IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received March 3, 2019, accepted April 11, 2019, date of publication July 29, 2019, date of current version December 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2912908

Transferring Ensemble Representations Using Deep Convolutional Neural Networks for Small-Scale Image Classification

SHUYIN XIA[1], (Member, IEEE), YULONG XIA[1], HONG YU[1], QUN LIU[1], YUEGUO LUO[2], GUOYIN WANG[1], (Senior Member, IEEE), AND ZIZHONG CHEN[3], (Senior Member, IEEE)

[1]Chongqing Key Laboratory of Computational Intelligence, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2]College of Big Data and Intelligent Engineering, Yangtze Normal University, Chongqing 408100, China
[3]Department of Computer Science and Engineering, University of California at Riverside, Riverside, CA 92521, USA

Corresponding author: Shuyin Xia (xiasy@cqupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61806030, Grant 61876027, Grant 61772096, and Grant 61533020, and in part by the National Key Research and Development Program of China under Grant 2016QY01W0200 and Grant 2016YFB1000905.

ABSTRACT The deep convolutional neural networks (DCNN) require large number of training data to avoid overfitting, which makes it unsuitable for processing small-scale image datasets. The transfer learning using DCNN (TCNN) reuses pre-trained layers to generate a mid-level image representation so that the optimization of more than millions CNN parameters can be avoided. By this way, overfitting problem in small-scale data can be alleviated. However, although now many public DCNNs have been trained and can be reused, the existing TCNNs are formed by only a single pre-trained DCNN structure and cannot make full use of multiple structures of pre-trained DCNNs. At the same time, the existing ensemble CNNs have not enough good representation ability. To address this problem, we combine the conventional ideas of ensemble CNNs and propose three ensemble TCNNs (TECNN). They are the voting method based on the combination of all TCNNs, the PickOver method by finding the optimal combination, and weighted method by finding weighted combination. Different from the existing ensemble CNNs, the proposed methods do not need to retrain the component CNNs and generate ensemble transferring representations by transferring the pre-trained mid-level parameters. The mathematical models of those three methods are also provided. Their versions of using fine-tuning are also compared in the experiments. In addition, we replace the Softmax classifier with ensemble linear classifiers in the full-connection layer. They outperform the current state of the art algorithms on Caltech ImageNet and some internet image data. All this research has released as an open source library called Transferring Image Ensemble Representations using Deep Convolutional Neural Networks (TECNN). The source codes and relevant datasets in different versions are available from: http://www.cquptshuyinxia.com/TECNN.html.

INDEX TERMS Convolutional neural networks, deep CNN, transferring CNN, transferring Learning.

I. INTRODUCTION

The object recognition represents an important part of the computer vision. Recently, the robust image descriptors have been developed significantly, such as SIFT [1] and HOG [2], bag of features image representations [3]–[6], deformable part models [7] and deep convolutional neural networks (DCNNs). An enabling factor is the development of increasingly large and realistic image datasets, providing an object annotation for training and testing, e.g. Caltech256 [8], Pascal VOC [9] and ImageNet [10]. The CNNs are high-capacity classifiers with a very large number of parameters that need to be optimized during the training process. CNNs have a long history in visual recognition and exhibit record-shattering results in computer vision [11], [12], image translations [13], optical character recognition [14]–[16] and many other various fields [17]–[21]. The early CNNs' performance was limited by a relatively small size of the

The associate editor coordinating the review of this manuscript and approving it for publication was Fanbiao Li.

standard object recognition datasets. However, this limitation has changed due to the appearance of the large-scale ImageNet dataset [10] and enhancement of the GPU computing power. Krizhevsky et al. achieved a performance leap in the image classification on the ImageNet 2012 Large-Scale Visual Recognition Challenge (ILSVRC-2012). They further improved the network performance by training with 15 million images and 22,000 ImageNet classes [22]. According to their works, a thorough evaluation of networks is made in terms of depth incensement by using an architecture with very small (3x3) convolution filters [23]. In addition, a significant improvement of the prior art configurations can be achieved by increasing of the depth to 16-19 layers. Although this result is promising and exciting, it is also worrisome as millions of annotated images are required to be collected for each visual recognition task. Namely, collection of a large corpus of annotated data to train the CNNs is nearly impossible in real applications, such as the robotics applications [24] and customized categories of applications [25]. In other words, the DCNN offers a large representation space and is very easy to lead to overfitting in processing small-scale datasets. Although the shallow CNNs including the ensemble CNNs can avoid overfitting in the processing of small-scale datasets, it suffers from poor representation ability due to the small number of parameters and layers.

To take advantage of the good representation ability of the DCNN and prevent overfitting by avoiding training too much parameters, researchers have studied the transfer image representations of DCNNs for visual recognition tasks with small sample size. Instead of directly training CNN for a specific task with a small-scale dataset, Oquab et al. designed a method that reuses the intermediate layers of a DCNN trained on the ImageNet dataset to generate a mid-level image representation of images in the PASCAL VOC dataset [26]. This transferred representation can significantly enhance classification accuracy in visual recognitions tasks with small sample size, such as [27]–[32]. However, the mentioned works almost used only one single pre-trained DCNN structure although many pre-trained DCNNs can be efficiently used for transfer learning.

To make full use of the existing pre-trained DCNNs, we propose here three methods to integrate multiple pre-trained DCNNs by introducing the ensemble methods of conventional CNNs.

The contributions of this paper are threefold as follows.

1) We introduce conventional ideas of ensemble CNNs into TCNNs and propose three ensemble TCNNs (TECNNs). They are the voting method based on the combination of all TCNNs, the PickOver method by finding the optimal combination, and weighted method by finding weighted combination. Different from the existing ensemble CNNs in which the component CNNs are retrained, the proposed methods do not need to retrain the component DCNNs and generate ensemble transferring representations by transferring the pre-trained mid-level parameters.

2) Their versions of using fine-tuning are also compared in the experiments, and the fine-tuning versions achieve a higher generalizability by using the ''root mean square prop'' method to fine-tune the last full-connected layer.

3) Except the ensemble method in the pre-trained DCNNs, we replace the Softmax classifier with ensemble linear classifiers in the full-connection layer, and the proposed methods achieve better performance on some datasets.

II. RELATED WORK

A. TRANSFERRING DCNN

The key idea of the existing transfer learning DCNN (TCNN) is that the internal layers of the CNN act as the extractors of a mid-level image representation. They can be hence pre-trained with the source dataset and then reused for other target tasks, as shown in Fig. 1 [26]. First, a network is trained on the source task (e.g. the ImageNet classification, top row) with a large amount of available labelled images. Then, the pre-trained parameters of the internal layers of the network (C1-FC7) are transferred to the target tasks (bottom row). To compensate different image statistics, e.g., objects types, typical viewpoints and imaging conditions, of the source and target data, an adaptation layer (fully connected layers FC1) is introduced and trained on the labelled data of the target task [26]. The TCNN has been widely used in various fields [33]–[35]. By transferring the pre-trained parameters of the internal layers, the TCNN is not required to train too many parameters and has deep representation ability. As a result, the TCNN not only exhibits outstanding representation ability of the deep CNN, but also alleviates overfitting for the DCNN process of small-scale datasets.

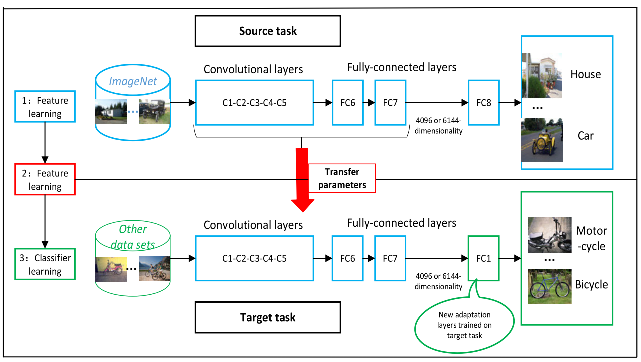


FIGURE 1. CNN Transferring parameters [26].

B. ENSEMBLE NEURAL NETWORK

Neural network ensemble is a learning strategy in which a limited number of neural networks receive the same task training [36]. It was derived from the work of Hansen and Salamon [37]. In general, two steps are required to construct a neural network integration including training a few component neural networks and combining them. The generalizability of the neural network system can be significantly improved by combining a series of neural networks.

This technology recently has become very popular in neural networks and machine learning community [38]. It has been successfully applied to various fields, such as the face recognition [39]–[41], medical diagnosis [42], image retrieval [43], [39] pedestrian detection [44], biological information processing [45] and medication safety [46]. Bagging and Boosting represent the most popular methods for training the component neural networks. The Bagging is based on the bootstrap sampling proposed by Breiman [47], [48] which generates several training sets from the original training set and then trains component neural networks from them. The Boosting was first proposed by Schapire [49] and then improved by Freund [50], Freund and Schapire [51], which produces a series of neural networks.

There are many other methods for training component neural networks. Hampshire and Waibel [52] use different target functions to train different neural networks. Liu [39] trains the network of components for different amounts of hidden units. Maclin and Shavlik [53] initialize component networks in different positions in the weight space. Krogh and Vedelsby [54] use cross-validation to create a component network. Opitz and Shavlik [55] use genetic algorithms to train different knowledge-based component networks. Yao and Liu [56] see all the individuals in the neural networks of evolution as component networks.

The most popular methods are plurality voting or majority voting [20] for classification tasks, simple average [57] or weighted average [58] for regression tasks. Wolpert [59] combine the learning system into component neural networks. Merz and Pazzani [60] use the principal component regression to determine the appropriate constraints of component network weights and combine them. Jimenez [61] uses dynamic weights that are determined by the confidence of the component networks to combine them. Ueda [62] uses the optimal linear weighting to combine the component neural networks based on the statistical pattern recognition theory. There are some ways to use neural networks to complete tasks in the style of divide-and-conquer [63]–[65].

Currently, however, few ensemble TCNNs are studied. Those existing ensemble CNNs are designed to retrain and integrate the CNN classifiers including a large number of parameters, leading to overfitting in small-scale datasets. In contrast, the TECNNs are not required to retrain a large number of parameters in the convolutional layers and can reuse several types of TCNNs. In this paper, we introduce three ensemble DCNN methods for transferring learning and verify their performance.

III. TRANSFERRING ENSEMBLE REPRESENTATIONS USING DEEP CONVOLUTIONAL NEURAL NETWORKS

A. THE FRAMEWORK OF TRANSFERRING IMAGE ENSEMBLE REPRESENTATIONS USING DEEP CNN (TECNN)

Fig. 2 shows the Framework of the Transferring Image Ensemble Representations using Deep CNN (TECNN). This framework is constituted of several pre-trained DCNNs, each of which has a corresponding TCNN. The TECNN is constituted of several TCNNs. Each convolutional layer of TCNN is generated by transferring the convolutional layers of the corresponded pre-trained DCNNs to the new DCNN. In addition, new adaptation layers are added into each TCNN and need to be retrained to compensate for different image statistics (type of objects, typical viewpoints, imaging conditions) of the source and target data. Moreover, an ensemble layer is added to integrate the results of the outputs of those TCNNs. More details and the mathematical model will be presented in Sec. 3.2.

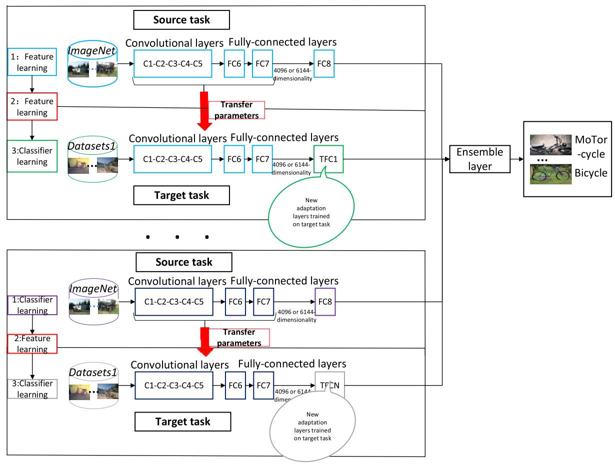


FIGURE 2. Transferring Image Ensemble Representations using DCNNs.

TABLE 1. Symbols used.

Symbol	The Meaning of the symbol
n	The number of training image samples;
N	The number of transferred CNNs;
w_i	the weights of the adaptive layer in the i-th transferred CNN ;
x_j	the j-th image sample;
y_j	the label of the j-th image sample;
$\mathrm{TCNN_i}$	The i-th transferred CNN;

B. CLASSIFICATION MODEL

Table 1 lists the symbols.

Take the binary classification problem as an example. A sample is labeled with +1 or −1. The loss function of the voting TECNN method can be expressed as follows:

$$\arg\min \sum_{j=1}^{n} \sum_{\substack{w_i \\ v_i \in [0,1], \\ i \in \{1..,N\},}} \left\{ ((f(x_j, w_i, TCNN_i) - y_j)^2 \right\} \tag{1}$$

The decision function of this method is expressed as follows:

$$\hat{f}(x) = \mathrm{sign}(\sum (label(f(x_j, w_i, TCNN_i)) - y_j)), \tag{2}$$

where sign(x) is a function described as follows:

$$\mathrm{sign}(x) = \begin{cases} 1, & if\ x > 0 \\ 1, & if\ x <= 0 \end{cases}$$

To optimize (1), each TCNN needs to be trained. In (2), the sign(.) function's value of the sum of the output labels of a sample in all TCNNs is considered as its predicted value when the voting TECNN method is used.

$$\arg\min \sum_{j=1}^{n} \sum_{\substack{w_i \\ v_i \in [0,1], \\ i \in \{1..,N\},}} \left\{ \begin{array}{l} ((f(x_j, w_i, TCNN_i) - y_j)^2 \\ + |sign(v_i * label(f(x_j, w_i, TCNN_i)) \\ -y_j)) \end{array} \right\}, \tag{3}$$

where $label\,(f(x_j, w_i, \mathrm{T}CNN_i))$ denotes the validation label of x_j in the i-th $\mathrm{T}CNN_i$. The loss function in (3) is constituted with two parts. In (3), the first half is first optimized and the second is then done. Consequently, the whole loss of (3) can be minimized. The first half denotes the loss function of each TCNN, so each TCNN should be optimized on their corresponding source dataset. The second half denotes the difference of combination output labels of the combination TCNNs and the true label. By optimizing the values of v_i, the value of which is set to 0 or 1, the candidate TCNNs are selected for ensemble.

In (3), some output probability values are lost in the ensemble process of labels. For example, the output probability values of a sample are respectively 0.7 and 0.4 in two TCNNs, so its labels are respectively 1 and -1 in binary classification problems. The ensemble results of the sample in the two TCNNs in (3) is equal to 0. If the output probability values of the sample are changed to be respectively 0.9 and 0.4, its ensemble result is the same with the above. Therefore, some output probability values are lost. Thus, (4) replaces the output label with the output probability value in the loss goal of (3). In addition, to show the different importance, in the third method, the test accuracy of a single TCNN is used as its weight to measure its importance in the ensemble representations. Therefore, (3) can be transformed into (4) as follows:

$$\arg\min \sum_{j=1}^{n} \sum_{\substack{w_i \\ v_i \in [0,1], \\ i \in \{1..,N\},}} \left\{ \begin{array}{l} ((f(x_j, w_i, TCNN_i) - y_j)^2 \\ +PA_i * |sign(v_i * (f(x_j, w_i, TCNN_i) \\ -y_j))| \end{array} \right\}, \tag{4}$$

where PA_i denotes the validation accuracy of the i-th transferred $\mathrm{T}CNN_\mathrm{i}$.

C. ALGORITHM DESIGN

To implement model (1), (2) and (3), three algorithms have been designed as Table 2, Table3 and Table 4.

TABLE 2. Training learning of the voting method.

Algorithm 1. Training learning of the Voting Method	
Input:	Input training image dataset D and test image dataset D', pre-traineded DCNNs
Output:	The labels of samples in D'
1	**For** i = 1 **to** the number of pre-trained DCNNs
2	Transfer the middle weights of $\mathrm{DCNN_i}$ to the transferred $\mathrm{TCNN_i}$;
3	Training and fine-tuning the w_i in the i-th $\mathrm{TCNN_i}$ to optimize the first part of (1);
4	Optimize the parameters v_i for each i according to the second part of (1);
5	**End**

TABLE 3. Training learning of the PickOver.

Algorithm 2. Training learning of the PickOver method	
Input:	Input training image dataset D and test image dataset D', pre-traineded DCNNs
Output:	The labels of samples in D'
1	**For** i = 1 **to** the number of pre-trained DCNNs
2	Transfer the middle weights of $\mathrm{DCNN_i}$ to the transferred $\mathrm{TCNN_i}$;
3	Training and fine-tuning the w_i in the i-th $\mathrm{TCNN_i}$ to optimize the first part of (2);
4	Optimize the parameters v_i for each i according to the second part of (2);
5	**End**

TABLE 4. Training learning of the weighted method.

Algorithm 3. Training learning of the weighted method	
Input:	Input training image dataset D and test image dataset D', pre-traineded DCNNs
Output:	The labels of samples in D'
1	**For** i = 1 **to** the number of pre-trained DCNNs
2	Transfer the middle weights of $\mathrm{DCNN_i}$ to the transferred $\mathrm{TCNN_i}$;
3	Training and fine-tuning the w_i in the i-th $\mathrm{TCNN_i}$ to optimize the first part of (3);
4	Optimize the parameters v_i for each i according to the second part of (3);
5	**End**

D. FINE-TUNING ENSEMBLE METHODS

In Sections III. A, B, C, the fine-tuning mechanism in pre-trained DCNNs is not used. Using the fine-tuning mechanism is good for improving the generalizability of TDCNN. However, it is easy to lead to overfitting in small-scale data sets if too much layers are fine-tuned. To utilize the advantage of the fine-tuning mechanism, at the same time, and optimize as few parameters as possible in the fine-tuning process, the last full-connected layer is fine-tuned by using the "root mean square prop" method. The "root mean square prop" method is proposed by Geoff Hinton in the Coursera. Although it is not published, it has been widely used in various fields. The weights of convolutional layers are fixed in this paper. The fine-turned version of Algorithm 1, 2, 3 are separately named by putting "+" after these letters.

E. USING VARIOUS LINEAR CLASSIFERS IN THE FULL-CONNECTION LAYER

The Softmax classifier is a common linear classifier in the full-connection layer, and some other classifiers are used to replace the Softmax classifier, such as SVM [65]. Few studies use the ensemble classifiers in the full-connection layer. In this paper, we will use the ensemble linear classifiers to achieve better classification generalizability. Sign(.) function's value of the sum of the output value of a sample in all TCNNs is considered as the output value of the sample, and its decision function can be expressed as follows:

$$\hat{f}(x) = \text{sign}(\sum (f(x_j, w_i, TCNN_i) - y_j))$$

IV. EXPERIMENTS

In this section we first describe details of the pre-trained CNNs. Next, we show the experimental results of the proposed transfer learning method on different datasets collected from the Google, Baidu's picture library and Caltech. Moreover, to demonstrate the superior efficiency of the proposed algorithms, we compare them with the TCNN method [26] and CNNs. The structure of the compared CNNs is set as follows. The size of the network inputs is 224 × 224 × 3 pixels. As the training set is not large, the structure only contains three convolutional layers. The full architecture corresponds to C(32,3,3)-R-P-C(32,3,3)-R-P-C(64,3,3)-R-P-FC(2048)-R-Dropout(0.5)-FC(48)-R-Dropout (0.5), where C(d,f,s) represents a convolutional layer with d filters with spatial size of f × f, applied to the input with strides. Here, FC(n) is a fully connected layer with n nodes, and the Dropout layer is used to alleviate the overfitting. Moreover, R indicates the activation layer using the RELU function. All pooling layers P pool spatially in non-overlapping 2 × 2 regions. The final layer is connected to a Softmax classifier with dense connections.

The experiments have been performed on a standalone desktop computer, configured as follows. We use a CPU from Intel Core i5-4460 3.20GHz CPU, 8.00GB RAM, 465GB hard drive; 64-bit Windows10 Enterprise Edition operating system, 64-bit Windows version of python3. 5.2, and JetBrains PyCharm Community Edition 2016.2 as the compiling software. The other parameters are same as the default system configuration.

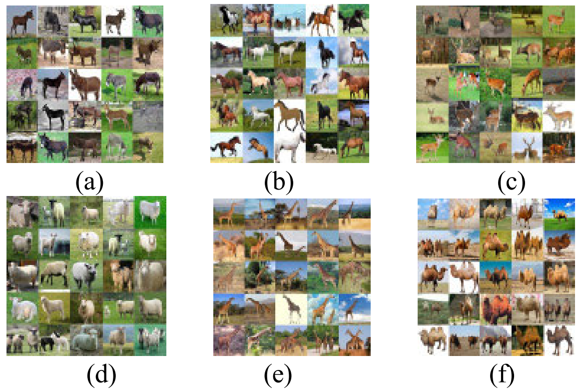


FIGURE 3. Some of the images in the data set. (a) ass, (b) horse, (c) cervus Nippon, (d) Bactrian camel, (e) giraffe, (f) sheep.

TABLE 5. Data sets details from Google and Baidu's picture library.

Data set	Data1	Data2	Data3	Data4	Data5	Data6
Firstclass	Horse	Horse	Horse	Horse	Horse	Sheep
Secondclass	BactrianCamel	Ass	Giraffe	Sheep	CervusNippon	Giraffe
Data7	**Data8**	**Data9**	**Data10**	**Data11**	**Data12**	
Sheep	Sheep	Giraffe	Giraffe	Sheep	Giraffe	
BactrianCamel	Ass	Ass	CervusNippon	CervusNippon	BactrianCamel	

A. PRE-TRAINED CNNs

We have used five pre-trained DCNNs based on the Keras framework, namely VGG16 [23], VGG19 [23], ResNet50 [66], InceptionV3 [67], and Xception [68]. Their structures have been trained by using the dataset ImageNet. The five per-trained models are combined with the transfer learning methods in [26] and named as TCNN_VGG16, TCNN_VGG19, TCNN_ResNet50, TCNN_InceptionV3 and TCNN_Xception, respectively. In all experiments, to stabilize the performance analysis of the compared algorithms, the test accuracy is achieved by averaging over 10 times. In each time, 80% samples are randomly selected from each dataset as the training set, and the remaining 20% as the test set. We have also used TensorFlow as the backend, where the parameters are set as the default values. The target objects in our datasets are not contained in the training dataset of the pre-trained DCNNs. So, the transferring representation ability of those algorithms can be checked.

B. IMAGE CLASSIFICATION ON INTERNET DATA

The experimental data sets have been randomly achieved from the Google and Baidu's picture library. There are six classes of picture data, each of which is composed 130 pictures. They include ass, horse, cervus nippon, bimodal camel, giraffe and sheep. Fig. 3 shows the experimental data. Each dataset is constituted of two classes, with 260 pictures in each class. Table 5 lists the experimental datasets. These datasets are available in http://pan.baidu.com/s/1mihu564.

The experimental results of all algorithms on all data sets are shown in Fig. 4. The DCNN has good representation

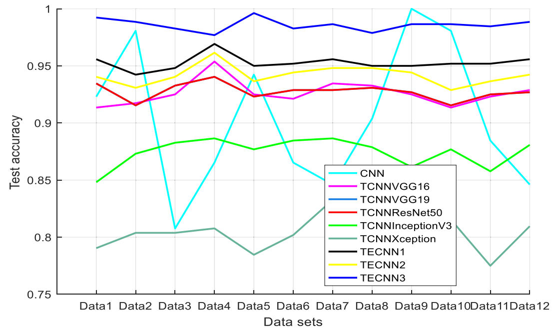


FIGURE 4. The comparison of test accuracy on different algorithms.

TABLE 6. Comparisons of test accuracy between different algorithms on the 12 data sets from internet.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
CNN	0.8269	0.8461	0.8269	0.8653	0.8076	0.9723
VGG16	0.9769	**0.9750**	**0.9712**	**0.9654**	**0.9846**	**0.9673**
VGG19	0.9788	0.9712	0.9692	0.9558	0.9808	0.9577
ResNet50	0.9788	0.9712	0.9692	0.9558	0.9808	0.9577
InceptionV3	0.8423	0.8481	0.8481	0.8212	0.8135	0.8346
Xception	0.7577	0.7654	0.7365	0.7500	0.7635	0.7385
Voting	0.9865	0.9750	0.9654	0.9635	0.9808	0.9654
PickOver	**0.9904**	**0.9923**	**0.9827**	**0.9769**	**0.9962**	**0.9846**
Weighted	**0.9904**	0.9885	**0.9827**	**0.9769**	**0.9962**	0.9827

TABLE 7. Comparisons of test accuracy between traditional transferred and ensemble transferred algorithms on the 12 Data Sets from Internet.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
CNN	0.9230	0.9038	0.9615	0.8653	0.9230	0.9615
VGG16	**0.9788**	0.9558	**0.9750**	0.9731	0.9692	0.9731
VGG19	0.9654	**0.9615**	0.9750	**0.9750**	**0.9692**	**0.9769**
ResNet50	0.9654	0.9615	0.9750	0.9750	0.9692	0.9769
InceptionV3	0.8519	0.8538	0.8481	0.8269	0.8231	0.8346
Xception	0.7385	0.7558	0.7615	0.7846	0.7365	0.7577
Voting	0.9769	0.9731	0.9788	0.9731	0.9635	0.9808
PickOver	**0.9865**	**0.9788**	0.9846	**0.9865**	**0.9865**	**0.9885**
Weighted	**0.9865**	**0.9788**	**0.9865**	**0.9865**	0.9846	**0.9885**

ability to describe an image; by contrast, the representation ability of CNNs is lower than the ability of DCNNs. Therefore, the transfer learning algorithms weighted method, PickOver method, voting method, TCNNVGG16, TCNNVGG19 and TCNNResNet50 exhibit higher test accuracy than the original CNN algorithm in most cases.

The proposed PickOver and Weighted methods present higher test accuracy than the CNN on all those datasets. InceptionV3 and Xception always exhibit the lowest accuracy. It indicates that these two TCNNs have not good transferring learning ability because of their relatively small original training data sets or simple structure or bad structure design. In addition, by integrating different TCNNs, Voting, PickOver and Weighted exhibit an obviously higher test accuracy than other TCNN algorithms.

Tables 6 and 7 provide the detail data, and the boldface is corresponded with the highest accuracy. As shown in Tables 6 and 7, it has the highest accuracy advantage when compared with other TCNN algorithms on the data2, i.e.1.73% higher than the most effective TCNN algorithm (i.e. VGG16) on data2.

TABLE 8. Comparisons of average test accuracy between different algorithms.

Algorithms	CNN	VGG16	VGG19	ResNet50
ACC	0.8907	0.9721	0.9697	0.9697
InceptionV3	**Xception**	**Voting**	**PickOver**	**Weighted**
0.8372	0.7538	0.9736	**0.9862**	0.9857

TABLE 9. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
CNN	0.8269	0.8461	0.8269	0.8653	0.8076	0.9723
VGG16+	**0.9942**	0.9692	**1.0000**	**1.0000**	0.9962	**0.9981**
VGG19+	0.9923	**0.9885**	**1.0000**	0.9981	**1.0000**	0.9923
ResNet50+	0.9923	0.9885	**1.0000**	0.9981	**1.0000**	0.9923
InceptionV3+	0.7385	0.7231	0.875	0.8135	0.7788	0.8827
Xception+	0.7308	0.7308	0.7962	0.825	0.7923	0.7500
Voting	0.9923	0.9788	**1.0000**	0.9981	**1.0000**	0.9923
PickOver	**0.9942**	**0.9885**	**1.0000**	**1.0000**	**1.0000**	**0.9981**
Weighted	0.9923	0.9788	**1.0000**	0.9981	**1.0000**	0.9923

TABLE 10. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
CNN	0.9230	0.9038	0.9615	0.8653	0.9230	0.9615
VGG16+	0.9865	0.9923	0.9942	**1.0000**	0.9942	0.9923
VGG19+	**0.9904**	**0.9962**	**1.0000**	**1.0000**	**0.9981**	**1.0000**
ResNet50+	**0.9904**	**0.9962**	**1.0000**	**1.0000**	**0.9981**	**1.0000**
InceptionV3+	0.8058	0.8038	0.85	0.8269	0.8558	0.8558
Xception+	0.7346	0.7692	0.8077	0.7192	0.7327	0.8308
Voting	0.9865	**0.9962**	**1.0000**	**1.0000**	**0.9981**	0.9981
PickOver	**0.9904**	**0.9962**	**1.0000**	**1.0000**	**0.9981**	**1.0000**
Weighted	**0.9904**	**0.9962**	**1.0000**	0.9981	**0.9981**	**1.0000**

TABLE 11. Comparisons of average test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	CNN	VGG16+	VGG19+	ResNet50+
ACC	0.8907	0.9931	0.9963	0.9963
InceptionV3+	**Xception+**	**Voting**	**PickOver**	**Weighted**
0.8175	0.7683	0.9950	**0.9971**	0.9954

Table 8 presents the average accuracies of the nine algorithms which are computed from Tables 6 and 7. As shown in Table 8, the proposed TECNNs have higher test accuracies than other algorithms, where the PickOver is the best. The average test accuracy provided by the PickOver is 9.55% higher than the CNN and 1.65% higher than the most effective TCNN algorithm (i.e. VGG16) in the experiments.

Tables 9-11 present the results of both the fine-tuning DCNNs and their TECNNs. Table 11 shows the average accuracies. Similar with the TECNNs that does not use fine-tuning, the PickOver exhibits the best performance in average. The voting and weighted methods can also achieve better performance on some cases.

TABLE 12. Comparisons of average test accuracy between from internet.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
VGG16	0.9769	**0.9750**	0.9712	0.9654	0.9846	0.9673
VGG16+	**0.9942**	0.9692	**1.0000**	**1.0000**	**0.9962**	**0.9981**
VGG19	0.9788	0.9712	0.9692	0.9558	0.9808	0.9577
VGG19+	**0.9923**	**0.9885**	**1.0000**	**0.9981**	**1.0000**	**0.9923**
ResNet50	0.9788	0.9712	0.9692	0.9558	0.9808	0.9577
ResNet50+	**0.9923**	**0.9885**	**1.0000**	**0.9981**	**1.0000**	**0.9923**
InceptionV3	**0.8423**	**0.8481**	0.8481	**0.8212**	**0.8135**	0.8346
InceptionV3+	0.7385	0.7231	**0.8750**	0.8135	0.7788	**0.8827**
Xception	**0.7577**	**0.7654**	0.7365	0.7500	0.7635	0.7385
Xception+	0.7308	0.7308	**0.7962**	**0.825**	**0.7923**	**0.7500**

TABLE 13. Comparisons of average test accuracy between from internet.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
VGG16	0.9788	0.9558	0.9750	0.9731	0.9692	0.9731
VGG16+	**0.9865**	**0.9923**	**0.9942**	**1.0000**	**0.9942**	**0.9923**
VGG19	0.9654	0.9615	0.9750	0.9750	0.9692	0.9769
VGG19+	**0.9904**	**0.9962**	**1.0000**	**1.0000**	**0.9981**	**1.0000**
ResNet50	0.9654	0.9615	0.9750	0.9750	0.9692	0.9769
ResNet50+	**0.9904**	**0.9962**	**1.0000**	**1.0000**	**0.9981**	**1.0000**
InceptionV3	**0.8519**	**0.8538**	0.8481	**0.8269**	0.8231	0.8346
InceptionV3+	0.8058	0.8038	**0.8500**	**0.8269**	**0.8558**	**0.8558**
Xception	**0.7385**	0.7558	0.7615	**0.7846**	**0.7365**	0.7577
Xception+	0.7346	**0.7692**	**0.8077**	0.7192	0.7327	**0.8308**

TABLE 14. The experimental results on comparison methods.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
T_SM	**0.9942**	0.9769	**1.0000**	0.9942	**1.0000**	**0.9962**
T__SVM	0.9885	**0.9827**	**1.0000**	**0.9981**	**1.0000**	0.9923
T_SM_SVM	**0.9944**	0.9769	**1.0000**	**0.9981**	**1.0000**	0.9923
Algorithms	**Data7**	**Data8**	**Data9**	**Data10**	**Data11**	**Data12**
T_SM	0.9942	**1.0000**	**1.0000**	**1.0000**	0.9962	**1.0000**
T_SVM	0.9942	0.9962	**1.0000**	0.9962	**0.9981**	0.9981
T_SM_SVM	**0.9985**	0.9962	**1.0000**	0.9962	**0.9981**	**1.0000**

Tables 12-13 show the comparison between the two methods of using fine-tuning and not using fine-tuning. The fine-tuning methods are suffixed with "+". It can be observed that the VGG16+, VGG19+ and Resnet50+ exhibit higher accuracies than the versions of not using fine-tuning on all datasets. The inceptionV3+ and Xception + achieve higher accuracies on part of datasets. In average, the fine-tuning methods exhibit higher generalizability than the version of not using fine-tuning. However, fine-tuning may lead to overfitting to an extent, so as shown in Table 13, the methods of fine-tuning have lower accuracies on few cases. That indicates the generalizability may be decreased. Table 14 compare the performance between methods of using different linear classifiers in the full-connection layer. T_SM uses the Softmax classifier in the full-connection layer, and T_SVM uses the SVM. T_SM_SVM combines the Softmax and SVM. It can be observed from Table 14 that, T_SVM and T_SM_SVM can achieve higher accuracy than the T_SM on some cases.

C. IMAGE CLASSIFICATION ON CALTECH

These experimental datasets are randomly selected from the image dataset Caltech256. The datasets are constituted of 17 classes of pictures, and each class contains 130 pictures. Each dataset is constituted of two classes of pictures with 260 pictures. Table 15 provides the specific information of those datasets.

TABLE 15. Data sets selected from the caltec.

Data set	Data1	Data2	Data3	Data4	Data5	Data6
First class	BaseballGlove	Bread maker	Hammock	Ladder	Lightning	Mattress
Second lass	Billiards	Grapes	Hot Tub	Lighthouse	Mars	Minaret
Data7	**Data8**	**Data9**	**Data10**	**Data11**	**Data12**	
Mussels	Teepee	Lighting	Billiards	Hammock	Ladder	
Raccoon	Treadmill	Clutter	&Mars	Mattress	Treadmill	

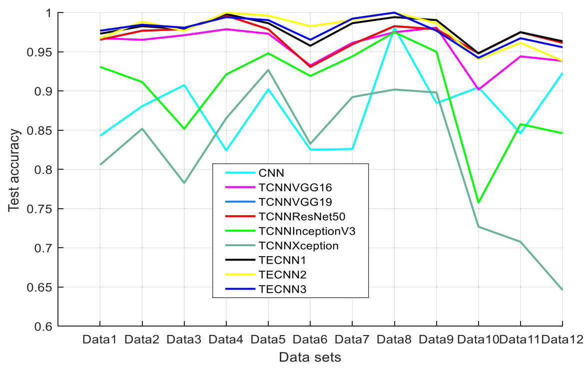


FIGURE 5. The comparison of test accuracy on different algorithms.

Fig. 5 presents the experimental results. Similar with Fig. 4, our proposed algorithms and other TCNN algorithms, TCNNVGG16, TCNNVGG19 and TCNNResNet50, exhibit higher test accuracies than the conventional CNN on most of the datasets except data2, data9 and data 10. The weighted method presents higher test accuracy than the conventional CNNs on all datasets only except data9. In addition, the TECNNs have higher test accuracies than the TCNN algorithms. Different from the case in Fig. 3, the weighted method almost has the highest test accuracy instead of the PickOver method. It shows that the proposed three TECNN algorithms have different ensemble advantages for different datasets. Tables 16 and 17 provide details. The boldface corresponds to the highest accuracy of the algorithms. Table 18 provides the average accuracies of the nine algorithms achieved from Tables 16 and 17. As shown in Table 18, the proposed three TECNNs have higher test accuracies than other algorithms, where the weighted method is the best. The proposed PickOver provides 5.85% higher accuracy than the most effective TCNN algorithm, TCNNVGG19.

The PickOver and Weighted methods have the similar mechanism to find the best combination of TECNNs. So, they present almost the same performance on many datasets. Despite of this, they exhibit different performance on some datasets, such as the experimental results in Table 16 and 17.

It can be observed from Tables 19, 20, 21 that, similar with the experimental results in Tables 9, 10, 11, the PickOver exhibits the best performance in comparison with other methods. The voting and weighted methods can achieve better

TABLE 16. Comparisons of test accuracy between different algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
CNN	0.9230	0.9807	0.8076	0.8653	0.9423	0.8653
VGG16	0.9135	**0.9173**	0.9250	**0.9538**	**0.9250**	0.9212
VGG19	**0.9346**	0.9154	**0.9327**	0.9404	0.9231	**0.9288**
ResNet50	0.9346	0.9154	0.9327	0.9404	0.9231	0.9288
InceptionV3	0.8481	0.8731	0.8827	0.8865	0.8769	0.8846
Xception	0.7904	0.8038	0.8038	0.8077	0.7846	0.8019
Voting	0.9558	0.9423	0.9481	0.9692	0.9500	0.9519
PickOver	0.9404	0.9308	0.9404	0.9615	0.9365	0.9442
Weighted	**0.9923**	**0.9885**	**0.9827**	**0.9769**	**0.9962**	**0.9827**

TABLE 17. Comparisons of test accuracy between different algorithms.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
CNN	0.8461	0.9038	1.0000	0.9807	0.8846	0.8461
VGG16	**0.9346**	**0.9327**	0.9250	0.9135	0.9231	**0.9288**
VGG19	0.9288	0.9308	**0.9269**	**0.9154**	**0.9250**	0.9269
ResNet50	0.9288	0.9308	0.9269	0.9154	0.9250	0.9269
InceptionV3	0.8865	0.8788	0.8615	0.8769	0.8577	0.8808
Xception	0.8327	0.7846	0.7923	0.8154	0.7750	0.8096
Voting	0.9558	0.9500	0.9500	0.9519	0.9519	0.9558
PickOver	0.9481	0.9481	0.9442	0.9288	0.9365	0.9423
Weighted	**0.9865**	**0.9788**	**0.9865**	**0.9865**	**0.9846**	**0.9885**

TABLE 18. Comparisons of average test accuracy between different algorithms.

Algorithms	CNN	VGG16	VGG19	ResNet50
ACC	0.9038	0.9261	0.9274	0.9274
InceptionV3	**Xception**	**Voting**	**PickOver**	**Weighted**
0.8745	0.8001	0.9527	0.9418	**0.9859**

TABLE 19. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
CNN	0.8461	0.9038	**1.0000**	0.9807	0.8846	0.8461
VGG16+	**0.9981**	**0.9942**	0.9981	0.9712	0.9923	0.9981
VGG19+	0.9942	0.9923	0.9885	**0.9923**	**1.0000**	**1.0000**
ResNet50+	0.9942	0.9923	0.9885	**0.9923**	**1.0000**	**1.0000**
InceptionV3+	0.8231	0.9173	0.7442	0.8212	0.9558	0.8635
Xception+	0.7500	0.9308	0.6981	0.7596	0.8865	0.8019
Voting	0.9923	0.9923	0.9942	0.9827	0.9923	**1.0000**
PickOver	**0.9981**	**0.9942**	**0.9981**	**0.9923**	**1.0000**	**1.0000**
Weighted	0.9904	0.9923	0.9981	0.9808	0.9942	**1.0000**

performance on some cases. Tables 22, 23 show that the T_SVM exhibits the highest classification accuracy than other two methods. Tables 24, 25 show that the fine-tuning methods achieve better generalizability; at the same time, fine-tuning may lead to overfitting on some cases, so classification accuracy is decreased.

D. IMAGE CLASSIFICATION ON ImageNet

In this section, to generate small data sets, two classes of data are randomly selected from the latest ImageNet to form each dataset, where the ImageNet is available at: http://www.image-net.org. The content of the ImageNet is continuously updated, and the generated datasets used in this section are not included in the original trained datasets for

TABLE 20. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
CNN	0.8461	0.9038	**1.0000**	0.9807	0.8846	0.8461
VGG16+	0.9981	**1.0000**	**1.0000**	**0.9981**	0.9788	**0.9462**
VGG19+	**1.0000**	**1.0000**	**1.0000**	**0.9981**	**0.9885**	**0.9462**
ResNet50+	**1.0000**	**1.0000**	**1.0000**	**0.9981**	**0.9885**	**0.9462**
InceptionV3+	0.6808	0.8538	0.9712	0.9365	0.7904	0.7404
Xception+	0.6385	0.8077	0.8981	0.9442	0.7577	0.7462
Voting	**1.0000**	**1.0000**	**1.0000**	**1.0000**	0.9865	0.9538
PickOver	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**0.9885**	**0.9635**
Weighted	**1.0000**	**1.0000**	**1.0000**	0.9981	0.9846	**0.9615**

TABLE 21. Comparisons of average test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	CNN	VGG16+	VGG19+	ResNet50+
ACC	0.9038	0.9894	0.9917	0.9917
InceptionV3+	**Xception+**	**Voting**	**PickOver**	**Weighted**
0.8415	0.8016	0.9912	**0.9946**	0.9917

TABLE 22. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
T_SM	0.9885	0.9865	**0.9962**	0.9808	**0.9942**	**1.0000**
T_SVM	0.9923	**0.9942**	0.9904	**0.9846**	**0.9942**	**1.0000**
T_SM_SVM	**0.9942**	**0.9942**	0.9904	0.9808	**0.9942**	**1.0000**
Algorithms	**Data7**	**Data8**	**Data9**	**Data10**	**Data11**	**Data12**
T_SM	0.9942	**1.0000**	**1.0000**	**1.0000**	**0.9865**	0.9519
T_SVM	**1.0000**	**1.0000**	**1.0000**	0.9981	0.9827	**0.9577**
T_SM_SVM	**1.0000**	**1.0000**	**1.0000**	0.9981	0.9846	0.9538

TABLE 23. Comparisons of average test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	T_SM	T_SVM	T_SM_SVM
ACC	0.9899	**0.9912**	0.9909

TABLE 24. Comparisons of average test accuracy between methods of using fine-tuning and not using fine-tuning.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
VGG16	0.9135	0.9173	0.9250	0.9538	0.9250	0.9212
VGG16+	**0.9981**	**0.9942**	**0.9981**	**0.9712**	**0.9923**	**0.9981**
VGG19	0.9346	0.9154	0.9327	0.9404	0.9231	0.9288
VGG19+	**0.9942**	**0.9923**	**0.9885**	**0.9923**	**1.0000**	**1.0000**
ResNet50	0.9346	0.9154	0.9327	0.9404	0.9231	0.9288
ResNet50+	**0.9942**	**0.9923**	**0.9885**	**0.9923**	**1.0000**	**1.0000**
InceptionV3	**0.8481**	0.8731	**0.8827**	**0.8865**	0.8769	**0.8846**
InceptionV3+	0.8231	**0.9173**	0.7442	0.8212	**0.9558**	0.8635
Xception	**0.7904**	0.8038	**0.8038**	**0.8077**	0.7846	**0.8019**
Xception+	0.7500	**0.9308**	0.6981	0.7596	**0.8865**	**0.8019**

the five TCNNs, which are trained by the 2014 version of ImageNet. Therefore, the original datasets and target datasets are different. Table 26 lists these datasets. Similar with the experiments, to generate small-scale datasets, each class contains 130 images that are randomly selected, and each dataset is formed by 260 images. Moreover, 80% of each dataset, i.e. 208, were used for training, and the remaining 20% were used

TABLE 25. Comparisons of average test accuracy between methods of using fine-tuning and not using fine-tuning.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
VGG16	0.9346	0.9327	0.9250	0.9135	0.9231	0.9288
VGG16+	**0.9981**	**0.9942**	**0.9981**	**0.9712**	**0.9923**	**0.9981**
VGG19	0.9288	0.9308	0.9269	0.9154	0.9250	0.9269
VGG19+	**0.9942**	**0.9923**	**0.9885**	**0.9923**	**1.0000**	**1.0000**
ResNet50	0.9288	0.9308	0.9269	0.9154	0.9250	0.9269
ResNet50+	**0.9942**	**0.9923**	**0.9885**	**0.9923**	**1.0000**	**1.0000**
InceptionV3	**0.8865**	0.8788	**0.8615**	**0.8769**	0.8577	**0.8808**
InceptionV3+	0.8231	**0.9173**	0.7442	0.8212	**0.9558**	0.8635
Xception	**0.8327**	0.7846	**0.7923**	**0.8154**	0.7750	**0.8096**
Xception+	0.7500	**0.9308**	0.6981	0.7596	**0.8865**	0.8019

TABLE 26. Experimental data sets from imagenet.

Data set	Data1	Data2	Data3	Data4	Data5	Data6
First class	Bicycle	Bicycle	Bicycle	Bicycle	Bicycle	Container
Second lass	Container House	IronNail	Masks	Necklace	Nipple	House IronNail

Data7	Data8	Data9	Data10	Data11	Data12
Container House Masks	Container House Necklace	Container House Nipple	IronNail Masks	IronNail Necklace	IronNail Nipple

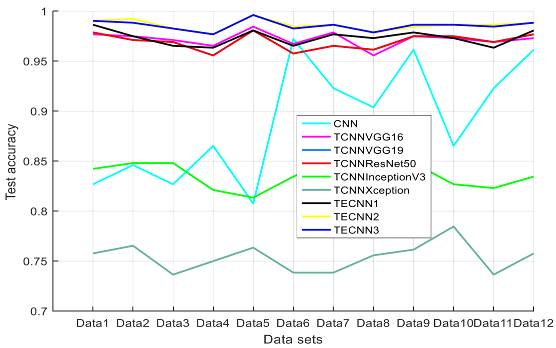


FIGURE 6. The comparison of test accuracy on different algorithms.

TABLE 27. Comparisons of test accuracy between different algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
CNN	0.8430	0.8807	0.9076	0.8243	0.9023	0.8253
VGG16	**0.9673**	0.9654	0.9712	0.9788	0.9731	**0.9327**
VGG19	0.9654	**0.9769**	**0.9788**	**0.9962**	**0.9788**	0.9308
ResNet50	0.9654	**0.9769**	**0.9788**	**0.9962**	**0.9788**	0.9308
InceptionV3	0.9308	0.9115	0.8519	0.9212	0.9481	0.9192
Xception	0.8058	0.8519	0.7827	0.8654	0.9269	0.8327
Voting	0.9731	0.9827	0.9788	0.9981	0.9865	0.9577
PickOver	0.9673	**0.9885**	0.9769	**1.0000**	**0.9962**	**0.9827**
Weighted	**0.9769**	0.9846	**0.9808**	0.9942	0.9904	0.9654

for the test. The training structure of the CNN is the same as the previous.

Fig. 6 presents the experimental results. It can be seen that the proposed algorithms and other TCNN algorithms, i.e., TCNNVGG16, TCNNVGG19 and TCNNResNet50, have higher test accuracies than the conventional CNNs on most datasets except data8. The test accuracies of the proposed TECNNs are higher than the conventional CNNs on all datasets. In addition, the proposed TECNNs provide higher test accuracies than the TCNNs.

Tables 27 and 28 provide the details. Table 29, derived from Tables 27 and 28, presents the average accuracies of the eight algorithms. As shown in Table 29, the proposed three TECNNs have higher test accuracies than other algorithms,

TABLE 28. Comparisons of test accuracy between different algorithms.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
CNN	0.8261	0.9807	0.8846	0.9047	0.8461	0.9230
VGG16	**0.9615**	0.9750	**0.9808**	0.9019	0.9442	0.9385
VGG19	0.9596	**0.9827**	0.9788	**0.9481**	**0.9750**	**0.9615**
ResNet50	0.9596	**0.9827**	0.9788	**0.9481**	**0.9750**	**0.9615**
InceptionV3	0.9442	0.9750	0.9500	0.7577	0.8577	0.8462
Xception	0.8923	0.9019	0.8981	0.7269	0.7077	0.6462
Voting	0.9865	0.9942	**0.9904**	**0.9481**	**0.9750**	**0.9635**
PickOver	0.9904	**1.0000**	0.9865	0.9404	0.9615	0.9385
Weighted	**0.9923**	**1.0000**	0.9769	0.9423	0.9673	0.9558

TABLE 29. Comparisons of average test accuracy.

Algorithms	CNN	VGG16	VGG19	ResNet50
ACC	0.8790	0.9575	0.9694	0.9694
InceptionV3	**Xception**	**Voting**	**PickOver**	**Weighted**
0.9011	0.8199	**0.9779**	0.9774	0.9772

TABLE 30. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
CNN	0.8430	0.8807	0.9076	0.8243	0.9023	0.8253
VGG16+	**0.9808**	0.9942	0.9750	0.9981	**0.9962**	0.9981
VGG19+	**0.9808**	**1.0000**	**0.9962**	**1.0000**	**0.9962**	**1.0000**
ResNet50+	**0.9808**	**1.0000**	**0.9962**	**1.0000**	**0.9962**	**1.0000**
InceptionV3+	0.8212	0.8519	0.8365	0.8712	0.9346	0.8577
Xception+	0.7442	0.8577	0.7154	0.8423	0.9096	0.7500
Voting	**0.9885**	0.9981	0.9904	**1.0000**	**0.9962**	**1.0000**
PickOver	**0.9885**	**1.0000**	**0.9962**	**1.0000**	**0.9962**	**1.0000**
Weighted	**0.9865**	0.9981	0.9865	**1.0000**	**0.9962**	0.9962

TABLE 31. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
CNN	0.8261	0.9807	0.8846	0.9047	0.8461	0.9230
VGG16+	0.9942	**1.0000**	**0.9981**	0.9885	0.9692	0.9654
VGG19+	**1.0000**	**1.0000**	**0.9981**	**0.9904**	**0.9885**	**0.9731**
ResNet50+	**1.0000**	**1.0000**	**0.9981**	**0.9904**	**0.9885**	**0.9731**
InceptionV3+	0.8404	0.9	0.8827	0.675	0.7288	0.6577
Xception+	0.7577	0.8	0.8019	0.6673	0.6442	0.6519
Voting	**1.0000**	**1.0000**	**0.9981**	**0.9904**	0.9788	0.9673
PickOver	**1.0000**	**1.0000**	**0.9981**	**0.9904**	**0.9885**	**0.9731**
Weighted	**1.0000**	**1.0000**	**0.9981**	0.9885	0.9827	**0.9750**

TABLE 32. Comparisons of average test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	CNN	VGG16+	VGG19+	ResNet50+
ACC	0.8790	0.9881	0.9936	0.9936
InceptionV3+	**Xception+**	**Voting**	**PickOver**	**Weighted**
0.8215	0.7619	0.9923	**0.9942**	0.9923

TABLE 33. Comparisons of test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
T_SM	**0.9962**	0.9981	**0.9942**	**1.0000**	**0.9962**	0.9923
T_SVM	0.9827	**1.0000**	**0.9942**	**1.0000**	**0.9962**	**0.9962**
T_SM_SVM	0.9827	0.9981	0.9923	**1.0000**	**0.9962**	**0.9962**
Algorithms	**Data7**	**Data8**	**Data9**	**Data10**	**Data11**	**Data12**
T_SM	**1.0000**	**1.0000**	0.9942	**0.9904**	**0.9904**	0.9692
T_SVM	**1.0000**	0.9962	**0.9981**	0.9827	0.9885	0.9712
T_SM_SVM	**1.0000**	**1.0000**	**0.9981**	0.9865	0.9846	**0.9750**

TABLE 34. Comparisons of average test accuracy between traditional algorithms and fine-tuning ensemble transferred algorithms on the 12 data sets from ImageNet.

Algorithms	T_SM	T_SVM	T_SM_SVM
ACC	**0.9934**	0.9921	0.9925

TABLE 35. Comparisons of average test accuracy between methods of using fine-tuning and not using fine-tuning.

Algorithms	Data1	Data2	Data3	Data4	Data5	Data6
VGG16	0.9673	0.9654	0.9712	0.9788	0.9731	0.9327
VGG16+	**0.9808**	**0.9942**	**0.9750**	**0.9981**	**0.9962**	**0.9981**
VGG19	0.9654	0.9769	0.9788	0.9962	0.9788	0.9308
VGG19+	**0.9808**	**1.0000**	**0.9962**	**1.0000**	**0.9962**	**1.0000**
ResNet50	0.9654	0.9769	0.9788	0.9962	0.9788	0.9308
ResNet50+	**0.9808**	**1.0000**	**0.9962**	**1.0000**	**0.9962**	**1.0000**
InceptionV3	**0.9308**	**0.9115**	**0.8519**	**0.9212**	**0.9481**	**0.9192**
InceptionV3+	0.8212	0.8519	0.8365	0.8712	0.9346	0.8577
Xception	**0.8058**	0.8519	**0.7827**	**0.8654**	**0.9269**	**0.8327**
Xception+	0.7442	**0.8577**	0.7154	0.8423	0.9096	0.7500

TABLE 36. Comparisons of average test accuracy between methods of using fine-tuning and not using fine-tuning.

Algorithms	Data7	Data8	Data9	Data10	Data11	Data12
VGG16	0.9615	0.9750	0.9808	0.9019	0.9442	0.9385
VGG16+	**0.9942**	**1.0000**	**0.9981**	**0.9885**	**0.9692**	**0.9654**
VGG19	0.9596	0.9827	0.9788	0.9481	0.9750	0.9615
VGG19+	**1.0000**	**1.0000**	**0.9981**	**0.9904**	**0.9885**	**0.9731**
ResNet50	0.9596	0.9827	0.9788	0.9481	0.9750	0.9615
ResNet50+	**1.0000**	**1.0000**	**0.9981**	**0.9904**	**0.9885**	**0.9731**
InceptionV3	**0.9442**	**0.9750**	**0.9500**	**0.7577**	**0.8577**	**0.8462**
InceptionV3+	0.8404	0.9000	0.8827	0.6750	0.7288	0.6577
Xception	**0.8923**	**0.9019**	**0.8981**	**0.7269**	**0.7077**	0.6462
Xception+	0.7577	0.8000	0.8019	0.6673	0.6442	**0.6519**

in which the voting method is the best. In particular, the proposed voting method provide 0.85% higher accuracy than other most effective TCNN algorithm (i.e. TCNN_VGG19) in this experiment. As shown in Tables 27 and 28, the proposed weighted method presents the highest accuracy in comparison with other TCNN algorithms on the data6, i.e. 5% higher than the most effective TCNN algorithm TCNN_VGG16 on data12.

Similar with the experimental results, Tables 30, 31, 32 still exhibit better generalizability of the PickOver in comparison with other methods. Tables 33-34 show that the T_SM has the highest classification accuracy in these ImageNet datasets. As shown in Tables 35-36, different from the experimental results on the Caltech, the two TCNNs, i.e. InceptionV3 and Xception, exhibit higher classification accuracy than the version of using fine-tuning. It indicates that fine-tuning lead to obvious overfitting.

V. CONCLUSIONS

To make full use of the existing multiple TCNNs and improve their generalizability, this study proposes three ensemble TCNNs by introducing the ensemble ideas. The experimental results show that these TECNNs exhibit better generalizability than the conventional CNNs and a single TCNN. In comparison with the CNN and five widely used TCNNs, the proposed TECNNs enhance the average test accuracy by 1.65%, 5.85% and 7.58% respectively on the internet datasets and two benchmark small-scale datasets, and the highest test accuracy by 1.73%, 7.12% and 8.85%.

Due to space limitations, only some common ensemble methods are combined into the proposed TECNNs. In the future, we will introduce more ensemble technologies to achieve better performance.

REFERENCES

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 886–893.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. Workshop Stat. Learn. Comput. Vis. (ECCV)*, 2004, vol. 44, no. 247, pp. 1–22.

[4] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial Pyramid matching for recognizing natural scene categories," in *Proc. CVPR*, 2006, pp. 2169–2178.

[5] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher Kernelfor large-scale image classification," in *Proc. ECCV*, 2010, pp. 143–156.

[6] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. ICCV*, Oct. 2003, pp. 1470–1477.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.

[8] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Inst. Technol., Pasadena, CA, USA, Tech. Rep., 2007.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman "The PASCAL visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, 2010.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255.

[11] F. Rosenblatt, "The perceptron: A perceiving and recognizing automaton," Project PARA, Cornell Aeronaut. Lab, Buffalo, NY, USA, Tech. Rep. 85-460-1, 1957.

[12] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *J. Physiol.*, vol. 148, no. 3, pp. 574–591, 1959.

[13] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[15] K. J. Lang and G. E. Hinton, "A time delay neural network architecture for speech recognition," CMU, Tech. Rep. CMUCS-88-152, 1988.

[16] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[18] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. IEEE Comput. Soc. Int. Conf. Document Anal. Recognit.*, Aug. 2003, pp. 958–963.

[19] M. Osadchy, Y. Le Cun, and M. L. Miller, "Synergistic face detection and pose estimation with energy-based models," *J. Mach. Learn. Res.*, vol. 8, pp. 1197–1215, Jan. 2007.

[20] R. Vaillant, C. Monrocq, and Y. Le Cun, "An original approach for the localization of objects in images," in *Proc. Int. Conf. Artif. Neural Netw. (IET)*, 1993, pp. 26–30.

[21] Y. Lecun, F. J. Huang, and L. Bottou, "Learning methods for generic object recognition with invariance to pose and lighting," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun./Jul. 2004, pp. II-97–II-104.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.

[24] D. Held, S. Thrun, and S. Savarese, "Robust single-view instance recognition," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 2152–2159.

[25] Y.-X. Wang and M. Hebert, "Model recommendation: Generating object detectors from few samples," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. IEEE Comput. Soc.*, Jun. 2015, pp. 1619–1628.

[26] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1717–1724.

[27] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. CVPRW*, 2014, pp. 512–519.

[28] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability for a generic ConvNet representation," *IEEE Pattern. Anal. Mach. Intell.*, vol. 38, no. 9, pp. 1790–1802, Sep. 2016.

[29] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 3320–3328.

[30] S. Yang and D. Ramanan, "Multi-scale recognition with DAG-CNNs," in *Proc. ICCV*, 2015, pp. 1215–1223.

[31] L. G. Hafemann, L. S. Oliveira, P. R. Cavalin, and R. Sabourin, "Transfer learning between texture classification tasks using convolutional neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2015, pp. 1–7.

[32] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2013, pp. 580–587.

[33] R. Zhang, Y. Zheng, T. W. C. Mak, R. Yu, S. H. Wong, J. Y. W. Lau, and C. C. Y. Poon, "Automatic detection and classification of colorectal polyps by transferring low-level CNN features from non-medical domain," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 41–47, Jan. 2016.

[34] Z. Li, Y. Song, I. Mcloughlin, and L. Dai, "Compact convolutional neural network transfer learning for small-scale image classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Mar. 2016, pp. 2737–2741.

[35] F. Liu, G. Lin, and C. Shen, "CRF learning with CNN features for image segmentation," *Pattern Recognit.*, vol. 48, no. 10, pp. 2983–2992, Oct. 2015.

[36] P. Sollich and A. Krogh, "Learning with ensembles: How over-fitting can be useful," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 8, 1996, pp. 190–196.

[37] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[38] A. J. Sharkey, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*. New York, NY, USA: Springer-Verlag, 1999.

[39] K. Liu, M. Zhang, and Z. Pan, "Facial expression recognition with CNN ensemble," in *Proc. IEEE Int. Conf. Cyberworlds*, Sep. 2016, pp. 163–166.

[40] G. Antipov, S. A. Berrani, and J.-L. Dugelay, "Minimalistic CNN-based ensemble model for gender prediction from face images," *Pattern Recognit. Lett.*, vol. 70, pp. 59–65, Jan. 2016.

[41] C. Ding and D. Tao, "Trunk-branch ensemble convolutional neural networks for video-based face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 1002–1014, Apr. 2016.

[42] T. Wan, J. Cao, J. Chen, and Z. Qin, "Automated grading of breast cancer histopathology using cascaded ensemble with combination of multi-level image features," *Neurocomputing*, vol. 229, pp. 34–44, Mar. 2017.

[43] H. K. Huang, C.-F. Chiu, C.-H. Kuo, Y.-C. Wu, N. N. Y. Chu, and P.-C. Chang, "Mixture of deep CNN-based ensemble model for image retrieval," in *Proc. IEEE Global Conf. Consum. Electron.*, Oct. 2016, pp. 1–2.

[44] Q. Hu, P. Wang, C. Shen, A. van den Hengel, and F. Porikli, "Pushing the limits of deep CNNs for pedestrian detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 6, pp. 1358–1368, Jun. 2016.

[45] Y. Li and T. Shibuya, "Malphite: A convolutional neural network and ensemble learning based protein secondary structure predictor," in *Proc. IEEE Int. Conf. Bioinformat. Biomed.*, Nov. 2015, pp. 1260–1266.

[46] L. Akhtyamova, A. Ignatov, and J. Cardiff, "A large-scale CNN ensemble for medication safety analysis," in *Proc. Int. Conf. Appl. Natural Lang. Inf. Syst.* Cham, Switzerland: Springer, 2017, pp. 247–253.

[47] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[48] B. Efron, *An Introduction to the Bootstrap*. London, U.K.: Chapman & Hall, 1995.

[49] R. E. Schapire, "The strength of weak learnability," in *Proc. Annu. Symp. Found. Comput. Sci.*, Oct./Nov. 1989, pp. 28–33.

[50] Y. Freund, "Boosting a weak learning algorithm by majority," 1990, pp. 202–216.

[51] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*. Berlin, Germany: Springer, 1995, pp. 119–139.

[52] J. B. Hampshire and A. H. Waibel, "A novel objective function for improved phoneme recognition using time," *IEEE Trans. Neural Netw.*, vol. 1, no. 2, pp. 216–228, Jun. 1990.

[53] R. Maclin and J. W. Shavlik, "Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 1995, pp. 524–530.

[54] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation and active learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 1994, pp. 231–238.

[55] Y. Jin and B. Sendhoff, "Reducing fitness evaluations using clustering techniques and neural network ensembles," in *Proc. Genetic Evol. Comput. Conf. Evol. Comput. (GECCO)*, Seattle, WA, USA, Jun. 2004, pp. 688–699.

[56] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Trans. Syst., Man, Cybern, B, Cybern.*, vol. 28, no. 3, pp. 417–425, Jun. 1998.

[57] D. W. Opitz and J. W. Shavlik, "Actively searching for an effective neural network ensemble," *Connection Sci.*, vol. 8, nos. 3–4, pp. 337–354, 1996.

[58] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," Tech. Rep., 1993, pp. 342–358.

[59] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.

[60] C. J. Merz and M. J. Pazzani, "Combining neural network regression estimates with regularized linear weights," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 1997, pp. 564–570.

[61] D. Jimenez, "Dynamically weighted ensemble neural networks for classification," in *Proc. IEEE Int. Joint Conf. Neural Netw. World Congr. Comput. Intell.*, vol. 1, May 1998, pp. 753–756.

[62] N. Ueda, "Optimal linear combination of neural networks for improving classification performance," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 2, pp. 207–215, Feb. 2000.

[63] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, 2012.

[64] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," in *Proc. IEEE/INNS Joint Conf. Neural Netw.*, Nagoya, Japanm, Oct. 1993, pp. 181–214.

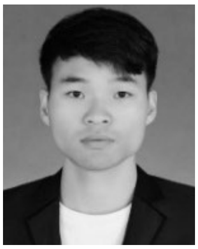
[65] A. F. Agarap, "An architecture combining convolutional neural network (CNN) and support vector machine (SVM) for image classification," Tech. Rep., 2017.

[66] Z. H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artif. Intell.*, vol. 137, nos. 1–2, pp. 239–263, 2002.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2015, pp. 770–778.

[68] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2818–2826.

SHUYIN XIA received the B.S. and M.S. degrees in computer science from the Chongqing University of Technology, China, in 2008 and 2012, respectively, and the Ph.D. degree from the College of Computer Science, Chongqing University, China. He is currently an Associate Professor with the College of Computer Science and Technology, Chongqing University of Posts and Telecommunications (CQUPT), where he is also the Executive Deputy Director with the Big Data and Network Security Joint Lab. His research results have been published at many prestigious journals, such as TKDE, *Information Science*. His research interests include classifiers and granular computing.

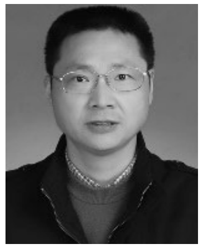


YULONG XIA received the bachelor's degree in computer science and technology from the Jingchu University of Technology, in 2018. He is currently pursuing the master's degree in computer science with the Chongqing University of Posts and Telecommunications. His research interests include machine learning and deep learning.

HONG YU received the B.S. degree in engineering mechanics from Nanchang Hangkong University, in 1994, the M.S. degree in computer science from the Chongqing University of Posts and Telecommunications, in 2002, and the Ph.D. degree in computer sciences from Chongqing University, in 2003. She is currently a Professor with the College of Computer Science and Technology, Chongqing University of Posts and Telecommunications. Her research interests include classifiers, rough sets, and granular computing.

QUN LIU received the B.S. degree in engineering mechanics from Xi'an Jiaotong University, in 1991, the M.S. degree in computer science from Wuhan University, in 2002, and the Ph.D. degree in computer science from Chongqing University, in 2008. She is currently a Professor with the College of Computer Science and Technology, Chongqing University of Posts and Telecommunications. Her research interests include classifiers, rough sets, and granular computing.

YUEGUO LUO received the Ph.D. degree from Chongqing University, China, in 2017. Since 2004, he has been a Teacher with Yangtze Normal University, Chongqing. He is currently an Associate Professor. His research interests include data mining and computational intelligence.



GUOYIN WANG (M'98–SM'03) received the B.E. degree in computer software, the M.S. degree in computer software, and the Ph.D. degree in computer organization and architecture from Xi'an Jiaotong University, Xi'an, China, in 1992, 1994, and 1996, respectively. He was with the University of North Texas, USA, and the University of Regina, Canada, as a Visiting Scholar, from 1998 to1999. Since 1996, he has been with the Chongqing University of Posts and Telecommunications, Chongqing, China, where he is currently a Professor, the Ph.D. Supervisor, the Director of the Chongqing Key Laboratory of Computational Intelligence, and the Dean of the Graduate School. His research interests include data mining, machine learning, rough set, granular computing, and cognitive computing. He is the Steering Committee Chair of the International Rough Set Society (IRSS), the Vice-President of the Chinese Association for Artificial Intelligence (CAAI), and a Council Member of the China Computer Federation (CCF).

ZIZHONG CHEN received the M.S. degree in engineering from Beijing Normal University, Beijing, in 1993, the B.S. degree from the Renmin University of China, in 1994, and the Ph.D. degree from the University of Tennessee, Knoxville, TN, USA, in 1996. He is currently an Associate Professor with the University of California at Riverside. His research interests include high performance computing, parallel, and distributed systems. His research has been supported by the National Science Foundation, Department of Energy, CMG Reservoir Simulation Foundation, Abu Dhabi National Oil Company, Nvidia, and Microsoft Corporation. He is a Life Member of the ACM. He also works as a Subject Area Editor for *Parallel Computing Journal* (Elsevier) and an Associate Editor for the IEEE Transactions on Parallel and Distributed Systems.