.g., [6, 10, 19]) can also provide a similar outcome, a trade-off value must be manually set in these methods to balance the domain alignment and intra-class variation minimization. The optimal choice for the trade-off value varies in different datasets and tasks, resulting in a labour-intensive exhaustive search each time but no guarantee of finding

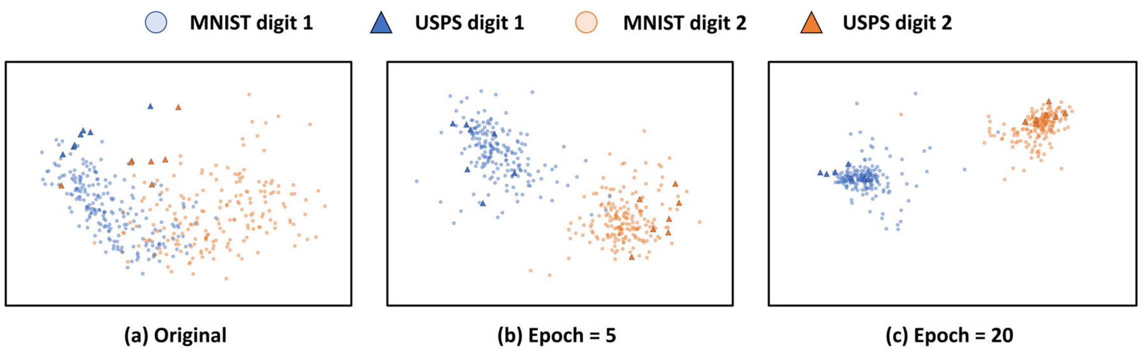


Fig. 1 The distribution of features generated by the model trained using the combination of softmax and CTL in a two-class toy problem with 200 source (MNIST) and 7 target (USPS) samples in each class. Training settings, expect for data and loss function, are the same as the toy example in [20]. The features are the output of the second last layer and dimensionally reduced 2 for visualization. (a) Features generated by the network without training. (b) Features generated by the network trained with only 5 epochs. (c) Features generated by the network trained with 20 epochs

the best value. Alternatively, our CTL can achieve both these two functionalities in different stages of training without the need to set a trade-off value to balance them.

To verify the effectiveness of our approach, we conduct extensive experiments on common DA benchmarks. The results show that our method achieves a better performance than current SOTAs. The remainder of this article is organized as follows. Section 2 introduces the related works. Section 3 explains the proposed method in detail. Section 4 presents the experiment protocols and results. The conclusion is drawn in Section 5.

2 Related works

SDA approaches focus on the specific scenario that the labeled target data are available in training, albeit very few samples per class. There are diverse SDA strategies targeted to different types of tasks, e.g., regression [22, 23], object detection [24, 25], and classification. Our study focuses on SDA methods in the classification problem.

Early SDA approaches for the classification task depend on the matrix-based mapping between domains and linear classifiers. Zhou et al. [26] proposed an SDA method called SHFR-ECOC by constructing a sparse feature transformation matrix to get invariant features between domains as inputs to a linear SVM classifier. Sukhija et al. [27] explored another SDA strategy that also learns a sparse feature transformation for the feature generation. This study used a random forest classifier instead. DL models have been rapidly developing in recent years. It offers an end-to-end way for the classification task and naturally arouses more researchers' interests.

In [18], source and sparely labeled target data are used to train a siamese network. The network learns domain invariant features using a soft label distribution matching loss. Similarly, Motiian et al. [6] raised a method called CCSA, also based on the siamese architecture. The model was trained by the joint supervision of categorical entropy (i.e., softmax loss) and point-wise contrastive loss introduced in [28]. The authors found that their method provided a fast convergence and a better performance in terms of classification accuracy. The same research group of CCSA proposed another SDA method (i.e., FADA [15]) using the adversarial training to attain the feature alignment. They carefully designed four kinds of paired data that the discriminator in the network is augmented to distinguish. In the same year, Piotr et al. [17] also presented a strategy, namely SoHoT, using the mixture of second or/and third scatter alignment measures between source and target domains. They aim to align within-class scatters of a two-stream network to a certain degree using bespoke loss and to keep a good separation of the between-class scatters.

More recently, another SDA method, called Domain Adaptation using Stochastic Neighborhood Embedding (d-SNE), was proposed by Xu et al. [19]. Interestingly, this approach only focused on minimizing the distance of the same-class pairs between source and target domains with the largest distance and maximizing the distance of the most nearby different-class pair. Alternatively, Hedegaard et al. [10, 29] utilized the graph embedding technique to learn a domain-invariant and semantically meaningful feature space. Similar to CCSA, a siamese network was also used in this study as a feature generator. Generated features are finally put into a linear discriminative analysis (LDA) classifier to perform the prediction. In addition, Tong et al. [9] presented a mathematical framework (MF) that considered DA as a convex optimization problem. This MF quantifies the transferability in the transfer learning problems based on the number of samples, model complexity, and Chi-square distance between source and target tasks. The authors also designed an SDA approach using this framework and achieved encouraging performance in the DA benchmarks. Nevertheless, the training of SDA methods above requires pair-up samples between $\mathcal{D}_s$ and $\mathcal{D}_t$. Our method is rather trainable by single-stream data based on the mini-batch strategy.

In addition, another study similar to ours is the center loss [20], which calculates the class centers using the data from both source and target domains. Given that the number of source samples is much larger than that of the target subject, the computation of the class centers is dominated by the source samples. Thus, the model trained by the center loss [20] only directly pulls the scarce samples of $\mathcal{D}_t$ to the centers of $\mathcal{D}_s$ without accounting for the feature distribution of $\mathcal{D}_t$. Generally, the center loss has a different insight from ours and can not address SDA classification tasks effectively.

3 Methodology

In this work, we are only concerned with the specific SDA problem in which a large-scale dataset in $\mathcal{D}_s$ and very few annotated samples in $\mathcal{D}_t$ are available. In other words, we have all data $\mathcal{D}_{all} = \{\mathbf{x}_i, y_i\}_{i=1}^{m+n}$ combined by the ones from source domain $\mathcal{D}_s = \{\mathbf{x}_i^s, y_i^s\}_{i=1}^{m}$ and from target domain $\mathcal{D}_t = \{\mathbf{x}_i^t, y_i^t\}_{i=1}^{n}$, respectively. $\mathbf{x}_i \in \mathbb{R}^d$ denotes the i^{th} feature in d dimensional space (vector size), and y_i is the corresponding label of $\mathbf{x}_i$ having a classes. The features $\mathbf{x}_i^s$ and $\mathbf{x}_i^t$ can be regarded as realization of random variables X^s and X^t, respectively. Note that $m >> n$ in the SDA scenario. Let's say that X^s represents the source domain and

X^t represents the target domain. In the absence of domain shift, we can simply train a DNN model using all the data directly from $\mathcal{D}_{all}$ with the softmax loss defined as

$$\mathcal{L}_S = -\sum_{i=1}^{k} \log \frac{e^{\mathbf{W}_{y_i}^{\top}\mathbf{x}_i + b_{y_i}}}{\sum_{j=1}^{a} e^{\mathbf{W}_j^{\top}\mathbf{x}_i + b_j}} \tag{1}$$

where $\mathbf{W}_j \in \mathbb{R}^d$ represents the j^{th} column of the weights $\mathbf{W} \in \mathbb{R}^{d\times a}$ in the last dense layer (features as input) and $\mathbf{b} \in \mathbb{R}^a$ denotes the bias term. k is the size of a mini-batch. However, the distributions of the two domains are mostly different. Directly using $\mathbf{x}_i^s$ for the training of classifier in $\mathcal{D}_t$ is naturally problematic. An alignment of features between different domains is therefore necessary. In UDA setting [7], it is assumed that labels are unavailable in $\mathcal{D}^t$. A common strategy for domain alignment is to introduce a distance loss of the marginal distribution between $\mathcal{D}^s$ and $\mathcal{D}^t$ (i.e. $p(X^s)$ and $p(X^t)$) as follows.

$$\mathcal{L}_{DIS} = D(p(X^s), p(X^t)) \tag{2}$$

where $D(\cdot,\cdot)$ is a certain metric between two distribution inputs which once aligned, a feature can no longer be recognized from the source or target domain. The UDA methods have a natural limitation that even the marginal distribution is perfectly aligned: there is no promise that the features belonging to the same class but in different domains are transformed into the same space. Such an alignment may not offer significant benefits on the DNN model with respect to a classification task. Alternatively, we have labeled data from the target domain in hand, albeit a small amount. It is practical to achieve a better alignment, where features from different domains but with the same class label are mapped in the nearby distribution by amending (2) as:

$$\mathcal{L}_{CDIS} = \sum_{i=1}^{a} D(p(X_i^s), p(X_i^t)) \tag{3}$$

Now, the core challenge is to find an appropriate metric $D(\cdot,\cdot)$. To this end, we minimize the Euclidean distance between features and the corresponding class centers of the opposite domain. Mathematically, the (3) can be reformulated as:

$$\mathcal{L}_{F-CTL} = \frac{1}{2}\sum_{i=1}^{m} \|\mathbf{x}_i^s - \mathbf{c}_{y_i}^t\|_2^2 + \frac{1}{2}\sum_{j=1}^{n} \|\mathbf{x}_j^t - \mathbf{c}_{y_j}^s\|_2^2 \tag{4}$$

where $\mathbf{c}_{y_i}^t \in \mathbb{R}^d$ denotes the y_ith class center of features in $\mathcal{D}^t$, and $\mathbf{c}_{y_j}^s \in \mathbb{R}^d$ represents the y_jth class center of features in $\mathcal{D}^s$. An intuitive example of (4) can be referred to the "Early stage in training" in Fig. 2. It is noted that m is usually much larger than n in SDA setting. In this case, the first half of (4) usually dominates the loss function without considering the contribution of the samples in $\mathcal{D}^t$ to the aligned latent space. To address this issue, we take the mean of distances instead of summing them up in each domain as follows

$$\mathcal{L}_{F-CTL} = \frac{1}{2m}\sum_{i=1}^{m} \|\mathbf{x}_i^s - \mathbf{c}_{y_i}^t\|_2^2 + \frac{1}{2n}\sum_{j=1}^{n} \|\mathbf{x}_j^t - \mathbf{c}_{y_j}^s\|_2^2. \tag{5}$$

Ideally, calculations of $\mathcal{L}_{F-CTL}$ and feature centers $\mathbf{c}$ should take all features of the whole training set into account. However, due to the large sample size of the source training set and a limited RAM storage, it is impractical to perform such an implementation. We implement the update for centers and features based on mini-batch. The $\mathcal{L}_{F-CTL}$ can be changed as (6).

$$\mathcal{L}_{CTL} = \frac{1}{2k^s}\sum_{i=1}^{k^s} \|\mathbf{x}_i^s - \mathbf{c}_{y_i}^t\|_2^2 + \frac{1}{2k^t}\sum_{j=1}^{k^t} \|\mathbf{x}_j^t - \mathbf{c}_{y_j}^s\|_2^2 \tag{6}$$

where k^s and k^t are number of samples from $\mathcal{D}^s$ and $\mathcal{D}^t$ in the mini-batch, respectively. Equation (6) ($\mathcal{L}_{CTL}$) is the ultimate form of the proposed center transfer loss (CTL) based on the mini-batch update strategy. From the equation, it is noticed that CTL can be optimized with one-stream input without relying on a two-stream network used in previous methods [6, 10, 15, 18, 19] for the loss optimization. The training of a two-stream network requires pairing up samples, leading to a quadratic increase in the sample size from the original dataset. Our loss is based on one-stream training and is able to avoid this problem. In addition, CTL contributes to a domain alignment at the initial training stage and increases the discriminative power of features afterwards by minimizing the intra-class variation (Fig. 2). Although previous methods introduce similar losses that also achieve a domain alignment and the increase of discriminative power. They require to manually set a trade-off value to balance these two functionalities in training to produce a good outcome. Alternatively, it is clear that the proposed loss, as shown in the equation, does not require such a manual trade-off value to balance them.

Some centers may not be updated in each iteration of training, as the training is conducted by the mini-batch strategy. The updating equations of class centers $\mathbf{c}^t$ and $\mathbf{c}^s$ are formulated as:

$$\Delta\mathbf{c}_h^t = \frac{\sum_{j=1}^{k^t}\delta(y_j = h)(\mathbf{c}_h^t - \mathbf{x}_j^t)}{\rho + \sum_{j=1}^{k^t}\delta(y_j = h)}; \quad h = 1,\ldots,a \tag{7}$$

$$\Delta\mathbf{c}_h^s = \frac{\sum_{i=1}^{k^s}\delta(y_i = h)(\mathbf{c}_h^s - \mathbf{x}_i^s)}{\rho + \sum_{i=1}^{k^s}\delta(y_i = h)}; \quad h = 1,\ldots,a \tag{8}$$

$$\mathbf{c}_h^t = \mathbf{c}_h^t - \alpha\Delta\mathbf{c}_h^t. \tag{9}$$

$$\mathbf{c}_h^s = \mathbf{c}_h^s - \alpha\Delta\mathbf{c}_h^s. \tag{10}$$

where $\delta(\cdot)$ is the indicator function, and it is equal to 1 if the condition is satisfied and equal to 0 otherwise. ρ is a small constant (i.e., 10^{-5}) to avoid the equation being divided

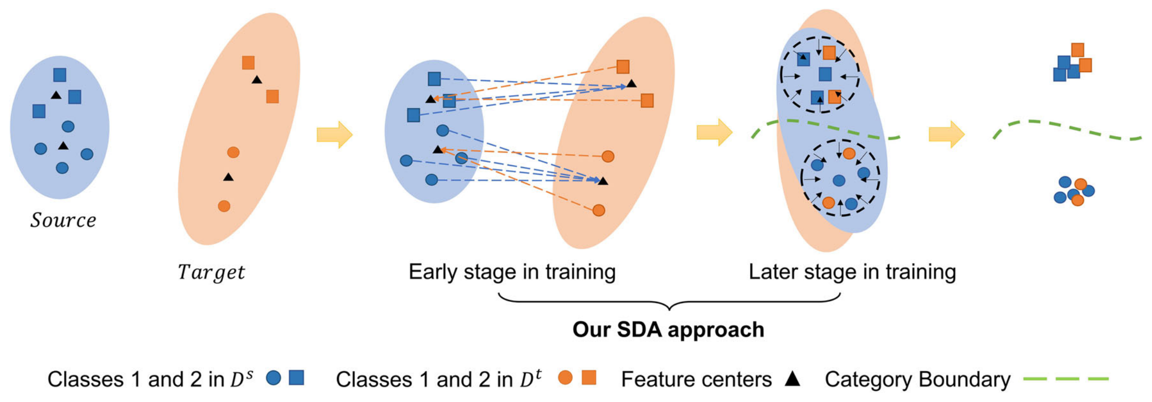


Fig. 2 An illustration of the proposed loss to achieve both a feature alignment and a decrease of the intra-class variation

by zero. A scalar $\alpha \in (0, 1]$ controls the learning rate of the centers, eliminating the noisy samples' negative impact. We adopt the joint supervision of softmax loss and CTL to train the DNN models for DA in the classification task. The objective function is given as follows.

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_{CTL} \tag{11}$$

where λ is a trade-off scalar to balance these two losses and is ranging from 0 to 5 in our experiments. We summarize the training strategy based on the mini-batch in Algorithm 1.

```
1:  Input: Training data $\{\mathbf{x}_i^s | i = 1, \dots, k^s\} \cup \{\mathbf{x}_j^t | j = 1, \dots, k^t\}$.
    Parameters $\theta$ in convolution layers and $\mathbf{W}$ in the last dense
    layer. Class centers $\{\mathbf{c}_h^s | h = 1, \dots, a\}$ and $\{\mathbf{c}_h^t | h =
    1, \dots, a\}$ initialized by the standard normal distribution.
    Hyper-parameters $\lambda$, $\alpha$ and learning rate $\mu$. The number of
    iteration $r = 0$.
2:  Output: Updated parameters $\theta$ and $\mathbf{W}$.
3:  while Not convergent do
4:      Compute $\mathcal{L}_S$, $\mathcal{L}_{CTL}$, and $\mathcal{L}$.
5:      Compute $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_i^s}$ in $\mathcal{D}_s$ for each $i$ by $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_i^s} = \frac{\partial \mathcal{L}_S}{\partial \mathbf{x}_i^s} + \lambda \frac{\partial \mathcal{L}_{CTL}}{\partial \mathbf{x}_i^s}$.
6:      Compute $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_j^t}$ in $\mathcal{D}_t$ for each $j$ by $\frac{\partial \mathcal{L}}{\partial \mathbf{x}_j^t} = \frac{\partial \mathcal{L}_S}{\partial \mathbf{x}_j^t} + \lambda \frac{\partial \mathcal{L}_{CTL}}{\partial \mathbf{x}_j^t}$.
7:      Update $\theta \leftarrow \theta - \mu(\sum_{i=1}^{k^s} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i^s} \cdot \frac{\partial \mathbf{x}_i^s}{\partial \theta} + \sum_{j=1}^{k^t} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_j^t} \cdot \frac{\partial \mathbf{x}_j^t}{\partial \theta})$.
8:      Update $\mathbf{W} \leftarrow \mathbf{W} - \mu \cdot \frac{\partial \mathcal{L}_S}{\partial \mathbf{W}}$.
9:      Update $\mathbf{c}_h^t \leftarrow \mathbf{c}_h^t - \alpha \Delta \mathbf{c}_h^t$.
10:     Update $\mathbf{c}_h^s \leftarrow \mathbf{c}_h^s - \alpha \Delta \mathbf{c}_h^s$.
11:     $r = r + 1$.
12: end while
13: end
```

Algorithm 1 Training of Center Transfer Loss.

4 Experimental protocols and results

We evaluate the proposed loss function on different common SDA benchmarks, including Office31 [30], Office-Caltech-10 [31], Office-Home [32], and digit transfer (MNIST [33], USPS [34], SVHN [35], and MNIST-M [36]). The visualization of features and sensitivity analyses on λ and α are also presented. The impact of the batch size on the effectiveness of CTL is introduced in the last part of this section. The source code for experiments is publicly available[1]. All data generated or analysed during this study are included.

4.1 Office31

Office31 is a classical benchmark collected for the evaluation of DA methods. It comprises 31 visual objectives from three separate domains, namely Amazon ($\mathcal{A}$), Webcam ($\mathcal{W}$), and DSLR ($\mathcal{D}$). Amazon is the largest dataset and contains 2,817 images. Webcam and DSLR are relatively compact and have 795 and 498 images, respectively. Examples of images in this dataset are shown in Fig. 3.

Our experiments follow the setting used in [9]. Six domain shifts, including $\mathcal{A} \rightarrow \mathcal{D}$, $\mathcal{A} \rightarrow \mathcal{W}$, $\mathcal{W} \rightarrow \mathcal{A}$, $\mathcal{W} \rightarrow \mathcal{D}$, $\mathcal{D} \rightarrow \mathcal{A}$, and $\mathcal{D} \rightarrow \mathcal{W}$ are examined in this dataset. All classes of the dataset and five-train-test-split validation scheme are used in the experiments. With respect to the source domain, 20 samples per class from $\mathcal{A}$, whereas 8 samples per class from $\mathcal{W}$ or $\mathcal{D}$ are randomly chosen to train the model for each split. For the target domain, 3 samples per class are randomly selected for training in each split. The remaining target samples are used for testing.

The convolutional layers of VGG16 [37] followed by two dense layers with output sizes of 1024 and 128, respectively, are used as the base network in the experiment. We use this architecture for a fair comparison to most published SDA approaches. The weights of convolutional layers were pre-trained by ImageNet [38]. The ones of dense layers are randomly initialized. All images are resized to 224 $\times$ 224, followed by normalization. The learning rates for convolutional and dense layers are set as 0.001 and 0.01,

[1]https://github.com/XiuyuHuangsmarthealth/Center-Transfer-for-Supervised-Domain-Adaptation

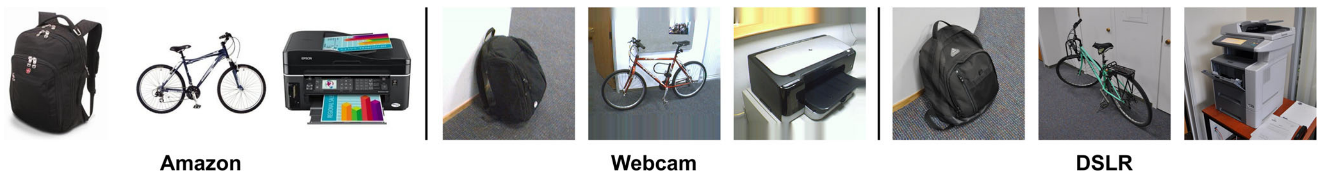


Fig. 3 Examples of Office31 dataset

respectively. The size of the mini-batch is 32. λ and α are fixed as 0.1 and 0.5, respectively. We compare our method with recent supervised domain adaptation SOTAs, including SDADT [18], CCSA [6], FADA [15], d-SNE [19], DAGE-LDA [10], and MF [9]. Three baselines, including (1) Model 1 trained by only source data using softmax loss, (2) Model 2 trained by source data and target samples using softmax loss, and (3) Model 3 trained by source data and target samples using a joint supervision of softmax loss and center loss, are also involved in the comparison. It is noted that all baselines use the same backbone (i.e., VGG16) as that used in the proposed method. We also compare our strategy with another SOTA approach, So-HoT [17]. Although the original paper of SoHoT reports the model performance using VGG16 on Office31, it only contains two cross-domain schemes (i.e., $\mathcal{A} \rightarrow \mathcal{D}$ and $\mathcal{A} \rightarrow \mathcal{W}$). Alternatively, the paper shows the performance of AlexNet [39] on all six schemes. Therefore, the proposed method based on AlexNet is also implemented for a fair comparison to So-HoT.

Table 1 shows the performance of different models on the Office31 dataset. Our experiments show that most SDA methods outperform Models 1 and 2, the baselines trained using the softmax loss without DA. It is worth knowing that several previous studies, such as [6, 15, 19], only compare their methods with a weak baseline (i.e., Model 1 trained by only source data using softmax loss). They claim that SDA methods have significant improvements (over 15%) in the classification task of Office31 dataset compared to this baseline. This may overstate the effectiveness of SDA methods. A stronger and fairer baseline, i.e., Model 2 trained by source and target data using softmax loss, should also be involved in the comparison. In Table 1, we can observe that SDA approaches only have less than 7% increments in the classification performance from the stronger baseline (Model 2), consistent with the findings in [10, 17]. Nevertheless, it is encouraging to see that our model has a higher performance than other recent SOTAs and baselines on the Offce31 dataset.

4.2 Office-caltech-10

Office-Caltech-10 dataset contains ten sharing categories (i.e., backpack, bike, calculator, headphones, keyboard, laptop-computer, monitor, mouse, mug, and projector) in Office31 and Caltech-256 [40]. The dataset contains 4 domains including, Amazon (958 images; $\mathcal{A}$), Webcam (295 images; $\mathcal{W}$), Dslr (157 images; $\mathcal{D}$), and Caltech (1123 images; $\mathcal{C}$). Therefore, we have 12 cross-domain tasks

Table 1 Average classification accuracy (%) of different methods on 31 classes of Office31 dataset

Method	Cross-domain scheme						
	$\mathcal{A} \rightarrow \mathcal{D}$	$\mathcal{A} \rightarrow \mathcal{W}$	$\mathcal{W} \rightarrow \mathcal{A}$	$\mathcal{W} \rightarrow \mathcal{D}$	$\mathcal{D} \rightarrow \mathcal{A}$	$\mathcal{D} \rightarrow \mathcal{W}$	Average
Model 1	67.6±0.7	61.3±0.9	60.1±0.6	95.7±1.1	61.00.9	95.8±0.3	73.6
Model 2	84.81.6	80.7±0.7	65.70.6	97.5±0.5	65.30.2	96.3±0.2	81.7
Model 3	86.2 1.6	82.4±2.9	68.01.4	97.8±0.7	68.21.2	95.7±1.0	83.0
SDADT [18]	86.11.2	82.7±0.8	65.00.5	97.6±0.2	66.2±0.3	95.70.5	82.2
CCSA [6]	89.0±1.2	88.2±1.0	72.1 1.0	97.6±0.4	71.8 0.5	96.4±0.8	85.8
FADA [15]	88.2±1.0	88.1±1.2	71.1±0.9	97.5±0.6	68.1±0.6	96.4±0.8	84.9
d-SNE [19]	91.4±0.2	**90.1±0.1**	71.1±0.2	97.1±0.1	71.7±0.4	97.5±0.2	86.5
DAG-LDA [10]	85.9±2.8	87.8±2.3	64.2±1.2	**99.5±0.5**	66.5±1.4	97.9±0.6	83.6
MF [9]	90.0±n.a.	87.3±n.a.	72.4±n.a.	96.5±n.a.	72.1±n.a.	97.2±n.a.	85.9
Ours	**92.1±1.5**	89.4±1.3	**74.2±1.0**	99.4±0.3	**74.4±1.0**	**98.3±0.7**	**88.0**
So-HoT* [17]	86.3±0.8	84.5±1.7	65.7±1.7	97.5±0.7	66.5±1.0	95.5±0.6	82.7
Ours*	87.6±0.9	86.2±1.3	66.6±0.8	98.5±0.6	66.3±1.3	97.9±1.3	83.9

Results marked with a star are based on AlexNet [39]. The others are based on VGG16 [37]. "n.a." represents that the original publication does not show standard deviations across the repetitive experiments

The best performance is highlighted in boldface

Table 2 Average classification accuracy (%) of different methods on the Office-Caltech-101 using DeCaF-fc6 features

Method	Within-Office31						
	$\mathcal{A} \rightarrow \mathcal{D}$	$\mathcal{A} \rightarrow \mathcal{W}$	$\mathcal{W} \rightarrow \mathcal{A}$	$\mathcal{W} \rightarrow \mathcal{D}$	$\mathcal{D} \rightarrow \mathcal{A}$	$\mathcal{D} \rightarrow \mathcal{W}$	Average
Model 1	73.7±2.8	79.7±3.5	62.7±3.7	95.1±2.2	71.0±2.0	91.8±0.8	79.0
Model 2	83.1±1.7	91.0±3.2	87.0±4.2	96.5±1.0	85.3±1.6	94.0±1.6	89.5
Model 3	85.6±1.8	91.6±3.5	84.4±3.4	95.3±1.1	84.2±0.2	95.4±1.9	89.4
CCSA [6]	91.0±0.9	90.6±2.8	87.0±1.9	**99.1±0.9**	**89.2±0.9**	97.5±1.6	92.4
d-SNE [19]	89.3±2.3	**94.9±3.5**	81.4±4.0	96.3±2.7	84.2±1.6	94.2±2.4	90.0
DAG-LDA [10]	89.4±2.6	91.6±1.2	85.5±2.8	98.6±1.2	87.4±0.8	96.8±1.2	91.6
MF [9]	92.0±1.4	92.5±5.8	83.5±0.9	95.6±2.9	85.6±4.1	93.1±1.5	90.4
Ours	**92.8±0.8**	94.7±1.0	**88.8±1.3**	98.6±0.6	88.4±1.4	**98.0±1.3**	**93.6**
	Office-Caltech						
	$\mathcal{A} \rightarrow \mathcal{C}$	$\mathcal{W} \rightarrow \mathcal{C}$	$\mathcal{D} \rightarrow \mathcal{C}$	$\mathcal{C} \rightarrow \mathcal{A}$	$\mathcal{C} \rightarrow \mathcal{W}$	$\mathcal{C} \rightarrow \mathcal{D}$	Average
Model 1	68.1±6.9	51.0±7.1	56.4±6.1	72.8±7.1	51.6±4.9	70.6±8.3	61.7
Model 2	82.7±0.7	78.4±2.4	78.3±0.9	87.3±0.7	87.1±1.7	87.2±3.5	83.5
Model 3	80.3±1.0	76.7±0.8	77.9±1.4	89.1±0.6	87.6±2.3	89.4±3.7	83.5
CCSA [6]	81.3±1.3	80.2±1.5	82.0±3.2	90.0±0.8	92.3±1.9	95.3±2.6	86.8
d-SNE [19]	78.0±1.8	70.2±6.3	63.3±8.4	86.5±2.7	89.8±2.3	94.8±1.4	80.4
DAG-LDA [10]	81.1±1.0	80.3±2.3	80.9±1.4	88.7±1.9	92.4±1.9	94.6±1.7	86.3
MF [9]	80.3±n.a.	72.9±n.a.	72.2±n.a.	88.4±n.a.	85.9±n.a.	83.5±n.a.	80.5
Ours	**84.3±0.3**	**82.8±0.8**	**82.8±1.4**	**91.5±0.9**	**95.2±1.1**	**97.5±1.0**	**89.0**

"n.a." represents that the original publication does not show standard deviations across repetitive experiments

The best performance is highlighted in boldface

formulated in this data collection. The same split-generation protocol in Office31 experiments is used but only applied to the ten classes above. Following the settings in [9], we implement our experiments using DeCAF-fc6 features [41] as model inputs.

Referring to the base architecture in [6, 9], we utilize two dense layers with output sizes of 1024 and 128 with PReLU activation as the feature embeddings and one fully-connected layer as a classifier. The learning rate is 0.001. The size of the mini-batch is 32. λ and α are fixed as 0.1 and 0.5, respectively. We compare the proposed method with CCSA, d-SNE, DAG-LDA, MF, and three baselines (Model 1, 2 and 3 mentioned in the Office31 experiment). We either report the results in the original publications or implement these recent SOTA methods based on their open-source codes.

Table 2 shows the performance of models using the DeCaF-fc6 features on the ten categories of the Office-Caltech-10 database. Again, our method gains a higher accuracy than other SDA approaches and baseline models on both Within-Office31 and Office-Caltech DA tasks. This result shows the advantage of the proposed strategy in DA classification tasks.

4.3 Office-home

Office-Home [32] is a relatively large-scale dataset for DA experiments. It contains 15500 images with 65 classes. The dataset has four different domains, i.e., Art ($\mathcal{A}r$), Clip Art ($\mathcal{C}a$), Product ($\mathcal{P}r$), and Real World ($\mathcal{R}w$). Thus, it contains 12 (4×3) different DA tasks.

The "S+T" evaluation protocol in [42] is implemented in our experiments. Specifically, labeled source images and three labeled target images per class are used to train the model in each DA task. The exact data splits can be found in the link below[2]. According to [42], AlexNet [39] pre-trained on ImageNet is used in our experiments. All images are resized to 227×227, followed by normalization. The size of the mini-batch is 32. λ and α are fixed as 0.001 and 0.5, respectively. We compare the proposed method with three recent SDA methods, i.e., CCSA [6], d-SNE [19], and DAGE-LDA [29]. The experiments of these three methods are conducted based on their open source codes. Three baselines, Model 1, 2 and 3, mentioned in Section 4.1 are also included in the comparison.

Table 3 presents the classification performance on twelve DA tasks of the 65-class Office-Home dataset. The proposed method achieves the highest accuracy in most DA tasks. It also obtains the best average accuracy across different DA tasks.

[2]https://github.com/VisionLearningGroup/SSDA_MME/tree/master/data/txt/office_home

Table 3 Average classification accuracy (%) of different methods on the Office-Home dataset

DA tasks	Method Model 1	Model 2	Model 3	CCSA [6]	d-SNE [19]	DAG-LDA [10]	Ours
$\mathcal{Ar} \to \mathcal{Ca}$	28.2	38.8	40.0	41.3	40.3	40.8	**42.9**
$\mathcal{Ar} \to \mathcal{Pr}$	39.5	57.6	58.4	57.3	58.2	55.3	**61.7**
$\mathcal{Ar} \to \mathcal{Rw}$	51.4	57.0	58.3	59.2	57.1	57.4	**62.5**
$\mathcal{Ca} \to \mathcal{Ar}$	32.0	37.5	40.3	42.5	41.5	41.3	**43.8**
$\mathcal{Ca} \to \mathcal{Pr}$	44.9	57.9	**59.7**	59.1	56.3	57.2	57.6
$\mathcal{Ca} \to \mathcal{Rw}$	47.1	54.3	55.1	**59.1**	58.2	58.4	59.0
$\mathcal{Pr} \to \mathcal{Ar}$	30.2	36.1	38.8	40.0	40.2	**42.2**	41.5
$\mathcal{Pr} \to \mathcal{Ca}$	28.7	44.4	40.7	44.1	43.9	44.0	**45.1**
$\mathcal{Pr} \to \mathcal{Rw}$	53.6	57.8	59.0	58.9	58.4	59.2	**60.4**
$\mathcal{Rw} \to \mathcal{Ar}$	43.4	47.7	47.3	46.4	46.2	48.2	**50.6**
$\mathcal{Rw} \to \mathcal{Ca}$	33.9	44.6	45.4	45.2	46.6	46.7	**48.0**
$\mathcal{Rw} \to \mathcal{Pr}$	60.6	66.7	67.1	66.9	68.2	67.4	**71.3**
Average	41.1	50.0	50.8	51.7	51.3	51.5	**53.7**

The best performance is highlighted in boldface

4.4 Digit transfer

The digit transfer dataset collection has also been popularly used to study the effectiveness of SDA approaches. We use four datasets, all containing hand-written digits from 0 to 9. These datasets include MNIST ($\mathcal{M}$), USPS ($\mathcal{U}$), SVHN ($\mathcal{S}$), and MNIST-M ($\mathcal{MM}$). MNIST consists of 70, 000 28×28 grayscale images; USPS contains 11,000 grayscale images with a 16×16 resolution; SVHN is a real-world image dataset having 99,280 RGB images extracted from street view house numbers; MNIST-M has 68,002 RGB images generated from MNIST by adding different backgrounds.

4.4.1 First experiment

The evaluation protocol in [10, 19] is performed in this experiment. We investigate the transfers from $\mathcal{M}$ to $\mathcal{MM}$, between $\mathcal{M}$ and $\mathcal{U}$, and between $\mathcal{M}$ and $\mathcal{S}$. Original train-test splits of the datasets are used in our experiments. With respect to the target domain, we randomly select 10 samples per class from the training split. The evaluation is repeated five times.

We use the same architecture as LetNets++ [20] according to [19]. Pre-processing techniques, including resize, normalization, and RGB-Greyscale transformation, are applied if necessary. The learning rate for the parameters of the network is 0.001. The size of the mini-batch is 64. λ and α are fixed as 0.75 and 0.5, respectively. We compared the proposed method with CCSA [6], d-SNE [19], DAGE-LDA [29] , and three baselines (Model 1, 2 and 3 mentioned in the Office31 experiment). The details of the architecture are given in Fig. 4.

As shown in Table 4, our method outperforms than other SOTAs and baselines. We notice that the proposed model has a similar performance with d-SNE when the domain shift is small, i.e. the cross-domain tasks $\mathcal{M} \to \mathcal{MM}$, $\mathcal{M} \to \mathcal{U}, \mathcal{U} \to \mathcal{M}$, and $\mathcal{S} \to \mathcal{M}$. When it comes to the $\mathcal{M} \to \mathcal{S}$ condition that has a relatively-large domain shift, our method demonstrates an evident advantage.

4.4.2 Second experiment

It is also interesting to see how the performance of models varies with even a smaller size of samples per class from the target domain. Therefore, we conduct another evaluation protocol in [6, 19, 29] applied to MNIST and USPS datasets. This protocol examines both $\mathcal{M} \to \mathcal{U}$ and $\mathcal{U} \to \mathcal{M}$ cross-domain tasks, where 2000 and 1800 images are randomly selected from MNIST and USPS, respectively. In addition, a small number (N) of labeled samples per category are randomly picked up from the target domain and used in the model training. The evaluation is repeated ten times for each N form 0 to 7. We use the same data splits generated in [6][3].

The implementation details are the same as those in the first experiment of digit transfer above. We compare the proposed method with CCSA [6], FADA [15] , d-SNE [19], and DAGE-LDA [29]. As reported in [10], there are discrepancies in the network architecture between the description in publications and public source codes of CCSA and d-SNE. Furthermore, differences in the model performance between the results reported in original articles and those reproduced by [10, 43] are also relatively

[3] https://github.com/samotiian/CCSA

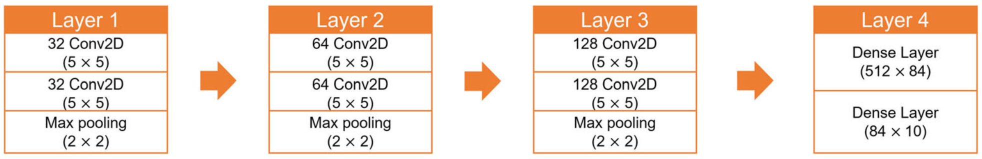


Fig. 4 CNN architecture used in MNIST-USPS experiments

large regarding CCSA and d-SNE. Therefore, for a fair comparison, we amend network architectures of CCSA, d-SNE, and DAGE-LDA as LetNets++ and rerun the experiments based on their publicly available codes. To our best knowledge, the authors of FADA may not release the source code, so we directly report the result in the original publication. Two baselines, Model 2 and Model 3 mentioned in the Office31 experiment, are also included in the comparison. Models 2 and 3 are trained by only source data when $N = 0$.

Table 5 shows the average classification accuracies for different approaches on the MNIST-USPS collection. The standard deviations of ten splits are minor for all methods, so we do not report them in the table. Clearly, SDA-based approaches (expect DAG-LDA) achieve better performance than baselines (Model 2 and 3) which do not incorporate the distribution alignment in training in our experiments. We also plot the performance of the proposed model against Models 2 and 3 on the $\mathcal{M} \rightarrow \mathcal{U}$ task with different N samples per class from $\mathcal{D}_t$ in Fig. 5(a). We observe that the proposed strategy significantly improves the classification performance over baselines when N is small. With an increase of N, the accuracy and the improvement gradually converge.

More importantly, Table 5 shows that our model outperforms other SDA methods in most scenarios, except that it has a slightly lower classification accuracy than FADA when $N = 5$. The proposed approach improves accuracy by at least 1% and 1.5% against other SOTA methods when $N = 7$. When $N = 1$, the proposed method loses its computational advantage against other SDA methods, as the number of pair-up samples is equal to the number of original samples. However, we still recognize an accuracy-wise superiority of our strategy. The superiority may come from the improvement of the discriminative power of the features by decreasing the intra-class variation in addition to the feature alignment between domains during the training.

4.5 Visualization of deep learning features

We visualize the deep features of models trained via multiple N ($N = 0, 1, 4$, or 7) on $\mathcal{U} \rightarrow \mathcal{M}$ to understand the proposed CTL better. We train the model only using softmax loss when $N = 0$, but using the joint supervision of softmax loss and CTL, otherwise. The models are trained by a random draw of 1800 samples from USPS and N samples per class from MNIST. Other settings are the same as those in the digit transfer experiments. The visualization is performed on another random draw of 1800 and 2000 samples from USPS and MNIST, respectively, to avoid visualization on training data. We apply the t-SNE technique [44] to transfer the high-dimension features into 2-D vectors for an easy illustration (Fig. 6).

Figure 6(a) shows that the features are not well aligned if no adaptation mechanism is involved. In addition, we notice that the features with the same label but in different domains stay close to each other even when $N = 1$, as shown in subfigure (b). The alignment gets even better with an increase of N using the proposed CTL loss by

Table 4 Average classification accuracy (%) of different methods on digit transfer tasks

Method	Cross-domain scheme					
	$\mathcal{M} \rightarrow \mathcal{MM}$	$\mathcal{M} \rightarrow \mathcal{U}$	$\mathcal{U} \rightarrow \mathcal{M}$	$\mathcal{M} \rightarrow \mathcal{S}$	$\mathcal{S} \rightarrow \mathcal{M}$	Average
Model 1	62.3±1.0	81.1±1.9	59.1±3.7	43.9±2.1	65.0±2.4	62.3
Model 2	72.1±1.9	93.1±0.8	89.3±2.5	57.6±1.2	82.3±1.2	78.9
Model 3	73.9±2.7	92.7±0.6	93.3±0.4	57.1±2.5	82.7±1.2	79.9
CCSA [6]	78.3±2.0	97.3±0.2	95.7±0.4	37.6±3.6	94.6±0.4	80.7
d-SNE [19]	**87.8±0.2**	**99.0±0.1**	98.5±0.4	61.7±0.5	96.5±0.2	88.7
DAG-LDA* [29]	72.5±1.5	96.5±0.3	93.7±0.7	57.4±0.9	89.5±0.4	81.9
Ours	85.2±1.1	97.7±0.4	**99.2±0.1**	**68.8±1.8**	**96.5±0.2**	**89.5**

[a]The experiments in DAG-LDA [29] are implemented in a rectified experimental setup. This setup is based on train-validation-test split. It is slightly different from the experimental protocol of other methods, which follow the traditional train-test split

The best performance is highlighted in boldface

Table 5 Average classification accuracy (%) of different methods on the MNIST-USPS collection

	Method	Number of training samples from D_t							
		0	1	2	3	4	5	6	7
MINST->USPS	Model 2	70.6	77.5	79.5	81.8	85.4	86.8	88.3	89.4
	Model 3	72.8	82.3	80.1	84.9	89.9	90.2	90.9	92.7
	CCSA [6]	n.a.	78.3	87.1	89.5	90.1	91.4	91.1	92.9
	FADA [15]	n.a.	89.1	91.3	91.9	93.3	**93.4**	94.0	94.4
	d-SNE [19]	n.a.	85.2	89.7	90.9	91.1	92.4	92.3	93.0
	DAG-LDA [29]	n.a.	80.8	85.7	86.5	88.8	89.2	89.2	89.5
	Ours	n.a.	**89.6**	**91.9**	**93.1**	**93.6**	93.3	**94.7**	**95.4**
USPS->MINST	Model 2	62.2	63.3	69.6	76.7	83.1	78.5	78.6	81.9
	Model 3	63.7	72.3	77.4	81.9	83.9	86.4	88.0	88.7
	CCSA [6]	n.a.	66.7	74.1	80.9	81.9	84.9	85.7	87.5
	FADA [15]	n.a.	81.1	84.2	87.5	89.9	91.1	91.2	91.5
	d-SNE [19]	n.a.	79.2	84.1	86.0	86.1	88.6	88.0	89.8
	DAG-LDA [29]	n.a.	64.7	66.2	70.8	73.2	74.7	74.5	77.2
	Ours	n.a.	**81.8**	**85.1**	**90.7**	**91.2**	**92.1**	**92.5**	**93.0**

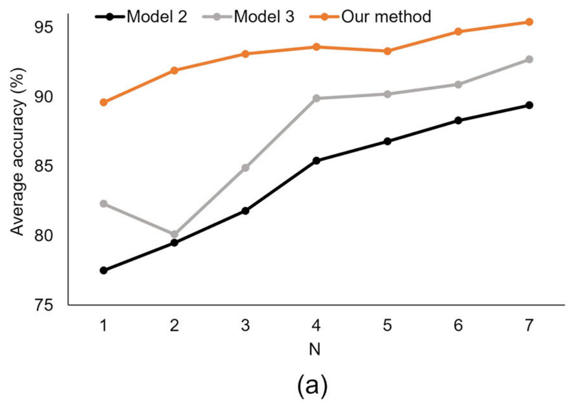
"n.a." refers to the training of the method requires samples from $\mathcal{D}_t$ and thus can not be implemented in the "0" condition
The best performance is highlighted in boldface

comparing the subfigures (b), (c), and (d). We can still observe discrepancies of features between $\mathcal{D}_s$ and $\mathcal{D}_t$ in several classes, e.g., classes 0 and 6 when N is equal to 1 or 4. However, the distributions of features between $\mathcal{D}_s$ and $\mathcal{D}_t$ nearly overlap with each other in the case when $N = 7$, as demonstrated in subfigure (d).

4.6 Sensitivity analysis on λ and α

The hyper-parameters λ and α control the adaptation rate between domains and the negative impact of noisy samples, respectively. They both are significant to the training of the DNN model. Therefore, we carry out analyses to demonstrate their sensitiveness.

The analyses consider four cross-domain tasks, i.e., $\mathcal{W} \rightarrow \mathcal{A}$ (Office-31), $\mathcal{C} \rightarrow \mathcal{A}$ (Office-Caltech-10), $\mathcal{Ar} \rightarrow \mathcal{Ca}$ (Office-Home), and $\mathcal{U} \rightarrow \mathcal{M}$ (Digit transfer). The experimental protocols of $\mathcal{W} \rightarrow \mathcal{A}$, $\mathcal{C} \rightarrow \mathcal{A}$, and $\mathcal{Ar} \rightarrow \mathcal{Ca}$ are the same as those described in previous sections, except that the values of λ and α are not fixed but vary in this sensitivity analysis. For $\mathcal{U} \rightarrow \mathcal{M}$ task, we sample 1800 images from USPS and 3 images per class from MNIST to form a training set. The evaluation is performed on 2000 samples randomly drawn from



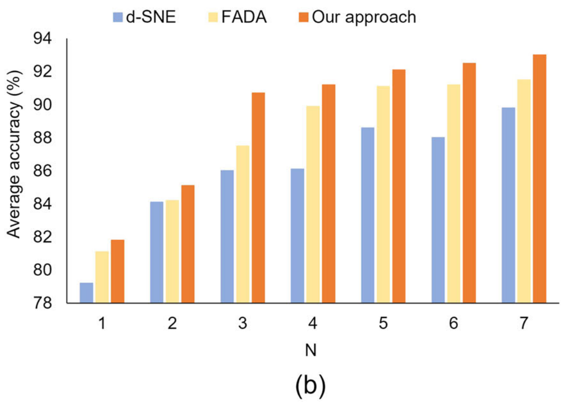


Fig. 5 Average classification accuracy for (a) $\mathcal{M} \rightarrow \mathcal{U}$ and (b) $\mathcal{U} \rightarrow \mathcal{M}$ tasks for different number (N) of labeled tagert samples per class from $\mathcal{D}_t$

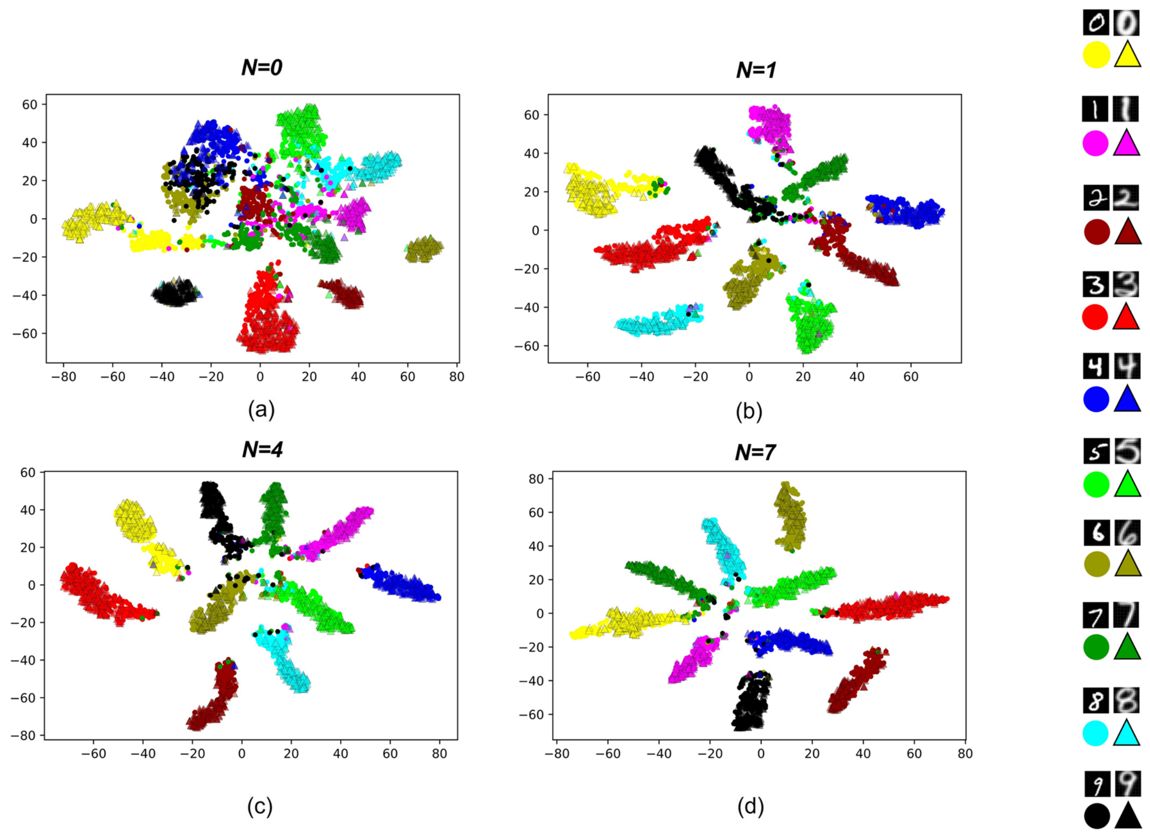


Fig. 6 Visualization of deep learning feature distributions varying N labeled samples per class from $\mathcal{D}_t$ on $\mathcal{U} \rightarrow \mathcal{M}$ task. (a) $N = 0$; without DA. (b) $N = 1$. (c) $N = 4$. (d) $N = 7$. Circles and triangles are from $\mathcal{D}_s$ and $\mathcal{D}_t$, respectively

MNIST, excluding the samples that have been selected for training. The implementation details are the same as those in Section 4.4.2, except that the values of λ and α vary in this analysis.

First, we fix the center step α as 0.5 and vary λ values to train different models. As CTL is based on $l2 - norm$, the dimensionality of the deep feature is positively related to the loss value, i.e., a higher dimension leads to a larger CTL. Dimensionalities of deep features for four cross-domain tasks are diverse. Therefore, we test different ranges of λ values for different tasks as follows.

1. $\mathcal{W} \rightarrow \mathcal{A}$: {0, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1}
2. $\mathcal{C} \rightarrow \mathcal{A}$: {0, 0.025, 0.05, 0.075, 0.1, 0.25, 0.5, 0.75, 1}
3. $\mathcal{A}r \rightarrow \mathcal{C}a$: {0, 0.00025, 0.0005, 0.00075, 0.001, 0.0002 5, 0.005, 0.0075, 0.01}
4. $\mathcal{U} \rightarrow \mathcal{M}$: {0, 0.1, 0.25, 0.5, 0.75, 1, 2.5, 5}

Figure 7 shows the evaluation accuracies of the proposed method with different values of λ. It is observed that simply using the softmax loss (when $\lambda = 0$) is not a good option, and DNN models have the lowest average classification accuracy. We can also observe a “Log” curve of the model performance in all four tasks when λ varies in the investigating ranges. This shows that our model is insensitive to λ values in a relatively large scope.

DNN models achieve the best performance when λ is equal to 0.1, 0.1, 0.001, and 0.75 for the evaluated four tasks, respectively. We then fix λ as these values and train DNN models with different values of α from 0.01 to 1 (the same range for all tasks). The evaluation accuracies of these models with different values α are shown in Fig. 8. We also

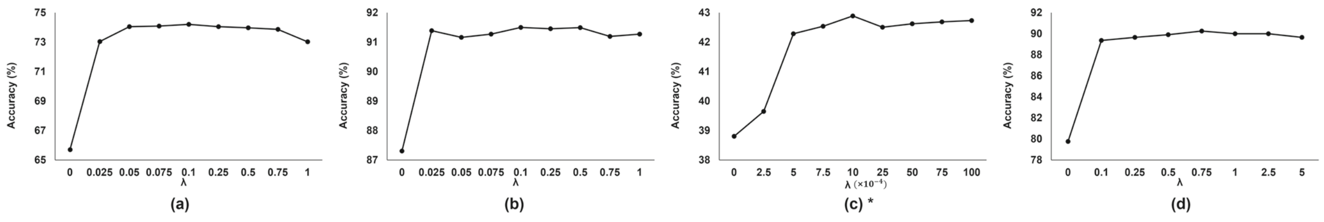


Fig. 7 Classification accuracies on DA tasks achieved by models with different λ and fixed $\alpha = 0.5$. (a) $\mathcal{W} \rightarrow \mathcal{A}$; (b) $\mathcal{C} \rightarrow \mathcal{A}$; (c) $\mathcal{A}r \rightarrow \mathcal{C}a$, *values on the x-axis are on a 10^{-4} basis; and (d) $\mathcal{U} \rightarrow \mathcal{M}$

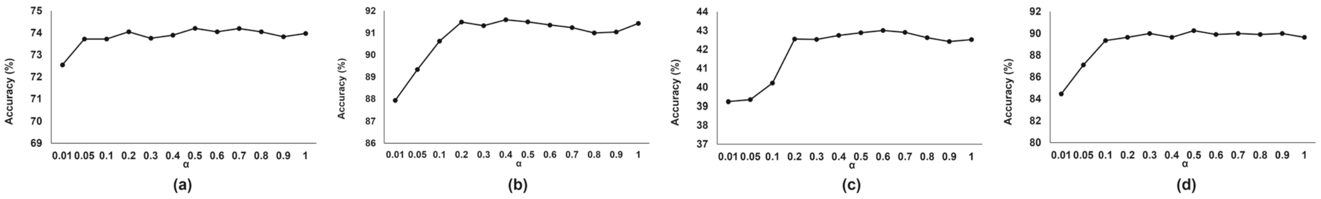


Fig. 8 Classification accuracies on DA tasks achieved by models with different α and fixed λ. (a) $\mathcal{W} \rightarrow \mathcal{A}$ ($\lambda = 0.1$); (b) $\mathcal{C} \rightarrow \mathcal{A}$ ($\lambda = 0.1$); (c) $\mathcal{A}r \rightarrow \mathcal{C}a$ ($\lambda = 0.001$); and (d) $\mathcal{U} \rightarrow \mathcal{M}$ ($\lambda = 0.75$)

observe stable performances among these models across different values of α, i.e., from 0.2 to 1.

4.7 Size of mini-batch

The proposed CTL can be trained by the mini-batch strategy. It is also interesting to explore how the size of mini-batch influences the effectiveness of CTL. We conduct experiments on a cross-domain task, $\mathcal{U} \rightarrow \mathcal{M}$. We sample 1800 images from USPS and 3 images per class from MNIST to form a training set. The evaluation is performed on 2000 samples randomly drawn from MNIST, excluding the samples that have been selected for training. The implementation details are the same as those in the digit transfer experiments, except that the batch size is not fixed as 64 at this time. Batch sizes that are multiples of powers of 2 are common in DL training. Thus, different values, i.e., {1, 2, 4, 8, 16, 32, 64, 128, 256, 512}, are tested in the analysis.

The accuracy of the proposed method using different values of batch size is shown in Fig. 9. It is observed that the performance is unsatisfactory when the batch size is small. We further plot the learning curve of CTL using different batch sizes to identify the minimization process of CTL (Fig. 10). The figure shows that the training of CTL is unstable when the batch size is small. The smaller the batch size is, the more volatile the training process becomes. Updating the centers based on very few samples each time naturally leads to great randomness and biases. When the batch size gets larger, the learning curves of CTL become stable. In addition, we also identify the accuracy drops in Fig. 9 when the batch size is relatively big (i.e., 256 and 512). It is consistent with the finding in [45]. The study states that a large batch size (over 10% of the full batch) may not be a good choice. A model trained using a larger batch size is more likely to converge to sharp minima, e.g., the model is reasonably good but does not offer the best solution to the classification task.

In sum, although the choice of batch size has an impact on the effectiveness of the proposed CTL, our loss is robust to the common options of batch size in DL training. Unless the batch size is either too small or too large, the model performance remains to be satisfactory and stable.

5 Conclusion

Domain adaptation has drawn considerable interest in the DL community recently. It aims to make use of the copious amount of accessible data from different domains. In this work, we propose a new loss function, referred to as CTL. It is trainable using a single-stream network based on the mini-batch strategy. By a joint supervision of the softmax loss and CTL, same-class features between source and target domains achieve a desirable degree of

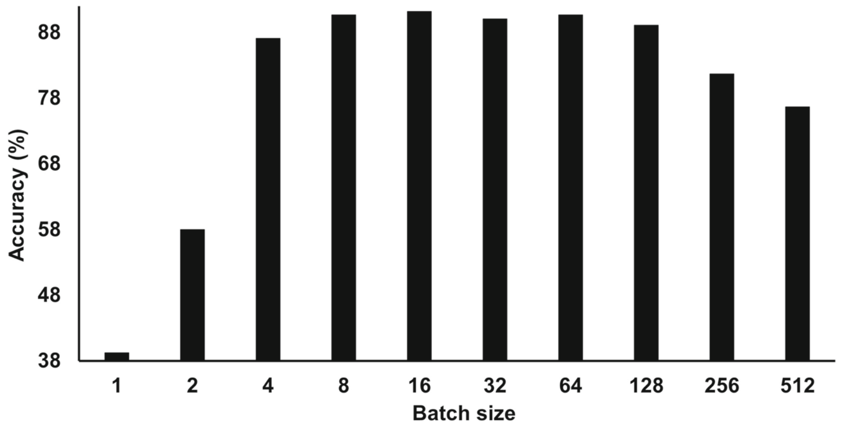


Fig. 9 Classification accuracies of the proposed method on $\mathcal{U} \rightarrow \mathcal{M}$ task using different batch sizes

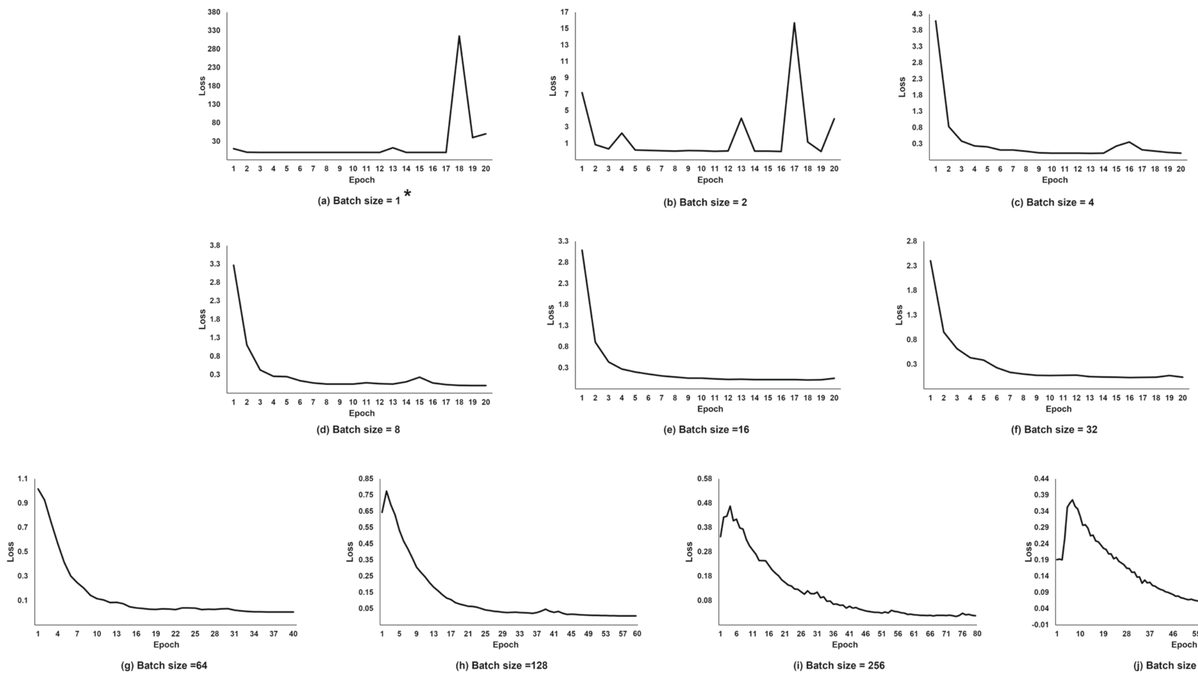


Fig. 10 Learning curves of CTL on $\mathcal{U} \rightarrow \mathcal{M}$ task using different values of batch sizes. *The learning curve in subfigure (a) seems to be flat before 18th epoch. This is because the loss at 18th epoch reaches over 300, and the losses at other epochs are much smaller than this value. In fact, the losses in this learning curve are volatile during the training process

alignment and a compact intra-class variation. At the same time, different-class features keep sufficiently separated. The usage of CTL results in both domain alignment and the minimization of intra-class variation subsequently in the early and latter training stages without the need to set trade-off values to balance these two functions. The "single-stream implementation" and "manual-balance-waived simultaneous achievement of domain alignment and intra-class variation minimization" are two main advantages of our approach compared to previous methods. Experiments in the present study show that our approach performs better than baselines and recent SOTAs under identical settings across standard DA benchmarks.

Although the proposed CTL offers an encouraging outcome, it is worthwhile to investigate whether its variants provide more promising performance. For example, referring to [46], we can try using only the nearest feature points of each class center instead of relying on all feature points to update the class centers in each iteration. This implementation may be able to decrease the negative impact of the out-of-distribution feature points in the center update. Moreover, in addition to using the $l2 - norm$ distance, other metrics, such as the cosine distance and other types of norms, are also valuable to be explored in future works.

Acknowledgements The work in this paper was supported in part by the Hong Kong Research Grants Council (PolyU 152006/19E).

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Li Z, Liu F, Yang W, Peng S, Zhou J (2021) A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans Neural Netw Learn Syst, 1–21, https://doi.org/10.1109/TNNLS.2021.3084827
2. Roy Y, Banville H, Albuquerque I, Gramfort A, Falk TH, Faubert J (2019) Deep learning-based electroencephalography analysis: a systematic review. J Neural Eng 16(5):051001
3. Aggarwal R, Sounderajah V, Martin G, Ting DS, Karthikesalingam A, King D, Ashrafian H, Darzi A (2021) Diagnostic accuracy of deep learning in medical imaging: a systematic review and meta-analysis. NPJ Digit Med 4(1):1–23
4. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22(10):1345–1359. https://doi.org/10.1109/TKDE.2009.191
5. Wang M, Deng W (2018) Deep visual domain adaptation: a survey. Neurocomputing 312:135–153. https://doi.org/10.1016/j.neucom.2018.05.083
6. Motiian S, Piccirilli M, Adjeroh DA, Doretto G (2017) Unified deep supervised domain adaptation and generalization. In: Proceedings of the IEEE international conference on computer vision (ICCV)
7. Wilson G, Cook DJ (2020) A survey of unsupervised deep domain adaptation. ACM Trans Intell Syst Technol, 11(5). https://doi.org/10.1145/3400066

8. Singh A (2021) CLDA: contrastive learning for semi-supervised domain adaptation. In: Beygelzimer A, Dauphin Y, Liang P, Vaughan JW (eds) Advances in neural information processing systems. https://openreview.net/forum?id=1ODSsnoMBav
9. Tong X, Xu X, Huang S-L, Zheng L (2021) A mathematical framework for quantifying transferability in multi-source transfer learning. In: Beygelzimer A, Dauphin Y, Liang P, Vaughan JW (eds) advances in neural information processing systems. https://openreview.net/forum?id=wQZWg82TWx
10. Hedegaard L, Sheikh-Omar OA, Iosifidis A (2021) Supervised domain adaptation : a graph embedding perspective and a rectified experimental protocol. IEEE Trans Image Proc 30:8619–8631. https://doi.org/10.1109/TIP.2021.3118978
11. Wang Y, Liu J, Ruan Q, Wang S, Wang C (2021) Cross-subject eeg emotion classification based on few-label adversarial domain adaption. Expert Syst Appl 115581:185. https://doi.org/10.1016/j.eswa.2021.115581
12. Sawyer D, Fiaidhi J, Mohammed S (2021) Few shot learning of covid-19 classification based on sequential and pretrained models: a thick data approach. In: 2021 IEEE 45Th annual computers, software, and applications conference (COMPSAC), pp 1832–1836. https://doi.org/10.1109/COMPSAC51774.2021.00276
13. Abdelwahab M, Busso C (2015) Supervised domain adaptation for emotion recognition from speech. In: 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 5058–5062. https://doi.org/10.1109/ICASSP.2015.7178934
14. Li X, He Y, Zhang JA, Jing X (2021) Supervised domain adaptation for few-shot radar-based human activity recognition. IEEE Sensors J 21(22):25880–25890. https://doi.org/10.1109/JSEN.2021.3117942
15. Motiian S, Jones Q, Iranmanesh SM, Doretto G (2017) Few-shot adversarial domain adaptation. In: Proceedings of the 31st international conference on neural information processing systems. NIPS'17, Curran Associates Inc, pp 6673–6683
16. Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML Deep Learning Workshop, vol 2. Lille, p 0
17. Koniusz P, Tas Y, Porikli F (2017) Domain adaptation by mixture of alignments of second-or higher-order scatter tensors. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 7139–7148. https://doi.org/10.1109/CVPR.2017.755
18. Tzeng E, Hoffman J, Darrell T, Saenko K (2015) Simultaneous deep transfer across domains and tasks, pp 4068–4076. https://doi.org/10.1109/ICCV.2015.463
19. Xu X, Zhou X, Venkatesan R, Swaminathan G, Majumder O (2019) D-sne: domain adaptation using stochastic neighborhood embedding. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 2492–2501. https://doi.org/10.1109/CVPR.2019.00260
20. Wen Y, Zhang K, Li Z, Qiao Y (2016) A discriminative feature learning approach for deep face recognition. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer vision – ECCV 2016, Springer, pp 499–515
21. Fernando B, Tommasi T, Tuytelaars T (2015) Joint cross-domain classification and subspace learning for unsupervised adaptation Pattern Recognition Letters, 65. https://doi.org/10.1016/j.patrec.2015.07.009
22. Teshima T, Sato I, Sugiyama M (2020) Few-shot domain adaptation by causal mechanism transfer. In: International conference on machine learning, PMLR, pp 9458–9469
23. Taskesen B, Yue M-C, Blanchet J, Kuhn D, Nguyen VA (2021) Sequential domain adaptation by synthesizing distributionally robust experts. In: International conference on machine learning, PMLR, pp 10162–10172
24. Corral-Soto ER, Nabatchian A, Gerdzhev M, Bingbing L (2021) Lidar few-shot domain adaptation via integrated cyclegan and 3d object detector with joint learning delay. In: 2021 IEEE international conference on robotics and automation (ICRA), pp 13099–13105. https://doi.org/10.1109/ICRA48506.2021.9561466
25. Zhong C, Wang J, Feng C, Zhang Y, Sun J, Yokota Y (2022) Pica: Point-wise instance and centroid alignment based few-shot domain adaptive object detection with loose annotations. In: 2022 IEEE/CVF winter conference on applications of computer vision (WACV), pp 398–407. https://doi.org/10.1109/WACV51458.2022.00047
26. Zhou JT, Tsang IW, Pan SJ, Tan M (2014) Heterogeneous domain adaptation for multiple classes. In: Kaski S, Corander J (eds) Proceedings of the seventeenth international conference on artificial intelligence and statistics. Proceedings of machine learning research, vol 33. PMLR, pp 1095–1103. https://proceedings.mlr.press/v33/zhou14.html
27. Sukhija S, Krishnan NC, Singh G (2016) Supervised heterogeneous domain adaptation via random forests. In: IJCAI
28. Hadsell R, Chopra S, LeCun Y (2006) Dimensionality reduction by learning an invariant mapping. In: 2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06), vol 2. pp 1735–1742. https://doi.org/10.1109/CVPR.2006.100
29. Morsing LH, Sheikh-Omar OA, Iosifidis A (2021) Supervised domain adaptation using graph embedding. In: 2020 25Th international conference on pattern recognition (ICPR), pp 7841–7847. https://doi.org/10.1109/ICPR48806.2021.9412422
30. Saenko K, Kulis B, Fritz M, Darrell T (2010) Adapting visual category models to new domains. In: Daniilidis K, Maragos P, Paragios N (eds) Computer vision – ECCV 2010, Springer, pp 213–226
31. Gong B, Shi Y, Sha F, Grauman K (2012) Geodesic flow kernel for unsupervised domain adaptation. In: 2012 IEEE conference on computer vision and pattern recognition, pp 2066–2073. https://doi.org/10.1109/CVPR.2012.6247911
32. Venkateswara H, Eusebio J, Chakraborty S, Panchanathan S (2017) Deep hashing network for unsupervised domain adaptation. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), pp 5385–5394
33. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324. https://doi.org/10.1109/5.726791
34. LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L, Handwritten digit recognition with a back-propagation network, Touretzky D (1989). In: Advances in neural information processing systems, vol 2. Morgan-Kaufmann, https://proceedings.neurips.cc/paper/1989/file/53c3bce66e43be4f209556518c2fcb54-Paper.pdf
35. Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY (2011) Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on deep learning and unsupervised feature learning 2011
36. 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(1):2096–2030
37. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) 3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. arXiv:1409.1556
38. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A, Fei-Fei L (2014) Imagenet large scale visual recognition challenge International Journal of Computer Vision 1150. https://doi.org/10.1007/s11263-015-0816-y

39. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems, vol 25. Curran Associates Inc
40. Griffin G, Holub A, Perona P (2007) Caltech-256 object category dataset CalTech Report
41. Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, Darrell T (2014) Decaf: a deep convolutional activation feature for generic visual recognition. In: Xing EP, Jebara T (eds) Proceedings of the 31st international conference on machine learning. Proceedings of machine learning research, vol 32. PMLR, pp 647–655. https://proceedings.mlr.press/v32/donahue14.html
42. Saito K, Kim D, Sclaroff S, Darrell T, Saenko K (2019) Semi-supervised domain adaptation via minimax entropy. In: 2019 IEEE/CVF international conference on computer vision (ICCV), IEEE Computer Society, pp 8049–8057. https://doi.org/10.1109/ICCV.2019.00814
43. Wang Z, Du B, Guo Y (2020) Domain adaptation with neural embedding matching. IEEE Trans Neural Netw Learn Syst 31(7):2387–2397. https://doi.org/10.1109/TNNLS.2019.2935608
44. van der Maaten L, Hinton G (2008) Visualizing data using t-sne. J Mach Learn Res 9(86):2579–2605
45. Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP (2017) On large-batch training for deep learning : generalization gap and sharp minima. In: International conference on learning representations. https://openreview.net/forum?id=H1oyRlYgg
46. Wang X, Zheng Z, He Y, Yan F, Zeng Z, Yang Y (2021) Soft person reidentification network pruning via blockwise adjacent filter decaying. IEEE Trans Cybern, 1–15. https://doi.org/10.1109/TCYB.2021.3130047

Xiuyu Huang finished his M.S. degree at Univesity of Sydney in 2020. He is now a PhD student at the Hong Kong Polytechnic University. His research interest includes brain-computer interfaces and deep learning. He publishes papers in good journals and conferces such as IEEE TNSRE and AAAI.

Nan Zhou received his Ph.D. degree from Center for Robotics, University of Electronic Science and Technology of China, Chengdu, Sichuan. He was a joint Ph.D. student in University of Alberta, Canada, in 2015. He was a Postdoc Fellow at the Centre for Smart Health, Hong Kong Polytechnic University during 2020-2021. He was a lecturer with Chengdu University of Information Technology during 2017-2022. He is now an Associate Professor with Chengdu University, China. His research interests include machine learning, feature learning, computer vision and sparse modeling.

Jian Huang is a M.S. student in Chengdu University of Information Technology, Chengdu, China. His research interest covers brain-computer interfaces and deep learning.

Huaidong Zhang is an Associate Professor in the School of Future Technology, South China University of Technology. He was a Postdoctoral Fellow at The Hong Kong Polytechnic University. He received his B.Eng. and Ph.D. degrees in Computer Science and Engineering from the South China University of Technology in 2015 and 2020 respectively. His research interests include computer vision, image processing, computer graphics and deep learning.

Witold Pedrycz (IEEE Life Fellow) is Professor in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. He is also with the Systems Research Institute of the Polish Academy of Sciences, Warsaw, Poland. Dr. Pedrycz is a foreign member of the Polish Academy of Sciences and a Fellow of the Royal Society of Canada. He is a recipient of several awards including Norbert Wiener award from the IEEE Systems, Man, and Cybernetics Society, IEEE Canada Computer Engineering Medal, a Cajastur Prize for Soft Computing from the European Centre for Soft Computing, a Killam Prize, a Fuzzy Pioneer Award from the IEEE Computational Intelligence Society, and 2019 Meritorious Service Award from the IEEE Systems Man and Cybernetics Society.

His main research directions involve Computational Intelligence, Granular Computing, and Machine Learning, among others.

Professor Pedrycz serves as an Editor-in-Chief of ***Information Sciences***, Editor-in-Chief of ***WIREs Data Mining and Knowledge Discovery*** (Wiley), and Co-editor-in-Chief of ***Int. J. of Granular Computing*** (Springer) and ***J. of Data Information and Management*** (Springer).

Kup-Sze Choi received his Ph.D. in Computer Science and Engineering from the Chinese University of Hong Kong. He is currently a professor at the Hong Kong Polytechnic University and the director of the Centre for Smart Health. He was also the founding programme leader of the Hong Kong's first M.Sc. programme in Health Informatics. His research interest is machine learning algorithms and their medical and healthcare applications. Thomas was a recipient of the National Natural Science Award 2017 of the Ministry of Education, China, and is among the World's Top 2% Most-Cited Scientists in the sub-field of artificial intelligence and image processing in 2022.

Affiliations

Xiuyu Huang[1,2] · **Nan Zhou**[3] · **Jian Huang**[4] · **Huaidong Zhang**[5] · **Witold Pedrycz**[2,6,7,8] · **Kup-Sze Choi**[1]

Nan Zhou
nzhouuestc@126.com

Jian Huang
jackyhuang_cuit@126.com

Huaidong Zhang
huaidongz@scut.edu.cn

[1] Center for Smart Health, The Hong Kong Polytechnic University, Hong Kong SAR, 999077, China

[2] Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, T6G 2R3, Canada

[3] School of Electronic Information and Electronic Engineering, Chengdu University, Chengdu, 610000, China

[4] College of Control Engineering, Chengdu University of Information Technology, Chengdu, 610101, China

[5] School of Computer Science and Engineering, South China University of Technology, Guangzhou, 510000, China

[6] Systems Research Institute, Polish Academy of Sciences, 00-901 Warsaw, Poland

[7] Department of Electrical and Computer Engineering Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia

[8] Faculty of Engineering and Natural Sciences, Department of Computer Engineering, Istinye University, Sariyer/Istanbul, Turkiye