Hindawi
Security and Communication Networks
Volume 2022, Article ID 3234078, 12 pages
https://doi.org/10.1155/2022/3234078

Research Article

Multisource Mobile Transfer Learning Algorithm Based on Dynamic Model Compression

Peng Gao,[1,2] **Jingmei Li**,[1] **and Changhong Ding**[3]

[1]*College of Computer Science and Technology, Harbin Engineering University, Harbin, China*
[2]*Converged Media Technology Department, Heilongjiang Broadcasting Station, Harbin, China*
[3]*Heilongjiang University of Chinese Medicine, Harbin, China*

Correspondence should be addressed to Peng Gao; gaopeng1979@hrbeu.edu.cn

Received 13 January 2022; Revised 12 February 2022; Accepted 17 February 2022; Published 16 March 2022

Academic Editor: Zhen Wang

With the development of the Internet of Things, the application of computer vision on mobile phones is becoming more and more extensive and people have higher and higher requirements for the timeliness of the recognition results returned and the processing capabilities of the mobile phone for image recognition. However, the processing capability and storage capability of the user terminal equipment cannot meet the needs of identifying and storing a large number of pictures, and the data transmission process will cause high energy consumption of the terminal equipment. At the same time, multisource deep transfer learning has outstanding performance in computer vision and image classification. However, due to the huge amount of calculation of the deep network model, it is impossible to use the existing excellent network model to realize image recognition and classification on the mobile terminal. In order to solve the abovementioned problems, we propose a multisource mobile transfer learning algorithm based on dynamic model compression, this algorithm considers the realization of multisource transfer learning computing in the case of multiple mobile device computing source domains, and the method also guarantees data privacy and security for each device (origin domain). Meanwhile, extensive experiments show that our method can achieve remarkable results in popular image classification datasets.

1. Introduction

Machine learning can achieve good results in computer vision, often based on the following assumptions: there are enough data samples in the training dataset to train a high-precision classifier and training data and test data come from the same feature space and the same distribution. However, obtaining sufficient labeled data for practical applications is often expensive and time-consuming. In this case, transfer learning [1] is a promising approach. It transfers knowledge from the tagged source domain to the target domain. At the same time, the emergence of convolutional neural networks has also accelerated the technical level of transfer learning models. Transfer learning generally assumes that training and testing data come from similar but different distributions [2]. For example, images of objects taken at different angles, backgrounds and lighting may result in different edge or conditional distributions. By observation, existing transfer learning methods mainly focus on distribution adaptation to alleviate the domain gap through distribution adaptation. For example, several unsupervised transfer learning methods [3–5] incorporate maximum mean difference loss into neural networks to reduce domain differences; other models introduce different learning modes to align source and target domains, including aligning second-order correlations [6, 7]. In practical applications, we are often faced with multiple source domains. Common multisource transfer learning [8] maps the data of these two domains into a common feature space, which describes the invariant features of the source and target domains by minimizing the difference in domain distribution [5, 9–12]. With the development of deep learning, many scholars have proposed transfer learning models based on deep learning [13–17].

At present, the global Internet of Things has entered the third wave of development. The mobile terminal generates a large amount of data. In the traditional computing architecture, the data needs to be centrally transmitted to the cloud for processing, which undoubtedly increases the network load and data transmission time. Therefore, mobile computing came into being. Processing deep learning on mobile devices has natural advantages over cloud computing. The entire workflow does not need to upload data to the cloud, which can run offline while avoiding the privacy risks caused by data transmission and off-site storage in the cloud computing process. Although deep learning models have excellent performance, the convolutional layers in CNNs consume a lot of computational and energy resources, posing severe challenges to devices. AlexNet [19], VGG [20], GoogleNet [21], and ResNet [22] are all excellent algorithm models of ILSVRC, but these models usually have tens of millions of parameters and require hundreds of megabytes or even 1G of memory. For multisource learning, multiple boxing neural networks are usually required to work together, so deploying multisource deep learning models on mobile devices is a very serious challenge, regardless of the amount of computation and memory overhead. GoogleNet and ResNet also have to face this problem.

Deploying multisource transfer learning on mobile devices faces two challenges: (1) How to effectively use mobile devices to deploy transfer learning models to mobile device side. (2) The model and calculation volume of multisource transfer learning are often very large and how to use the advantages of the cloud and mobile terminals to achieve the purpose of optimizing computing.

In this paper, a novel multisource mobile transfer learning algorithm based on dynamic model compression is proposed by combining the advantages of mobile computing, convolutional neural networks, and multisource transfer learning.(-Multi-source Mobile Transfer Learning Algorithm Based on Dynamic Model Compression, MMTLDMC). MMTLDMC first performs BN pruning on the convolutional neural network. The classification loss and MMD loss is then computed on the device side (source domain). The classifier alignment loss is then calculated server-side. Then, the parameters are updated by minimizing the objective function. Finally, the model results are obtained, and the data classification is completed.

Compared with previous work, the contributions of this work are as follows:

(1) Different from the previous multisource transfer learning algorithms, we consider the needs of mobile computing and propose a new multi-source transfer learning algorithm MMTLDMC algorithm, which combines the advantages of both to accelerate the multisource transfer learning model.

(2) Use data on a device as a source domain. For samples on multiple devices, calculations are only performed on the device side, and the server and device side only synchronize parameters and models, taking into account the security of device data.

(3) Experiments on real datasets show that the proposed algorithm outperforms or at least comparable to state-of-the-art benchmark algorithms in classification accuracy. At the same time, the speed is much better than the existing transfer learning algorithm.

The rest of the paper is organized as follows: Section 2 reviews related work on convolutional neural network pruning, mobile computing, and multisource transfer learning; In Section 3, a multisource mobile transfer learning algorithm based on dynamic model compression is proposed; Section 4 verifies the effectiveness of the algorithm on SVHN, USPS, MINIST, Office-31, Caltech-256, and DomainNet; Section 5 summarizes the main work of this paper.

2. Brief Review of Related Work

2.1. Convolutional Neural Network Pruning and Mobile Computing. The powerful feature sampling performance of convolutional neural networks comes from a large number of parameters and a complex multilayer structure. Modern convolutional neural networks are also deepening and widening the classical structure to improve the accuracy. For example, VGG [20] increases the number of layers from 8 to 19 on the basis of AlexNet [21]; GoogleNet [22] is increased to 22 layers, adopts inception structure design, and uses average pooling to replace the full connection layer. ResNet [23] only learns the relationship between the residual and the input through the residual identity mapping, and builds a model that is easier to optimize. The depth has also developed from 152 layers of the standard to thousands of layers. The traditional pruning method is divided into three steps: baseline training, pruning, and fine tuning. In terms of pruning granularity, it can be divided into two categories: unstructured pruning [24] and structured pruning. Unstructured pruning means directly pruning individual weights. The weight matrix formed is sparse, which is not easy to achieve the compression and acceleration effect of general and easy deployment. Structured pruning refers to pruning for convolution core [25], channel [26, 27] or layer [28]. In the direction of convolution kernel pruning, literature [29] improved the pruning strategy based on convolution kernel weight ranking, and adopted the sum of the absolute values of the regular term L1 of convolution kernel as the pruning weight. Layer pruning needs to cut the complete layer with a poor flexibility and high precision risk. It is generally used to cut the deep network structure [30]. Channel pruning is one of the most studied and widely used structural pruning methods. Reference [31] proposed a channel pruning method for BN layer for model compression. By using the weight of BN layer to evaluate the score of input channels, the author sets a threshold to filter out the channels with low scores. When connecting, the neurons of these channels with too small scores do not participate in the connection, and then prune layer by layer. The pruning technology of convolutional neural network reduces the actual deployment of the model in the mobile terminal. Figure 1 shows the pruning model of BN channel factor.

2.2. Multisource Transfer Learning. Multisource transfer learning as a research direction of transfer learning has very important practical values. In the process of real life and practical application, there are often multiple source domains. Although each source domain has a different

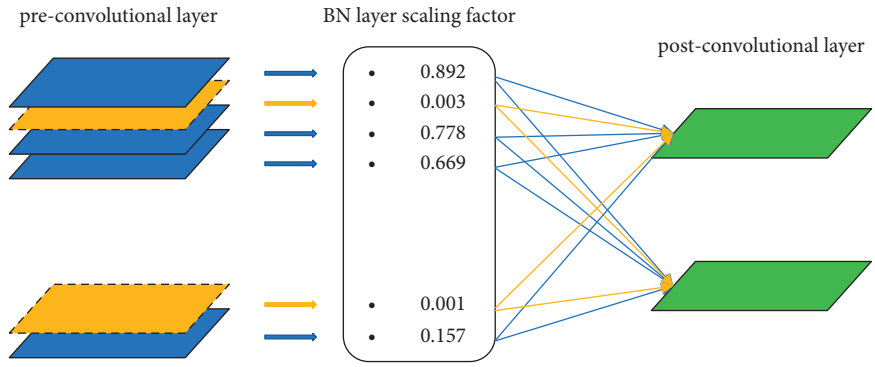

Figure 1: Pruning model of the BN channel factor.

similarity with the target, these source domains can still be used for knowledge transfer. Moreover, multisource transfer learning contains more knowledge, which can make the effect of the model better. At the same time, transfer learning also has a theoretical basis. Crammer [28] first proposed the expected loss boundary condition of multisource transfer learning. Later, Mansour [29] proved that the distribution weighted combination rule can reduce the instantaneous function between the source domain and the target domain. Ben-David [30] gave two learning boundaries of minimizing empirical risk by introducing the distance between the target domain and the source domain.

In recent years, a lot of work has been centered on multisource transfer learning and deep learning. Zhuang et al. [31] proposed the Deep Cocktail Network (DCTN), which uses a single domain discriminator and a classifier for each source domain and target domain. The domain discriminator is used to align the feature distribution, and the classifier outputs the predicted probability distribution. Based on the output of the domain discriminator, DCTN designed a method of voting by multiple classifiers. Carmmer et al. [32] proposed a moment matching multisource domain adaptation (M^3SDA) method, which not only considers the alignment between the source domain and the target domain but also aligns the feature distribution of different source domains. Zhu proposed a framework named aligning domain-specific distribution and classifier for cross-domain classification from multiple sources (MFSAN) [33]. However, the current deep multisource transfer learning algorithms often only consider marginal probability distribution or consider the marginal probability distribution and the conditional probability distribution separately. In this paper, multisource transfer learning is based on balanced distribution adaptation, which considers the joint probability distribution to improve the accuracy of the algorithm. However, the multisource deep transfer learning algorithm is not combined with mobile computing because of its large amount of computation. This paper proposes a multisource mobile transfer learning algorithm based on dynamic model compression, which aims to train the multisource transfer model on the mobile terminal.

2.3. Problem. In mobile computing or edge computing, we can take the data collected by each device as a source domain, but for privacy and security reasons, the data of each device cannot be completely uploaded to the server for model construction. According to this restriction, we redefine multisource transfer learning. In multisource transfer learning, there are N source domains(clients), and their labeled sample data can be represented as $X_C^{s_i} = \left\{\left(x_j^{s_i}, y_j^{s_i}, e_j^{s_i}\right)\right\}_{j=1}^{N_i}$, where $\left\{\left(x_j^{s_i}\right)\right\}_{j=1}^{N_i}$ represents the i-th sample data in the j-th source domain, and $\left\{\left(y_j^{s_i}\right)\right\}_{j=1}^{N_i}$ represents the i-th source domain in the j-th source domain, $\left\{\left(e_j^{s_i}\right)\right\}_{j=1}^{N_i} \in \{0,1\}$ represents whether the j-th sample tag in the i-th source domain can be synchronized to the server, 0 indicates that it cannot be synchronized, and 1 indicates that it can be synchronized. The joint probability distribution of N different domain can be expressed as $\left\{P^{s_i}(x,y)\right\}_{i=1}^{N_i}$, where the marginal probability can be expressed as $\left\{P^{s_i}(x)\right\}_{i=1}^{N_i}$, and the conditional probability can be expressed as $\left\{P^{s_i}(y|x)\right\}_{i=1}^{N_i}$. Similarly, we give the definition of the target domain, the sample of the target domain can be expressed as $X^t = \left\{\left(x_j^t\right)\right\}_{j=1}^{N_t}$, and the probability distribution can be expressed as $P^t(x,y)$. This paper mainly considers the problem that the source domain data cannot be shared with the server, but the target domain data can be shared with the server which means $X_C^{s_i} = \left\{\left(x_j^{s_i}, y_j^{s_i}, 0\right)\right\}_{j=1}^{N_i}$, $X^t = \left\{\left(x_j^t, 1\right)\right\}_{j=1}^{N_t}$.

In recent years, some papers have defined the objective function of multisource deep transfer learning. They first map all domains to the same target space, then use the common domain invariant representation in the common feature space for learning all domains. Zhu [33] gave a definition of the loss function:

$$\min_{F,C} \sum_{i=1}^{N} \mathrm{E}_{x\sim X_{s_i}} J\left(C_i\left(H_i\left(F\left(x_j^{s_i}\right)\right), y_j^{s_i}\right) + \lambda \sum_{i=1}^{N} \hat{D}\left(H_i\left(F\left(X_{s_i}\right), F\left(X_t\right)\right)\right) + \gamma L_{di\ sc}. \tag{1}$$

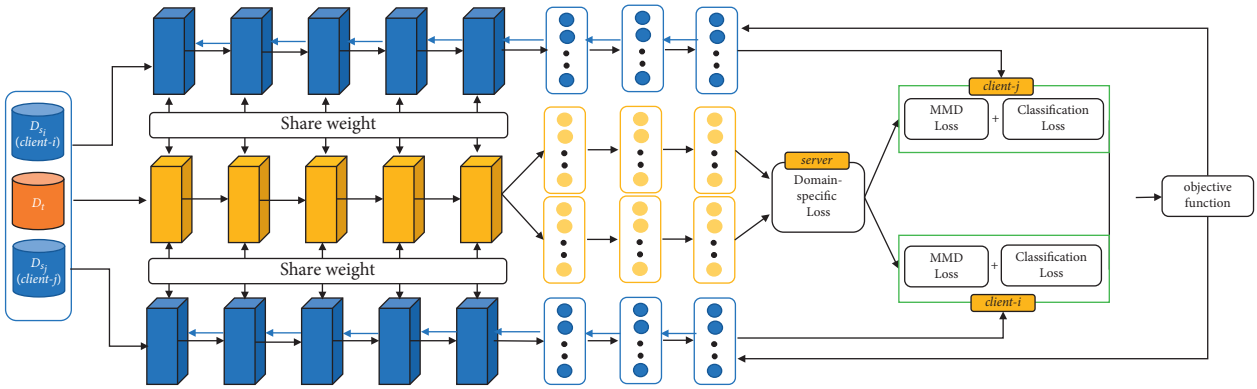

Figure 2: Framework of multisource mobile transfer learning algorithm based on dynamic model compression (MMTLDMC). Our model consists of two parts: (i) sever: calculating loss function and updating parameters and (ii) client: calculating MMD loss and Classification loss.

The first term represents the loss of the classification function, the general classification loss is the cross-entropy loss, and the second term represents the statistical measurement of the source domain and the target domain. Nowadays, the commonly used metrics are MMD [34, 35] loss, CORAL loss [5], and confusion loss [14, 15]. The third item, Zhu defines it as a specific difference loss. At present, there is no relevant method to consider the combination of multisource transfer learning and mobile computing. Most of these algorithms have high requirements for GPU and CPU of computing devices and cannot be calculated on the mobile end. At the same time, due to the concept of user privacy, it is very unfriendly to synchronize user data to the cloud. Therefore, how to realize the calculation of a large-scale model on the mobile terminal, such as multisource migration learning, is our concern.

In order to solve the above problems, we first map multiple source domains and target domains into the same subspace, and then prune the BN channel of the convolutional neural network in the space to reduce the amount of computation at the mobile end. Then, in order to ensure data security, we calculate the classification loss and MMD loss at the mobile terminal. Then, the calculated loss is synchronized to the server, and the server calculates the loss of the alignment classifier. Finally, we optimize the parameters by minimizing the loss function to obtain the optimal solution.

3. Multisource Mobile Transfer Learning Algorithm Based on Dynamic Model Compression

In order to solve the impact of mobile computing with memory and CPU on the existing multisource transfer learning algorithms. In this chapter, we introduce the multisource mobile transfer learning algorithm based on dynamic model compression. We use the model compression strategy proposed in literature [31] to compress the deep learning model.

3.1. Algorithm Structure. Our algorithm structure consists of two parts. The first part is the server side, including the calculation of minimization loss function and parameter update; The second part is the client (source domain), which mainly calculates the loss function and updates the parameters according to the results of server, as shown in Figure 2.

3.1.1. Preparation-BN Channel Pruning. We extract the source domain features into the same feature space in the form of shared parameters. Before extraction, we prune the BN channel of the feature extraction network according to the previously trained migration learning model. The purpose here is to reduce the parameters of the mobile terminal network model. We prune according to the literature [31], and the main method is to directly use the gamma parameters of BN layer for pruning evaluation.

The loss function after introducing the channel factor of BN layer is as follows:

$$L_{bn} == \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{\gamma \in \Gamma} g(\gamma), \tag{2}$$

where (x, y) represents training data and labels, and W represents parameters. The first half is CNN's original loss function, and the second half is the introduced penalty term. λ is the sparsity factor used to balance the formula term.

The mean and variance of BN activation values are calculated as follows:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i,$$
$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu)^2. \tag{3}$$

Then, the calculation process of BN layer output can be expressed as

$$\widehat{z} = \frac{z_{in} - \mu}{\sqrt{\sigma^2 + \varepsilon}},$$
$$z_{\text{out}} = \gamma \widehat{z} + \beta. \tag{4}$$

Among them λ And β Are BN layer linear transformation parameters that can be trained. We refer to the method given in reference [31] to pretrain and fine tune the

existing RESNET network. The initial convolution neural network parameters are formed.

Client: In order to ensure the data security of the client (source domain), we separate the loss computation of the multisource transfer learning objective function proposed in [17]. Put the classification loss and MMD loss into the client side to calculate. A set of images were given: x_j^s from source domain $X_C^s = \{(x_j^s, y_j^s)\}$ and a set of images x^t from target domain $\{(x_j^t)\}$. The features of these specific fields are mapped to the same feature space through a common feature extractor. Some are specifically expressed as source domain mapping feature $f(x_j^s)$, target domain mapping characteristics $f(x^t)$. Therefore, we can get N feature extractor $h(\cdot)$ corresponding to specific source domains $\{(x_j^s, y_j^s)\}$. We use the pruned convolutional neural network as our classifier, we define C_i as the classifier of N source domains. According to experience, our classification loss is crossing entropy loss, and the loss function is $J(\cdot,\cdot)$.

Server: From Figure 2, we can see that the server needs to calculate the alignment loss of domain-specific classifiers proposed in the literature [17]. At the same time, on server side, we need to calculate the minimization objective function and the nonshared parameters of the neural network in each client.

3.1.2. Objective Function. According to Figure 2, we define the final objective function of the algorithm as

$$\begin{aligned} L &= L_{cls} + \lambda L_{mm\ d} + \gamma L_{di\ sc}, \\ L &= L_{cls}^{sever} + \lambda L_{mm\ d}^{sever} b + \gamma L_{di\ sc}^{sever} = \sum L^{client-i} + \gamma L_{di\ sc}^{sever}. \end{aligned} \tag{5}$$

Classification loss L_{cls}-the loss caused by a specific domain classifier, according to Figure 2, we can see that the variable x_j in source domain(client) i undergoes a three-step transformation, first get $F(x_j^{Client-i})$ through the public feature extractor, then get $H_i(F(x_j^{Client-i}))$ through domain-specific feature extractor, and finally get $C_i(H_i(F(x_j^{Client-i})))$ through the CNN classification after pruning. The final loss of the i-th client classification is:

$$\begin{aligned} L_{cls}^{Client-i} &= \mathrm{E}_{x\sim X_{Client-i}} J\left(C_i\left(H_i\left(F\left(x_j^{Client-i}\right)\right), y_j^{Client-i}\right)\right), \\ L_{cls}^{sever} &= \sum_{i=1}^{N} L_{cls}^{Client-i} = \sum_{i=1}^{N} \mathrm{E}_{x\sim X_{Client-i}} J\left(C_i\left(H_i\left(F\left(x_j^{Client-i}\right)\right), y_j^{Client-i}\right)\right). \end{aligned} \tag{6}$$

MMD loss L_{mmd}-specific domain classification loss, maximum mean discrepancy (MMD) is a commonly used method to estimate the difference of distribution measurement. It is a commonly used two sample test method in statistics. From two probability distributions p and q, first assume $p = q$, and then we can decide to accept or reject this hypothesis according to the results calculated or observed by MMD. Generally speaking, we can measure the difference between the two distributions according to the value of MMD.

$$d_k^2(p,q) \coloneqq \left\| \mathrm{E}_{x\sim p}[\phi(x)] - \mathrm{E}_{x\sim q}[\phi(x)] \right\|_H^2, \tag{7}$$

where H is the reproducing kernel Hilbert space (RKHS) endowed with a characteristic kernel k. Here, $\phi(\cdot)$ denotes some feature map to map the original samples to RKHS and the kernel k means $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$,where $\langle \cdot,\cdot \rangle$ represents inner product of vectors. The main theoretical result is that $p = q$ if and only if $d_k^2(p,q) = 0$. In practice, an estimate of the MMD compress the square distance between the empirical kernel embedding is obtained, that is, the MMD loss of the client.

$$\hat{D}_H(p,q) = \left\| \frac{1}{n_s} \sum_{x_a \in X^s} \phi(x_a) - \frac{1}{n_t} \sum_{x_b \in X^t} \phi(x_b) \right\|_H^2. \tag{8}$$

Equation (8) defines the difference estimation between single source domain and target domain. Therefore, we can give the MMD loss on the server side.

$$L_{\text{mmd}}^{\text{server}} = \frac{1}{N} \sum_{i=1}^{N} \hat{D}\left(B_i\left(F\left(X^{s_i}\right), F\left(X^t\right)\right)\right). \tag{9}$$

Classifier alignment loss L_{disc}-The target samples near the class boundary are more likely to be misclassified by the classifiers learned from the source samples. The classifier is trained in different source domains, so there may be differences in the prediction of target samples, especially near the class boundary. Intuitively, the same target samples predicted by different classifiers should get the same prediction. Therefore, we implement the classifier alignment strategy between domains on the server. Inspired by reference [17], we define the server side and the classifier alignment strategy as follows:

$$L_{\text{disc}}^{\text{server}} = \frac{2}{N \times (N-1)} \sum_{j=1}^{N} \sum_{j=1}^{N} \mathrm{E}_{x\sim X_t} \left\| C_{client-i}\left(H_{client-i}\left(F\left(x_k\right)\right)\right) - C_{client-j}\left(H_{client-j}\left(F\left(x_k\right)\right)\right) \right\|. \tag{10}$$

The final objective function is

$$L^{client-i} = L_{cls}^{client-i} + \lambda L_{mm\ d}^{client-i}$$

$$L^{sever} = L_{cls}^{sever} + \lambda L_{mm\ d}^{sever} + \gamma L_{di\ sc}^{sever} = \sum_{i=1}^{N} L^{client-i} + \gamma L_{di\ sc}^{sever} = \sum_{i=1}^{N} L_{cls}^{client-i} + \frac{\lambda}{N} \sum_{i=1}^{N} L_{mm\ d}^{client-i} + \gamma L_{di\ sc}^{sever}. \tag{11}$$

In summary, the specific process steps of MMTLDMC algorithm are shown in Algorithm 1 θ_0^G, $\theta^{\mathscr{L}}$ is the model parameter.

4. Experimental Results

In order to test the effectiveness and generalization of the MMTLDMC algorithm, we test it on two types of image datasets which are shown in Tables 1 and 2. The first type is a digital classification dataset including SVHN[36] dataset, USPS[37] dataset, and MNIST[38] dataset; the second category is image classification dataset including Office31 [39] dataset and Caltech [40] dataset; the third category is network dataset named DomainNet [16].

The experiment will compare the multisource transfer learning algorithms DCTN, MFSAN, and MultiSource TrAdaBoost and give the results of the experiment under different pruning rates. For the fairness of the experiments, a 5-fold cross-validation strategy was selected for all experiments, and the experimental results of repeating this strategy twice were used as the final comparison results. In the experiment, we use the average classification accuracy [41] and recall rate of each algorithm after running 10 times as the evaluation criteria. The recall rate reflects how many positive examples in the sample are predicted correctly. The form of expression of classification accuracy and recall are defined as follows:

Classification accuracy: Accuracy = $(|x: x \in X \wedge f(x) = y(x)|/|x: x \in X|)$

Recall rate: $R = (\mathrm{FP}/\mathrm{TP} + \mathrm{FN}) \times 100\%$

Among them, TP represents the number of positive samples that are correctly classified as positive, FP represents the number of negative samples that are incorrectly classified as positive, TN represents the number of negative samples that are correctly classified as negative, and FN represents the number of positive samples that are incorrectly classified as negative.

X represents the target domain number test dataset, $f(x)$ is the sample x-class label predicted by the classifier, and $y(x)$ is the reality-class label of the sample x.

4.1. Digital Classification Dataset

4.1.1. Dataset Introduction. Both the USPS dataset and the MNIST dataset contain handwritten digits $0''$–$''9''$, the former is composed of 9298 16×16 images, and the latter is composed of 70,000 28×28 images. Street View House Number (SVHN) comes from Google. Each picture contains a group of Arabic numerals' 0–$9'$, which contains 73257 digits and the image pixel is 32×32. Figure 3 shows examples of USPS, MNIST, and SVHN. We can see that the distributions of USPS and MNIST are different, but they contribute the same feature space. SVHN datasets are different from their distribution and feature space. We extract 9000 images from MNIST and SVHN as two domains. Because USPS has only 9298 pictures, we regard the whole dataset as a domain. Due to the limitation of the mobile terminal, 3000 images are extracted from MNIS, SVHN, and USPS as a domain, respectively.

4.1.2. Experimental Data. In this part, we compare some multisource transfer learning algorithms such as DCTN and M^3SDA MultiSource TrAdaBoost with our algorithm MMTLDMC.

It can be seen from Table 3 that in the three cross-domain tasks, the MMTLDMC algorithm has an accuracy of 79.87%, 97.68%, and 95.49% when the pruning rate is 90%, which is higher than the comparison algorithms DCTN, MultiSource TrAdaBoost, and M^3SDA. Compared with the data without pruning, in the tasks U.M- > S, U.S- > M, M.S- > U, the accuracy of our algorithm only drops by 1.36%, 1.45%, and 1.46%. However, when the pruning rate of the MMTLDMC algorithm is 95%, the accuracy dropped sharply. Because our algorithm runs on the mobile terminal, the pruning of 30% is not considered for the algorithm MMTLDMC.

4.2. Image Classification Dataset

4.2.1. Dataset Introduction. The Office-31 dataset is a commonly used standard transfer learning dataset. It contains 4652 sample pictures collected from different areas named Amazon (A), Webcam (W), and DSLR (D), these pictures can be divided into 31 categories. Among them, Amazon's samples are from https://amazon.com, and the samples in Webcam and DSLR are obtained through web cameras and digital SLR cameras in different environments. Caltech-256 [40] is a standard database for object recognition. The database has 30607 images and 256 categories. In these experiments, we used the dataset as office-31+Caltech published by Gong [39] et al., as shown in Figure 4. Specifically, we have four domains, C(caltech-256), A (Amazon), W (webcam), and D (DSLR). We randomly select three domains as the source domain and the remaining one as the target domain, that is, (A, C, D- > W), (A, C, W- > D), (A, D, W- > C), (C, D, W- > A).

Sever:
(1) initialize θ_0^G with the BN pruning model
(2) for each round $r = 1, 2, 3, \ldots,$ do
(3) for each client $i = 1, 2, 3, \ldots,$ do
(4) $\mathscr{L}_i^{\text{Client}} \leftarrow \text{client}(\mathscr{L}_{ck} + \mathscr{L}_{\text{mmd}})$
(5) end for
(6) calculate $\mathscr{L}_{\text{disc}}$
(7) $\mathscr{L}^{\text{sever}} \leftarrow \min(\sum \mathscr{L}_{\text{client}} + \lambda \mathscr{L}_{\text{disc}})$
(8) Update θ_0^G, $\theta^{\mathscr{L}i}$ by minimizing $\mathscr{L}^{\text{sever}}$
(9)end for
Client: Run on client
(1) Update $\theta^{\mathscr{L}i}$ by sever
(2) for each local epoch do
(3) Calculate $L_{\text{client}} = L_{\text{cls}} + L_{mm\ d}$
(4) end for
(5) return $\mathscr{L}_{\text{client}}$

Algorithm 1: MMTLDMC algorithm training MMTLDMC steps.

Table 1: Description of digital datasets and image datasets.

Dataset	Digital dataset			Image dataset			
Name	USPS	MNIST	SVHN	Amazon	Office31 Webcam	DSLR	Caltech
Short	U	M	S	A	W	D	C

SVHN USPS MNIST

Figure 3: Example of USPS MNIST and SVHN pictures.

DSLR Amazon Webcam Caltech-256

Figure 4: Example of Office31 and Caltech-256 pictures.

Table 2: Description of network datasets.

Dataset	Network dataset (DomainNet)					
Name	Clipart	Infograph	Painting	Quickdraw	Real	Sketch
Short	clp	inf	pnt	qdr	rel	skt

Table 3: Pruning rate, accuracy (%), and decrease point of the algorithm on the digit dataset.

Algorithms	U.M- > S			U.S- > M			M.S- > U		
	Pruning (%)	Accuracy (%)	Decrease Point	Pruning (%)	Accuracy (%)	Decrease Point	Pruning (%)	Accuracy (%)	Decrease Point
DCTN	0	78.02		0	97.58		0	93.64	
	30	77.88	0.14	30	97.47	0.11	30	93.51	0.13
	50	77.7	0.32	50	97.39	0.19	50	93.39	0.25
	70	77.6	0.42	70	97.27	0.31	70	93.18	0.46
	90	76.49	1.53	90	96.14	1.44	90	92.31	1.33
	95	49.56	28.46	95	69.22	28.36	95	64.52	29.12
MultiSource TrAdaBoost	0	78.83		0	96.41		0	93.82	
	30	78.72	0.11	30	96.28	0.13	30	93.5	0.32
	50	78.62	0.21	50	96.2	0.21	50	93.44	0.38
	70	78.26	0.57	70	96.15	0.26	70	93.29	0.53
	90	77.4	1.43	90	95.05	1.36	90	92.41	1.41
	95	48.57	30.26	95	66.43	29.98	95	63.71	30.11
M^3SDA	0	79.28		0	99.05		0	95.93	
	30	79.21	0.07	30	98.99	0.06	30	95.82	0.11
	50	79.13	0.15	50	98.96	0.09	50	95.7	0.23
	70	79.05	0.23	70	98.89	0.16	70	95.61	0.32
	90	77.72	1.56	90	97.37	1.52	90	94.27	1.66
	95	49.92	29.36	95	69.59	29.46	95	66.95	28.98
MMTLDMC	0	81.23		0	99.13		0	96.95	
	X	X	X	X	X	X	X	X	X
	50	81.1	0.13	50	99.07	0.06	50	96.79	0.16
	70	81.07	0.16	70	98.99	0.14	70	96.69	0.26
	90	79.87	1.36	90	97.68	1.45	90	95.49	1.46
	95	52.24	28.99	95	69.92	29.21	95	66.68	30.01

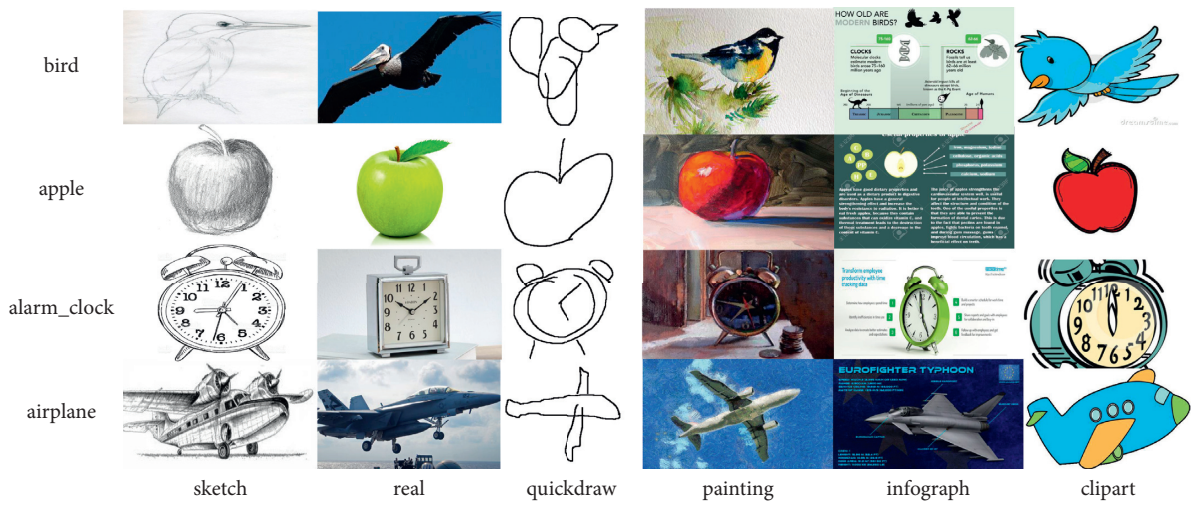

Figure 5: Example of DomainNet pictures.

Table 4: Pruning rate, accuracy (%), and decrease point of the algorithm on the image dataset.

Algorithms	A.W.D- > C			A.W.C- > D			A.D.C- > W			W.D.C- > A		
	Pruning(%)	Accuracy(%)	Decrease point	Pruning (%)	Accuracy (%)	Decrease point	Pruning (%)	Accuracy (%)	Decrease point	Pruning (%)	Accuracy (%)	Decrease point
DCTN	0	90.48		0	98.7		0	98.31		0	91.22	
	30	90.32	0.16	30	98.65	0.05	30	98.28	0.03	30	91.1	0.12
	50	90.19	0.29	50	98.62	0.08	50	98.21	0.1	50	90.97	0.25
	70	90.07	0.41	70	98.49	0.21	70	98.05	0.26	70	90.79	0.43
	90	89.26	1.22	90	97.69	1.01	90	96.75	1.56	90	89.77	1.45
	95	65.85	24.63	95	68.48	30.22	95	68.06	30.25	95	61.59	29.63
MultiSource TrAdaBoost	0	90.32		0	98.42		0	97.89		0	90.62	
	30	90.07	0.25	30	98.3	0.12	30	97.81	0.08	30	90.41	0.21
	50	90.01	0.31	50	98.23	0.19	50	97.77	0.12	50	90.28	0.34
	70	89.87	0.45	70	98.08	0.34	70	97.69	0.2	70	90.1	0.52
	90	88.11	2.21	90	97.2	1.22	90	96.41	1.48	90	88.66	1.96
	95	62.01	28.31	95	71.68	26.74	95	69.58	28.31	95	62.7	27.56
M^3SDA	0	92.55		0	99.29		0	99.06		0	94.08	
	30	92.37	0.18	30	99.2	0.09	30	99.03	0.03	30	93.97	0.11
	50	92.35	0.2	50	99.15	0.14	50	98.99	0.07	50	93.89	0.19
	70	92.32	0.23	70	99.11	0.18	70	98.93	0.13	70	93.82	0.26
	90	91.29	1.26	90	98.06	1.23	90	97.51	1.55	90	92.07	2.01
	95	65.22	27.33	95	73.92	25.37	95	68.93	30.13	95	64.84	29.24
MMTLDMC	0	91.88		0	99.46		0	99.52		0	95.11	
	X	X	X	X	X	X	X	X	X	X	X	X
	50	91.75	0.13	50	99.4	0.06	50	99.49	0.03	50	95.05	0.06
	70	91.67	0.21	70	99.33	0.13	70	99.47	0.05	70	95.01	0.1
	90	90.77	1.11	90	98.48	0.98	90	98.52	1.25	90	94.13	0.98
	95	67.67	24.21	95	75.14	24.32	95	73.21	26.31	95	67.15	27.96

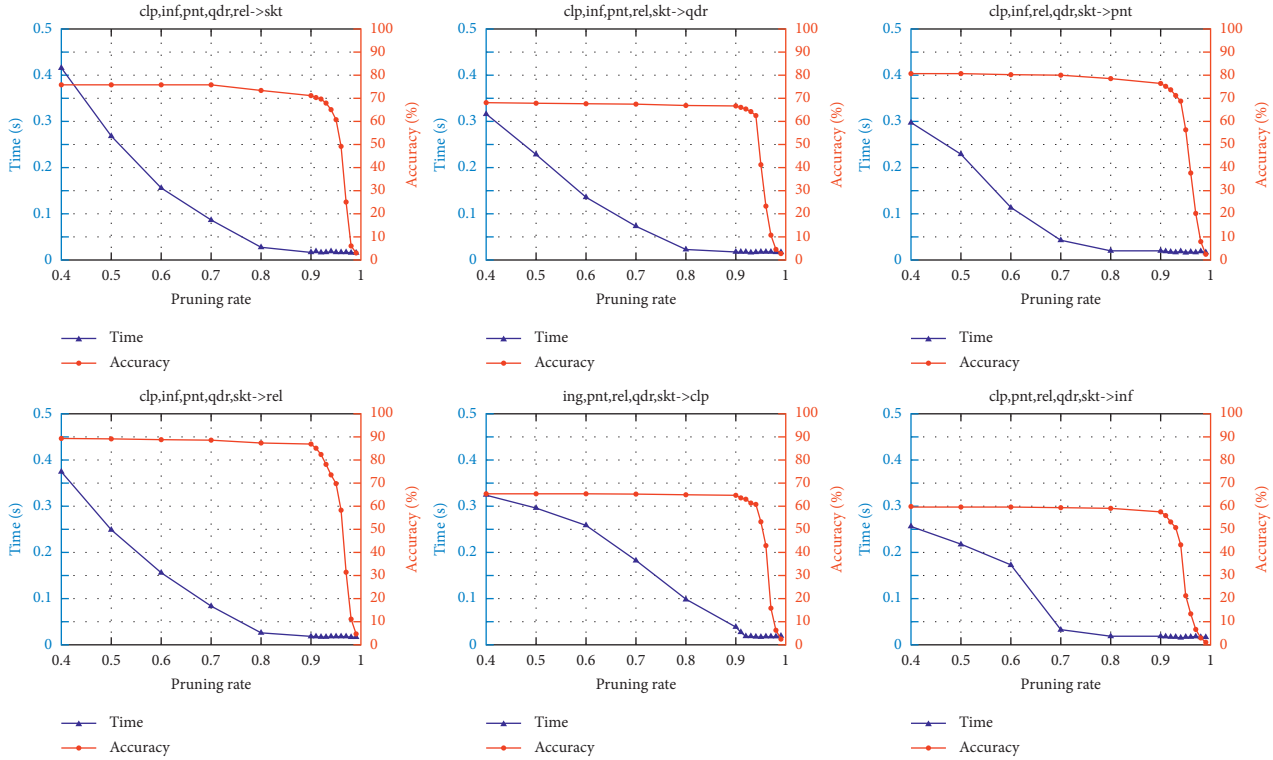

Figure 6: Comparison on DomainNet.

4.2.2. Experimental Data. In this section, we compare MMTLDMC with multisource transfer learning algorithms such as DCTN, M^{3}SDA, and MultiSource TrAdaBoost.

It can be seen from Table 4 that among the four cross-domain tasks, when the pruning rate of the MMTLDMC algorithm is 90%, the accuracy rates are 90.77%, 98.48%, 98.52%, and 94.13%, which are higher than the comparative algorithms DCTN and MultiSource TrAdaBoost. At the same time, under the tasks of A.W.C- > D and W.D.C- > A, the accuracy only drops by 0.98%. However, when the pruning rate of the MMTLDMC algorithm is 95%, the accuracy dropped sharply. Because our algorithm runs on the mobile terminal, the pruning of 30% is not considered for the algorithm MMTLDMC.

4.3. Effect of the Pruning Rate on Computation Time. In order to demonstrate the advantages of the influence of our algorithm category, we choose the network dataset DomainNet proposed by the literature [14] (as shown in Figure 5). We randomly sample 20 classes from each domain, 3000 data as our training data.

4.3.1. Experimental Data

(1) Effect of the Pruning Rate on Computation Time and Iterations. As can be seen from Figure 6, the algorithm MMTLDMC is under the dataset DomainNet:a) The model calculation time will gradually decrease with the increase of the pruning rate. (b) The computational accuracy of the model will gradually decrease with the increase of the pruning rate. (c) From the figure, we can clearly see that when the pruning rate is 94%, the calculation accuracy will drop significantly. (d) When the pruning rate reaches 90%, the accuracy generally only drops by 2–4%. (e) When the pruning rate reaches 80%, the model calculation time will stabilize.

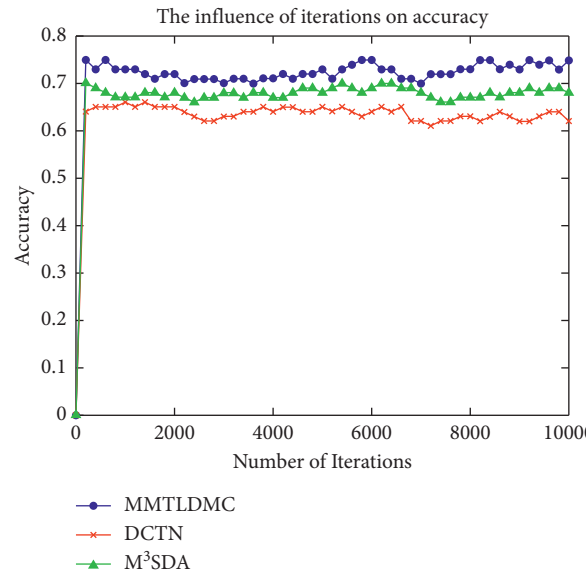

Figure 7: Effect of iteration times on accuracy.

(2) Influence of Iterations. Figure 7 shows the effect of the number of iterations on the accuracy. (a) When the number of iterations exceeds 1000, the accuracy of the algorithm

tends to be stable. (b) At the same time, MMTLDMC has better results.

5. Conclusion

In this paper, aiming at the problem of multisource transfer learning for mobile computing, a multisource mobile transfer learning algorithm based on dynamic model compression is proposed. This method combines the advantages of mobile computing and multisource learning to complete the image classification on the mobile terminal. This method first performs BN pruning on the pretrained network, and then calculates the classification loss and MMD loss of the multisource transfer model on the mobile side, and calculates the losses of different classifiers on the server side. Finally, the parameters of the minimized objective function are synchronized to the client. The experimental results on the SVHN, USPS, MNIST, Office31, Caltech, and DomainNet show that MMTLDMC outperforms the benchmark algorithms in both classification accuracy and training efficiency. Although the experimental results show that the MMTLDMC algorithm is better than the benchmark algorithm, further research is still needed in the following aspects: through data encryption, partial data sharing, and dynamic adjustment of the BN pruning strategy on the server side to optimize the model classification accuracy; continuing to reduce the number of parameters to achieve faster client computing.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been supported by China's national key Research and Development plan (2016YFB0801004). This work has been supported by Science and Technology Major Special Project of Heilongjiang (CN) (2020ZX14A02).

References

[1] 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, 2010.

[2] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, 2016.

[3] M. Long, Z. Han, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conferenceon Machine Learning, ICML 2017*, pp. 2208–2217, Sydney, NSW, Australia, August 2017.

[4] J. Kang, R. Fernandez-Beltran, and P. Duan, "Deep unsupervised embedding for remotely sensed images based on spatially augmented momentum contrast[J]," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 99, pp. 2598–2610, 2020.

[5] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research*, F. Bach and D. Blei, Eds., PMLR, Lille, France, pp. 97–105, Jul 2015.

[6] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," *AAAI*, vol. 6, p. 8, 2016.

[7] X. Peng and K. Saenko, "Synthetic to real adaptation with generative correlation alignment networks," in *Proceedings of the 2018 IEEE Winter Conference on Applications of Computer Vision*, pp. 1982–1991, Lake Tahoe, NV, USA, March 12-15, 2018.

[8] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, p. 4, Honolulu, HI, USA, 21-26 July 2017.

[9] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning, volume 37 of Proceedings of Machine Learning Research*, F. Bach and D. Blei, Eds., PMLR, Lille, France, pp. 1180–1189, Jul 2015.

[10] Y. Zhu, F. Zhuang, J. Wang, J. Ke, J. Bian, and H. Xiong, "Deep subdomain adaptation network for image classification[J]," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 99, pp. 1713–1722, 2021.

[11] B. Sun, J. Feng, and K. Saenko, "Return of frustratingly easy domain adaptation," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, vol. 6, p. 8, 2016.

[12] P. Gao, W. Wu, and J. Li, "Multi-source fast transfer learning algorithm base on support vector machine [J]," *Applied Intelligence*, vol. 52, pp. 1–15, 2021.

[13] H. Zhao, S. Zhang, G. Wu, J. M. Moura, J. P. Costeira, and G. J. Gordon, "Adversarial multiple source domian adpation," in *Proceedings of the Conference on Neural Information Processing Systems*, pp. 8559–8570, 2018.

[14] R. Xu, Z. Chen, and W. Zuo, "Deep Cocktail network: multi-source unsupervised domain adaptation with category shift," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2018.

[15] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[16] Y. Zhu, F. Zhuang, and D. Wang, "Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5989–5996, 2019.

[17] Y. Lin, H. Zhao, X. Ma, Y. Tu, and M. Wang, "Adversarial attacks in modulation recognition with convolutional neural networks," *IEEE Transactions on Reliability*, vol. 70, no. 1, pp. 389–401, 2021.

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

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks [J]," *Advances in Neural Information Processing Systems*, vol. 25, no. 2, pp. 1097–1105, 2012.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations*, pp. 1021–1026, San Diego, CA, USA, 2015.

[21] C. Szegedy, W. Liu, and Y. Jia, "Going deeper with convolutions," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pp. 1–9, IEEE, Bosto, MA, USA, 2015.

[22] K. He, X. Zhang, and S. Ren, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) , pp. 770–778, IEEE, Las Vegas, NV, USA, 2016.

[23] Y. Yang, D. Xu, F. Nie, S. Yan, and Y. Zhuang, "Image clustering using local discriminant models and global integration," *IEEE Transactions on Image Processing*, vol. 19, no. 10, pp. 2761–2773, 2010.

[24] J. F. Wang, J. D. Wang, and J. K. Song, "Optimized cartesian K-Means," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 180–192, 2014.

[25] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Handbook of systemic autoimmune diseases*, vol. 1, no. 4, pp. 1024–1031, 2009.

[26] J. Deng, W. Dong, and R. Socher, "Imagenet: a large-scale hierarchical image database," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, IEEE, Miami, Florida, USA, 2009.

[27] D. G. Lowe, "Distinctive image features from scale-invariant k," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[28] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: speeded up robust features," in *Proceedings of the 9th European conference on Computer Vision-Volume Part I. [S.l.]*, pp. 404–417, Springer-Verlag, 2006.

[29] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," *Computer Vision - ECCV 2018* in *Proceedings of the European Conference on Computer Vision*, vol. 11218, pp. 139–156, Munich. Germany, 2018.

[30] J. Chang, L. Wang, G. Meng, S. Xiang, and C. Pan, "Deep adaptive image clustering," in *Proceedings of the 2017 IEEE International Conference on Computer Vision*, pp. 5880–5888, IEEE, Venice, Italy, 2017.

[31] L. Zhuang, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming[J]," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 22-29 Oct. 2017.

[32] K. Crammer, M. Kearns, and J. Wortman, "Learning from multiple sources," *Journal of Machine Learning Research*, vol. 9, pp. 1757–1774, 2008.

[33] Y. Mansour, M. Mohri, and A. Rostamizadeh, "Domain adaptation with multiple sources," *Advances in Neural Information Processing Systems*, pp. 1041–1048, 2008.

[34] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer feature learning with joint distribution adaptation," in *Proceedings of the 2013 IEEE International Conference on Computer Vision*, pp. 2200–2207, IEEE, Sydney, NSW, Australia, 1-8 Dec. 2013.

[35] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Sch¨olkopf, and A. Smola, "A kernel two-sample test," *JMLR*, vol. 13, no. Mar, pp. 723–773, 2012.

[36] X. Li, H. Xiong, H. Wang, Y. Rao, L. Liu, and J. Huan, "DELTA: deep learning transfer using feature map with attention for convolutional net- works," in *Proceedings of the 2019 International Conference on Learning Representations*, New Orleans, Louisiana, 2019.

[37] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," *Nips Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[38] X. Li, M. Fang, and J.-J. Zhang, "Projected Transfer Sparse Coding for cross domain image representation," *Journal of Visual Communication and Image Representation*, vol. 33, pp. 265–272, 2015.

[39] M. Long, J. Wang, G. Ding, S. J. Pan, and P. S. Yu, "Adaptation regularization: a general framework for transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1076–1089, 2014.

[40] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE*, pp. 2066–2066, Providence, RI, USA, 2012.

[41] G. Griffin, A. Holub, and P. Perona, *Caltech-256 Object Category Dataset*, Technical report, Caltech, 2007.