A Self-Paced Category-Aware Approach For Unsupervised Adaptation Networks

Wenzhen Huang*†‡, Peipei Yang† and Kaiqi Huang*†‡§
*NLPR, CASIA, Beijing, China †CRIPAC, CASIA, Beijing, China
‡University of Chinese Academy of Sciences, Beijing, China
§CAS Center for Excellence in Brain Science and Intelligence Technology, Beijing, China
{wenzhen.huang, ppyang, kqhuang}@nlpr.ia.ac.cn

***Abstract*—The success of deep neural networks usually relies on a large number of labeled training samples, which unfortunately are not easy to obtain in practice. Unsupervised domain adaptation focuses on the problem where there is no labeled data in the target domain. In this paper, we propose a novel deep unsupervised domain adaptation method that learns transferable features. Different from most existing methods, it attempts to learn a better domain-invariant feature representation by performing a category-wise adaptation to match the conditional distributions of samples with respect to each category. A self-paced learning strategy is used to bring the awareness of label information gradually, which makes the category-wise adaptation feasible even if the labels are unavailable in target domain. Then, we give detailed theoretical analysis to explain how the better performance is obtained. The experimental results show that our method outperforms the current state of the arts on standard domain adaptation datasets.**

I. Introduction

In recent years, deep learning has shown its great power in solving computer vision problems and becomes a standard tool in computer vision [1]. However, its success seriously relies on a large number of labeled training samples, which may be unavailable in practice.

Domain adaptation is proposed to address this problem. It utilizes the knowledge discovered from some related domains and thus reduces the number of required training samples. Unsupervised domain adaptation is an important topic which focuses on the scenario where label information is unavailable in the target domain [2]. For instance, we might expect to adapt an object recognition model trained in one scene (the source domain) for some other scenes (the target domain). Even if we get only some unlabeled samples in each new scene, the trained model can still adapt well to the new scenes.

There have been a large amount of unsupervised domain adaptation approaches proposed during the past decade. Most of the researchers devote their efforts to matching the marginal distributions of the two domains [3]–[8]. However, these approaches do not consider the label information which plays an important role in learning the transformation.

To make use of the label information, we propose a category-aware adaptation network for unsupervised domain adaptation which makes the features *with respect to the same class* inseparable across domains. In this way, a better feature transformation is obtained by aligning the conditional distributions of each class. It is notable that there is no labeled data in the target domain, and thus the conditional distributions cannot be aligned directly. To overcome this obstacle, we use the classifier trained on the labeled data in the source domain to predict a pseudo label for each sample in the target domain (both on the learned domain-invariant features), which is then used to learn the features that align the conditional distributions. During the training process, the adaptation neural network and the pseudo labels are updated alternately.

However, such an alternating optimization process is not guaranteed to converge to its optimal solution, which obviously depends on the initial point. One way to deal with this problem is to use the technique of self-paced learning(SPL) [9]. The key idea of SPL is to firstly use the easy examples and then gradually include the difficult examples, which helps to avoid bad local minimum. With such the self-paced approach, the process converges to a good solution.

Then we give a detail theoretical analysis to explain the benefits of our method. From the theoretical perspective, one key issue of unsupervised domain adaptation problem is approximately minimizing the target domain generalized error of a classifier through evaluating and minimizing its upper bound. Thus, the domain adaptation method will benefit from tightening the corresponding upper bound. In this paper, we show that the upper bound given by our method is tighter than the bound proposed by Ben-David *et al*. [10], which further demonstrates the effectiveness of our method.

We evaluate our method on several popular adaptation datastes. The experimental results show that by matching the conditional distributions, the classification accuracies are effectively increased.

II. Related Work

A. Unsupervised Domain Adaptation

There have been a large amount of unsupervised domain adaptation approaches proposed during the past decade.

Transfer Component Analysis (TCA) [4] and Geodesic Flow Kernel (GFK) [5] are two typical methods for shallow models. TCA uses Maximum Mean Discrepancy(MMD) as the evaluation criterion of the similarity between two distributions and learns the transfer components in a Reproducing Kernel Hilbert Space. GFK derives an infinite number of intermediate subspaces to bridge the source and target domains by constructing the geodesic flows.

Since the shallow models' capability is limited by their poor feature representation abilities, many researchers devote their efforts to combine the adaptation strategies above with deep models. Deep Domain Confusion (DDC) [6] and Deep Adaptation Network (DAN) [7] learn the transferable features by matching distributions of single or multiple task-specific layers using MMD. Reverse Gradient (RevGrad) [8] aligns the distributions by making them indistinguishable for a discriminative domain classifier. Joint Adaptation Network (JAN) [11] learns transferable features by matching the joint distributions of features and labels using MMD.

The central idea of the above methods is to align the marginal or joint distributions of both source and target domains. However, the method of aligning the marginal distributions cannot make use of the label information in source domain. While matching the joint distributions cannot be effectively met when the label distributions of two domains are different. In this paper, we design a deep model to learn domain-invariant features by matching the conditional distributions.

B. Self-Paced Learning

Inspired by the learning process of human being, Kumar *et al*. [9] proposes self-paced learning (SPL) model , the key idea of which is to firstly learn the easy examples and then gradually include the difficult examples. By virtue of generality of SPL, it has been successfully applied to a series of problems, such as category discovery [12], long term tracking [13] and matrix factorization [14].

In this work, we use SPL strategy to alleviate the bad local optimum problem of the alternating optimization process.

III. Category-Aware Adaptation Networks

A. Problem Definition and Notations

Consider a classification problem with input space $\mathcal{X}$ and label space $\mathcal{Y}$. Assuming that there are C categories, the label space should be the set of them $\mathcal{Y} = \{1, \cdots, C\}$. In the unsupervised domain adaptation problem, we are given an unlabeled sample set in the target domain $\mathcal{D}_t = \{(\mathbf{x}_j^t)\}_{j=1}^{n_t}$, a labeled sample set in the source domain $\mathcal{D}_s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{n_s}$. The distributions of them are different. The task is to predict the labels of the samples in the target domains.

To make a simpler description, we define the union set including all samples $\{\mathbf{x}_1^s, \cdots, \mathbf{x}_{n_s}^s, \mathbf{x}_1^t, \cdots, \mathbf{x}_{n_t}^t\}$ as $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_k, \cdots, \mathbf{x}_n\}$, where $n = n_s + n_t$. A binary variable d_k is used to indicate whether the k-th sample $\mathbf{x}_k \in \mathcal{D}$ is from the source domain ($d_k = 0$) or target domain ($d_k = 1$).

B. A Self-Paced Alternating Algorithm For Category Awareness Adaptation

It is intuitive that matching the conditional distributions of source and target domains is an efficient method for domain adaptation. However, this method requires the labels of the samples in target domain, Y_t, which are unavailable in unsupervised domain adaptation problems.

To solve this problem, we use the label predictor trained on $\mathcal{D}_s$ to estimate a pseudo label $\hat{y}_i^t$ for each target sample $\mathbf{x}_i^t$. Denoting the feature extractor as $\phi(\cdot)$ which maps the input space to the feature space, the optimization problem to match the conditional distributions of two domains can be written as

$$\min_{\phi} \sum_{c=1..C} L_d^{(c)}(\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_n), w_1^c, \cdots, w_n^c) \qquad (1)$$

where $L_d^{(c)}$ evaluates whether the marginal distributions of source and target samples re-weighted by $\{w_1^c, \cdots, w_n^c\}$ are similar. w_i^c is set to $\mathbb{1}(y_i^s = c)$ if $\mathbf{x}_i$ is the source sample and $\mathbb{1}(\hat{y}_i^t = c)$ otherwise. $\mathbb{1}(y = c)$ is set to 1 if y is equal to c and 0 otherwise.

Due to the discrepancy between the two domains, there are some pseudo labels incorrectly labeled. Nevertheless, as long as the majority of the pseudo labels are correct, it will help to learn a better feature. Furthermore, the better feature makes the pseudo labels more accurate. The estimation of pseudo labels and the learning of features are performed alternately and their accuracies are improved mutually.

However, it is obvious that such an approach is not guaranteed to converge to a good solution and the initial steps are critical to the final results. To solve this problem, we implement the strategy of SPL [9] which can avoid the iteration procedure trapping in a bad minimum. The core idea of SPL is training the model with "easy" samples firstly, and incrementally involving more "difficult" samples into learning.

In previous SPL methods, the difficulty of a sample depends on the value of the loss function on it. However, in our model, L_d is determined by the whole of all samples in two domains instead of any individual sample, and thus those methods cannot be used directly. A suitable criterion of judgment is whether the sample is easy to be determined by the classifier trained on the source samples. Take softmax regression for example, and denote the probability of a sample $\mathbf{x}_i$ belongs to c-th class as p_i^c. If the sample $\mathbf{x}_i$ is classified to be of the c-th class and p_i^c is the large, then the samples $\mathbf{x}_i$ is easy for the problem of matching the distributions of source and target samples in c-th class. As a result, the weight w_i^c for $\mathbf{x}_i$ should be larger. If a sample $\mathbf{x}_i$ is determined to be not of the c-th class, then the less p_i^c is, the easier samples $\mathbf{x}_i$ is, and the less weight w_i^c should be.

Therefore, a simple SPL strategy is to use p_i^c as the weight w_i^c for the i-th sample with label c. The new optimization problem to match the conditional distributions of two domains is formalized as

$$\min_{\phi} \sum_{c=1..C} L_d^{(c)}(\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_n), p_1^c, \cdots, p_n^c)$$

where p_i^c is the sample weight of $\mathbf{x}_i$ in $L_d^{(c)}$. It is not only coincide with the intuition, but also has deep theoretical support (shown in Section III-D).

C. Network Architecture For Category-Wise Adaptation

Although there are a variety of choices for the loss function evaluating the difference between distributions L_d, we choose

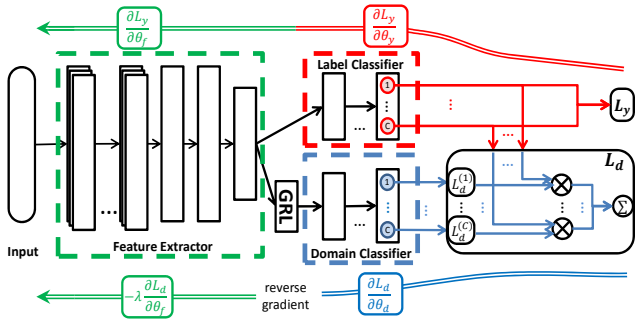


Fig. 1: The figure shows the feed-forward network architecture of our method.

the domain classifier loss [8] in this paper because it shows a good performance and is easy to implement. The architecture of our category-wise adaptation network is composed of three parts as shown in Figure 1.

The first part is a multi-layer feed-forward neural network which maps the input image $\mathbf{x}$ to a D-dimensional feature vector $\mathbf{f} \in \mathbb{R}^D$. We denote it as $G_f(\mathbf{x}; \theta_f)$ where θ_f is the set of network parameters.

The second part $G_y(\mathbf{f}; \theta_y)$ is a softmax regression which maps the feature to the label distribution. It learns the discriminative information between different classes during the training procedure. The objective of this part is to minimize the prediction loss of $G_y(G_f(\mathbf{x}))$ on the labeled samples :

$$L_y^i(\theta_f, \theta_y) = L_y(G_y(G_f(\mathbf{x}_i^s, \theta_f), \theta_y); y_i^s). \tag{2}$$

The third part is used to learn the domain-invariant features. In *RevGrad* [8], the feature is learned by making the samples in two domains indistinguishable for a discriminative domain classifier. Our purpose is to make the learned feature to align samples of the same category. To achieve the purpose, this part is composed of C domain classifiers where c-th one $G_d^{(c)}(\mathbf{f}; \theta_d)$ is used to judge whether a sample $\mathbf{x}_i$ is from source domain or target domain, and the loss outputted by the classifier is re-weighted by $p_i^c = G_y^{(c)}(G_f(\mathbf{x}_i, \theta_f), \theta_y)$ as mentioned in Section III-B. The loss function of domain classifiers for the sample $\mathbf{x}_i$ can be written as

$$L_d^i(\theta_f, \theta_d) = \sum_{c=1..C} p_i^c L_d^{(c)}(G_d^{(c)}(G_f(\mathbf{x}_i, \theta_f), \theta_d); d_i). \tag{3}$$

During the training process, each domain classifier captures the discrepancy between the features belonging to a specific class from two domains. Then the feature extractor learns to eliminate the discrepancy by maximizing the domain classfier losses.

We define the total loss function as

$$E(\theta_f, \theta_y, \theta_d) = \sum_{\substack{i=1..n \\ d_i=0}} L_y^i(\theta_f, \theta_y) - \mu \sum_{i=1..n} L_d^i(\theta_f, \theta_d). \tag{4}$$

We obtain θ_f and θ_y through minimizing $E(\theta_f, \theta_y, \theta_d)$, and obtain θ_d through maximizing $E(\theta_f, \theta_y, \theta_d)$.

To minimize and maximize $E(\theta_f, \theta_y, \theta_d)$ simultaneously, the gradient reversal layer (GRL), proposed by Ganin and Lempitsky [8] is introduced. We add GRL between the first part (the feature extractor) and the third part (the domain classifiers). With the help of it, we only need to minimize $(L_y^i + \mu L_d^i)$ at the end of the network.

In the forward propagation process of our network, the samples' weight will be estimated by label classifier. And in the backward propagation process, the feature extractor will be updated to match the conditional distribution of two domains.

D. Theoretical Analysis

To analyze our method, we first propose a new upper bound on the target domain generalization error of a classifier, which is tighter than the bound proposed by Ben-David *et al.* [10]. Then we use it to explain the merits of our method.

1) Triangle Inequality: A hypothesis is defined as a function $h : \mathcal{X} \to (p_1, \ldots, p_c, \ldots, p_C)^T$, where p_c means the probability of the input sample x belong to c-th class estimated by h. And the labeling functions g_s and g_t, which project the sample to its truth label, also can be seen as hypotheses. Then we denote the label distribute for input $\mathbf{x}$ predicted by the hypothesis h as $\mathbf{h}(\mathbf{x}) = (h^1(\mathbf{x}), \ldots, h^c(\mathbf{x}), \ldots, h^C(\mathbf{x}))^T$.

The probability according to the distribution P_S that a hypothesis h_0 disagrees with other hypothesis h_1 is defined as

$$\epsilon_S(h_0, h_1) = \int [1 - h_0^T(\mathbf{x}) h_1(\mathbf{x})] \, \mathrm{d}P_S(\mathbf{x}). \tag{5}$$

Especially, replacing h_0 and h_1 in (5) with the hypothesis h and labeling functions g_s, we can get the source risk of h, $\epsilon_S(h, g_s)$. And $\epsilon_S(h, g_s)$ is denoted as $\epsilon_S(h)$ for short. We use similar notation $\epsilon_T(h_0, h_1)$, $\epsilon_T(h, g_t)$ and $\epsilon_T(h)$ for the target domain.

Before obtaining the upper bound of $\epsilon_T(h)$, we prove that the following triangle inequality for classification error is always true for any domain,

$$\epsilon(h_0, h_1) \le \epsilon(h_0, h_2) + \epsilon(h_1, h_2). \tag{6}$$

Proof. Assume the largest item in the vector $\mathbf{h}_0(\mathbf{x}) + \mathbf{h}_1(\mathbf{x})$ is $h_0^m(\mathbf{x}) + h_1^m(\mathbf{x})$. For any input $\mathbf{x} \in \mathcal{X}$,

$$\begin{aligned} 1 - \mathbf{h}_0^T(\mathbf{x})\mathbf{h}_1(\mathbf{x}) &= 1 - \sum_{l=1}^{L} h_0^l(\mathbf{x}) h_1^l(\mathbf{x}) \\ \le 1 - h_0^m(\mathbf{x}) h_1^m(\mathbf{x}) &\le 2 - [h_0^m(\mathbf{x}) + h_1^m(\mathbf{x})] \\ = 2 - \sum_{c=1}^{C} &[h_0^m(\mathbf{x}) + h_1^m(\mathbf{x})] h_2^c(\mathbf{x}) \\ \le 2 - \mathbf{h}_0^T(\mathbf{x})\mathbf{h}_2(\mathbf{x}) &- \mathbf{h}_1^T(\mathbf{x})\mathbf{h}_2(\mathbf{x}) \end{aligned}.$$

Supporting from the aforementioned inequality, we can get

$$\begin{aligned} \epsilon(h_0, h_1) &= \int [1 - \mathbf{h}_0^T(\mathbf{x})\mathbf{h}_1(\mathbf{x})] \, \mathrm{d}P(\mathbf{x}) \\ &\le \int [2 - \mathbf{h}_0^T(\mathbf{x})\mathbf{h}_2(\mathbf{x}) - \mathbf{h}_1^T(\mathbf{x})\mathbf{h}_2(\mathbf{x})] \, \mathrm{d}P(\mathbf{x}) \\ &= \epsilon(h_0, h_2) + \epsilon(h_1, h_2). \end{aligned}$$

Thus the triangle inequality (6) holds. □

Though this triangle inequality (6) is similar as the one given by Ben-David *et al*. [10] in the form, the former one can be applied to the case of multi-classification while the latter is applicable only to binary classification problem. Therefore, this triangle inequality and the later proofs based on it would fit the real classification problem better.

2) Error Upper Bound: We define the ideal hypothesis as the hypothesis which minimizes the combined error

$$h^* = \arg\min_{h\in\mathcal{H}} \epsilon_S(h) + \epsilon_T(h)$$

and denote the combined error of the ideal hypothesis by $\lambda = \epsilon_S(h^*) + \epsilon_T(h^*)$.

Now we can give a tighter upper bound on target error.

Theorem 1. *Let $\mathcal{H}$ and $\mathcal{H}'$ be two hypothesis space, and $h^* \in \mathcal{H}'$ is the ideal hypothesis. The target error of $h_0 \in \mathcal{H}$ satisfying the following inequality,*

$$\epsilon_T(h_0) \le \epsilon_S(h_0) + \lambda' + \sup_{h_1\in\mathcal{H}'} (\epsilon_T(h_0,h_1) - \epsilon_S(h_0,h_1)) \tag{7}$$

where $\lambda' = \min_{h\in\mathcal{H}'} \epsilon_S(h) + \epsilon_T(h)$.

Proof. Using the triangle inequality (6), we have

$$\begin{aligned}\epsilon_T(h_0) &\le \epsilon_T(h^*) + \epsilon_T(h_0,h^*)\\ &= \epsilon_T(h^*) + \epsilon_S(h_0,h^*) + \epsilon_T(h_0,h^*) - \epsilon_S(h_0,h^*)\\ &\le \epsilon_S(h_0) + \lambda' + \epsilon_T(h_0,h^*) - \epsilon_S(h_0,h^*)\\ &\le \epsilon_S(h_0) + \lambda' + \sup_{h_1\in\mathcal{H}'} (\epsilon_T(h_0,h_1) - \epsilon_S(h_0,h_1)).\end{aligned}$$

□

The error bound given by Ben-David *et al*. [10] is as following:

$$\epsilon_T(h_0) \le \epsilon_S(h_0) + \lambda + \sup_{h_1,h_2\in\mathcal{H}} |\epsilon_T(h_2,h_1) - \epsilon_S(h_2,h_1)|, \tag{8}$$

where $\lambda = \arg\min_{h\in\mathcal{H}} \epsilon_S(h) + \epsilon_T(h)$, $h_2 \in \mathcal{H}$ is a hypothesis, and other symbol definitions are the same as Theorem 1. Comparing with the error bound (8), our error bound benefits from several advantages as follows:

Firstly, in Theorem 1 the hypothesis space $\mathcal{H}$ of the candidate hypothesis h_0 and the hypothesis space $\mathcal{H}'$ of the ideal hypothesis h^* can be different. Practically speaking, the ideal hypothesis h^* could possibly be outside the selected hypothesis space $\mathcal{H}$, thus our theorem is consistent with the real situation better.

Secondly, our error bound is tighter than their error bound when $\mathcal{H}$ and $\mathcal{H}'$ are identical,

$$\begin{aligned}&\sup_{h_1\in\mathcal{H}} (\epsilon_T(h_0,h_1) - \epsilon_S(h_0,h_1))\\ \le &\sup_{h_1,h_2\in\mathcal{H}} (\epsilon_T(h_2,h_1) - \epsilon_S(h_2,h_1))\\ \le &\sup_{h_1,h_2\in\mathcal{H}} |\epsilon_T(h_2,h_1) - \epsilon_S(h_2,h_1)|.\end{aligned}$$

In the problem of unsupervised domain adaptation, one key issue is minimizing the target domain generalized error of a classifier, $\epsilon_T(h_0)$, but it can't be evaluated or minimized directly. A reasonable approach to approximately minimize $\epsilon_T(h_0)$ is minimizing the upper bound of it. The upper bound is tighter, and the bound is closer to the truth value of $\epsilon_T(h_0)$. Then during the process of minimizing the tighter upper bound, the truth value are more likely to decrease obviously.

3) The Relationship Between Our Method and This Theory: The tighter error bound of the hypothesis h_0 in (7) can be attained by seeking a hypothesis $h_1 \in \mathcal{H}$ to maximize $(\epsilon_T(h_0,h_1) - \epsilon_S(h_0,h_1))$. Let's keep a close watch on this term:

$$\begin{aligned}&\epsilon_T(h_0,h_1) - \epsilon_S(h_0,h_1)\\ =&\frac{1}{n_t}\sum_{i=1}^{n_t}[1 - h_0^T(\mathbf{x}_i)h_1(\mathbf{x}_i)] - \frac{1}{n_s}\sum_{j=1}^{n_s}[1 - h_0^T(\mathbf{x}_j)h_1(\mathbf{x}_j)]\\ =&\sum_{c=1}^{C}\Big\{\frac{1}{n_t}\sum_{i=1}^{n_t} h_0^c(\mathbf{x}_i)[1 - h_1^c(\mathbf{x}_i)] + \frac{1}{n_s}\sum_{j=1}^{n_s} h_0^c(\mathbf{x}_j)[h_1^c(\mathbf{x}_j)]\Big\}\\ &- 1\end{aligned} \tag{9}$$

From the formula (9), we can get several important conclusions as follows: Assuming the outputs of the hypothesis h_0 only are 0 or 1 to facilitate understanding, then maximizing $[\epsilon_T(h_0,h_1) - \epsilon_S(h_0,h_1)]$ is equal to finding a hypothesis h_1 which classifies the source samples the same as h_0, but classifies the target samples different from h_0. In the more general case that the outputs of h_0 are the probabilities of sample belonging to each class, the condition $h1$ should satisfy is nearly identical except that the samples have been reweighted by the outputs of h_0.

Directly finding a multi-class classifier h_1 to satisfy the condition proposed above is hard, so we replace h_1 by C binary-class classifiers. The c-th classifier determines whether a sample belongs to the c-th class. In this case, maximizing (9) is equal to minimizing the domain loss function (3). Leaving the error of the ideal hypothesis λ out of account, minimizing the total loss function (4) can be regarded as minimizing the tighter error bound (7).

IV. Experiments

A. Datasets

Office-31 dataset [15] is a standard benchmark for domain adaptation, comprising three real world object domains: Amazon (images downloaded from amazon.com), Webcam (low-resolution images captured by a web camera) and DSLR (high-resolution images captured by a digital SLR camera). Figure 2 shows sample images from these data sets. By selecting two different domains as the source domain and target domain respectively, we construct $3 \times 2 = 6$ cross-domain object recognition datasets: $A \to W$, $A \to D$, $W \to A$, $W \to D$, $D \to A$, $D \to W$.

MNIST, MNIST-M and SVHN are three databases used for digit classification. The MNIST database [16] consist of handwritten digits which are sampled from a larger set,

Amazon Webcam DSLR

Fig. 2: The samples from Amazon, Webcam and DSLR

NIST. The MNIST-M database [8] is a synthetic database in which the samples are otained by blending digits from MNIST over the patches randomly extracted from color photos from BSDS500 [17]. Street-View House Numbers (SVHN) database [18] is obtained from house numbers in Google Street View images. Figure 3 shows sample images from these data sets and clearly highlights the differences between them. For these datasets, we construct three cross-domain classification datasets: MNIST→MNIST-M, MNIST-M→MNIST, SVHN→MNIST.

MNIST MNIST-M SVHN

Fig. 3: The samples from MNIST, MNIST-M and SVHN

B. Setup

Baselines. We compare our approach with six state-of-the-art transfer learning and deep learning methods: TCA [4], GFK [5], AlexNet [19], DDC [6], DAN [7], RevGrad [8] and JAN [11]. We have introduced these methods in the Section II. The output of fc7 in Convolutional Neural Network(CNN) is taken as the original features to domain adaptation for the shallow models, including TCA and GFK. The method of "AlexNet" means fine-tuning AlexNet only using source-domain data. For other methods, we use the classification accuracy which is published by the original papers. For several absent results of RevGrad, we obtained them with the help of the source code released by the authors.

Network Architectures. We implement our deep model based on the Caffe framework. For Office dataset we fine-tune from CNN model of AlexNet [19] which is pre-trained on the ImageNet dataset. As mentioned previously, the conv1-fc7 layers are considered as feature extractor and fc8 is seen as the label classifier. We use the three fully connected layers ($\mathbf{x} \rightarrow 1024 \rightarrow 1024 \rightarrow 31$) for domain classifier connected to the fc7 layer. And for the digit recognition tasks, we use the same architecture of Ganin and Lempitsky [8], expect that their domain loss is replaced by ours. For loss functions, we set L_y and $L_d^{(l)}$ to be the logistic regression loss and the binomial cross-entropy respectively.

The model is trained on 128-sized batches. A half of each batch is the samples from source domain, the rest is the ones of target domain. Images are preprocessed by the mean subtraction. We use the adaptation factor μ instead of fixing to suppress noisy signal from the domain classifier at the early stages of the training procedure. μ changes from 0 to μ_0 gradually by using the following strategy, $\mu = \mu_0 \left[\frac{2}{1+exp(-\gamma * p)} - 1\right]$, where p is the iteration numbers, μ_0 and γ were set to 0.1 and 10 respectively in all experiments.

We set stochastic gradient descent (SGD) with 0.9 momentum and the learning rate annealing strategy implemented in Caffe, using the following formula, $\mathrm{lr} = \mathrm{lr}_0 * (1 + \gamma' * p)^{(-\beta)}$. In the object recognition tasks, the inital learning rate $\mathrm{lr}_0 = 0.001$, $\gamma' = 0.0001$ and $\beta = 0.75$. While in the digit recognition tasks, $\mathrm{lr}_0 = 0.01$, $\gamma' = 0.00001$ and $\beta = 0.75$.

Evaluation Protocols. For these two datasets, we follow the evaluation protocols for unsupervised domain adaptation [8].

C. Results and Discussion

TABLE I: Accuracy evaluation of different domain adaptation approaches on Office-31 dataset

Method	Source	A	D	W	A	D	W
	Target	W	W	D	D	A	A
TCA [4]		0.610	0.932	0.952	0.608	0.516	0.509
GFK [5]		0.604	0.956	0.950	0.606	0.524	0.481
AlexNet [19]		0.642	0.961	0.978	0.668	0.516	0.498
DDC [6]		0.618	0.950	0.985	0.644	0.521	0.522
DAN [7]		0.685	0.960	0.990	0.670	0.540	0.531
RevGrad [8]		0.730	0.964	0.992	0.729	0.550	0.526
JAN [11]		0.749	0.966	0.995	0.718	0.583	0.551
Ours Method		**0.765**	**0.972**	**0.996**	**0.759**	**0.589**	**0.580**

Accuracy Evaluation on Office Dataset. The classification accuracy results on the Office-31 dataset are shown in Table I. Several points can be concluded from the experimental results. Firstly, the performances of JAN and our method are significantly better than the other ones. Making the best of the pseudo labels of target samples is the main common point of JAN and our method which distinguishes from other deep adaptation methods obviously. This point illustrates that the pseudo labels, despite being imprecise, are especially valuable to unsupervised domain adaptation. Secondly, our method is more effective than JAN. This may be attributed to the reasons as follows: (1) The method based on $\mathcal{A}$-distance is more compatible to the theoretical error bound of target domain deduced in Section III-D; (2) The SPL strategy prevents our algorithm falling into a bad local minima successfully.

Overall, our method outperforms all comparison methods on all transfer tasks of the dataset. These results prove the validity of the approach of matching conditional distributions estimated by pseudo labels and the directive importance of the tighter error bound.

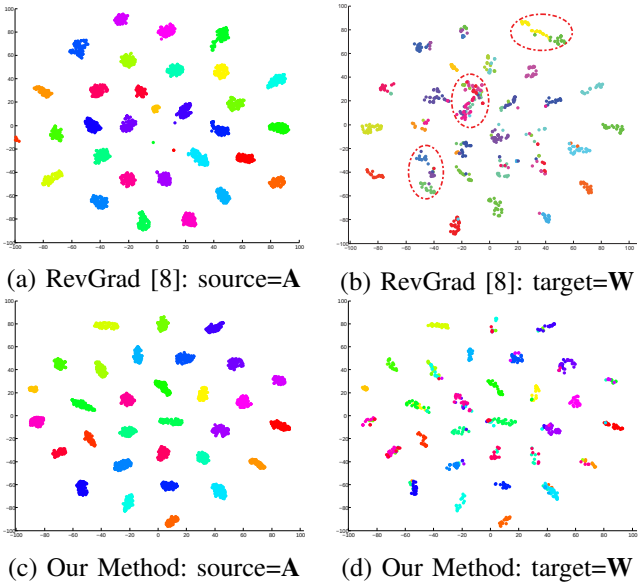
Accuracy for Digit Image Classifications. The classification accuracy results on digit image datasets are shown in Table II. We compare the classification accuracy of our network against RevGrad and the model without domain adaptatin. Our method still outperforms other methods whether the tasks are hard or easy. The more significant improvement might benefit from the large numbers of training samples in source and target

TABLE II: Classification accuracies of digit image classifications for different source and target domains.

METHOD	Source / Target	MNIST / MNIST-M	MNIST-M / MNIST	SVHN / MNIST
Source Only		0.523	0.972	0.549
RevGrad		0.767	0.975	0.739
Ours Method		**0.860**	**0.987**	**0.828**

domains which can help us estimate and match the conditional distributions more precisely.

D. Empirical Analysis



(a) RevGrad [8]: source=**A** (b) RevGrad [8]: target=**W**

(c) Our Method: source=**A** (d) Our Method: target=**W**

Fig. 4: The t-SNE visualization of the network activations generated by RevGrad and our method, respectively.

Feature Visualization We use t-SNE projection [20] to visualize the distributions of the feature learned by RevGrad and our method in Figure 4, where each category is represented by a type of color. For these two methods, the distributions of source and target samples are similar and the samples tend to aggregate into clusters. But comparing the results in Figure 4(b) and Figure 4(d), the distances of some clusters in the former is smaller, especially the two clusters closed in the dashed ellipses are almost connected. This phenomenon leads to that the samples in different classes are confused with the RevGrad features and suggests that our method is a more effective approach to unsupervised domain adaptation.

V. Conclusion

In this paper, we have proposed a new approach for unsupervised domain adaptation which matches the conditional distributions in source and target domains. The label information in target domain and the domain-invariant transformation are learned alternately, and their accuracies are enhanced mutually through self-paced learning. We also give a theoretical analysis of our method to show that it is equal to optimizing a tighter error bound. Experimental results on standard domain adaptation benchmarks show that our model is more effective than previous methods.

Acknowledgment

This work is funded by the National Key Research and Development Program of China (2016YFB1001005), the National Natural Science Foundation of China (Grant No. 61673375, Grant No. 61403388 and Grant No. 61473290), and the Projects of Chinese Academy of Science (Grant No. QYZDB-SSW-JSC006, Grant No. 173211KYSB20160008).

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[2] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE signal processing magazine*, vol. 32, no. 3, pp. 53–69, 2015.

[3] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[4] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

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

[6] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.

[7] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 97–105.

[8] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 1180–1189.

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

[10] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[11] M. Long, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," *arXiv preprint arXiv:1605.06636*, 2016.

[12] Y. J. Lee and K. Grauman, "Learning the easy things first: Self-paced visual category discovery," in *Computer Vision and Pattern Recognition, 2011 IEEE Conference on*. IEEE, 2011, pp. 1721–1728.

[13] J. S. Supancic and D. Ramanan, "Self-paced learning for long-term tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2379–2386.

[14] Q. Zhao, D. Meng, L. Jiang, Q. Xie, Z. Xu, and A. G. Hauptmann, "Self-paced learning for matrix factorization." in *AAAI*, 2015, pp. 3196–3202.

[15] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *European conference on computer vision*. Springer, 2010, pp. 213–226.

[16] 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.

[17] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[18] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.