Research Article

Learning Transferable Convolutional Proxy by SMI-Based Matching Technique

Wei Jin^{1,2} and Nan Jia³

¹Beijing Academy of Science and Technology, Beijing 100094, China

²Beijing Institute of New Technology Applications, Beijing 100094, China

³School of Informatics and Cyber Security, People's Public Security University of China, Beijing 100038, China

Correspondence should be addressed to Wei Jin; jinwei@bjast.ac.cn

Received 20 August 2020; Revised 22 September 2020; Accepted 28 September 2020; Published 14 October 2020

Academic Editor: Chaoqun Duan

Copyright © 2020 Wei Jin and Nan Jia. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Domain-transfer learning is a machine learning task to explore a source domain data set to help the learning problem in a target domain. Usually, the source domain has sufficient labeled data, while the target domain does not. In this paper, we propose a novel domain-transfer convolutional model by mapping a target domain data sample to a proxy in the source domain and applying a source domain model to the proxy for the purpose of prediction. In our framework, we firstly represent both source and target domains to feature vectors by two convolutional neural networks and then construct a proxy for each target domain sample in the source domain space. The proxy is supposed to be matched to the corresponding target domain sample convolutional representation vector well. To measure the matching quality, we proposed to maximize their squared-loss mutual information (SMI) between the proxy and target domain samples. We further develop a novel neural SMI estimator based on a parametric density ratio estimation function. Moreover, we also propose to minimize the classification error of both source domain samples and target domain proxies. The classification responses are also smoothened by manifolds of both the source domain and proxy space. By minimizing an objective function of SMI, classification error, and manifold regularization, we learn the convolutional networks of both source and target domains. In this way, the proxy of a target domain sample can be matched to the source domain data and thus benefits from the rich supervision information of the source domain. We design an iterative algorithm to update the parameters alternately and test it over benchmark data sets of abnormal behavior detection in video, Amazon product reviews sentiment analysis, etc.

1. Introduction

1.1. Background. With the rapid development in Internet technologies, more and more people are using the Internet and generating large-scale sets of behavior data [1, 2]. Nowadays, machine learning technologies are widely used for the automatic annotation of the data to extract insights into the purpose of decision-making [3, 4]. To learn the machine learning models for this purpose, we need sufficient data and the corresponding labels so that the model can learn from the data-label pairs to build the mapping from the data to the labels. Moreover, the size of such data-label pairs should be large enough to cover most of the patterns of the data. However, the labeling of the data is sometimes expensive. When the data are generated, it is usually not labeled automatically, and thus, the label is

missing. To fill this gap, human label work is required to provide the label. In many cases, the labeling of data is time-consuming and costly. As a result, we can only label a limited amount of data and train the model with a large amount of unlabeled data with a small amount of labeled data [5, 6]. The lack of labeled data is a bottleneck for the machine learning tasks over big data. To solve this problem, various methods are developed, among which the most popular ones are the semisupervised learning and transfer learning [7, 8].

Remark 1. The research method of this paper is a new method of transfer learning, and the application objects include many different applications. Their close relations are explained as follows.

Transfer learning aims to borrow the sufficiently labeled data from another domain to help the learning of the model in a domain where the data are not sufficient. This problem is defined over a source domain with sufficient labeled data and a target domain with insufficient labeled data, but a large set of unlabeled data. The source domain and target domain have the same data space and label space. However, the data distribution of the two domains is different; thus, it is not suitable to directly merge them to train a model for the target domain. Due to the mismatch of the data distributions of source and target domains, it is important to transfer the knowledge from the source domain to the target domain. Transfer learning algorithms are designed for this purpose. For example, in the problem of video anomaly detection, the problem is to detect the anomalies from the frames of the video, and the anomalies include cycle, skater, truck, car, wheelchair, and baby cart. In the real world, the data of video are usually collected from subway stations, communities, shopping centers, etc., which are treated as the target domain. Moreover, these data are usually not labeled at all, or only a small part is labeled. Meanwhile, we can use a fully labeled data set collected and annotated by the University of California and San Diego (UCSD) [9, 10] as the source domain data to help the training of the target domain model. However, the video of UCSD is usually captured on the campus of the university, which is different from the target domain's locations, such as shopping centers and subway stations. Thus, the data of source and target domains do not match perfectly, and the mismatching of the two domain's video requires transferring the model trained over the source domain to the target domain to adapt the target domain data. Otherwise, the model trained over the source domain cannot fit the target domain.

Another example is the Amazon product review sentiment analysis [11, 12]. This task is to detect the sentiment (positive/negative) from the text content of a review of a product. For different product types, the buyers may have different review styles. Some product types have sufficient labeled data, while other types are in lack of labeled data. Thus, we want to use the well-labeled product reviews as a source domain to help the learning of sentiment detectors for the limited-labeled reviews of other products. Again, the domain transfer is needed to adopt the model from the source domain product to the target domain product.

In the transfer learning models, the deep convolutional neural network (CNN) [13–17] is the most popular base models. It is composed of multiple convolutional layers and max-pooling layers. The convolutional layers use a filter bank to extract the location features by sliding over the input data, such as image or sentence. This model can extract both local and hierarchical features from data and thus can perform automatic engineering effectively. In this paper, we study the problem of transfer learning over a source domain and a target domain. We proposed a novel transfer learning method based on a convolutional neural network (CNN) [13] and use the squared-loss mutual information (SMI) [18] to measure the matching between the source and the target domain.

1.2. Existing Works. In traditional methods of transfer learning, to match the source and target domains, various methods are applied, such as mutual information minimization (MIM) [19], Hilbert Schmidt independence criterion (HSIC) maximization [20], and maximum mean discrepancy (MMD) minimization [21, 22]. Moreover, to train the transfer learning models, the most popular base model is the deep CNN model, which is also adapted in our paper. In this paper, we briefly introduce some state-of-the-arts relevant to our work of CNN-based transfer learning.

Long et al. [21, 23] proposed a deep CNN model for transfer learning. This model is a two-branch network. The first four convolutional layers are shared by both source and target domains, and the last three full-connection layers are domain-specific. However, in the last three layers, the output features are forced to be in the same spaces by the MMD minimization. The MMD minimization measures mismatch in the features of each layer of two domain domains by the squared ℓ_2 distance of the means of two domains.

Zhang et al. [14] proposed a CNN model learning method for transfer learning to present both data features and attributes. The structure of this network is composed of a shared attribute-embedding CNN across domains, a domain-independent CNN, and a domain-dependent CNN for the embedding of the original data of each data sample. The outputs of these three CNN models are concatenated as the overall features of one sample and used to predict the class label. The attribute-embedding CNN is imposed to approximate the transformed attributes accurately, and the domain-independent CNN outputs are supposed to minimize the MMD of two domains.

Wang et al. [19] developed a novel CNN model learning method for both source and target domains. The CNN outputs are imposed to be independent of the domain; i.e., from the CNN model output of a data sample, we should not be able to tell which domain it is from. To measure the independence of the CNN outputs and the domain indicator, the mutual information is applied. The CNN models are learned by minimizing the mutual information between the CNN outputs and the domain indicators to obtain the domain-independent CNN representations.

Geand and Yu [24] designed a learning method of deep CNN model for transfer learning. This method is based on the source-target selective joint fine-tuning. This method is not to use all the data samples of the source domain but only use a subset of the training samples of the source domain. The selected subset is similar to the target domain samples at the low-level characteristics. To this end, the algorithm constructs the descriptors from the responses of the filter bank over the training samples. These descriptors are used to search for the subset of training samples used for the learning problem.

1.3. Our Contributions. In this paper, we propose a novel CNN-based deep learning framework for transfer learning. Our work is motivated by a phenomenon: the mismatch between the source domain and target domain is usually at

the entire data set level but at the data sample level. Moreover, it is also difficult to find the one-to-one matching between the samples of the source and target domains. Thus, we propose to construct a proxy of each target domain in the source domain. This proxy is constructed by a linear combination of the source domain samples. We also argue that the CNN model is a powerful tool to extract the features of both source and target domains and an optimal choice for the proxy construction. Thus, we first extract the convolutional features from both source and target samples and then apply the proxy construction. We hope the constructed proxy of each target domain can match the original target sample convolutional representation as well as possible. In this paper, we propose to use an information theory measure to measure the quality of matching between the proxy and convolutional features of target domain samples. This measure is the squared-loss mutual information (SMI) [25, 26], which measures the Pearson divergence of the joint probability of proxies and convolutional features and the product of the independent profanities of proxies and convolutional features. We further release the calculation of the SMI by firstly designing a parametric density ratio estimator of proxy and convolutional feature of target domain samples and then approximate it as the squared matching error of the true density ratio against the density ratio estimator.

To learn the proxy construction coefficients, we model the learning problem as a minimization problem. In this minimization problem, we minimize the SMI, the classification loss of labeled samples, and the classification response entropy of the unlabeled samples of both domains. Moreover, the manifold regularization and the model complexity are also considered in the minimization problem. The contributions of this paper are listed as follows:

- (1) We build a novel transfer learning schema, which firstly represents the source and target domains by CNN models and then constructs proxies for the target samples from the source domain. The construction is guided by the SMI minimization. In this way, the matching of the two domains is directly applied at the proxy level, while the CNN representations can still keep the domain characteristics. A novel SMI approximation is proposed as the squared matching error between the density ratio function and its parametric estimator.
- (2) We model the problem as a minimization problem with SMI, classification loss, classification entropy, manifold regularization, and model complexity regularization. This minimization is a joint learning framework to optimize the CNN model parameters and proxy construction coefficients simultaneously.
- (3) We develop an iterative algorithm to solve the minimization problem based on the alternating direction method of multipliers (ADMM) [27]. Moreover, the parameter of the density ratio estimator of the SMI is also updated in this algorithm. With this algorithm, the SMI parameter is also

automatically approximated together with the other variables of the objective.

1.4. Paper Organization. This paper is organized as follows: in Section 2, we introduce the proposed transfer learning method based on CNN and proxy learning; in Section 3, we evaluate the proposed method experimentally over some benchmark data sets; in Section 4, we give the conclusion of this paper and some future works.

2. Proposed Method

In this section, we will propose the novel transfer convolutional neural network learning method. Firstly, we will build the objective function for the learning of the source and target domain convolutional network, and then, we will discuss how to minimize this objective function and finally develop an iterative algorithm based on the optimization results.

2.1. Objective Function. Suppose we have a training set of two domains. In the target domain, we have n training samples $\{x_1^t, \dots, x_n^t\}$, where x_i^t is the i -th training sample of the target domain. To represent a target domain sample x^t , we use a deep CNN model, f , to convert it to a d_t -dimensional vector:

$$f = f(x^t; \Theta) \in R^{d_t}. \quad (1)$$

Meanwhile, we have a set of source domain training set $\{x_1^s, \dots, x_m^s\}$, which has m samples and x_j^s is the j -th source domain training sample. Similar to the target domain, we use another deep CNN model, g , to convert a source domain sample, x^s to a d_s -dimensional vector:

$$g = g(x^s; \Phi) \in R^{d_s}. \quad (2)$$

To learn the network parameters Θ and Φ , we discuss the following problems.

2.1.1. Squared-Loss Mutual Information. Due to the mismatch of the source and target domains, we propose to leverage the two domains by constructing a proxy for each target domain sample in the source domain space. To this end, for a target domain sample, x_i^t , we denote its proxy $z_i \in R^{d_s}$, as a d_s -dimensional vector. Moreover, we impose that it can be reconstructed by a learning combination of the convolutional representations of source domain samples:

$$z_i = \sum_{j=1}^m \alpha_{ij} g_j, \quad \text{s.t.} \quad \sum_{j=1}^m \alpha_{ij} = 1, \quad 1 \geq \alpha_{ij} \geq 0, \quad (3)$$

where $g_j = g(x_j^s; \Phi)$ is the convolutional vector of the j -th source domain sample and α_{ij} is the nonnegative weight of the j -th source domain sample for the construction of the i -th proxy.

To encourage the matching of the source and target domains, we argue to measure their quality of matching by the density ration of the proxies and convolutional vectors of

the target domain samples z and f . For this purpose, we firstly measure the density ratio of z and f , denoted as $r(f, z)$. According to the definition of density ratio [18],

$$r(f, z) = \frac{p(f, z)}{p(f)p(z)}, \quad (4)$$

where $p(f, z)$ is the joint probability of f and z , $p(f)$ is the probability of f , and $p(z)$ is the probability of z . Since it is difficult to directly estimate the density ratio, we propose to learn a parametric density ratio estimator function:

$$\hat{r}(f, z; \phi) = h\left(\phi^\top \begin{bmatrix} f \\ z \\ f - z \end{bmatrix}\right), \quad (5)$$

where $p(f, z)$ is the joint probability of f and z , $p(f)$ is the probability of f , and $p(z)$ is the probability of z . Since it is difficult to directly estimate the density ratio, we propose to

learn a parametric density ratio estimator function. $\begin{bmatrix} f \\ z \\ f - z \end{bmatrix}$

is the concatenation of the f , z , and their element-wise difference vector, $\phi \in \mathbb{R}^{3d_s}$ is the parameter vector, and $h(x) = 1/(1 + \exp(-x))$ is the Sigmoid activation function. To learn ϕ , we minimize the SMI between $r(f, z)$ and $\hat{r}(f, z; \phi)$, defined as follows:

$$\begin{aligned} SMI(\phi) &= \int_f \int_z [r(f, z) - \hat{r}(f, z; \phi)]^2 p(f)p(z) df dz \\ &= \int_f \int_z [r(f, z)^2 - 2r(f, z)\hat{r}(f, z; \phi) + \hat{r}(f, z; \phi)^2] p(f)p(z) df dz \\ &= \int_f \int_z r(f, z)^2 p(f)p(z) df dz - 2 \int_f \int_z r(f, z)\hat{r}(f, z; \phi) p(f)p(z) df dz \\ &\quad + \int_f \int_z \hat{r}(f, z; \phi)^2 p(f)p(z) df dz \\ &= \int_f \int_z r(f, z)^2 p(f)p(z) df dz - 2 \int_f \int_z p(f, z)\hat{r}(f, z; \phi) df dz \\ &\quad + \int_f \int_z \hat{r}(f, z; \phi)^2 p(f)p(z) df dz. \end{aligned} \quad (6)$$

To estimate the SMI given the training samples of target domain and their proxies, we collect the set of convolutional vectors and corresponding proxies $\{(f_1, z_1), \dots, (f_n, z_n)\}$. We assume the distributions of f , z , and (f, z) are uniform distributions, which lead the probability functions as

$$\begin{aligned} p(f) &= \frac{1}{n}, \\ p(z) &= \frac{1}{n}, \\ p(f, z) &= \frac{1}{n}. \end{aligned} \quad (7)$$

With probabilities in (7) and the empirical approximation, we rewrite the second and third terms of (6) as follows:

$$\begin{aligned} \int_f \int_z p(f, z)r(f, z; \phi) df dz &= \frac{1}{n} \int_f \int_z r(f, z; \phi) df dz = \frac{1}{n} \sum_{i=1}^n r(f_i, z_i; \phi), \\ \int_f \int_z \hat{r}(f, z; \phi)^2 p(f)p(z) df dz &= \frac{1}{n^2} \sum_{i,j=1}^n \hat{r}(f_i, z_i; \phi)\hat{r}(f_j, z_j; \phi). \end{aligned} \quad (8)$$

By substituting (8) to (6), we have the approximated SMI as

$$\text{SMI}(\phi) = C - \frac{2}{n} \sum_{i=1}^n r(f_i, z_i; \phi) + \frac{1}{n^2} \sum_{i,i'=1}^n r(f_i, z_i; \phi) r(f_{i'}, z_{i'}; \phi), \quad (9)$$

where $C = \int \int r(f, z)^2 p(f) p(z) df dz$ is a constant. To obtain a good quality estimator, we propose to minimize the squared-loss mutual information learned from the parameter of $\hat{r}(f, z; \phi)$:

$$\phi^* = \text{argmin}_{\phi} \text{SMI}(\phi). \quad (10)$$

With this optimal ϕ^* , we have to use the SMI as a measure of the matching quality between f and z . Using this measure as a term of loss for the purpose of learning the convolutional representations f and proxy z , we organize these parameters as matrices as $F = [f_1, \dots, f_n]$ and $Z = [z_1, \dots, z_n]$. With these variables and the SMI as the matching measure, we seek to learn them to maximize the SMI. In other words, we minimize a loss function derived from SMI, defined as $\mathcal{M}(F, Z; \phi^*)$:

$$\begin{aligned} \mathcal{M}(F, Z; \phi^*) &= \frac{2}{n} \sum_{i=1}^n \hat{r}(f_i, z_i; \phi^*) - \frac{1}{n^2} \sum_{i,i_j=1}^n \hat{r}(f_i, z_i; \phi^*) \hat{r}(f_{i_j}, z_{i_j}; \phi^*) \\ &= -\frac{2}{n} \sum_{i=1}^n h(\phi^{*\top} [f_i z_i f_i - z_i]) \\ &\quad + \frac{1}{n^2} \sum_{i,i_j=1}^n h(\phi^{*\top} [f_i z_i f_i - z_i]) h(\phi^{*\top} [f_{i_j} z_{i_j} f_{i_j} - z_{i_j}]). \end{aligned} \quad (11)$$

We rewrite

$$\phi^{*\top} [f_i z_i f_i - z_i] = \phi_1^\top f_i + \phi_2^\top z_i + \phi_3^\top (f_i - z_i) = (\phi_1 + \phi_3)^\top f_i + (\phi_2 - \phi_3)^\top z_i, \quad (12)$$

where $\phi^* = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}$ accordingly.

$$\begin{aligned} \mathcal{M}(F, Z; \phi^*) &= -\frac{2}{n} \sum_{i=1}^n h((\phi_1 + \phi_3)^\top f_i + (\phi_2 - \phi_3)^\top z_i) \\ &\quad + \frac{1}{n^2} \sum_{i,i_j=1}^n h((\phi_1 + \phi_3)^\top f_i + (\phi_2 - \phi_3)^\top z_i) h((\phi_1 + \phi_3)^\top f_{i_j} + (\phi_2 - \phi_3)^\top z_{i_j}). \end{aligned} \quad (13)$$

The squared-loss mutual information regularization problem is modelled as

$$\min_{F, Z} \mathcal{M}(F, Z; \phi^*). \quad (14)$$

By minimizing this objective, we seek to learn the convolutional representation vectors of both source and target domain samples, target domain sample proxies, and other parameters to match the two domains as much as possible. Please note that, in this problem, we are not matching the source and target domain samples directly but seek to match target domain samples and the proxies of target domain samples in the source domain space.

2.1.2. Semisupervised Classification. In both the source and target domains, there are both labeled and unlabeled data samples. To learn from them, we use the semisupervised method. We discuss the semisupervised learning problems in both source and target domains as follows.

Source Domain. To approximate the class label vector, $y^s \in \{1, 0\}^c$, of a source domain sample, x^s , from its convolutional feature vector, g , we design a linear classification function:

$$y^s = W^\top g, \quad (15)$$

where $W \in \mathbb{R}^{d_s \times c}$ is the parameter vector of the classifier and $\hat{y}^s \in \mathbb{R}^c$ is the classification response. To learn the classifier parameter W , we utilize both the labeled and unlabeled

samples of the source domain. We denote the set of the labeled samples of source domain as $\mathcal{L}^s$ and the set of unlabeled as $\mathcal{U}^s$. For the samples in $\mathcal{L}^s$, we minimize their classification errors measured by the squared loss:

$$\min_{W,G} \left\{ O_1(W,G) = \sum_{j: x_j^s \in \mathcal{L}^s} \|y_j^s - W^\top g_j\|_2^2 \right\}, \quad (16)$$

where y_j^s is the class label vector of the x_j and $G = [g_1, \dots, g_m]$. For the samples in $\mathcal{U}^s$, even we do not have the available labels to constrain the classification responses in $\hat{y}^s$, and we still can regularize the learning of the parameter by its entropy. For a sample x_j^s , we hope the uncertainty of its classification responses in $\hat{y}_j^s$ is as small as possible. The uncertainty is measured by its entropy:

$$H(y_j^s) = - \sum_{k=1}^c y_{jk}^s \log(y_{jk}^s) = - \sum_{k=1}^c (w_k^\top g_j) \log(w_k^\top g_j), \quad (17)$$

where $\hat{y}_{jk}^s = w_k^\top g_j$ is the k -th dimension $\hat{y}_j^s$ and w_k is the k -th column of W . We minimize the entropy of the source samples in $\mathcal{U}^s$ as follows:

$$\min_{W,G} \left\{ O_2(W,G) = - \sum_{j: x_j^s \in \mathcal{U}^s} \left(\sum_{k=1}^c (w_k^\top g_j) \log(w_k^\top g_j) \right) \right\}. \quad (18)$$

Target Domain. In the target domain, we also have labeled set denoted as $\mathcal{L}^t$ and unlabeled set denoted as $\mathcal{U}^t$. To use the labeled samples of the target domain, instead of learning a classifier in the target domain space, we design a classifier in the target domain sample proxy space. For a proxy z , we apply a linear classifier function to approximate its label vector:

$$\hat{y}^t = V^\top z, \quad (19)$$

where $V \in R^{d_t \times c}$ is the classifier parameter and $\hat{y}^t \in \mathbb{R}^c$ is the estimated class label response vector. For a labeled sample, $x_i^t \in \mathcal{L}^t$, its class label vector $y_i^t \in \{1, 0\}^c$ is known, and we minimize the classification loss measured by the squared loss between y_i^t and $\hat{y}_i^t$:

$$\min_{V,Z} \left\{ O_3(V,Z) = \sum_{qi: x_i^t \in \mathcal{L}^t} \|y_i^t - V^\top z_i\|_2^2 \right\}. \quad (20)$$

Meanwhile, for the samples in the target domain $\mathcal{U}^t$, we also hope their uncertainty can be minimized, and accordingly, we minimize the entropy of the classification responses:

$$\min_{V,Z} \left\{ O_4(V,Z) = - \sum_{i: x_i^t \in \mathcal{U}^t} \left(\sum_{k=1}^c (v_k^\top z_i) \log(v_k^\top z_i) \right) \right\}. \quad (21)$$

2.1.3. Manifold Regularization. Moreover, we also hope the classification responses of samples in both source and target domains can be smooth over their manifold. To be specific, the classification responses of two neighboring samples should also be similar. To this end, we first construct the nearest neighbor graphs for both the source and target domains and then use the graphs to regularize their classification responses.

Source Domain. We firstly build a convolutional graph from the convolutional representations of the source domain samples $\{g_1, \dots, g_m\}$. For each sample x_j^s , its nearest neighbor set is defined as $\mathcal{N}_j^s$, and we calculate its affinity between itself and each neighbor $x_{j'}^s \in \mathcal{N}_j^s$ based on a Gaussian kernel:

$$A_{jj'}^s = \begin{cases} \frac{\psi(g_j - g_{j'})}{\sum_{j'': x_{j''}^s \in \mathcal{N}_j^s} \psi(g_j - g_{j''})}, & \text{if } x_j^s \in \mathcal{N}_j^s, \\ 0, & \text{otherwise,} \end{cases} \quad (22)$$

where $\psi(x) = \exp(-x^2/\sigma^2)$ is the Gaussian kernel function. Based on the neighborhood affinity, we hope the squared ℓ_2 norm distance between g_j and its neighbors can be minimized so that the neighborhood structure can be kept in the convolutional representation space:

$$\min_G \left\{ O_5(G) = \sum_{j,j'=1}^m A_{jj'}^s \|g_j - g_{j'}\|_2^2 \right\}. \quad (23)$$

Target Domain. In the target domain, we firstly construct the graph from the original convolutional representations of the target samples $\{f_1, \dots, f_n\}$, not the proxies. Given a target sample x_i^t , we find its nearest neighbors and denote the set of nearest neighbors $\mathcal{N}_i^t$. We calculate the similarity between x_i^t and a sample $x_{i'}^t \in \mathcal{N}_i^t$ also by kernel function:

$$A_{ii'}^t = \begin{cases} \frac{\psi(f_i - f_{i'})}{\sum_{i'': x_{i''}^t \in \mathcal{N}_i^t} \psi(f_i - f_{i''})}, & \text{if } x_i^t \in \mathcal{N}_i^t, \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

With this affinity measure, different from the source domain, we use it to regularize the learning of the proxies of the target domain, $\{z_1, \dots, z_n\}$, i.e.,

$$\min_Z \left\{ O_6(Z) = \sum_{i,i'=1}^m A_{ii'}^t \|z_i - z_{i'}\|_2^2 \right\}. \quad (25)$$

2.1.4. Overall Objective Function. Our overall object is the combination of the above objectives:

$$\begin{aligned}
O(F, G, Z, A, W, V, \theta, \Phi) &= M(F, G, Z, A; \phi^*) + C_1 O_1(W, G) + C_2 O_2(W, G) \\
&+ C_3 O_3(V, Z) + C_4 O_4(V, Z) + C_5 O_5(G) + C_6 O_6(Z) + C_7 (|W|_2^2 + |V|_2^2 + |\Theta|_2^2 + |\Phi|_2^2) \\
&= \frac{2}{n} \sum_{i=1}^n (h(\phi_1 + \phi_3)^T f_i + (\phi_2 - \phi_3)^T Z_i) \\
&+ \frac{1}{n^2} \sum_{i,i'=1}^n (h(\phi_1 + \phi_3)^T f_i + (\phi_2 - \phi_3)^T Z_i) h(\phi_1 + \phi_3)^T f_{i'} \\
&+ (\phi_2 - \phi_3)^T + C_1 \sum_{j: x_j^s \in L^s} |y_j^s - W^T - g_j|_2^2 - C_2 \sum_{j: x_j^s \in U^s} \left(\sum_{k=1}^c (W_k^T g_j) \log(W_k^T g_j) \right) \\
&+ C_3 \sum_{i: x_i^t \in L^t} |y_i^t - V^T Z_i|_2^2 - C_4 \sum_{i: x_i^t \in U^t} \left(\sum_{k=1}^c (V_k^T Z_i) \log(V_k^T Z_i) \right) + C_5 \sum_{j,j'=1}^m A_{jj'}^s |g_j - g_{j'}|_2^2 \\
&+ C_6 \sum_{i,i'=1}^m A_{ii'}^t |Z - Z_{i'}|_2^2 + C_7 (|W|_2^2 + |V|_2^2 + |\Theta|_2^2 + |\Phi|_2^2), \\
\text{s.t. } \forall i = 1, \dots, n: f_i &= f(x_i^t; \Theta), z_i = \sum_{j=1}^m \alpha_{ij} g_j, \sum_{j=1}^m \alpha_{ij} = 1, \text{ and } 1 \geq \alpha_{ij} \geq 0, \\
\forall j = 1, \dots, m: g_j &= g(x_j^s; \Phi),
\end{aligned} \tag{26}$$

where $A = \begin{bmatrix} \alpha_{11}, & \dots, & \alpha_{n1} \\ \vdots & \ddots & \vdots \\ \alpha_{1m}, & \dots, & \alpha_{nm} \end{bmatrix} \in \mathbb{R}_+^{m \times n}$ and $C_i, i = 1, \dots, 7$ are

the tradeoff parameters, which weight the loss terms of the objective. The last term is the squared ℓ_2 norms of the mode parameters to prevent the overfitting problem. In this objective, we impose the following condition:

- (1) In the target domain, we hope the SMI between the samples and their proxies constructed from the source domain can be maximized so that the source and target domains can be matched well.
- (2) In both target and source domains, we hope the classification function can approximate the ground

truth of labeled well, while the uncertainty of classification results of unlabeled can be minimized.

- (3) Again, in both source and target domains, we hope the classification results can be similar among the neighborhoods.
- (4) Finally, we hope the overall complexity of the model can be minimized as much as possible; thus, we minimize the squared ℓ_2 norm of the parameters of the model in the last term.

With this objective, we model the learning problem as a minimization problem:

$$\begin{aligned}
\min_{F, G, Z, A, W, V, \Theta, \Phi} & \mathcal{O}(F, G, Z, A, W, V, \Theta, \Phi), \\
\text{s.t. } \forall i = 1, \dots, n: f_i &= f(x_i^t; \Theta), z_i = \sum_{j=1}^m \alpha_{ij} g_j, \sum_{j=1}^m \alpha_{ij} = 1, \text{ and } 1 \geq \alpha_{ij} \geq 0, \\
\forall j = 1, \dots, m: g_j &= g(x_j^s; \Phi).
\end{aligned} \tag{27}$$

By solving this problem, we can learn the source and target CNN models W and V and the classification layer parameters Θ and Φ . Moreover, the proxy parameters of the target samples, A , are also optimized together with the model parameters to match the source and target domains.

2.2. Problem Optimization. To solve the problem at (27), we adopt the ADMM algorithm, and according, we have the following argued Lagrangian function:

$$\begin{aligned}
& \mathcal{L}(F, G, Z, A, W, V, \Theta, \Phi, \pi, \Psi, \Xi, \varsigma) \\
&= \frac{2}{n} \sum_{i=1}^n h((\phi_1 + \phi_3)^T f_i + (\phi_2 + \phi_3)^T z_i) \\
&+ \frac{1}{n^2} \sum_{i,i'=1}^n h((\phi_1 + \phi_3)^T f_i + (\phi_2 + \phi_3)^T z_i) h((\phi_1 + \phi_3)^T f_{i'} + (\phi_2 + \phi_3)^T z_{i'}) \\
&+ C_1 \sum_{j: x_j^s \in L^s} |y_j^s - W^T g_j|_2^2 + C_2 \sum_{j: x_j^t \in U^s} \left(\sum_{k=1}^c (W_k^T g_j) \log(W_k^T g_j) \right) \\
&+ C_3 \sum_{i: x_i^t \in L^T} |y_i^t - V^T z_i|_2^2 \\
&- C_4 \sum_{i: x_i^t \in U^T} \sum_{k=1}^c (V_k^T z_i) \log(V_k^T z_i) + C_5 \sum_{j,j'=1}^m A_{jj'}^s |g_j - g_{j'}|_2^2 \\
&+ C_6 \sum_{i,i'=1}^m A_{ii'}^t |g_z - g_{z'}|_2^2 + C_7 (|W|_2^2 + |V|_2^2 + |\Theta|_2^2 + |\Phi|_2^2) \\
&+ \sum_{i=1}^n \pi_i^T (f_i - f(x_i^t; \Theta)) + \frac{\rho_1}{2} \sum_{i=1}^n |f_i - f(x_i^t; \Theta)|_2^2 + \sum_{j=1}^m \omega_j^T (g_j - g(x_j^s; \Phi)) \\
&+ \frac{\rho_2}{2} \sum_{j=1}^n |g_j - g(x_j^s; \Phi)|_2^2 + \sum_{i=1}^n \tau_i^T \left(z_i - \sum_{j=1}^m \alpha_{ij} g_j \right) \\
&+ \frac{\rho_3}{2} \sum_{i=1}^n \left| z_i - \sum_{j=1}^m \alpha_{ij} g_j \right|_2^2 + \sum_{i=1}^n \varsigma_i^T \left(\sum_{j=1}^m \alpha_{ij} - 1 \right) \\
&+ \frac{\rho_4}{2} \sum_{i=1}^n \left| \sum_{j=1}^m \alpha_{ij} - 1 \right|_2^2 \\
&s.t. \forall i = 1, \dots, n, j = 1, \dots, m: 1 \geq \alpha_{ij} \geq 0,
\end{aligned} \tag{28}$$

where the following holds:

- (i) π_i is the dual variable of constraint $f_i = f(x_i^t; \Theta)$, ω_j is the dual variable of constraint $g_j = g(x_j^s; \Phi)$, τ_i is the dual variable of constraint $z_i = \sum_{j=1}^m \alpha_{ij} g_j$, and ς_i is the dual variable of constraint $\sum_{j=1}^m \alpha_{ij} = 1$.
- (ii) $\Pi = [\pi_1, \dots, \pi_n]$, $\Psi = [\omega_1, \dots, \omega_m]$, $\Xi = [\tau_1, \dots, \tau_n]$, and $\varsigma = [\varsigma_1, \dots, \varsigma_n]^T$.

- (iii) $\rho_k, k = 1, \dots, 4$ are the corresponding penalty parameters of the constraints.

According to the ADMM algorithm, we optimize the parameters and the dual variables iteratively as follows.

Optimizing F . To optimize the target convolutional vectors of F , we update f_i one by one. The subgradient of $\mathcal{L}$ with regard to f_i is

$$\begin{aligned} \nabla_{f_i} \mathcal{L}(f_i) = & \left(\frac{1}{n^2} \sum_{i'=1}^n h((\phi_1 + \phi_3)^\top f_{i'} + (\phi_2 - \phi_3)^\top z_{i'}) - \frac{2}{n} \right) \\ & \times \nabla h((\phi_1 + \phi_3)^\top f_i + (\phi_2 - \phi_3)^\top z_i)(\phi_1 + \phi_3) + \pi_i + \rho_1(f_i - f(x_i^t; \Theta)), \end{aligned} \quad (29)$$

where $\nabla h(x) = h(x) \times (1 - h(x))$.

The subgradient descent step to update f_i is

$$f_i \leftarrow f_i - \eta \nabla_{f_i} \mathcal{L}(f_i). \quad (30)$$

Optimizing G. Similar to F, we also optimize g_j of G one by one, and the following subgradient algorithm is used to update g_j :

$$\begin{aligned} g_j & \leftarrow g_j - \eta \nabla_{g_j} \mathcal{L}(g_j), \\ \text{where } \nabla_{g_j} \mathcal{L}(g_j) = & 2C_1 I(x_j^s \in \mathcal{Z}^s) W(y_j^s - W^\top g_j) - C_2 I(x_j^s \in \mathcal{U}^s) \left(\sum_{k=1}^c (\log(w_k^\top g_j) + (w_k^\top g_j) \log(w_k^\top g_j)) w_k \right) \\ & + 2C_5 \sum_{j'=1}^m \omega_j + \rho_2 \sum_{j'=1}^m A_{jj'}^s (g_j - g_{j'}) (g_j - g(x_j^s; \Phi)) - \sum_{i=1}^n \alpha_{ij} \tau_i + \frac{\rho_3}{2} \sum_{i=1}^n \left(\sum_{j'=1}^m \alpha_{ij} \alpha_{ij'} g_{j'} - 2\alpha_{ij} z_i \right), \end{aligned} \quad (31)$$

where $I(x) = 1$ if x is true and 0 otherwise.

Optimizing Z. We use the subgradient descent algorithm to update z_i one by one, and the updating rule is

$$\begin{aligned} z_i & \leftarrow z_i - \eta \nabla_{z_i} \mathcal{L}(z_i) \\ \text{where } \nabla_{z_i} \mathcal{L}(z_i) = & \left(\frac{1}{n^2} \sum_{i'=1}^n h((\phi_1 + \phi_3)^\top f_{i'} + (\phi_2 - \phi_3)^\top z_{i'}) - \frac{2}{n} \right) \times \nabla h((\phi_1 + \phi_3)^\top f_i + (\phi_2 - \phi_3)^\top z_i)(\phi_2 - \phi_3) \\ & - 2C_3 I(x_i^t \in L^t) V(y_i^t - V^\top z_i) - C_4 I(x_i^t \in U^t) \left(\sum_{k=1}^c (\log(v_k^\top z_i) + (v_k^\top z_i) \log(v_k^\top z_i)) v_k \right) \\ & + 2C_6 \sum_{i'=1}^m A_{ii'}^t (z_i - z_{i'}) + \tau_i + \rho_3 \left(z_i - \sum_{j=1}^m \alpha_{ij} g_j \right). \end{aligned} \quad (32)$$

Optimizing Θ . The gradient descent algorithm is applied to update Θ :

$$\begin{aligned} \Theta & \leftarrow \Theta - \eta \nabla_{\Theta} \mathcal{L}(\Theta), \\ \text{where } \nabla_{\Theta} \mathcal{L}(\Theta) = & \sum_{i=1}^n \nabla_{f(x_i^t; \Theta)} \mathcal{H}(f(x_i^t; \Theta)) \times \nabla_{\Theta} f(x_i^t; \Theta) + 2C_7 \Theta, \mathcal{H}(f(x_i^t; \Theta)) = \pi_i^\top (f_i - f(x_i^t; \Theta)) \\ & + \frac{\rho_1}{2} \|f_i - f(x_i^t; \Theta)\|_2^2, \nabla_{f(x_i^t; \Theta)} \mathcal{H}(f(x_i^t; \Theta)) = -\pi_i - \rho_1 (f_i - f(x_i^t; \Theta)), \end{aligned} \quad (33)$$

and $\nabla_{\Theta} f(x_i^t; \Theta)$ is the gradient function of the convolutional network function with regard to the network parameters, usually the convolutional filters.

Optimizing Φ . We use the gradient descent algorithm to optimize Φ to minimize $\mathcal{L}(\Phi)$:

$$\Phi \leftarrow \Phi - \eta_{\Phi} \nabla_{\Phi} \mathcal{L}(\Phi),$$

$$\text{where } \nabla_{\Phi} \mathcal{L}_3(\Phi) = \nabla_{g(x_j^s; \Phi)} \mathcal{G}(g(x_j^s; \Phi)) \times \nabla_{\Phi} g(x_j^s; \Phi) + 2C_7 \Phi \mathcal{G}(g(x_j^s; \Phi)) = \bar{\omega}_j^T (g_j - g(x_j^s; \Phi)) + \frac{\rho_2}{2} |g_j - g(x_j^s; \Phi)|_2^2,$$

$$\nabla_{g(x_j^s; \Phi)} \mathcal{G}(g(x_j^s; \Phi)) = - \sum_{j=1}^m \bar{\omega}_j - \rho_2 \sum_{j=1}^m (g_j - g(x_j^s; \Phi)).$$

(34)

Optimizing W . To optimize the classifier parameter in W , we also update the columns one by one. To optimize the k -th column, we use the following subgradient descent step:

$$w_k \leftarrow w_k - \eta \nabla_{w_k} \mathcal{L}(w_k),$$

$$\text{where } \nabla_{w_k} \mathcal{L}(w_k) = -2C_1 \sum_{j: x_j^s \in L^s} (y_j^s(k) - w_k^T g_j) g_j - C_2 \sum_{j: x_j^s \in U^s} ((w_k^T g_j + 1) \log(w_k^T g_j)) g_j + 2C_7 w_k. \quad (35)$$

Optimizing V . To optimize V , we also update the columns one by one. To update the k -th column v_k , we have the following subgradient descent step:

$$v_k \leftarrow v_k - \eta \nabla_{v_k} \mathcal{G}(v_k),$$

$$\text{where } \nabla_{v_k} \mathcal{G}(v_k) = -2C_1 \sum_{j: x_j^t \in L^t} (y_j^t(k) - v_k^T z_j) z_j - C_2 \sum_{j: x_j^t \in U^t} ((v_k^T z_j + 1) \log(v_k^T z_j)) z_j + 2C_7 v_k. \quad (36)$$

Optimizing A . The optimization of the proxy construction coefficient matrix is a constrained minimization problem as follows:

$$\min_A \left\{ L(A) = \sum_{i=1}^n \tau_i^T \left(z_i - \sum_{j=1}^m \alpha_{ij} g_j \right) + \frac{\rho_3}{2} \sum_{i=1}^n \left\| z_i - \sum_{j=1}^m \alpha_{ij} g_j \right\|_2^2 + \sum_{i=1}^n \zeta_i^T \left(\sum_{j=1}^m \alpha_{ij} - 1 \right) + \frac{\rho_4}{2} \sum_{i=1}^n \left\| \sum_{j=1}^m \alpha_{ij} - 1 \right\|_2^2 \right\}, \quad (37)$$

$$\text{s.t. } \forall i = 1, \dots, n, j = 1, \dots, m: 1 \geq \alpha_{ij} \geq 0.$$

Again, we optimize the columns of A one by one. To optimize the i -th column α_i of A , we have the following subproblem:

$$\min_{\alpha_i} \left\{ \tau_i^T (z_i - G \alpha_i) + \frac{\rho_3}{2} |z_i - G \alpha_i|_2^2 + \zeta_i^T (1^T \alpha_i - 1) + \frac{\rho_4}{2} |1^T \alpha_i - 1|_2^2 \right\}, \quad (38)$$

$$\text{s.t. } 1 \geq \alpha_i \geq 0,$$

where $\mathbf{1}$ is an all-one vector and $\mathbf{0}$ is an all-zero vector. This is a typical quadratic programming (QP) problem, and we use the active set algorithm to solve it.

Optimizing Π, Ψ, Ξ , and ς . To optimize the dual variables, we have the following maximization problem:

$$\begin{aligned} \max_{\Pi, \Psi, \Xi, \varsigma} \left\{ \mathcal{L}(\Pi, \Psi, \Xi, \varsigma) = \sum_{i=1}^n \pi_i^\top (f_i - f(x_i^t; \Theta)) \right. \\ + \sum_{j=1}^m \omega_j^\top (g_j - g(x_j^s; \Phi)) + \sum_{i=1}^n \tau_i^\top \left(z_i - \sum_{j=1}^m \alpha_{ij} g_j \right) \\ \left. + \sum_{i=1}^n \varsigma_i^\top \left(\sum_{j=1}^m \alpha_{ij} - 1 \right) = \text{Tr}(\Pi^\top N) + \text{Tr}(\Psi^\top J) + \text{Tr}(\Xi^\top D) + \varsigma^\top e \right\}, \end{aligned} \quad (39)$$

where $N = [(f_1 - f(x_1^t; \Theta)), \dots, (f_n - f(x_n^t; \Theta))]$,
 $J = [(g_1 - g(x_1^s; \Phi)), \dots, (g_m - g(x_m^s; \Phi))]$,
 $D = \left[\left(z_1 - \sum_{j=1}^m \alpha_{1j} g_j \right), \dots, \left(z_n - \sum_{j=1}^m \alpha_{nj} g_j \right) \right]$, and
 $e = \left[\left(\sum_{j=1}^m \alpha_{1j} - 1 \right), \dots, \left(\sum_{j=1}^m \alpha_{nj} - 1 \right) \right]$,

are the residual matrices and vector and $\text{Tr}(X)$ is the trace of matrix X . To solve this problem, we use the gradient-ascent algorithm, and the updating rules are

$$\begin{aligned} \Pi &\leftarrow \Pi + \epsilon \nabla_{\Pi} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma), \nabla_{\Pi} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma) = N, \\ \Psi &\leftarrow \Psi + \epsilon \nabla_{\Psi} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma), \nabla_{\Psi} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma) = J, \\ \Xi &\leftarrow \Xi + \epsilon \nabla_{\Xi} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma), \nabla_{\Xi} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma) = D, \\ \varsigma &\leftarrow \varsigma + \epsilon \nabla_{\varsigma} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma), \text{ and } \nabla_{\varsigma} \mathcal{L}(\Pi, \Psi, \Xi, \varsigma) = e, \end{aligned} \quad (40)$$

where ϵ is the ascent step size.

Besides solving the minimization problem, we also need to solve the parameter of the SMI term of (13) ϕ^* . To learn the parameter of **SMI**, ϕ^* , according to (10), we have the following minimization problem:

$$\phi^* = \min_{\phi} \left\{ S(\phi) = -\frac{2}{n} \sum_{i=1}^n r(f_i, z_i; \phi) + \frac{1}{n^2} \sum_{i, i'=1}^n r(f_i, z_i; \phi) r(f_{i'}, z_{i'}; \phi) = -\frac{2}{n} \sum_{i=1}^n h(\phi^\top r_i) + \frac{1}{n^2} \sum_{i, i'=1}^n h(\phi^\top r_i) h(\phi^\top r_{i'}) \right\}, \quad (41)$$

where $r_i = [f_i; z_i; f_i - z_i]$.

The gradient function of $\mathcal{S}(\phi)$ regarding ϕ is

$$\nabla_{\phi} \mathcal{S}(\phi) = -\frac{2}{n} \sum_{i=1}^n \nabla_{\phi} h(\phi^\top r_i) r_i + \frac{1}{n^2} \sum_{i, i'=1}^n \left(\nabla_{\phi} h(\phi^\top r_i) h(\phi^\top r_{i'}) r_i + h(\phi^\top r_i) \nabla_{\phi} h(\phi^\top r_{i'}) r_{i'} \right). \quad (42)$$

We use the gradient descent algorithm to update ϕ to minimize $\mathcal{S}(\phi)$:

$$\phi \leftarrow \phi - \eta \nabla_{\phi} \mathcal{S}(\phi). \quad (43)$$

3. Experiments

In this section, we experimentally evaluate the proposed algorithm over benchmark data sets.

3.1. Benchmark Data Sets. In our experiment, we use the following benchmark data sets:

- (i) Amazon review data set. This data set has four domains of four products, including books, DVD, and music. For each domain, there are 2,000 positive reviews and 2,000 negative reviews. To fit the content of the review to the CNN network, we firstly tokenize the review to a sequence of words and secondly represent each word to by word embedding and then use the 1D-CNN as the CNN model.
- (ii) UCSD anomaly detection data set. This data set has 70 sequences of video, which contains 13,900 frames. This data set is used as a source domain. Meanwhile, we also collected 122 sequences of videos from subway stations, malls, and communities in China as a target domain. The target domain has 24,400 frames. Both target and source domain data sets have 6 classes of anomalies.
- (iii) TRECVID video concept detection data set. This data set has two domains of data. One domain is the data of TRECVID 2005, which has 61,901 frames from video, while the other one domain is the data of TRECVID 2007, which has 21,532 frames. The classification problem is to categorize one frame to one of the 36 semantic concepts.
- (iv) Protein subcellular localization data set. The last data set is a protein data set. It is composed of three domains. The first domain is the MultiLoc data set which has 5,859 proteins, the second one is the BaCelLoc data set with 4,286 proteins, and the last data set Euk-mPLoc has 5,618 proteins. The problem of this data is to predict the subcellular locations of each protein from the amino acid sequence. To this end, we first map each amino acid by embedding technology and then use the 1D-CNN model to extract the features.

The statistics of the data sets are summarized in Table 1.

3.2. Experimental Setting. To perform the experiment, we design a leave-one-out protocol to use each domain as a target domain, while the other domains are combined as one single source domain. With each target domain-source domain configuration, we use the 10-fold cross-validation protocol to perform the training test process. The target domain data set is split into ten folds, one fold is used as the test set, and the remaining nine folds are combined with the source domain as the training set. The model is learned over the training set and tested over the test set. To measure the performance of the method, we calculate the average classification accuracy over different target domains.

TABLE 1: Summary of data sets.

Data set	# domain	# sample	Data type
Amazon	3	12,000	Text
UCSD	2	38,300	Video
TRECVID	2	83,433	Video
Protein	3	15,763	Protein

3.3. Experimental Results. In our experiments, we firstly study the properties of the proposed method experimentally and then compare it to the other state-of-the-art methods.

3.4. Convergence Analysis. Since the algorithm is an iterative algorithm, we are interested in the convergence of the algorithm. We plot the curves of the objective values against the iteration number in Figure 1. From this figure, we can see that, for all four benchmark data sets, the algorithm converges well at 50 iterations. For the protein data set, the object value keeps decreasing but cannot reach a lower value after 50 iterations. The possible reason is that the algorithm finds a local minimum object instead of a global minimum. The overall conclusion for the convergence of the algorithm is for all the tested data sets, and the algorithm can converge at certain number of iterations. This is due to the effective alternate optimization method.

3.5. Tradeoff Parameter Analysis. We have seven tradeoff parameters in the objective, which control the weights of different objective terms. We also study how the performance of the algorithm changes with the change in these tradeoff parameters. The classification accuracies with different values of the tradeoff parameter values are shown in Figure 2.

From this figure, we have the following observations:

- (i) The increasing C_1 and C_3 improve the performance significantly since it is the supervision term of the labeled data. This means the supervision of the labeled plays an important role in the learning of predictive models in the transfer learning problem.
- (ii) The increasing C_2 and C_4 also boost the performance; however, it is not as significant as the class error terms' tradeoff parameters. The reduction of entropy of the unlabeled samples' classification responses from both domains also improves the performance.
- (iii) C_5 and C_6 are the tradeoff parameters of two domains' manifold regularization terms. Their increasing values also improve the accuracy of the target domain prediction. This means the neighborhood smoothness is also important for the transfer learning problem.
- (iv) The performance is stable to the change in C_7 , the tradeoff value of the squared ℓ_2 norm term, which

FIGURE 1: Convergence curves.

FIGURE 2: Tradeoff parameter analysis: (a) UCSD and (b) protein.

means this term did not play a critical role in the learning problem.

3.6. Comparison to Other Methods. We compare our proposed algorithm against several state-of-the-art CNN-based transfer learning methods, including the methods developed by Haque et al. [11, 12], Zhang et al. [14], Long et al. [21], and Pan and Yang [7]. The comparison results are shown in Figure 3.

From this figure, we can see the proposed method always outperforms the other methods in all cases. The second best method is Long et al. method. The results show the advantage of our method, especially the SMI-guided domain-

transfer CNN learning and the proxy mapping mechanism. Among the four data sets, the most difficult one is the TRECVID, where all the classification accuracies of different methods are below 0.75. However, in this data set, our methods give the most significant improvements over the other methods. It is the only method which achieves an accuracy higher than 0.72.

4. Conclusions

In this paper, we proposed a novel CNN-based transfer learning method. We construct a proxy for each target domain sample in the source domain space and use the SMI

FIGURE 3: Comparison to state-of-the-arts.

as a measure to match the constructed proxy and the convolutional representation of a target domain sample. The learning of the proxy construction parameters and the CNN parameters are learned simultaneously by a unified learning framework. The estimation and parameter learning of the SMI is also driven by the proxy and CNN outputs. Thus, this framework can optimize the CNN, proxy, and SMI parameters jointly and automatically. Experimental results show the advantages of the proposed framework and algorithm.

Data Availability

All the data sets used in this paper to produce the experimental results are publicly accessed online.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the work reported in this paper.

Acknowledgments

This work was supported by the Beijing Natural Science Foundation (no. 9194027), National Key R&D Program of China (Grant no. 2018YFC0704800), 2018-2019 Excellent Talent Program, Xicheng District, Beijing, and National Natural Science Foundation of China (no. 71904095).

References

- [1] P. Russom, "Big data analytics," *TDWI Best Practices Report*, vol. 19, no. 4, pp. 1–34, 2011.
- [2] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.
- [3] T. M. Mitchell, *Machine Learning*, Springer, Berlin, Germany, 1997.
- [4] E. Alpaydin, *Introduction to Machine Learning*, MIT press, Cambridge, MA, USA, 2020.
- [5] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, no. 1, pp. 1–130, 2009.
- [6] X. J. Zhu, *Semi-supervised Learning Literature Survey*. Tech. Rep., University of Wisconsin-Madison Department of Computer Sciences, Madison, Wisconsin, 2005.
- [7] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.
- [8] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.
- [9] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1975–1981, San Francisco, CA, USA, June 2010.
- [10] S. Bansod and A. Nandedkar, "Transfer learning for video anomaly detection," *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 3, pp. 1967–1975, 2019.
- [11] T. U. Haque, N. N. Saber, and F. M. Shah, "Sentiment analysis on large scale amazon product reviews," in *Proceedings of the IEEE International Conference on Innovative Research and Development (ICIRD)*, pp. 1–6, Bangkok, Thailand, 2018.
- [12] A. Bhatt, A. Patel, H. Chheda, and K. Gawande, "Amazon review classification and sentiment analysis," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 6, pp. 5107–5110, 2015.
- [13] T. N. Sainath, A. Mohamed, B. Kingsbury, and B. Ramabhadran, "Deep convolutional neural networks for lvcsr," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8614–8618, Vancouver, Canada, May 2013.
- [14] G. Zhang, G. Liang, F. Su, F. Qu, and J.-Y. Wang, "Cross-domain attribute representation based on convolutional neural network," in *Proceedings of the International Conference on Intelligent Computing*, pp. 134–142, Wuhan, China, October 2018.
- [15] Y. Geng, R. Z. Liang, W. Li et al., "Learning convolutional neural network to maximize pos@ top performance measure," in *Proceedings of the ESANN 2017 - Proceedings*, pp. 589–594, Bruges, Belgium, April 2017.
- [16] G. Zhang, G. Liang, W. Li et al., "Learning convolutional ranking-score function by query preference regularization," in *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, pp. 1–8, Guilin, China, October 2017.
- [17] Y. Geng, G. Zhang, W. Li et al., "A novel image tag completion method based on convolutional neural transformation," in *Proceedings of the International Conference on Artificial Neural Networks*, pp. 539–546, Sardinia, Italy, September 2017.
- [18] M. Yamada, L. Sigal, M. Raptis, M. Toyoda, Y. Chang, and M. Sugiyama, "Cross-domain matching with squared-loss mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1764–1776, 2015.
- [19] K. Wang, J. Liu, and J.-Y. Wang, "Learning domain-independent deep representations by mutual information minimization," *Computational Intelligence and Neuroscience*, vol. 2019, pp. 1–14, Article ID 9414539, 2019.
- [20] M. Xiao and Y. Guo, "Feature space independent semi-supervised domain adaptation via kernel matching," *IEEE*

- Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 54–66, 2014.
- [21] M. Long, Y. Cao, Z. Cao, J. Wang, and M. I. Jordan, “Transferable representation learning with deep adaptation networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 12, pp. 3071–3085, 2018.
 - [22] L. Duan, I. W. Tsang, and D. Xu, “Domain transfer multiple kernel learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 3, pp. 465–479, 2012.
 - [23] M. Long, H. Zhu, J. Wang, and M. I. Jordan, “Deep transfer learning with joint adaptation networks,” in *Proceedings of the 34th International Conference on Machine Learning*, pp. 2208–2217, Sydney, Australia, August 2017.
 - [24] W. Ge and Y. Yu, “Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1086–1095, Honolulu, HI, USA, July 2017.
 - [25] T. Suzuki and M. Sugiyama, “Sufficient dimension reduction via squared-loss mutual information estimation,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 804–811, Sardinia, Italy, May 2010.
 - [26] G. Niu, W. Jitkrittum, B. Dai, H. Hachiya, and M. Sugiyama, “Squared-loss mutual information regularization: A novel information-theoretic approach to semi-supervised learning,” in *Proceedings of the International Conference on Machine Learning*, pp. 10–18, Atlanta, GA, USA, June 2013.
 - [27] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.