Hindawi
Computational Intelligence and Neuroscience
Volume 2019, Article ID 9414539, 14 pages
https://doi.org/10.1155/2019/9414539

Research Article

Learning Domain-Independent Deep Representations by Mutual Information Minimization

Ke Wang[1], **Jiayong Liu**[2] **and Jing-Yan Wang**[3]

[1]*College of Mathematics, Sichuan University, Chengdu 610065, China*
[2]*College of Cybersecurity, Sichuan University, Chengdu 610065, China*
[3]*New York University Abu Dhabi, Abu Dhabi, UAE*

Correspondence should be addressed to Ke Wang; wangke1845@outlook.com

Received 4 January 2019; Revised 1 May 2019; Accepted 21 May 2019; Published 16 June 2019

Academic Editor: Friedhelm Schwenker

Domain transfer learning aims to learn common data representations from a source domain and a target domain so that the source domain data can help the classification of the target domain. Conventional transfer representation learning imposes the distributions of source and target domain representations to be similar, which heavily relies on the characterization of the distributions of domains and the distribution matching criteria. In this paper, we proposed a novel framework for domain transfer representation learning. Our motive is to make the learned representations of data points independent from the domains which they belong to. In other words, from an optimal cross-domain representation of a data point, it is difficult to tell which domain it is from. In this way, the learned representations can be generalized to different domains. To measure the dependency between the representations and the corresponding domain which the data points belong to, we propose to use the mutual information between the representations and the domain-belonging indicators. By minimizing such mutual information, we learn the representations which are independent from domains. We build a classwise deep convolutional network model as a representation model and maximize the margin of each data point of the corresponding class, which is defined over the intraclass and interclass neighborhood. To learn the parameters of the model, we construct a unified minimization problem where the margins are maximized while the representation-domain mutual information is minimized. In this way, we learn representations which are not only discriminate but also independent from domains. An iterative algorithm based on the Adam optimization method is proposed to solve the minimization to learn the classwise deep model parameters and the cross-domain representations simultaneously. Extensive experiments over benchmark datasets show its effectiveness and advantage over existing domain transfer learning methods.

1. Introduction

1.1. Background. Transfer learning is a machine learning problem which deals with data from two domains [1–6]. One domain is target domain, and in this domain, we aim to learn an effective machine learning model for prediction. Another domain is source domain, in which we have sufficient labeled data points. Usually, in the target domain, the labeled data points are of a small number, which is not sufficient to learn an effective model. Thus domain transfer learning tries to transfer the knowledge in the source domain to the target domain to help the learning in the target domain. Although the target domain and source domain share the same input and output space, the distribution of the input data points of two domains is of significant difference. For example, in the problem of text topic categorization, the newspaper article is a source domain, where almost all the articles are labeled well, and the personal communication message is a target domain. Usually, the message texts are not labeled, or only a small number of them are labeled. It is natural to use the newspaper articles and the corresponding labels to help the learning of model for the categorization of the message texts. However, the newspaper articles are written normally, while the personal messages are usually written casually. Thus, the usage of words and the writing styles are very different. This leads to the significant difference between the distributions

of the source domain (newspaper article) and the target domain (personal message). Transfer learning aims to build a predictive model for the target domain by utilizing the data points of both domains, even though they are of different distributions.

In this case, it is very necessary to map the data points of both domains to a common data space so that they lie in the same distribution, and we can directly train a model for the target domain by using both domains' data points' representations. Another solution is to learn a model for the source domain first and then adapt it to the target domain. In this paper, we focus on the first solution where the data points are mapped to a common space. This solution aims to learn domain transferable representations for data points in different domains. Different representation learning methods have been applied for domain transferable representation learning, including multikernel learning [7–10], deep learning [11–20], nonnegative matrix factorization [21–24], sparse coding [25–28], etc. For the domain distribution matching-based domain transfer learning, the most popular method is based on the maximum mean discrepancy criterion. It calculates the means of the representations of data points of source and target domain and minimizes the squared ℓ_2 norm distance to match the two domains.

In this paper, we study the problem of learning domain transfer representations. However, we do not consider the distribution matching of two domains but consider learning representations which can be directly generalized to two domains.

1.2. Related Works. In this section, we briefly introduce the state-of-the-art methods for transferable representation learning.

Theauthors in [4] present a novel method to learn deep networks for domain adaptation. The proposed method maps the outputs of all layers of deep networks to reproduce kernel Hilbert spaces and tries to match the distributions of these layers' outputs of target and source domains. Moreover, the kernel space mapping is conducted by applying multikernel learning, where the optimal kernel function is a weighted linear combination of multiple kernels. This is different from the conventional transfer learning methods, which only matches the distributions of outputs of the last layer of source and target domains. The mismatching of the two distributions is measured by the maximum mean discrepancy criterion, which actually minimizes the squared Euclidean distance between the mean outputs of the source and target domains of the corresponding layer.

The authors in [6] proposed to learn a domain transfer learning model for feature space independent domains. In this case, the source domain and target domain have completely different feature space. The source domain data points are mapped to target domain data points. The mapping guarantees that any source domain data point is mapped to a target domain point with the same class label. The target domain and source domain (mapped to target domain) are both represented by kernel matrices. To measure how well the two domains are allied, the Hilbert Schmidt Independence Criterion is applied. It calculates the trace of the product of the target domain kernel matrix and the mapped source domain kernel matrix. By maximizing the trace of the product of target and source, the distributions of the source and target domains are aligned and matched.

The authors in [1] proposed a novel method for transfer learning. It selects the data points from the source domain for the target domain learning problem. To be specific, it assigns a weight to each source domain data point, which plays the role of selective weight. This weight has two functions. The first function is to select important source domain data points to represent the source domain to match the target domain. Instead of calculating the mean vector of the source domain features, this method calculates the weighted mean and matches it to the target domain by the maximum mean discrepancy criterion. The second function is to weight the loss function of the target domains. The target domain data points' weights and the classifier parameters are also learned simultaneously in an iterative algorithm.

The authors in [2] developed a novel multikernel classifier for domain transfer learning. It constructs the kernel function by multikernel combination with learned weights. Meanwhile, the kernel weights and classifier parameters are learned simultaneously. To match the source domain and target domain, the data points of two domains are mapped to a nonlinear Hilbert space, and their distributions are matched in this space. The learning algorithm minimizes the classification losses over the labeled data points of both domains and the squared Euclidean distance between the mean multikernel representations of tow domains under the maximum mean discrepancy criterion.

1.3. Our Contributions

1.3.1. Motive. All the above methods are based on the matching of two domains' distributions of the data representations. The two key components of this method are the representation of distributions and the metric of the mismatching of the two distributions. In this paper, we give up this framework and propose a completely different framework for domain transfer learning. We observed that for an ideal representation model across two different domains, from its output of one data point, we cannot tell which domain it is from. At the same time, we can separate it from its true class and the other classes according to its output of the cross-domain representation model. It means that the representation of a data point is independent of its domain, but closely relevant to its class. Thus, instead of measuring the mismatch of source and target domain distributions, we measure the independence of the representations and domain-belonging indicators of the data points. To measure

the dependency between the representation and the domain indicator, we employ the mutual information. By minimizing the mutual information between them, we learn the domain-independent representations. Meanwhile, we also propose to maximize the margin of each data point so that it can be separated from data points from other classes and kept close to the data points from the same classes.

1.3.2. Our Method. Motivated by the above ideas, we propose a novel deep learning model for the representation of data points of transfer learning problems. Firstly, to enhance the ability to discriminate data points of different classes, we propose to learn a unique deep convolutional network for each class, named classwise convolutional representation model. This is different from traditional domain transfer representation models, which learns a common model for all classes. To make the outputs of this model independent from the domain indicators, we propose to minimize the mutual information between the representation model outputs and the domain indicators. The mutual information estimation is based on the probability of representations and conditional probability of domain indicators given representations. We develop novel estimators for the conditional probability of domain indicators given representations. The estimator is defined over the neighborhood of the data point of the given representation, and it calculates the normalized summation of the soft weights of the data points from the input domain. To make the outputs of the model to be discriminate, we proposed to maximize the margin of each data point in the corresponding class. The margin is defined as the difference between intraclass dissimilarity and the interclass dissimilarity. The intraclass dissimilarity is defined in an intraclass neighborhood which contains a set of neighboring data points from the same class, while the interclass dissimilarity is defined in an interclass neighborhood which contains a set of neighboring data points from the other classes. To learn the representation model parameters, we build a unified learning framework. The objective function is defined by combining the margins, mutual information, and a squared ℓ_2 norm term to control the complexity of the model. An iterative algorithm based on Adam is developed to solve the problem.

Remark. The overall diagram of the proposed learning framework for each class is given in Figure 1. As we can see from the figure, for each CNN model, its outputs are regularized by two types of auxiliary information: the domain indicator and the class label. Our framework calculates the mutual information between the CNN representations and the indicators of domains and minimizes it. Meanwhile, it calculates the margin from class label and maximizes it. In this way, this framework enables the CNN model to be discriminative and insensitive to the diversities of domains.

Our contributions are of three folds:

(1) For the first time, the idea of learning cross-domain representations which are independent of domains is proposed for transfer learning. Instead of learning representations and making the distributions of two domains' representations to match each other, we directly learn representations which are independent of their domain-belonging indicators. The mutual information is used to measure such dependency of representations and domain indicators, and it is minimized to seek the domain-independent representations.

(2) We develop a novel and practical representation learning method to minimize the mutual information between the data points' representations and domain indicators. The mutual information between representations and domain indicators of data points is estimated according to the probability of representation and the conditional probability of domain indicator given representation. We estimate the conditional probability of a data point's domain indicator given its representation over its neighborhood. It is calculated as a summation of the normalized Gaussian kernel based similarity measured of the data points in the neighborhood but from the considering domain.

(3) We propose a novel transfer learning framework for learning domain transfer deep representation models. It is a classwise model and we learn the parameters by simultaneously maximizing the margin of each data point of this class and minimizing the mutual information between the data points' representations and their domain indicators. An iterative algorithm is developed to learn the optimal representations and the parameters of the model to output these representations.

1.4. Paper Organization. The paper is organized as follows: in Section 2, we introduce the proposed method in detail, including its mathematical modeling, problem optimization, and iterative algorithm design. In Section 3, we evaluate the proposed method over several transfer learning benchmark datasets to compare it against state-of-the-art transfer learning methods. In Section 4, we give the conclusion of this paper and some future works.

2. Methods

2.1. Definition of Symbols. In this section, we give a list of detailed definitions of the symbols used in the following sections.

2.2. Problem Modeling. We assume we have a set of n training data points, denoted as $\mathcal{X} = \{X_1, \ldots, X_n\}$, where $X_i = \{\mathbf{x}_{i1}, \ldots, \mathbf{x}_{in_i}\}$ is the i-th data point, which is composed of n_i instances, and $\mathbf{x}_{ij}$ is the j-th instance of the i-th data point. For the computer vision problem, a data point is an image, and an instance is an image patch, while for the natural language process problem, a data point is a sentence, and an instance is the embedding vector of a word.

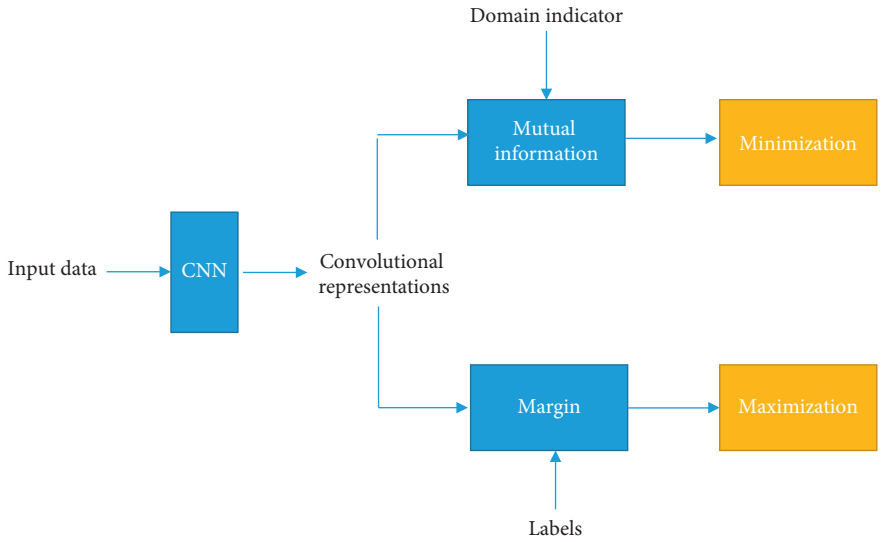


Figure 1: Overall diagram of the proposed learning framework for minimum mutual information domain transfer representation.

2.2.1. Large Margin Class-Specific Convolutional Representations. We consider a classification problem of L classes, the training set can be divided into subsets of L classes and a set of unlabeled data points whose class labels are not known yet. The training set can be denoted as follows:

$$\mathcal{X} = \mathcal{X}_1 \cup \cdots \cup \mathcal{X}_L \cup \mathcal{X}_U, \tag{1}$$

where $\mathcal{X}_l$ is the subset of the l-th class and $\mathcal{X}_U$ is the subset of the unlabeled data points.

For the l-th class, we learn a class-specific deep CNN model to represent the data point X,

$$f_l: X \longrightarrow \mathbf{z} = f_l(X; W_l) \in \mathbb{R}^m, \tag{2}$$

which outputs a vector of m dimensions as the class-specific convolutional vector, and W_l presents the parameters of the model.

Remark. We choose to learn the convolutional representations due to the following two reasons:

(1) CNN model is good at extracting local patterns by utilizing a large number of sliding local filters, while in most domain transfer applications discussed in this paper, the local patterns play the most important role. For example, in the cross-domain image categorization task, for two images of different domains but containing the same object, the CNN model can capture the local region of the object with some local filters while ignoring the contexts which may vary in different domains. Another example is the text-related tasks; long sentences of the same topic may have different linguistic styles of different domains, but still contains short phrase, which could be captured by the CNN model effectively by its sliding local filters to extract features from short phrases.

(2) CNN model, compared to the other deep learning models, such as recurrent neural network (RNN), has a more efficient training process. The CNN model has a parallel structure, and the responses of a sliding filter are calculated independently from each other; thus, its computing can be easily paralleled by GPU. This is different from the RNN model which has a sequential structure, where the response of a node is calculated based on the response of the previous nodes, which makes its computing time longer than the CNN model.

Naturally, we hope the class-specific convolutional representations can separate the data points of the l-th class and the other classes as far as possible so that the classification performance can be improved. To this end, we propose to learn to discriminate convolutional representations for the data points of l-th class, by maximizing the local margin of each data point in this class. For the local margin of a data point in the l-th class, $X_i \in \mathcal{X}_l$ is defined by its intraclass neighborhood and interclass neighborhood. The intraclass neighborhood is the set of κ nearest neighboring data points in the same class, $\mathcal{X}_l$:

$$\mathcal{N}_i^+ = \kappa - \arg\min_{X_j : X_j \in \mathcal{X}_l} \left\| \mathbf{z}_i - \mathbf{z}_j \right\|_F^2, \tag{3}$$

where the ℓ_2 norm distance between their l-th class-specific convolutional representations is used to measure the distance of neighbors. Meanwhile, the interclass neighborhood is the set of κ nearest neighboring data points from a different class:

$$\mathcal{N}_i^- = \kappa - \arg\min_{X_{j'} : X_{j'} \in \mathcal{X}_{l'}, l' \neq l} \left\| \mathbf{z}_i - \mathbf{z}_{j'} \right\|_F^2. \tag{4}$$

Note that to search for the interclass neighbors of X_i, we use the convolutional representations of its class, the l-th class, even for the data points of the other classes. We further calculate an affinity measure for X_i and a data point X_j from $\mathcal{N}_i^+$, according to their class-specific convolutional representations and a Gaussian kernel function:

$$A_{ij}^+ = \frac{g(\mathbf{z}_i - \mathbf{z}_j)}{\sum_{X_{j'} \in N_i^+} g(\mathbf{z}_i - \mathbf{z}_{j'})}, \tag{5}$$

where $g(x) = \exp(-\|x\|_F^2 / 2\sigma^2)$.

Similarly, we also calculate the interclass affinity between X_i and data points in $\mathcal{N}_i^-$:

$$A_{ij'}^- = \frac{g\left(\mathbf{z}_i - \mathbf{z}_{j'}\right)}{\sum_{X_{j'} \in N_i^-} g\left(\mathbf{z}_i - \mathbf{z}_{j'}\right)}. \tag{6}$$

The local margin of X_i is defined as the difference between the weighted intraclass convolutional dissimilarity and the interclass dissimilarity:

$$m_l(X_i) = \sum_{X_j \in \mathcal{N}_i^-} A_{ij'}^- \left\|\mathbf{z}_i - \mathbf{z}_{j'}\right\|_F^2 - \sum_{X_j \in \mathcal{N}_i^+} A_{ij'}^+ \left\|\mathbf{z}_i - \mathbf{z}_j\right\|_F^2. \tag{7}$$

We proposed to maximize the local margin to improve the ability to separate data points of the l-th class from the other classes. To this end, we minimize the following objective function of margins over the data points of the l-th class to learn the convolutional representation network parameters:

$$\min_{W_l} \sum_{X_i \in \mathcal{X}_l} \left(\sum_{X_j \in \mathcal{N}_i^+} A_{ij'}^+ \left\|\mathbf{z}_i - \mathbf{z}_j\right\|_F^2 - \sum_{X_j \in \mathcal{N}_i^-} A_{ij'}^- \left\|\mathbf{z}_i - \mathbf{z}_{j'}\right\|_F^2 \right). \tag{8}$$

2.2.2. Minimum Mutual Information Domain Adaptation. Since we are considering the problem of domain transfer learning, the training data points are from a source domain and a target domain, and we denote the source domain training set as $\mathcal{X}_S$ and $\mathcal{X}_T$; thus, $\mathcal{X} = \mathcal{X}_S \cup \mathcal{X}_T$. We introduce a domain indicator for each data point X_i to present which domain it is from, $\pi_i \in \{\text{source}, \text{target}\}$, where π_i = source indicates that X_i is a source domain data point, while π_i = target indicates that it is a target domain data point. Naturally, we hope the classwise convolutional representations of the source and target domains are mapped to a common space of the same distribution. To this end, we impose that the representations of the data points and their domain indicators are independent of each other so that from the presentation, we cannot measure which domain it is from. To measure the mutual dependence between the classwise representation $\mathbf{z}$ and the domain indicator π, we proposed to use the mutual information between them, $I(\pi; \mathbf{z})$.

Remark. According to the probability theory and information theory, the mutual information between two variables is a measure of the mutual dependence between them. For two variables, x, and y, the definition of mutual information of x and y is calculated by the double integral as follows:

$$I(x; y) = \int_y \int_x p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right) dx\, dy, \tag{9}$$

where $p(x, y)$ is the joint probability function of x and y and $p(x)(p(y))$ is the probability function of $x(y)$. For the discrete variables, the mutual information is calculated by the double sum:

$$I(x; y) = \sum_y \sum_x p(x, y) \log\left(\frac{p(x, y)}{p(x)p(y)}\right). \tag{10}$$

According to the mutual information's relation to Kullback–Leibler divergence,

$$\begin{aligned} I(x; y) &= \sum_y p(y) D_{\mathrm{KL}}(p(x \mid y)p(x)) \\ &= \sum_y p(y) \sum_x p(x \mid y) \log\left(\frac{p(x \mid y)}{p(x)}\right), \end{aligned} \tag{11}$$

where $D_{\mathrm{KL}}(p(x \mid y)p(x))$ is the Kullback–Leibler divergence between $p(x \mid y)$ and $p(x)$ and $p(x \mid y)$ is the conditional probability of x given y. Following equation (11), the mutual information between π and $\mathbf{z}$ is defined as follows:

$$I(\pi; \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) \sum_{\pi} p(\pi \mid \mathbf{z}) \log\left(\frac{p(\pi \mid \mathbf{z})}{p(\mathbf{z})}\right). \tag{12}$$

To estimate the mutual information over the training set, we propose to recalculate $I(\pi; \mathbf{z})$ as follows:

$$I(\pi; \mathbf{z}) = \sum_{i: X_i \in \mathcal{X}} p(\mathbf{z}_i) \sum_{\pi \in \{\text{source}, \text{target}\}} p(\pi \mid \mathbf{z}_i) \log\left(\frac{p(\pi \mid \mathbf{z}_i)}{p(\mathbf{z}_i)}\right). \tag{13}$$

In the following, we discuss how to estimate the conditional probability of domain indicator given the convolutional representation, $p(\pi \mid \mathbf{z}_i)$, and the probability of the convolutional representation, $p(\mathbf{z}_i)$.

2.2.3. Estimation of $p(\pi \mid \mathbf{z}_i)$. To estimate the probability of π given a data point, we propose to calculate the density of π over the neighborhood of X_i. $\mathcal{X}_i$ is the set of k-nearest neighbors,

$$\mathcal{N}_i = k - \arg\min_{X_j \in \mathcal{X}} \left\|\mathbf{z}_i - \mathbf{z}_j\right\|_F^2, \tag{14}$$

and the probability of π over $\mathcal{X}_i$ is calculated as the empirical distribution,

$$p(\pi \mid \mathbf{z}_i) = \frac{1}{|\mathcal{N}_i|} \sum_{X_j \in \mathcal{N}_i} \delta(\pi_j = \pi), \tag{15}$$

where $\delta(x) = 1$ if x is true, otherwise 0. According to equation (15), $p(\pi \mid \mathbf{z}_i)$ is the weighted summation of $\delta(\pi_j = \pi)$ over $\mathcal{N}_i$, and the weights are hard weight $(1/|\mathcal{N}_i|)$. We release the calculation of the weights as soft weight according to Gibbs distribution as follows:

$$\begin{aligned} p(\pi \mid \mathbf{z}_i) &= \sum_{j: X_j \in \mathcal{N}_i} \omega_{ij} \delta(\pi_j = \pi), \\ \text{where } \omega_{ij} &= \frac{g\left(\mathbf{z}_i - \mathbf{z}_j\right)}{\sum_{j' \in \mathcal{N}_i} g\left(\mathbf{z}_i - \mathbf{z}_{j'}\right)}. \end{aligned} \tag{16}$$

The weights satisfy the constraints of $\sum_{j \in \mathcal{N}_i} \omega_{ij} = 1$ and $\omega_{ij} \geq 0$.

2.2.4. Estimation of $p(\mathbf{z}_i)$. We assume the convolutional representations are evenly distributed; thus, we use a simple empirical distribution function to calculate the probability of $\mathbf{z}_i$ as follows:

$$p(\mathbf{z}_i) = \frac{1}{n}. \tag{17}$$

Substituting equations (17) and (15) in (13), we rewrite the mutual information between variables $\mathbf{z}$ and π as follows:

$$\begin{aligned} I(\pi;\mathbf{z}) = \frac{1}{n} \sum_{i:X_i\in\mathcal{X}} \sum_{\pi\in\{\text{source},\text{target}\}} \left(\sum_{j:X_j\in\mathcal{N}_i} \omega_{ij}\delta(\pi_j = \pi) \right) \\ \cdot \left(\log\left(\sum_{j:X_j\in\mathcal{N}_i} \omega_{ij}\delta(\pi_j = \pi) \right) + \log(n) \right). \end{aligned} \tag{18}$$

To simplify the equations, we introduce the following variables:

$$\begin{aligned} h_i &= \sum_{j:X_j\in\mathcal{N}_i} g(\mathbf{z}_i - \mathbf{z}_j), \\ h_i^s &= \sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{source}} g(\mathbf{z}_i - \mathbf{z}_j), \\ h_i^t &= \sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{target}} g(\mathbf{z}_i - \mathbf{z}_j), \end{aligned} \tag{19}$$

so that

$$\begin{aligned} h_i &= h_i^s + h_i^t \\ \sum_{j:X_j\in\mathcal{N}_i} \omega_{ij}\delta(\pi_j = \text{source}) &= \frac{h_i^s}{h_i}, \\ \sum_{j:X_j\in\mathcal{N}_i} \omega_{ij}\delta(\pi_j = \text{target}) &= \frac{h_i^s}{h_i}. \end{aligned} \tag{20}$$

We rewrite equation (18) with h_i, h_i^s, and h_i^t as follows:

$$\begin{aligned} I(\pi;\mathbf{z}) &= \frac{1}{n} \sum_{i:X_i\in\mathcal{X}} \left[\frac{h_i^s}{h_i}\log\left(\frac{h_i^s}{h_i}\right) + \frac{h_i^s}{h_i}\log(n) \right. \\ &\quad \left. + \frac{h_i^s}{h_i}\log\left(\frac{h_i^s}{h_i}\right) + \frac{h_i^s}{h_i}\log(n) \right] \\ &= \frac{1}{n} \sum_{i:X_i\in\mathcal{X}} \left[\frac{h_i^s\log(h_i^s) + h_i^t\log(h_i^t)}{h_i} + \log(n) - \log(h_i) \right] \\ &= \frac{1}{nh_i} \sum_{i:X_i\in\mathcal{X}} \left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{source}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad \cdot \log\left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{source}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad + \frac{1}{nh_i} \sum_{i:X_i\in\mathcal{X}} \left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{target}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad \cdot \log\left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{target}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad - \frac{1}{n} \sum_{i:X_i\in\mathcal{X}} \log\left(\sum_{j:X_j\in\mathcal{N}_i} g(\mathbf{z}_i - \mathbf{z}_j) \right) + \log(n). \end{aligned} \tag{21}$$

To learn a cross-domain representation to map the data of both domains to common space, we reduce the dependency of the domain indicator and the convolutional representation variables as much as possible. Since the mutual information measures the dependency, we minimize $I(\pi;\mathbf{z})$ as follows:

$$\min_{W_l} I(\pi;\mathbf{z}). \tag{22}$$

In this way, we hope that the learned representations are independent from the domains as much as possible so that it can be generalized to adapt to both domains.

To construct the learning framework for the domain adaptation problem based on the classwise deep CNN representation model, we combine the objects of equations (8) and (22) for the minimization problem:

$$\begin{aligned} \min_{W_l} \Bigg\{ o(W_l) &= \sum_{X_i\in\mathcal{X}_l} \left(\sum_{X_j\in\mathcal{N}_i^+} A_{ij'}^+ \left\|\mathbf{z}_i - \mathbf{z}_j\right\|_F^2 - \sum_{X_j\in\mathcal{N}_i^-} A_{ij'}^- \left\|\mathbf{z}_i - \mathbf{z}_{j'}\right\|_F^2 \right) \\ &\quad + C_1 I(\pi;\mathbf{z}) + C_2\|W\|_{lF}^2 \\ &= \sum_{X_i\in\mathcal{X}_l} \left(\sum_{X_j\in\mathcal{N}_i^+} A_{ij'}^+ \left\|\mathbf{z}_i - \mathbf{z}_{j'}\right\|_F^2 - \sum_{X_j\in\mathcal{N}_i^-} A_{ij'}^- \left\|\mathbf{z}_i - \mathbf{z}_{j'}\right\|_F^2 \right) \\ &\quad + \frac{C_1}{nh_i} \sum_{i:X_i\in\mathcal{X}} \left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{source}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad \cdot \log\left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{source}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad + \frac{C_1}{nh_i} \sum_{i:X_i\in\mathcal{X}} \left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{target}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad \cdot \log\left(\sum_{j:X_j\in\mathcal{N}_i,\pi_j=\text{target}} g(\mathbf{z}_i - \mathbf{z}_j) \right) \\ &\quad - \frac{C_1}{n} \sum_{i:X_i\in\mathcal{X}} \log\left(\sum_{j:X_j\in\mathcal{N}_i} g(\mathbf{z}_i - \mathbf{z}_j) \right) + C_2\|W_l\|_{lF}^2 \Bigg\}, \end{aligned} \tag{23}$$

where $\|W_l\|_F^2$ term is used to control the complexity of the model to prevent the overfitting problem and C_1 and C_2 are the tradeoff parameters. In the objective, the first term is a large margin corresponding term, while the second and third terms are corresponding to the entropies of location distribution of convolutional representations over the neighborhood specified by source and target domains. The fourth them is corresponding to the entropy of the overall location distribution of representations.

2.3. Optimization. It is difficult to solve the problem of equation (23) because the classwise representations $\mathbf{z}_i, i = 1, \ldots, n$ are the outputs of a deep CNN function, f_l, while it also defines the neighborhoods and affinities. To solve the problem of equation (23), we treat the representation $\mathbf{z}_i$ as slack variables and introduce the following optimization problem:

$$\begin{aligned}\min_{W_l,\mathbf{z}_1,\ldots,\mathbf{z}_n} \Bigg\{ & o(W_l, \mathbf{z}_1, \ldots, \mathbf{z}_n) = \sum_{X_i \in \mathcal{X}_l} \Bigg(\sum_{X_j \in \mathcal{N}_i^+} A_{ij'}^+ \left\| \mathbf{z}_i - \mathbf{z}_j \right\|_F^2 \\ & - \sum_{X_j \in \mathcal{N}_i^-} A_{ij'}^- \left\| \mathbf{z}_i - \mathbf{z}_{j'} \right\|_F^2 \Bigg) \\ & + \frac{C_1}{nh_i} \sum_{i:X_i \in \mathcal{X}} \Bigg(\sum_{j:X_j \in \mathcal{N}_i, \pi_j = \text{source}} g(\mathbf{z}_i - \mathbf{z}_j) \Bigg) \\ & \cdot \log \Bigg(\sum_{j:X_j \in \mathcal{N}_i, \pi_j = \text{source}} g(\mathbf{z}_i - \mathbf{z}_j) \Bigg) \\ & + \frac{C_1}{nh_i} \sum_{i:X_i \in \mathcal{X}} \Bigg(\sum_{j:X_j \in \mathcal{N}_i, \pi_j = \text{target}} g(\mathbf{z}_i - \mathbf{z}_j) \Bigg) \\ & \cdot \log \Bigg(\sum_{j:X_j \in \mathcal{N}_i, \pi_j = \text{target}} g(\mathbf{z}_i - \mathbf{z}_j) \Bigg) \\ & - \frac{C_1}{n} \sum_{i:X_i \in \mathcal{X}} \log \Bigg(\sum_{j:X_j \in \mathcal{N}_i} g(\mathbf{z}_i - \mathbf{z}_j) \Bigg) + C_2 \|W_l\|_F^2 \Bigg\} \\ \text{s.t.} \quad & \mathbf{z}_i = f_l(X_i; W_l), \ i = 1, \ldots, n. \end{aligned} \tag{24}$$

To solve this problem, we use the ADMM algorithm. Following ADMM, we have the following optimization problem:

$$\begin{aligned}\min_{W_l,\mathbf{z}_1,\ldots,\mathbf{z}_n} \Bigg\{ & o(W_l, \mathbf{z}_1, \ldots, \mathbf{z}_n) + \frac{\rho}{2} \sum_{i=1}^{n} \left\| f_l(X_i; W_l) - \mathbf{z}_i \right\|_F^2 \\ & + \sum_{i=1}^{n} \boldsymbol{\alpha}_i^\top \left(f_l(X_i; W_l) - \mathbf{z}_i \right) \Bigg\}, \end{aligned} \tag{25}$$

where $\boldsymbol{\alpha}_i$ is a dual variable for the constraint $\mathbf{z}_i = f_l(X_i; W_l)$ and ρ is its penalty parameter. We solve this problem by alternately updating the variables in an iterative algorithm.

2.3.1. Updating of $\mathbf{z}_i$. The updating of $\mathbf{z}_i$ is conducted by solving the following ionization problem:

$$\begin{aligned}\min_{\mathbf{z}_i} \Bigg\{ & o_1(\mathbf{z}_i) = o(\mathbf{z}_i) + \frac{\rho}{2} \left\| f_l(X_i; W_l) - \mathbf{z}_i \right\|_F^2 \\ & + \boldsymbol{\alpha}_i^\top \left(f_l(X_i; W_l) - \mathbf{z}_i \right) \Bigg\}, \end{aligned} \tag{26}$$

and we solve it by a gradient descent method:

$$\mathbf{z}_i \longleftarrow \mathbf{z}_i - \rho \nabla_{\mathbf{z}_i} o_1(\mathbf{z}_i), \tag{27}$$

where ρ is a descent step parameter and $\nabla_{\mathbf{z}_i} o_1(\mathbf{z}_i)$ is the gradient function regarding $\mathbf{z}_i$,

$$\nabla_{\mathbf{z}_i} o_1(\mathbf{z}_i) = \nabla_{\mathbf{z}_i} o(\mathbf{z}_i) - \rho \left(f_l(X_i; W_l) - \mathbf{z}_i \right) - \boldsymbol{\alpha}_i. \tag{28}$$

2.3.2. Updating of W_l. Updating of W_l is conducted by solving the following minimization problem:

$$\begin{aligned}\min_{W_l} \Bigg\{ & o_2(W_l) = C_1 \|W_l\|_F^2 + \frac{\rho}{2} \left\| f_l(X_i; W_l) - \mathbf{z}_i \right\|_F^2 \\ & + \boldsymbol{\alpha}_i^\top \left(f_l(X_i; W_l) - \mathbf{z}_i \right) \Bigg\}. \end{aligned} \tag{29}$$

We also use the backprorogation algorithm to solve this problem, based on the chain rule:

$$\begin{aligned} W_l & \longleftarrow W_l - \rho \nabla_{W_l} o_2(W_l), \\ \nabla_{W_l} o_2(W_l) & = 2C_1 W_l + \rho \left(f_l(X_i; W_l) - \mathbf{z}_i \right)^\top \\ & \quad \cdot \nabla_{W_l} f_l(X_i; W_l) + \boldsymbol{\alpha}_i^\top \nabla_{W_l} f_l(X_i; W_l). \end{aligned} \tag{30}$$

2.3.3. Updating of α_i. The dual variable is updated by gradient ascent:

$$\alpha_i \longleftarrow \alpha_i + \rho f_l(X_i; W_l). \tag{31}$$

2.4. Overall Learning Algorithm of MMITR. In this section, we give the overall iterative learning algorithm of the proposed minimum mutual information transfer representation (MMITR) method. In this algorithm, it has an updating strategy similar to the expectation-maximization (EM) algorithm. In each iteration, we firstly fix domain transfer representations to update the inter- and intraclass affinity metrics in an E-step and then fix the inter- and intraclass affinity metrics to update the CNN parameters and the representations in an M-step. The iterations are stopped until a maximum iteration number is reached or the objective value reaches a threshold. The overall algorithm is described in Algorithm 1.

2.5. Prediction of a New Data Point. When we have a new data point, X, to classify it, we calculate its classwise representation and corresponding margin regarding each class:

Input: Training set of L classes and unlabeled data points $\mathcal{X}_1 \cup \cdots \cup \mathcal{X}_L \cup \mathcal{X}_U$;
Input: Domain indicators of training points $\pi_1, \ldots, \pi_n$;
Input: Tradeoff parameters C_1 and C_2;
Input: Maximum number of iterations, η;
Input: Objective value threshold, ε.
Initialize iteration indicator $\iota = 1$.
Initialize model parameters and objective value $o^0 = 0$.
while $\iota \leq \eta$ or objective value $|o^\iota - o^{\iota-1}| \leq \varepsilon$ **do**
  E-step: Update the inter- and intraclass affinities for each data point, according to equations (5) and (6).
  M-step: Iterating the ADMM updating steps.
  for $t = 1, \ldots, T$ **do**
    Update the domain transfer representations by iterating the gradient descent steps of equation (27).
    Update the CNN model parameters by iterating the backprorogation steps of equation (30).
    Update the dual variables by iterating the gradient ascent steps of equation (31).
  end for
  $\iota = \iota + 1$.
end while
Output: W and $\mathbf{z}_1, \ldots, \mathbf{z}_n$.

Algorithm 1: Iterative learning algorithm of MMITR.

$$\begin{aligned} \mathbf{z} &= f_l(X; W_l), \\ m_l(X) &= \sum_{X_j \in \mathcal{N}_X^-} A_{ij'}^- \left\| \mathbf{z}_i - \mathbf{z}_{j'} \right\|_F^2 - \sum_{X_j \in \mathcal{X}_X^+} A_{ij'}^+ \left\| \mathbf{z}_i - \mathbf{z}_{j'} \right\|_F^2, \\ l &= 1, \ldots, L, \end{aligned} \tag{32}$$

where the intra- and interclass neighborhood and affinity are calculated according to the classwise representations. The new data point is assigned to a class which gives the maximum margin:

$$y^* = \arg\max_{l=1}^{L} m_l(X). \tag{33}$$

3. Experiments

3.1. Datasets. In our experiments, we use the following datasets as benchmark datasets:

Office-31: this dataset has 4,652 images of 31 classes. The images are from three different domains: Amazon (images downloaded from http://www.amazon.com), Webcam (photos taken by web camera), and DSLR (photos taken by digital SLR camera).

ImageCLEF-DA: this dataset is composed of images of 12 classes of four domains. Each domain is a unique database, including Caltech-256, ImageNet ILSVRC 2012, Pascal VOC 2012, and Bing.

Email spam: this dataset has email texts of spam and nonspam. The data are collected from three users, and each user has 2,500 emails. Each user is treated as a domain.

Extended Cohn-Kanade (CK+): this dataset is an image set for facial expression recognition. It has images of 123 subjects, and each subject is treated as a domain. This dataset has 593 videos in total, and for each video, there are about 20 frames. Each image of a face belongs to one of the 7 expression classes.

Amazon: this dataset is a text classification task dataset. The texts are from three different domains, and each domain is the review for products of books, DVD, and music. For each domain, there are 2,000 positive review texts and 2,000 negative review texts.

3.2. Experimental Setting. In this experiment, we use each domain of a dataset as a target domain in turn and the remaining domains as the source domains. For each test domain, we use the leave-one-out protocol to split it into a training set and a test set. Each data point of a target domain is used as a test data point in turn, and the remaining data points are combined to form a training set. The training set of the target domain is randomly split to an unlabeled set and a labeled set, with equal size. The data points of the source domains are always treated as labeled in our setting. Our algorithm is performed over the training set to learn the parameters of the classwise representation model and the domain-independent representations and then used to classify the test data points. The classification accuracy is used to evaluate the performance of the algorithm.

3.3. Results. In our experiment, we first study the properties of the algorithm experimentally, including its sensitivity to the tradeoff parameters and its convergence property to iteration numbers. Then, we compare its performance to state-of-the-art domain transfer learning algorithms.

3.3.1. Algorithm Property Evaluation

(1) Sensitive to Tradeoff Parameters. There are two tradeoff parameters in our algorithm: C_1 and C_2. They are the weights of the mutual information term and the complexity reduction term in our object. We plot the accuracy of our algorithm regarding different values of C_1, as shown in Figure 2. From this figure, we observe that the accuracy

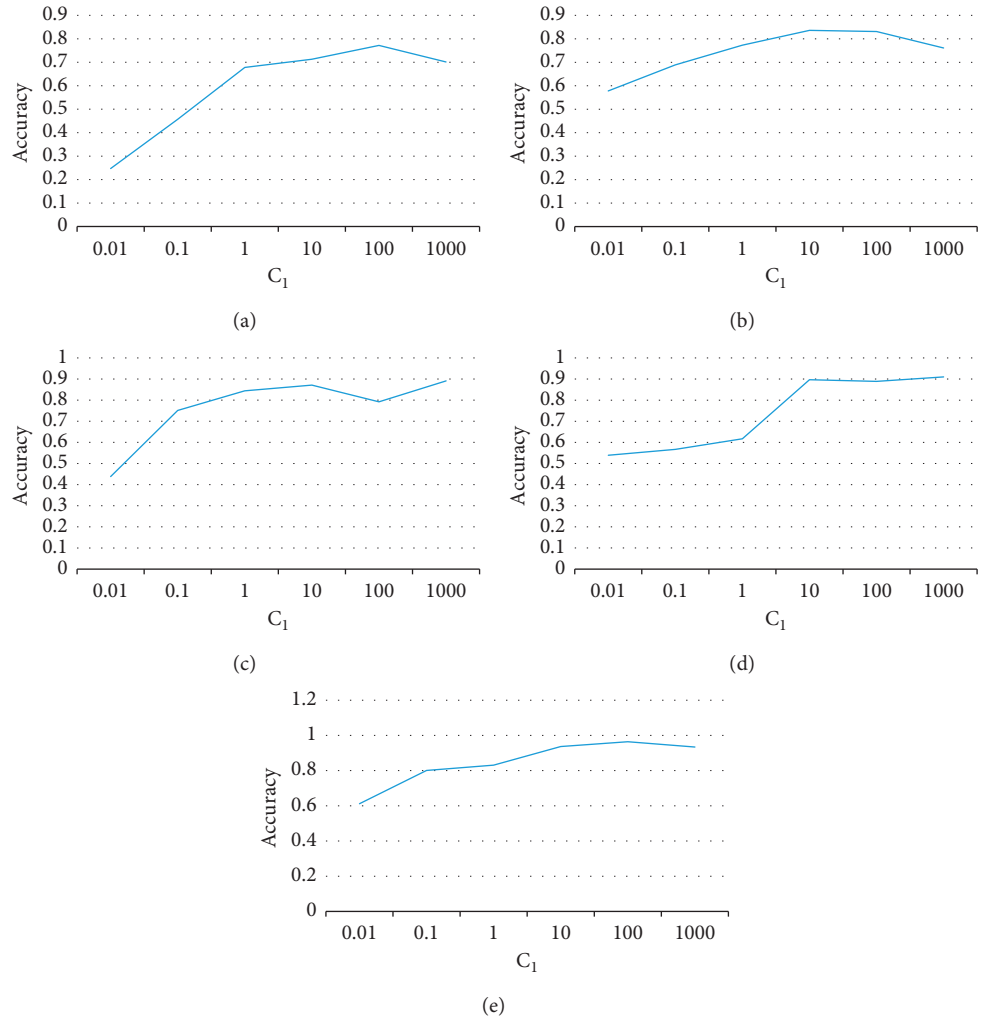


Figure 2: Tradeoff parameter sensitivity study of C_1. (a) Office-31. (b) ImageCLEF-DA. (c) Email spam. (d) CK+. (e) Amazon.

improves in most cases when the value of C_1 increases. Since C_1 is the weight of the mutual information term to measure how dependent the representation is from the domain, this indicates that a more domain-independent representation helps the classification in the target domain. Actually, the more independent the representation is from the domain, the better the data of different domains are merged. Thus, the source domain can benefit the learning problem in the target domain more. This phenomenon is even more obvious in the CK + dataset; when C_1 grows from 1 to 10, the accuracy is boosted significantly. This is a piece of strong evidence how the minimum mutual information improves the transfer learning.

The accuracy curves of classification with different values of C_2 are shown in Figure 3. From this figure, we can see that the proposed algorithm is stable to the change of C_2. Since the algorithm is not sensitive to the change of C_2, the tuning of this parameter will be easy for a specific dataset. One only exception is the case when C_2 varies between 1 and 10, the accuracy changes dramatically.

(2) Convergence Analysis. Since our algorithm is an iterative algorithm, it is critical to know when to stop the iterations. We study the convergence of the algorithm by plotting the accuracy over different datasets with varying numbers of iterations in Figure 4. According to the curves in the figure, in most datasets, the algorithm gives a better accuracy when the iteration number grows and then becomes stable after about 100 iterations. For the email spam dataset, the algorithm converges at 50 iterations.

Remark. To solve a minimization problem in our algorithm, we employed the ADMM algorithm. To verify if the ADMM algorithm solves the optimization of the minimization problem effectively, we plot the object values of the learning problem with increasing numbers of iterations in Figure 5. As we can see from the curves, the object value decreases stably as the number of iterations increases, until it reaches a convergency, and then the changing of object values becomes small. This is a strong evidence that ADMM algorithm solves the optimization problems effectively (Table 1).

3.3.2. Comparison to State of the Arts. We compare our algorithm, MMITR, to several state-of-the-art transfer learning algorithms, including the Deep Adaptation Network (DAN) [4], Selective Transfer Machine (STM) [1],

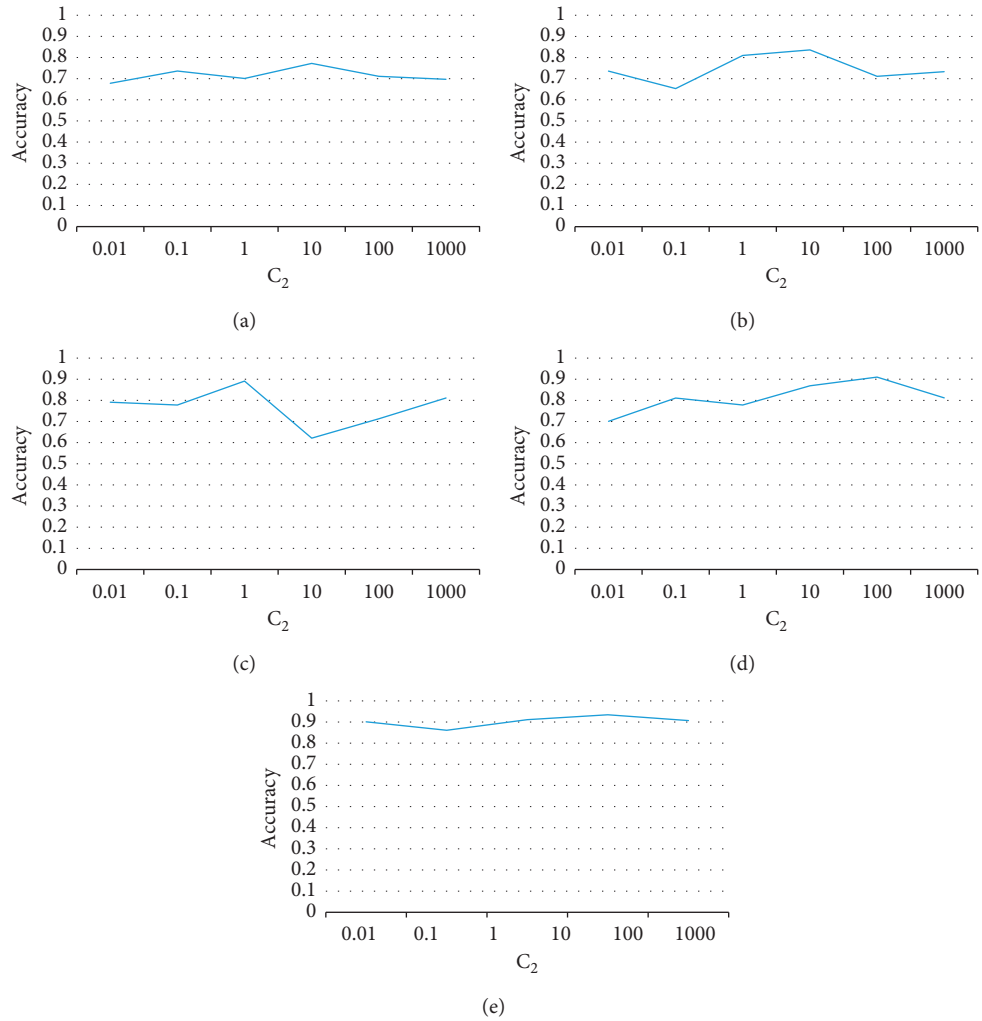


Figure 3: Tradeoff parameter sensitivity study of C_2. (a) Office-31. (b) ImageCLEF-DA. (c) Email spam. (d) CK+. (e) Amazon.

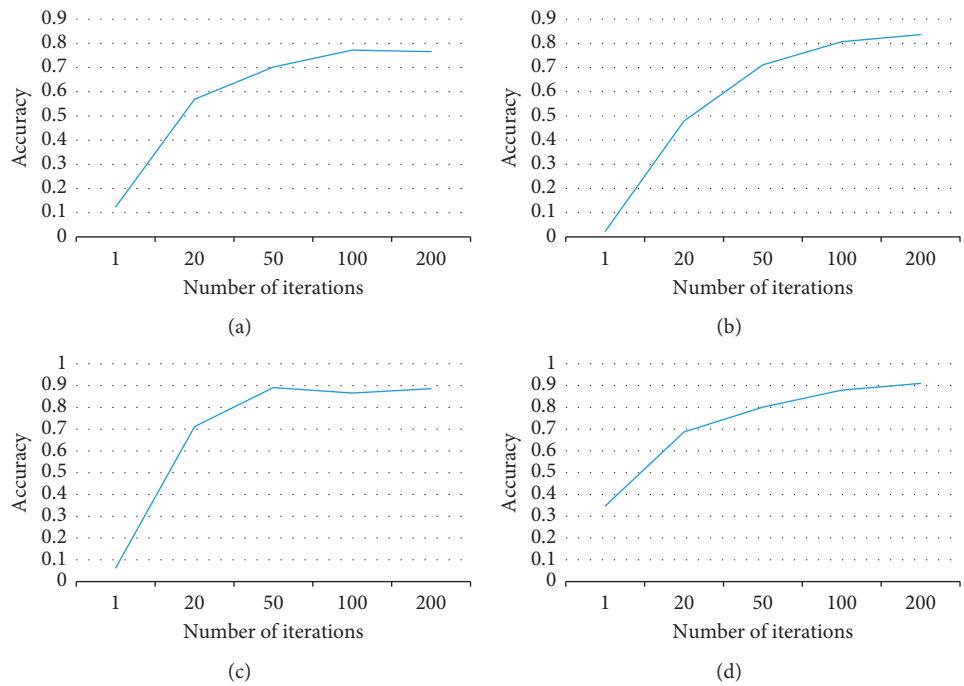
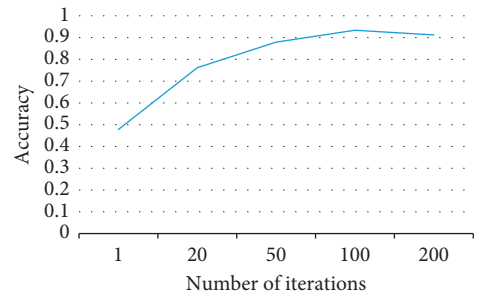


Figure 4: Continued.



(e)

Figure 4: Convergence analysis. (a) Office-31. (b) ImageCLEF-DA. (c) Email spam. (d) CK+. (e) Amazon.

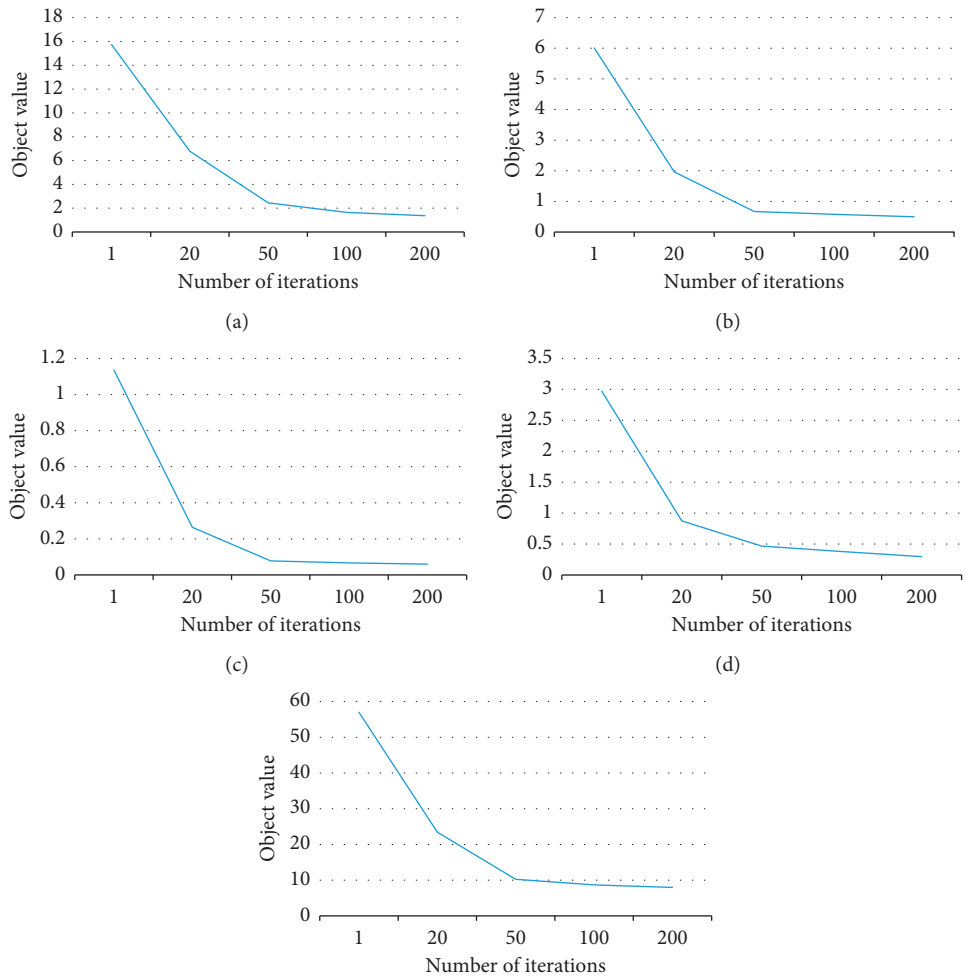


Figure 5: Objective value curve of ADMM algorithm. (a) Office-31. (b) ImageCLEF-DA. (c) Email spam. (d) CK+. (e) Amazon.

Semisupervised Kernel Matching Domain Adaptation (SSKMDA) [6], and Domain Transfer Multiple Kernel Learning (DTMKL) [2]. In Table 2, we have provided a detailed list of algorithms compared in the experiment, regarding the aspect of data representation components and domain matching criteria.

The comparison of accuracy results is given in Figure 6. In this figure, we can observe that the proposed method outperforms the other methods in four experiments out of five. In experiments over three datasets (Office-31, ImageCLEF-DA, and Amazon), our algorithm outputs the second best method, DAN, by a large margin. For the dataset of CK+, DAN outperforms our method by a slight amount. Both DAN and our method MMITR are based on deep learning model, but our method tries to learn domain-independent deep representations, while DAN tries to

Table 1: Definitions of symbols.

Symbol	Definition
X_i	i-th data point
$\mathcal{X}_l$	Set of data points of the l-th class
$\mathcal{X}_U$	Set of unlabeled data points
$\mathbf{z}$	Domain transfer representation of a data point
W_l	Parameters of the CNN model of the l-th class
N_i^+	Intraclass neighborhood of the i-th data point
N_i^-	Interclass neighborhood of the i-th data point
A_{ij}^+	Intraclass affinity between the i-th and j-th data points
$A_{ij'}^+$	Interclass affinity between the i-th and j'-th data points
π_i	Domain indicator of the i-th data point
ω_{ij}	Soft weight of the the j-th data point in the i-th neighborhood

Table 2: Compared methods.

Method	Data representation	Domain matching criterion
MMITR	CNN	Mutual information minimization
DAN	CNN	Maximum mean discrepancy (MMD)
STM	—	Kernel mean-matching (KMM)
SSKMDA	Multikernel learning	Hilbert schmidt independence criterion (HSIC)
DTMKL	Multikernel learning	MMD

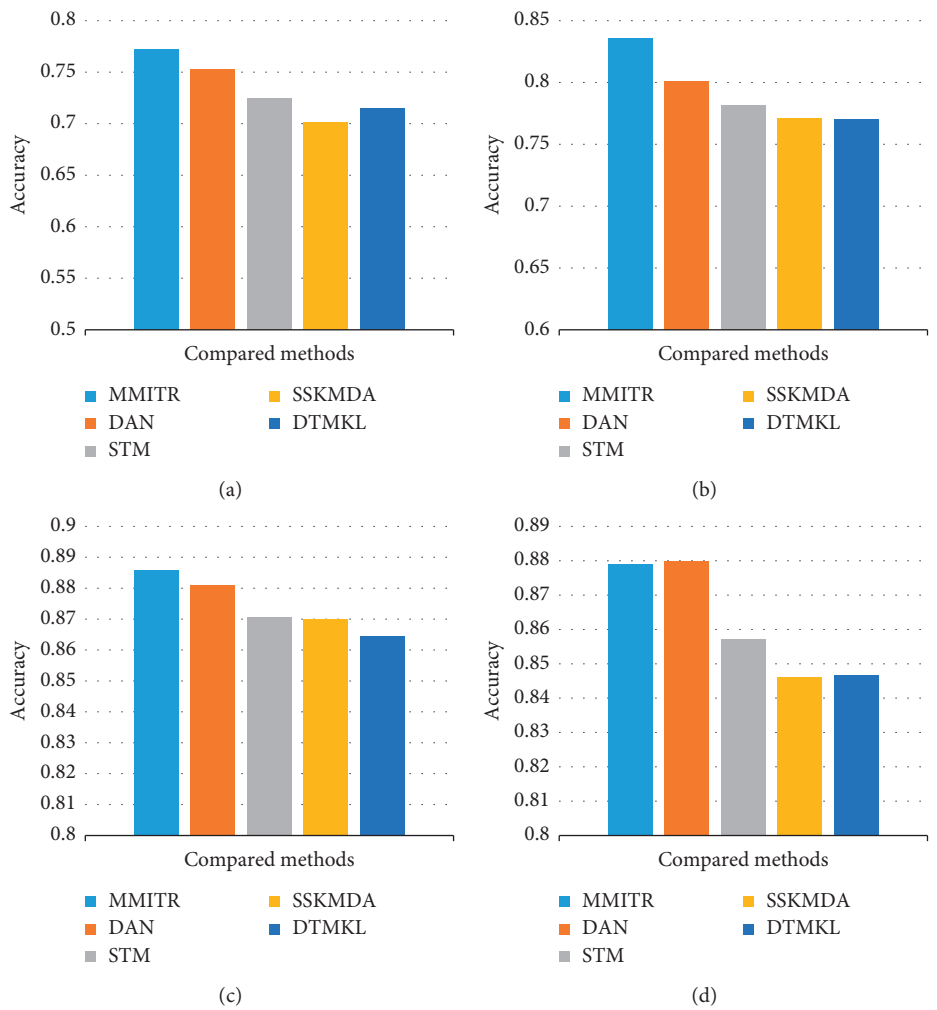
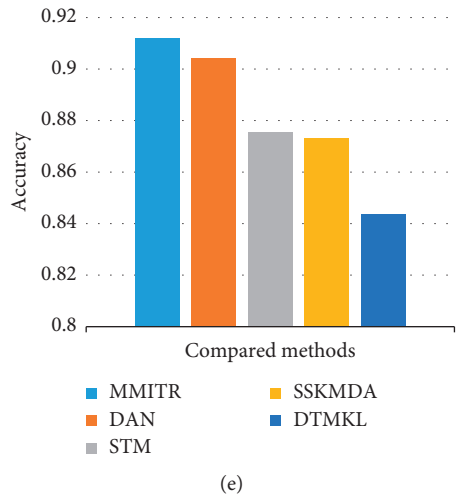


Figure 6: Continued.



(e)

Figure 6: Comparison of accuracy of state of the arts. (a) Office-31. (b) ImageCLEF-DA. (c) Email spam. (d) CK+. (e) Amazon.

learn a deep learning model to represent the data points so that mean of representations of source domain and target domain can be similar to each other. According to the results, MMITR outperforms DAN in most cases; we conclude that domain-independent deep representation is more suitable for domain transfer learning than domain-mean matched representation. The other methods are also based on mean-matching of domain transfer representations, but using a shallow model instead of a deep model; thus, they are not able to explore hierarchal deep features. This again verifies the effectiveness of the deep model.

Remark. The conditions of the methods of the results reported in 6 are described in details as follows. For the DAN algorithm, it has two hyperparameters: the MMD Penalty λ and the Entropy Penalty γ, and we set their values to 1 and 0.1, respectively. For the STM algorithm, there are two hyperparameters: C for the tradeoff between maximal margin and training loss and λ for the tradeoff between the SVM empirical risk and the domain mismatch loss. In this experiment, we set both their values to 1. For the SSKMDA algorithm, it has five tradeoff parameters between the model components: $\mu, \eta, \beta, \gamma_s, \gamma_t$, and their value setting in our experiment are 10, 2, 0.1, 0.1, and 1, respectively. The DTMKL algorithm has only one hyperparameter: the regularization parameter C, and we set it to 0.5 in the experiments. For our algorithm MMITR, it has two tradeoff parameters, C_1 and C_2; for each benchmark dataset, we report the best results among the results obtained by using different values of C_1 and C_2.

4. Conclusions and Future Works

In this paper, we proposed a novel framework for transfer learning. Not like the traditional transfer learning which tries to match the representations of source domain and target domain, we proposed to learn domain-independent representations. We argue to measure the dependency of learned deep representations and domain by mutual information and learn the domain-independent deep representations by minimizing the mutual information. We also proposed a practical estimation method for the mutual information between domain and deep representations. A classwise deep representation neural network work is trained under this framework and used to classify new data points. Experiments over benchmark datasets for transfer learning verify the effectiveness of the proposed method.

Remark. The new concept proposed in this paper is a novel domain transfer learning framework which minimizes the mutual information between the domain transfer representations and the domain indicators so that the gaps among domains can be effectively leveraged and a common representation space is learned. The new method developed in this is a novel iterative learning algorithm to learn the domain transfer representations based on CNN models.

Data Availability

All the datasets used in this paper are publicly accessed.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] W.-S. Chu, F. D. L. Torre, and J. F. Cohn, "Selective transfer machine for personalized facial expression analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 3, pp. 529–545, 2017.

[2] 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.

[3] I. Hossain, A. Khosravi, I. Hettiarachchi, and S. Nahavandi, "Multiclass informative instance transfer learning framework for motor imagery-based brain-computer interface,"

Computational Intelligence and Neuroscience, vol. 2018, Article ID 6323414, 12 pages, 2018.

[4] 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*, 2018.

[5] A. Mousavi, B. Nadjar Araabi, and M. Nili Ahmadabadi, "Context transfer in reinforcement learning using action-value functions," *Computational Intelligence and Neuroscience*, vol. 2014, Article ID 428567, 10 pages, 2014.

[6] 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, 2015.

[7] S. Ding, L. Guo, and Y. Hou, "Extreme learning machine with kernel model based on deep learning," *Neural Computing and Applications*, vol. 28, no. 8, pp. 1975–1984, 2017.

[8] X. Dong, F. Wu, and X. Y. Jing, "Semi-supervised multiple kernel intact discriminant space learning for image recognition," *Neural Computing and Applications*, pp. 1–18, 2018.

[9] X. Li, W. Mao, and W. Jiang, "Multiple-kernel-learning-based extreme learning machine for classification design," *Neural Computing and Applications*, vol. 27, no. 1, pp. 175–184, 2016.

[10] F. Lin, J. Wang, N. Zhang, J. Xiahou, and N. McDonald, "Multi-kernel learning for multivariate performance measures optimization," *Neural Computing and Applications*, vol. 28, no. 8, pp. 2075–2087, 2017.

[11] J. Dhanaraj and A. Kumar Sangaiah, "Elephant detection using boundary sense deep learning (BSDL) architecture," *Journal of Experimental and Theoretical Artificial Intelligence*, pp. 1–16, 2018.

[12] Y. Geng, G. Zhang, W. Li et al., "A novel image tag completion method based on convolutional neural transformation," *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10614 LNCS*, pp. 539–546, Springer, Berlin, Germany, 2017.

[13] P. Guha, T. Bhatnagar, I. Pal, U. Kamboj, and S. Mishra, "Prediction of properties of wheat dough using intelligent deep belief networks," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 29, no. 6, pp. 1283–1296, 2017.

[14] V. Gurupur, S. Kulkarni, X. Liu, U. Desai, and A. Nasir, "Analysing the power of deep learning techniques over the traditional methods using medicare utilisation and provider data," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 31, no. 1, pp. 99–115, 2018.

[15] C. Tong, L. Medsker, L. Cheng, A. Jafari, and X. Wang, "Introduction to the special issue on deep learning for biomedical and healthcare applications," *Neural Computing and Applications*, vol. 30, no. 7, pp. 2015-2016, 2018.

[16] W. Yuan, C. Li, D. Guan, G. Han, and A. M. Khattak, "Socialized healthcare service recommendation using deep learning," *Neural Computing and Applications*, vol. 30, no. 7, pp. 2071–2082, 2018.

[17] F. Zeng, S. Hu, and K. Xiao, "Research on partial fingerprint recognition algorithm based on deep learning," *Neural Computing and Applications*, pp. 1–10, 2018.

[18] G. Zhang, G. Liang, W. Li et al., "Learning convolutional ranking-score function by query preference regularization," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10585 LNCS*, pp. 1–8, Springer, Berlin, Germany, 2017.

[19] G. Zhang, G. Liang, F. Su, F. Qu, and J. Y. Wang, "Cross-domain attribute representation based on convolutional neural network," in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 10956 LNAI*, pp. 134–142, Springer, Berlin, Germany, 2018.

[20] H. Zhang, Y. Ji, W. Huang, and L. Liu, "Sitcom-star-based clothing retrieval for video advertising: a deep learning framework," *Neural Computing and Applications*, pp. 1–20, 2018.

[21] C. Dorffer, M. Puigt, G. Delmaire, and G. Roussel, "Informed nonnegative matrix factorization methods for mobile sensor network calibration," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 4, pp. 667–682, 2018.

[22] S. Huang, P. Zhao, Y. Ren, T. Li, and Z. Xu, "Self-paced and soft-weighted nonnegative matrix factorization for data representation," *Knowledge-Based Systems*, vol. 164, pp. 29–37, 2019.

[23] X. Peng, D. Chen, and D. Xu, "Semi-supervised least squares nonnegative matrix factorization and graph-based extension," *Neurocomputing*, vol. 320, pp. 98–111, 2018.

[24] W. Ye, H. Wang, S. Yan, T. Li, and Y. Yang, "Nonnegative matrix factorization for clustering ensemble based on dark knowledge," *Knowledge-Based Systems*, vol. 163, pp. 624–631, 2019.

[25] W. Chu, H. Xue, C. Yao, and D. Cai, "Sparse coding guided spatiotemporal feature learning for abnormal event detection in large videos," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 246–255, 2019.

[26] Z. Ji, Y. Ma, Y. Pang, and X. Li, "Query-aware sparse coding for web multi-video summarization," *Information Sciences*, vol. 478, pp. 152–166, 2019.

[27] M. Kalluri, M. Jiang, N. Ling, J. Zheng, and P. Zhang, "Adaptive RD optimal sparse coding with quantization for image compression," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 39–50, 2019.

[28] J. Tong, Y. Zhao, P. Zhang, L. Chen, and L. Jiang, "MRI brain tumor segmentation based on texture features and kernel sparse coding," *Biomedical Signal Processing and Control*, vol. 47, pp. 387–392, 2019.