DC-FUDA: Improving Deep Clustering via Fully Unsupervised Domain Adaptation

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

16-07-2021 / 23-07-2021

CITATION

Yang, Zhimeng; Wu, Zirui; Zeng, Ming; Ren, Yazhou; Pu, Xiaorong; Xu, Jie; et al. (2021): DC-FUDA: Improving Deep Clustering via Fully Unsupervised Domain Adaptation. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.14995086.v1>

DOI

[10.36227/techrxiv.14995086.v1](https://doi.org/10.36227/techrxiv.14995086.v1)

DC-FUDA: Improving Deep Clustering via Fully Unsupervised Domain Adaptation

Zhimeng Yang, Yazhou Ren*, Zirui Wu, Ming Zeng, Jie Xu, Xiaorong Pu, Yang Yang, *Senior Member, IEEE*

Abstract—By transferring knowledge from a source domain, the performance of deep clustering on an unlabeled target domain can be improved. When achieving this, traditional approaches make the assumption that adequate amount of labeled data is available in a source domain. However, this assumption is usually unrealistic in practice. The source domain should be carefully selected to share some characteristics with the target domain, and it can not be guaranteed that rich labeled samples are always available in the selected source domain.

We propose a novel framework to improve deep clustering by transferring knowledge from a source domain without any labeled data. To select reliable instances in the source domain for transferring, we propose a novel adaptive threshold algorithm to select low entropy instances. To transfer important features of the selected instances, we propose a feature-level domain adaptation network (FeatureDA) which cancels unstable generation process. With extensive experiments, we validate that our method effectively improves deep clustering, without using any labeled data in the source domain. Besides, without using any labeled data in the source domain, our method achieves competitive results, compared to the state-of-the-art methods using labeled data in the source domain.

Index Terms—transfer learning, Unsupervised domain adaptation, deep clustering, GAN.

I. INTRODUCTION

WITH deep neural networks as feature extractors, deep clustering methods can achieve higher accuracy in unsupervised learning tasks, compared to traditional clustering methods. Deep clustering has received massive attentions and can be widely applied to the scenarios without sufficient supervised information such as motion segmentation and face clustering [1], acoustic source separation [2], and reducing the manufacturing process quality variations [3]. The performance of deep clustering methods on an unlabeled target domain can be further improved, by transferring knowledge from a relative domain (*i.e.*, source domain) [4], [5]. To achieve this, traditional approaches, such as unsupervised domain adaptation (UDA) [4], [5], are developed to transfer information from a source domain to improve accuracy in the target task.

However, when developing these traditional approaches [4], [5] transferring knowledge from source domains, an unrealistic assumption is often made that rich labeled samples are available in the source domain. In practice, the source domain should be carefully selected to share some characteristics

Fig. 1: The first line shows 5 samples (digit ‘2’) in USPS data and the second line gives samples (label ‘mouse’) in OFFICE-31 data. The computed entropy of each sample increases from left to right. It is easy to see that low entropy instances are typically with high confidence.

with the target domain. If the source domain is not carefully selected, the accuracy in the target domain will decrease. More importantly, it can not be guaranteed that rich labeled samples are always available in the carefully selected source domain. Labeling data in a particular domain can require a lot of human labor or expertise, which is very expensive. When there are limited labeled data samples in the source domain, traditional approaches can not be applied.

To overcome the above assumption made by traditional approaches, we present the first work and propose a novel framework to improve the deep clustering performance by transferring knowledge from a source domain without any labeled data. First, we apply deep clustering methods on source domain to obtain the cluster soft assignments (*i.e.*, pseudo labels) of the data instances in the source domain. Then, we select instances with high confidences, and use these instances to transfer knowledge from the source domain to the target domain. Specifically, with these selected instances, we train a generative adversarial network (GAN) [6] to generate fake features, which share similar distribution with the learned features in the target domain. Finally, the selected instances and generated fake features are used to train a classifier, to further predict the final cluster assignments of the data instances in the target domain.

It is non-trivial to select the appropriate instances and transfer information from these instances in our approach. We need to (i) select the reliable instances with important characteristics which can be shared between the source domain and target domain, and (ii) transfer important features of these instances from the source domain to enhance the clustering

Zhimeng Yang, Yazhou Ren, Zirui Wu, Jie Xu, Xiaorong Pu and Yang Yang are with the School of Computer Science and Engineer, University of Electronic Science and Technology of China, Chengdu 611731, China. Ming Zeng is with the Department of Electrical Computer Engineering, Carnegie Mellon University, PA 15213, USA.
Yazhou Ren is the corresponding author. E-mail: yazhou.ren@uestc.edu.cn.

performance in the target domain. To (i), information entropy is an intuitive measurement for the instances' confidence. Construct a model which uses one-hot encoding labels as supervision, the output vectors after normalization can be explained as the probability for the instances to every cluster, and then we can compute the entropy of the instances. The problem is how to choose a cutoff entropy value so that the instances below it will be selected for transfer. Inspired by the work [7], we use a low-entropy object for the clustering network to get entropy values which can be easily divided. To (ii), the Existing GAN-based UDA nets considers both image generation and classification, which means the vast majority of computing power would be used for the generation task rather than classification. Some works use the features extracted by the existing classification network as the input to decrease the computing complexity which ignores the training process of the feature extractor. We propose a new model which asks for the source and target image as input and generates features following the target image features' distribution.

In summary, we make three major contributions.

- We identify an unrealistic assumption made by traditional approaches, and present the first work to improve the deep clustering performance by transferring knowledge from a source domain without any labeled data.
- To select reliable instances in the unlabeled source domain for transferring, we propose a novel adaptive threshold algorithm to select low entropy instances. To transfer important features of the selected instances, we propose a feature-level domain adaptation network for clustering.
- It is empirically verified that our method effectively improves deep clustering, when no label information in the source domain is provided. In addition, compared to existing domain adaptation methods which need labeled data of source domain, our method can obtain competitive performance.

II. RELATED WORK

In our method, first we need a traditional deep clustering algorithm to label the instances based on instances' entropy, second, we ask for an UDA method. So we will introduce the works about traditional deep clustering algorithms and UDA algorithms in this part.

A. Deep Clustering

For the clustering methods based on a deep neural network, the most important and difficult part is to set a suitable objective. Find a practical and can be widely-used objective is the accelerator for unsupervised learning. In recent years, amounts of deep clustering methods with different objectives have been proposed. Autoencoder(AE) [8] has been the basic net frame for its' ability to learning high non-linear mapping functions. [9] proposed a deep clustering method based on autoencoder (AE) considering both data reconstruction and compactness. To learn the cluster-oriented feature representations, [10] imposed locality-preserving constraint and group sparsity when learning the deep autoencoder. [11] proposed

a deep subspace clustering which enhances autoencoder by considering the structure prior to samples. Density-based clustering algorithms are also extended to deep learning, e.g., [12] used autoencoder to learn the low-dimensional feature representations and then proposed a density-based method to partition the learned features. Inspired by t-SNE [13], [7] proposed a joint framework to optimize feature learning and clustering objective which is named deep embedded clustering (DEC). DEC also uses the features exposed by the autoencoder and then sets a low entropy objective for the encoder. As variations of DEC, improved deep embedded clustering (IDEC) with local structure preservation [14] and deep embedded clustering with data augmentation (DEC-DA) [15] are proposed. IDEC optimizes the weighted clustering loss and the reconstruction loss of autoencoder jointly. DEC-DA uses data augmentation in deep embedded clustering. GAN can also be used in clustering tasks. [16] proposed adversarial deep embedded clustering (ADEC) which addresses the feature randomness and feature drift using adversarial training. Except the model objective, different training strategies are imposed into deep clustering, e.g., self-paced learning [17], [18], multi-view learning [19], [20] and semi-supervised learning [21], [22].

B. Unsupervised Domain Adaptation

The objective of domain adaptation is not only based on the data we want to cluster or generate, it's also based on another dataset or module.

Unsupervised domain adaptation (UDA) aims at transferring knowledge from labeled data of the source domain to enhance the prediction performance on the target domain with only unlabeled data. The objectives of UDA methods are always based on the discrepancy of domains [5], [23]. [23] used maximum mean discrepancy (MMD) [24] to calculate the discrepancy between domains. After this, a number of methods based on MMD have been proposed. The main optimization way is to choose different versions of MMD, such as joint adaption network (JAN) [5] and weighted deep adaption network (WDAN) [25]. JAN choose joint MMD which measures the Hilbert-Schmidt norm between kernel mean embedding of empirical joint distributions of source and target data. WDAN is proposed to solve the question about imbalanced data distribution by introducing an auxiliary weight for each class in the source domain. The methods based on reconstruction use AE as the base model. [26] uses the same encoder to source data and target data. The source features extracted by the encoder are delivered to a classifier and the target data would be reconstructed by the decoder. [27] proposes Contrastive Adaptation Network (CAN) optimize the intra-class domain discrepancy and the inter-class domain discrepancy alternately. UDA methods based on GAN also attracted peoples' attention. They generally use the generator to generate the fake samples and use a discriminator to judge whether each generated instance is sampled from the target domain. Once the discriminator is fixed, the source data and target data can be projected to the same feature space with the generator. [28] used noise vectors and source data to generate

fake data for the discriminator. [29] provided independent generator and discriminator. The domain adaptation is completed by sharing the weights in the generator's first layer and the discriminator's last layer. [30] proposed an attention module in the training process of GAN which allows the discriminator to distinguish the transferable regions among the source and target images. [31] proposes an unsupervised domain adaptation model which can generate image-label pairs in the target domain and achieve class-level transfer.

How to pick an appropriate source domain is always ignored. Recently, UDA methods without source domain are proposed. [32] proposes a collaborative class conditional generative adversarial net to bypass the dependence on the source data. [33] proposes a new framework called SHOT (Source HypOthesis Transfer) which freezes the classifier module of the source model and learns the target-specific feature extraction module by exploiting both information maximization and self-supervised pseudo-labeling to implicitly align representations from the target domains to the source hypothesis.

III. PROPOSED METHOD

In this section, we introduced our deep clustering approach via fully unsupervised domain adaptation (DC-FUDA). In order to transfer knowledge from unlabeled source domain to target domain, we select the high confidence pseudo-labeled data from source domain for target domain deep clustering with a feature domain adaptation network (FeatureDA).

The proposed method mainly comprises two modules: Instances selection with adaption threshold module and feature-level domain adaptation with generative adversarial network. Fig. 2 shows the whole model. The upper part is instances selection with adaption threshold module. In training process, the source domain instances are input to the feature extractor and then the features are optimized for getting lower entropy. Then, with our adaptive threshold instance selection algorithm, we can transfer low-entropy instances to the nether part of Fig. 2: FeatureDA. In FeatureDA, generator G generates fake features share the distribution with features of target data. The labels of the fake features are the same with the instances used to generate them. Then, the source low-entropy features (extract from the feature extractor) and fake features are used to optimize the classifier C . In the test phase, the source and target images are all handled to the classifier.

Consider the source domain with N^s instances, $X^s = \{\mathbf{x}_i^s\}_{i=1}^{N^s}$, and N^t instances in the target domain, $X^t = \{\mathbf{x}_i^t\}_{i=1}^{N^t}$. We aim at clustering instances from both source and target domain into k clusters. The whole model consists of the following components:

- **Feature extractor:** $E(\mathbf{x}_i^s; \theta_I)$, where θ_I is model parameters. It iteratively maps the input X^s to a feature space and assign pseudo labels to the source instances, resulting in $X = \{\mathbf{x}_i, y_i^{ps}\}_{i=1}^{N^s}$, where y_i^{ps} is the pseudo label of instances i .
- **Low entropy instance selector:** It selects N^{le} ($N^{le} < N^s$) high confident (i.e., low entropy) instances $X^{le} = \{\mathbf{x}_i, y_i^{ps}\}_{i=0}^{N^{le}} \subseteq X^s$ for transferring.

Fig. 2: The overview of the model architecture. It's composed of two parts: The upper part is used to select instances with low-entropy. It takes the source instances as input and output the low-entropy instances with pseudo label. The under part is a feature adaptation network. It's used to transfer the information from source domain to target domain in feature level.

- **Generator:** $G(\mathbf{x}^{le}, \mathbf{z}; \theta_G) \rightarrow F^{fake}$, parameterized by θ_G , $F^{fake} = \{\mathbf{f}_i^{fake}, y_i^{fake}\}_{i=1}^{N^{fake}}$ is to generate fake features share the distribution with target feature. $\mathbf{z}$ is the random Gaussian noise, $\mathbf{f}_i^{fake}$ is the fake features generated from G , and y_i^{fake} is the same with y_i^{ps} .
- **Discriminator:** $D(\mathbf{x}; \theta_D) \rightarrow (1, 0)$, parameterized by θ_D , is used to discriminate the features of target data and fake data.
- **Classifier:** $C(\mathbf{x}; \theta_C) \rightarrow \hat{y}$, parameterized by θ_C , $\hat{y}$ is the predicted labels. C is the classifier to predict the labels of source data and target data.

We will introduce the detail of each component in the next sub-sections.

A. Selecting Low Entropy Source Instances with Adaptive Threshold

To select the source domain samples that are useful in target domain clustering, we extract the features of source domain instances and use a standard clustering algorithm (e.g. k -means) to obtain labeled data with high confidence. We use autoencoder [15] and pretrained networks Inception-V3 [34] as the feature extractor. The label confidence is represented by the entropy, which can be obtained in the last layer of the feature extractor:

$$H(E(x_i)) = H(p_i) = - \sum_{j=1}^n p_{ij} \log p_{ij}. \quad (1)$$

$H(E(x_i)) = h_i$ where h_i is the entropy of x_i . The number of neurons in the last layer equals to the number of clusters. So p_{ij} is the output of the last layer after normalization and can be considered as the probability of the instance i belongs to the cluster j .

Once we label the source domain data, we propose an adaptive threshold algorithm to select the instances with low entropy.

Fig. 3: The instances selected with different thresholds n_1, n_2, n_3 , ($n_1 < n_2 < n_3$). (1) is the entropy distribution in original feature space of dataset Amazon's first cluster. (2) is the entropy distribution in optimized feature space (with Eq. (5)) of dataset Amazon's first cluster.

Here we introduce a parameter Δ based on the entropy:

$$\Delta_j = \frac{\max_i (H(p_{ij})) - \min_i (H(p_{ij}))}{|p_{ij}|}, \quad (2)$$

The threshold of instance selection for the cluster j is defined as $n \times \Delta_j$ ($n=1,2,3, \dots$). It indicates an entropy interval: $[N_{min}, n\Delta]$, where N_{min} is the minimum number of instances will be selected. In the experiment, we set it as 50% of the instances for every cluster. The instances in this interval will be used in target domain clustering. If n is too small, we cannot have enough instances to transfer source domain knowledge. For example, small n_1 in Fig. 3(2) results in a small area S_1 . The generated fake features or instances would have weak variety as the source instances. If n is too large as n_3 , it might include too much noise to obtain ideal performance. Therefore, we aim at finding an appropriate n , such as n_2 in Fig. 3, for the clusters where the area does not include the instances with a large entropy gap. The threshold selection process is introduced in the following steps: 1) Initialize the adaptive threshold as N_{min} . This is the minimum number of instances we should choose in each cluster, otherwise it will result in serious data imbalance problems. 2) Increase n from 1 to N^s (The final value of n is less than 50 every time in our experiments), once every thresholds add for one Δ and the number of instances in all the scopes (every cluster has its own scope, the union of these scope is marked as S) is less than the number of clusters k , n is confirmed. In our method, every cluster uses the same n , because of the different Δ , the thresholds' scopes are different in the clusters.

Nevertheless, the adaptive threshold algorithm cannot handle all the entropy distributions. As shown in Fig. 3 (1), it is difficult to find a precise cutoff because the distribution of entropy is almost uniform. Fig. 3 (2) is an ideal distribution for adaptive threshold algorithm for which we can see the entropy of the instances out of scope S_2 increase abnormally. Inspired by DEC, we iteratively refine E to strengthen pseudo labels' confidence.

The original feature extractor E is represents as $F^s = E(\mathbf{x}^s; \theta^E)$, $F^s = \{\mathbf{f}_i^s, y_i^{le}\}_{i=0}^{N^{le}}$. F^s represents the features of instances in source dataset and the pseudo labels are assigned by a shallow clustering method like k -means [35]. Then net E

will be optimized to produce features following a distribution which is convenient to identify the instances with low entropy.

Following DEC [7], we choose Student's t -distribution [36] as a kernel to measure the similarity between every embedded point and cluster centroid pair and then normalize the similarity (the centroid of every cluster is initialized with k -means [35] method in our experiments.):

$$q_{ij} = \frac{(1 + \|f_i - \mu_j\|^2)^{-1}}{\sum_{j'} (1 + \|f_i - \mu_{j'}\|^2)^{-1}}. \quad (3)$$

In Eq. (3), q_{ij} presents the probability of assigning sample i to cluster j . At the beginning of the training process, the entropy of the q_{ij} distribution is difficult to find an abnormal increment (This conclusion is shown in the experiment). To differentiate the probability distribution and refine the cluster results into a high confident mode, we set a target distribution to the probability distribution:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_{j'}}, \quad (4)$$

where $f_j = \sum_i q_{ij}$. The loss function of module Low-entropy feature extractor is set as the KL divergence loss between the soft assignments q_{ij} and the target distribution p_{ij} :

$$\mathcal{L}_s = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}. \quad (5)$$

B. PixelDA to FeatureDA

After getting instances with low entropy and pseudo labels, we can leverage them in target domain clustering. We choose PixelDA as the base domain adaptation model. PixelDA is a generative net which can generate images following sampled from source domain and random noise. The generated image has the same label with the image that generates it. But the original target for PixelDA is not only UDA, so some constraints of PixelDA is not necessary for our target: clustering. In clustering issue, what we want to do is identification instead of generation. Hence we optimize PixelDA model to FeatureDA model which can handle classification task with lower complexity especially when the input data of generator has high dimension. The nether part of Fig. 2 describes the architecture of FetureDA: G generates fake features with noise and low-entropy source instances, the fake features have the same label as the source instance used for generating. D is used to judge the fake features and the target features. Theoretically speaking, with different noise vector, an image from source domain can produces different fake features which means the model won't be under fitting. For broader applicability, we adopt the base model of PixelDA, which is optimized by three objective loss: $\mathcal{L}_G$, $\mathcal{L}_D$ and $\mathcal{L}_C$. As shown in Fig. 2, our goal is to minimize the objective Eq. (6):

$$\min_{\theta_G, \theta_C} \max_{\theta_D} \alpha \mathcal{L}_d(D, G) + \beta \mathcal{L}_c(G, C), \quad (6)$$

where $\mathcal{L}_d$ is a routine loss of generative adversarial network, and $\mathcal{L}_c$ is the loss of the classifier and generator, it's shown in

Fig. 4: An example showing the differences between the setting with or without low entropy source data to train the classifier. (1) shows the images from target domain. (2) shows the images generated when the low entropy source data is not used to train the classifier. (3) shows the images generated when the low entropy source data is used to train the classifier.

Eq. (8). The cross entropy loss of the classifier is calculated by both source instances and generated instances.

$$\mathcal{L}_d(D, G) = \mathbb{E}_{\mathbf{x}^t} [\log D(\mathbf{x}^t; \theta_D)] + \mathbb{E}_{\mathbf{x}^s, \mathbf{z}} [\log (1 - D(G(\mathbf{x}^s, \mathbf{z}; \theta_G); \theta_D))], \quad (7)$$

$$\mathcal{L}_c(G, C) = \mathbb{E}_{\mathbf{x}^s, \mathbf{y}^s, \mathbf{z}} [-\mathbf{y}^{s\top} \log C(G(\mathbf{x}^s, \mathbf{z}; \theta_G); \theta_C)] - \mathbb{E}_{\mathbf{x}^s, \mathbf{y}^s, \mathbf{z}} [\mathbf{y}^{s\top} \log C(\mathbf{x}^s; \theta_C)]. \quad (8)$$

The influence of whether the low entropy instances should be used to train the classifier is shown in Fig. 4. Train the classifier with both source features and fake features accelerates G to generate instances or features which are more plain to be classified.

Our algorithm is summarized in Algorithm 1. N, M, k are numbers of source domain instances, target domain instances, clusters respectively. Let L_1 denote the maximum number of neurons in feature extractors' hidden layers. The complexity of pretrain and optimize the feature extractor are $O(NL_1^2)$. The complexity to select the low entropy instances is $O(N \log N)$ because the sort algorithm is used here. Let L_2 denote the maximum number of neurons in G, D, C 's hidden layers. The complexity of training FeatureDA is $O(NL_2^2)$. So, the complexity of our model is $O(N \log N)$ to the data size.

IV. EXPERIMENTS

A. Experimental Settings

1) *Datasets*: In this section, we evaluate our method in both quantitative and qualitative ways. We compared our method on following datasets: **MNIST** [37], **MNIST-M** [38], **USPS** [39] which are digital images and **OFFICE-31** [40] which contains office environment images. The details of these datasets is listed in Table I. To evaluate the proposed methods, we construct the following dataset pairs: **MNIST**→**MNIST-M**, **USPS**→**MNIST-M**, **A**→**W**, **D**→**W**, **W**→**D**, **A**→**D**, **D**→**A**, **W**→**A**. For OFFICE-31, we choose 5 classes in **A**, **D** and **W**, i.e., keyboard, laptop computer, mouse, printer and scissors, which have larger number of samples among the 31 classes. Since the number of instances of OFFICE-31 is small, we

Algorithm 1 DC-FUDA

Input:

X^s, X^t, k, N_{min} and N_{epoch}, α, β

Output:

Cluster assignments y_{source} and y_{target}

- 1: Pretrain the original feature extractor E
- 2: Initialize μ_j by k -means
- 3: Predict y_{ps} by E
- 4: **repeat**
- 5: $y_{old} = y_{ps}$
- 6: Update E by Eq. (5)
- 7: Predict y_{ps} by E
- 8: **until** $y_{ps} == y_{old}$
- 9: Calculate h_i with Eq. (1), $\forall i \in \{1, \dots, N_s\}$
- 10: Calculate Δ_j with Eq. (2), $\forall i \in \{1, \dots, k\}$
- 11: $S = N_{min}, n = 0$
- 12: **repeat**
- 13: $S_{old} = S, n = n + 1$
- 14: **for** $j = 1$ to k **do**
- 15: $S = S + \{ \mathbf{x}_i \mid y_i = j, (n-1)\Delta_j < e_i < n\Delta_j \}$
- 16: **end for**
- 17: **until** $S - S_{old} < k$
- 18: $X^{le} = \{ \}$
- 19: **for** $j = 1$ to k **do**
- 20: $X^{le} = X^{le} \cup \{ \mathbf{x}_i \mid y_i = j, e_i < n\Delta_j \}$
- 21: **end for**
- 22: **for** 1 to N_{epoch} **do**
- 23: Update D and G on X^{le} , according to Eq. (7)
- 24: Update C and G on X^{le} , according to Eq. (8)
- 25: **end for**
- 26: Output y_{source} and y_{target} predicted by C

applied data augmentation (i.e., noise addition, random rotation and intensification) such that **A**, **D** and **W** all have 500 samples, each class providing 100 samples.

2) *Implementation Details*: The algorithm is implemented with Python. For the digits datasets, convolutional auto-encoder (CAE) is applied as the feature extractor whose encoder's structure is Input—Conv₃₂⁵—Conv₆₄⁵—Conv₁₂₈³—Fc₁₀. For OFFICE-31 dataset, Inception-V3 with weight ImageNet is used as the feature extractor for all methods. For all the comparing methods, the suggested parameters are adopted. For proposed method, we suggest N_{min} can be set as 50% of the instances for every cluster, α and β are set as 1, N_{epoch} is set as 80.

Dataset		# Instances	Image size	# Classes
MNIST		70000	$28 \times 28 \times 1$	10
MNIST-M		70000	$32 \times 32 \times 3$	10
USPS		5000	$16 \times 16 \times 1$	10
OFFICE-31	A (Amazon)	500	$299 \times 299 \times 3$	5
	D (Dslr)	500	$299 \times 299 \times 3$	5
	W (Webcam)	500	$299 \times 299 \times 3$	5

TABLE I: The details of datasets.

3) *Evaluation Metrics*: Cluster accuracy (ACC), normalized mutual information (NMI), and adjusted rand index (ARI) are the clustering metrics. The reported results are average values of 10 independent runs.

	MNIST $\rightarrow$ MNIST-M			USPS $\rightarrow$ MNIST-M		
	ACC	NMI	ARI	ACC	NMI	ARI
k -means (only target)	22.60	15.74	6.28	22.60	15.74	6.28
k -means (mixture of source and target)	14.15	05.98	00.22	16.72	10.96	00.99
DEC-DA (only target)	22.20	19.53	11.24	22.20	19.53	11.24
DEC-DA (mixture of source and target)	17.43	09.63	00.91	11.84	00.44	00.08
CC (only target)	32.34	26.68	16.50	32.34	26.68	16.50
CC (mixture of source and target)	33.97	15.95	9.70	29.36	14.63	7.92
DC-FUDA (ours)	95.63	89.68	88.12	59.94	41.16	37.93
DAN	74.04	54.03	50.43	54.99	42.93	31.56
MCD	95.00	87.74	89.30	57.57	46.76	32.78
PixelDA	98.27	95.11	96.24	59.40	38.53	32.30

TABLE II: Quantitative comparison results on digital datasets. DAN, MCD, and PixelDA are all UDA methods with labeled source data. For the clustering methods, "only target" means only use the MNIST-M dataset for training and validating so the results are same.

	A $\rightarrow$ D			A $\rightarrow$ W			D $\rightarrow$ A			D $\rightarrow$ W			W $\rightarrow$ A			W $\rightarrow$ D		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
k -means(only target)	82.14	71.16	67.64	72.88	55.96	50.51	75.22	54.59	52.67	72.88	55.96	50.51	75.22	54.59	52.67	82.14	71.16	67.64
k -means(mixture of source and target)	81.92	65.85	63.21	69.64	49.61	46.78	7.18	49.91	57.58	71.96	55.71	51.69	73.98	55.04	51.40	78.38	66.34	63.61
DEC-DA(only target)	84.28	75.20	71.39	76.20	63.16	57.39	77.66	60.84	58.15	76.20	63.16	57.39	77.66	60.84	58.15	84.28	75.20	71.39
DEC-DA(mixture of source and target)	85.86	76.78	71.61	75.54	64.79	59.10	79.16	65.01	60.46	28.52	17.78	10.59	72.52	62.21	52.46	29.32	21.13	12.75
CC (only target)	64.20	50.41	39.77	76.4	60.14	54.37	64.00	50.62	43.70	76.4	60.14	54.37	64.00	50.62	43.70	64.20	50.41	39.77
CC (mixture of source and target)	63.00	54.34	39.51	55.00	37.87	26.02	70.80	63.16	55.58	63.40	51.82	41.43	65.00	57.56	47.94	62.00	52.81	41.93
DC-FUDA (ours)	92.84	83.21	84.53	94.00	83.84	85.39	88.50	73.68	74.70	99.60	98.61	98.99	88.51	72.59	75.05	100.00	100.00	100.00
DAN	85.60	74.59	70.98	93.60	84.08	84.38	83.80	65.41	64.94	99.20	97.56	98.01	84.20	67.86	66.15	99.40	98.32	98.51
MCD	92.80	82.39	82.24	78.20	58.99	54.22	81.40	60.55	60.53	99.20	97.22	98.01	87.60	71.61	72.49	99.80	99.30	96.04
PixelDA	93.00	81.68	83.30	92.80	81.98	83.05	80.80	57.79	59.03	99.20	97.22	98.01	84.80	66.72	67.35	97.00	93.71	99.50

TABLE III: Quantitative comparison results on OFFICE-31 datasets. DAN, MCD, and PixelDA are all UDA methods with labeled source data. Their results are reported for reference.

Fig. 5: The changing of the entropy in different iteration of Amazon dataset. The points with different color represents instances from different clusters. The brief lines during the points represent the thresholds of every class with $n = 18$.

	A $\rightarrow$ D			A $\rightarrow$ W			D $\rightarrow$ A			D $\rightarrow$ W			W $\rightarrow$ A			W $\rightarrow$ D		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
All-DC-FUDA	87.60	75.34	73.84	88.80	76.49	74.80	82.80	62.31	63.69	99.20	97.38	97.99	83.40	61.92	63.90	98.20	94.91	95.56
DC-FUDA (ours)	92.84	83.21	84.53	94.00	83.84	85.39	88.50	73.68	74.70	99.60	98.61	98.99	88.51	72.59	75.05	100.00	100.00	100.00
FeatureDA	90.80	79.96	79.89	94.60	85.22	85.41	87.00	70.13	71.17	99.20	97.56	98.01	87.20	68.80	70.94	99.60	98.61	99.00

TABLE IV: Quantitative comparison results on OFFICE-31 datasets based on FeatureDA and InceptionV3 features.

	A $\rightarrow$ D			A $\rightarrow$ W			D $\rightarrow$ A			D $\rightarrow$ W			W $\rightarrow$ A			W $\rightarrow$ D		
	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI
SPL	94.05	84.78	86.08	94.98	85.74	87.82	88.98	72.37	74.63	98.29	94.68	95.76	87.96	70.50	72.87	97.18	91.78	93.14
SPL with All-DC-FUDA	89.95	78.05	77.53	93.69	83.66	85.04	86.14	67.12	69.00	97.09	91.09	92.84	89.00	72.21	74.92	93.89	87.27	86.08
SPL with DC-FUDA	94.12	85.97	86.95	92.19	81.30	81.79	89.27	72.11	75.51	97.47	92.67	93.70	89.07	72.43	74.98	99.13	97.12	97.90

TABLE V: Quantitative comparison results on OFFICE-31 datasets based on SPL and ResNet50 features.

B. Comparison with State-of-the-art Methods

We compare our method with the classic and state of the arts. For our method is a combination of clustering problem and UDA problem, we choose 4 clustering method and UDA method as comparison methods which have labeled source dataset. All the clustering methods are evaluated on two modal: "only target" and "mixture of source and target" where "only target" means the model is trained on target dataset and

tested on target dataset, "mixture of source and target" means the modal is trained on both target and source dataset and tested on target dataset. The quantitative comparison results for digits datasets and OFFICE-31 are shown in Table II and Table III, respectively. In Table II, It is shown that our method performs the best on MNIST $\rightarrow$ MNIST-M, USPS $\rightarrow$ MNIST-M dataset pairs. To the traditional clustering methods, which rely on the feature extractor too much so the 3-layers autoencoder

limited the performance of these methods. The UDA methods, which transfer the knowledge of MNIST and USPS to MNIST-M can get huge improvement to the performance. In Table III, our method performs better than most of the comparison methods even the UDA methods have labeled source data. We can see that the results in experiments with "mixture of source and target", perform worse than "only target" on most dataset pairs, which shows mix different dataset straightforwardly can't improve the clustering power.

C. Entropy Optimization Experiments

We visualize the process of optimizing the entropy distribution with Eq. (5), as shown in Fig. 5. To show the thresholds in every cluster clearly, we fix $n = 18$ (n is automatically computed after the optimization in our algorithm). At the beginning of the optimization, the entropy of the instances is hard to get clear thresholds. In iteration 0 of Fig. 5, the first and third clusters have no thresholds. At the end of the training, every cluster can be divided with a threshold and most of the instances' entropy in the same clusters is compressed into a narrow range. With the training process, the entropy of the dataset also declines. Fig. 5 shows that the first part of our method "electing Low Entropy Source Instances with Adaptive Threshold" can produce a clear threshold according to the instances' entropy. The results in Ablation Studies show the selected instances are reliable.

Fig. 6: DC-FUDA is our method. "DC-FUDA with fake instances generation" is our instances selection method with PixelDA network.

D. Ablation Studies

In our methods, we mainly have two strategies to enhance the performance, the first one is instance selection and another one is FeatureDA. In this part, we will respectively show the influence of both parts.

Table 1 shows the results of three methods: FeatureDA uses image instances to generate features and performs clustering operation on feature level, our method DC-FUDA and DC-FUDA with all source instances transferred called "All-DA-FUDA". Compared with All-DA-FUDA and FeatureDA, DC-FUDA gets better results especially when the source domain is noisy, which is shown in "A→D" and "A→W" dataset pairs. The dataset "A" has some images wrong labeled in the OFFICE-31 dataset, and our method only

chooses the credible instances from the source domain which relieves the destructiveness of the noisy instances. To illustrate the influence of data selection and exclude the influence of PixelDA, we use SPL [41] model (which also needs labeled data in the source domain) as the domain adaptation network. SPL is a selective pseudo-labeling strategy based on structured prediction. The experimental results are shown in Table V (Notice, for its high complexity we only validate it on OFFICE-31 dataset). In Table V, we can see the promotion with data selection. "SPL with DC-FUDA" performs better than "SPL with All-DC-FUDA" on every dataset setting. "SPL with DC-FUDA" even performs better than SPL which has a labeled source domain. The main reason is that there are several wrong labeled instances in the OFFICE-31 dataset which could cause the negative transfer. In contrast, "SPL with DC-FUDA" only chooses the low entropy instances with reliable pseudo labels.

The difference between FeatureDA and PixelDA is whether G needs to generate instances. Fig. 6 shows that FeatureDA focuses more on the feature generation and can improve the prediction performance in every dataset pairs.

V. CONCLUSIONS

In this paper, we present the first work and propose a novel framework to improve the deep clustering performance by transferring knowledge from a source domain without any labeled data. To select reliable instances in the source domain for transferring, we propose a novel adaptive threshold algorithm to select low entropy instances. To transfer important features of the selected instances, we propose a feature-level domain adaptation network for clustering. Experimental results demonstrate that our method can significantly enhance deep clustering performance, without using any label information from the source domain. Besides, compared to those state-of-the-art methods which need labeled data of the source domain, our method can achieve competitive results.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation of China (No. 61806043), Sichuan Science and Technology Program (Nos. 2021YFS0172 and 2020YFS0119), and Fundamental Research Funds for the Central Universities of China (No. ZYGX2021YGLH022).

REFERENCES

- [1] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *TPAMI*, vol. 35, no. 11, pp. 2765–2781, 2013.
- [2] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2016, pp. 31–35.
- [3] J. Hwang and H. Kim, "Variational deep clustering of wafer map patterns," *IEEE Transactions on Semiconductor Manufacturing*, vol. 33, no. 3, pp. 466–475, 2020.
- [4] F. Pizzati, R. d. Charette, M. Zaccaria, and P. Cerri, "Domain bridge for unpaired image-to-image translation and unsupervised domain adaptation," in *WACV*, 2020, pp. 2990–2998.
- [5] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *ICML*. PMLR, 2017, pp. 2208–2217.

- [6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *NIPS*, vol. 27, pp. 2672–2680, 2014.
- [7] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *ICML*, 2016, pp. 478–487.
- [8] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. 12, 2010.
- [9] C. Song, F. Liu, Y. Huang, L. Wang, and T. Tan, "Auto-encoder based data clustering," in *ICPR*. Springer, 2013, pp. 117–124.
- [10] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *ICPR*. IEEE, 2014, pp. 1532–1537.
- [11] X. Peng, S. Xiao, J. Feng, W.-Y. Yau, and Z. Yi, "Deep subspace clustering with sparsity prior," in *IJCAI*, 2016, pp. 1925–1931.
- [12] Y. Ren, N. Wang, M. Li, and Z. Xu, "Deep density-based image clustering," *Knowledge-Based Systems*, vol. 197, p. 105841, 2020.
- [13] L. Van Der Maaten, "Learning a parametric embedding by preserving local structure," in *JMLR*, 2009, pp. 384–391.
- [14] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *IJCAI*, 2017, pp. 1753–1759.
- [15] X. Guo, E. Zhu, X. Liu, and J. Yin, "Deep embedded clustering with data augmentation," in *ACML*, 2018, pp. 550–565.
- [16] N. Mrabah, M. Bouguessa, and R. Ksantini, "Adversarial deep embedded clustering: on a better trade-off between feature randomness and feature drift," *TKDE*, 2020.
- [17] Y. Ren, S. Huang, P. Zhao, M. Han, and Z. Xu, "Self-paced and auto-weighted multi-view clustering," *Neurocomputing*, vol. 383, pp. 248–256, 2020.
- [18] P. Zhou, L. Du, X. Liu, Y.-D. Shen, M. Fan, and X. Li, "Self-paced clustering ensemble," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 4, pp. 1497–1511, 2020.
- [19] J. Xu, Y. Ren, G. Li, L. Pan, C. Zhu, and Z. Xu, "Deep embedded multi-view clustering with collaborative training," *Information Sciences*, vol. 573, pp. 279–290, 2021.
- [20] J. Xu, Y. Ren, H. Tang, Z. Yang, L. Pan, Y. Yang, and X. Pu, "Self-supervised discriminative feature learning for multi-view clustering," *arXiv preprint arXiv:2103.15069*, 2021.
- [21] Y. Ren, K. Hu, X. Dai, L. Pan, S. C. Hoi, and Z. Xu, "Semi-supervised deep embedded clustering," *Neurocomputing*, vol. 325, pp. 121–130, 2019.
- [22] Z. X. Yong and T. T. Torrent, "Semi-supervised deep embedded clustering with anomaly detection for semantic frame induction," in *Proceedings of The 12th Language Resources and Evaluation Conference*, 2020, pp. 3509–3519.
- [23] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*. PMLR, 2015, pp. 1180–1189.
- [24] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 723–773, 2012.
- [25] H. Yan, Y. Ding, P. Li, Q. Wang, Y. Xu, and W. Zuo, "Mind the class weight bias: Weighted maximum mean discrepancy for unsupervised domain adaptation," in *CVPR*, 2017, pp. 2272–2281.
- [26] J. Kim, J. Kwon Lee, and K. Mu Lee, "Deeply-recursive convolutional network for image super-resolution," in *CVPR*, 2016, pp. 1637–1645.
- [27] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4893–4902.
- [28] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *CVPR*, 2017, pp. 3722–3731.
- [29] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *NIPS*, 2016, pp. 469–477.
- [30] W. Chen and H. Hu, "Generative attention adversarial classification network for unsupervised domain adaptation," *Pattern Recognition*, vol. 107, p. 107440, 2020.
- [31] R. Li, W. Cao, S. Wu, and H.-S. Wong, "Generating target image-label pairs for unsupervised domain adaptation," *IEEE Transactions on Image Processing*, vol. 29, pp. 7997–8011, 2020.
- [32] R. Li, Q. Jiao, W. Cao, H.-S. Wong, and S. Wu, "Model adaptation: Unsupervised domain adaptation without source data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9641–9650.
- [33] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6028–6039.
- [34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *CVPR*, 2016, pp. 2818–2826.
- [35] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *J. R. Stat. Soc. Ser. C-Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.
- [36] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *JMLR*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [37] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [38] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *JMLR*, vol. 17, no. 1, pp. 2096–2030, 2016.
- [39] J. J. Hull, "A database for handwritten text recognition research," *TPAMI*, vol. 16, no. 5, pp. 550–554, 1994.
- [40] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *ECCV*. Springer, 2010, pp. 213–226.
- [41] Q. Wang and T. Breckon, "Unsupervised domain adaptation via structured prediction based selective pseudo-labeling," in *AAAI*, vol. 34, no. 04, 2020, pp. 6243–6250.