IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 29, 2019, accepted February 25, 2019, date of publication March 7, 2019, date of current version March 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2903550

Using an Ensemble of Incrementally Fine-Tuned CNNs for Cross-Domain Object Category Recognition

XUESONG ZHANG[1,2], FEI YAN[1], YAN ZHUANG[1], HUOSHENG HU[3], (Senior Member, IEEE), AND CHUNGUANG BU[4]

[1]School of Control Science and Engineering, Dalian University of Technology, Dalian 116024, China
[2]Software Technology Institute, Dalian Jiaotong University, Dalian 116028, China
[3]School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K.
[4]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China

Corresponding author: Fei Yan (fyan@dlut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61503056 and Grant U1508208, in part by the Fundamental Scientific Research Project of Liaoning Provincial Department of Education under Grant JDL2017017, in part by the National Science Foundation of Liaoning Province of China under Grant 20180551020, and in part by the State Key Laboratory of Robotics under Grant 2017-O15.

ABSTRACT When the training data is inadequate, it is difficult to train a deep Convolutional Neural Network (CNN) from scratch with randomized initial weights. Instead, it is common to train a source CNN model on a very large data set beforehand, and then use the learned source CNN model as an initialization to train a target CNN model. In deep learning realm, this procedure is called fine-tuning a CNN. This paper presents an experimental study on how to combine a collection of incrementally fine-tuned CNN models for cross-domain and multi-class object category recognition tasks. A group of fine-tuned CNN models is trained on the target data set by incrementally transferring parameters from a source CNN model trained on a large data set initially. The last two fully-connected (FC) layers of the source CNN model are eliminated, and two New FC layers are added to make the learned new CNN model suitable for the target task. Based on Caltech-101 and Office data sets, the experimental results demonstrate the effectiveness and good performance of the proposed methods. The proposed method is more suitable for the object recognition task when there is inadequate target training data.

INDEX TERMS Convolutional neural network, object category recognition, ensemble learning, transfer learning.

I. INTRODUCTION

In recent years, deep Convolutional Neural Network (CNN) has been successfully used in object category detection and recognition tasks [1]–[6], [10]–[17]. When the training data is inadequate, it is difficult to train a deep CNN from scratch with randomized initial weights. It is common to train source CNN models on a large-scale data set initially, and then further train them on the inadequate and different distribution data to obtain the target CNN models for target tasks. This procedure is often called fine-tuning in the deep learning realm [5].

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

Since training a deep CNN model needs to estimate millions of parameters, it remains difficult to fine-tune a source CNN model on the target domain when we have inadequate target training data. In order to avoid CNN model overfitting, several methods, such as data augmentation [7], dropout [8] and weight decay [9] *etc.*, have been presented to train a deep CNN model from scratch with inadequate training data. In addition, Oquab *et al.* [10] proposed an approach to transfer parameters from some layers of a source CNN model.

Inspired by the work in [10] and ensemble learning, we study the problem of transferring parameters from a group of medium depth CNN models for cross domain object recognition tasks. A group of source CNN models are trained on the ImageNet data set in advance and then incrementally fine-tuned for the target task. The last two Fully-Connected (FC)

layers of each source CNN model have been removed and two New-FC layers are added to the end of each pruned source CNN to make the learned target CNN model suitable for the target task.

The proposed method fine-tunes the source CNN models in an incremental manner, which differs to the work in [10]. In addition, these fine-tuned CNN models are combined to make an ensemble prediction. The CNN-S, M, F architecture are used to train some source CNN models on the ImageNet data set [6]. Although other methods can be used to improve the ensemble diversity, such as bootstrap sampling the target training data or using different base learning algorithms, we focus on CNN weight parameter transfer and ensemble learning in this work. The main contribution of this paper can be summarized here.

1) A novel method is proposed to train a collection of incrementally fine-tuned CNN models for cross-domain object recognition with inadequate training data.
2) A horizontal/vertical/selective ensemble of the fine-tuned CNN models is empirically studied on two data sets. In particular, a clustering-based selective ensemble algorithm is proposed to select user specified number of models.
3) A Matlab software package is provided for research usage, which can be used to reproduce the experimental results of this paper.

The rest of this paper is organized as follows. Section II reviews some related research works. The proposed method is described in Section III. Experiments are conducted in Section IV to demonstrate the effectiveness and good performance of the proposed method. Finally, a conclusion and future work is presented in Section V.

II. RELATED WORK

Some previous works of using deep CNN for object category recognition have been studied. Krizhevsky *et al.* [11] trained a deep CNN to classify the 1.2 million high-resolution images into the 1000 different classes in the ImageNet LSVRC-2010 contest. Chatfield *et al.* [6] presented an empirical evaluation of CNN-based approaches for image classification, along with a comparison against several traditional shallow feature encoding methods. Simonyan *et al.* [12] studied the effect of the CNN

depth on its accuracy in the large-scale image recognition task. Szegedy *et al*. [13] presented a deep CNN architecture named GoogLeNet which achieved good performance in the ImageNet LSVRC-2014 contest. He *et al*. [14] proposed a residual CNN learning framework to ease the training of much deeper CNN and presented a CNN architecture named ResNet. An ensemble of ResNets won the 1st place on the ImageNet LSVRC-2015 image classification contest.

Although training a much deeper CNN is helpful to improve the accuracy on the object recognition task, it requires a very large training image set. In order to alleviate CNN overfitting, some researchers resort to CNNs combination. Kumar *et al*. [16] proposed a method for classifying medical images that use an ensemble of different CNN architectures. Wen *et al*. [17] presents an ensemble of CNNs method with probability-based fusion for facial expression recognition, where the architecture of CNN was adapted by using the convolutional rectified linear layer as the first layer and multiple hidden maxout layers. Zhao *et al.* [18] presented a deeply merge-and-run fused network based on the very deep ResNet architecture.

Other than CNN combination, adapting a pre-trained source CNN model for the target tasks has also attracted much more attentions. This procedure is known as parameter transfer in transfer learning realm. For the case of CNN parameter transfer, we assume that there are no training data but a pre-trained CNN model in the source domain and a small-scale training data set in the target domain. Girshick *et al.* [1] fine-tuned a CNN under their famous Region-based CNN (R-CNN) framework for object detection. Hoffman *et al*. [15] proposed an approach to adapt CNN-based object detectors trained on RGB images to effectively leverage depth images at test time to boost detection performance. Zhou *et al.* [19] presented a method called AIFT to naturally integrate active learning and transfer learning into a single framework. AIFT starts with a pre-trained CNN to seek "worthy" samples from the unannotated for annotation, and the fine-tuned CNN is continuously fine-tuned by incorporating newly annotated samples in each iteration. Wani et al. [20] propose a hybrid approach that integrates gain parameter based back propagation algorithm and the dropout technique and evaluate its effectiveness in the fine tuning of deep neural networks on three benchmark datasets. Tan *et al.* [21] proposed a method on the top of DenseNet called Sequential Fine-Tuning in a computer-aided diagnosis system for lung diseases. Wang *et al.* [22] proposed an image-specific fine tuning method to make a CNN model adaptive to a specific test image, which can be either unsupervised (without additional user interactions) or supervised (with additional scribbles).

III. THE PROPOSED APPROACH

A. CNN ARCHITECTURE

In this study, three pre-trained source CNN models, namely CNN-S, CNN-M and CNN-F in [6], are fine-tuned by different manners to generate a group of CNN classifiers for the target task. As illustrated in Fig. 1(a), each of the source CNN models contains 5 convolutional layers (C1-C5) and 3 fully-connected layers (FC6-FC8). Given an input tensor x, evaluating such a plain architecture source CNN model is just a matter of evaluating all the layers from left to right.

For all the source CNN models, the computational units in each layer are different, including the number of convolution filters and their receptive field size, the convolution stride and spatial pooling size etc. Hence, the learned parameter values of the three source CNN models are different on the same data set. Note that each of the source CNN models used in this work takes an image with $224 \times 224 \times 3$ pixels and produces

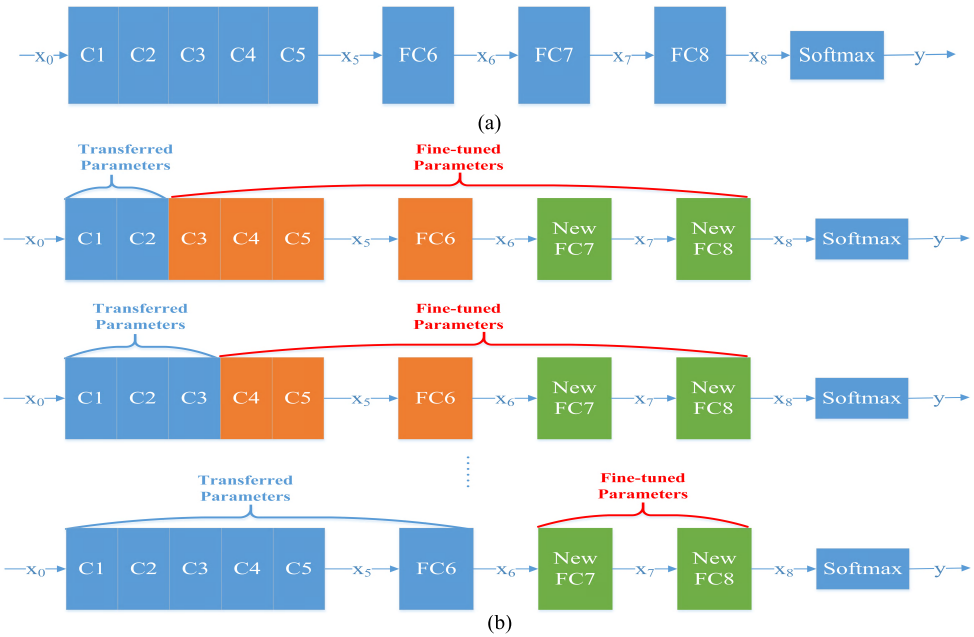


FIGURE 1. Ensemble learning with incremental parameter transfer. (a) Architecture of the source CNN model. (b) An ensemble of incrementally fine-tuned CNN models for the target task.

a distribution over the 1000 object categories on the large-scale ImageNet dataset. Please refer to [6] for the detailed description of the architecture of the CNN-S, M, F models.

For the target task, the amount of object categories is usually different from the source task. For instance, there are 101 object classes (ignoring background class) in the Caltech-101 data set rather than 1000 in ImageNet data set. Inspired by the work of [10], we remove the FC7 and FC8 layers from the source CNN model and add a novel domain adaptation module composed of New-FC7 and New-FC8 layers that use the output of FC6 layer. Note that the proposed architecture is slightly different from the CNN architecture used in [10] where Oquab *et.al.* removed the FC8 layer from the source CNN model. Here, we remove both the FC7 and FC8 layers from the source CNN model. In addition, Oquab, *et.al.* transferred the parameters in the C1-C5 and FC6-FC7 of a given source CNN model once a time. But we incrementally transfer the weight parameters of a given source CNN model several times under the ensemble learning framework, as illustrated in Fig. 1(b).

As shown in Fig. 2, the fourth and third dimensions of the convolution filters in New-FC7 and New-FC8 lays are separately set to be 2048, 1024 and 512 pixels to further improve the ensemble diversity. Note that $h\times w\times c\times n$ is used to denote a set of |n| convolution filters, which has the size h×w×c for each filter. Moreover, we presented the dimension of the convolution filters used in the New-FC7 and New-FC8 layers while ignored the bias term. Note that the bias term is an n-dimensional vector which should be learned when training a CNN model. Taking the type A of domain adaptation modules as an example, the internal units of the New-FC7 and New-FC8 layers are detailed in Fig. 3.

B. TRAINING AN ENSEMBLE OF FINE-TUNED CNN MODELS

Using the notations similar to [23], a plain architecture of CNN can be represented as a function g composed of a sequence of simpler functions f_l as follows:

$$g(x_0; w_1, \ldots, w_L) = f_L(\ldots f_2(f_1(x_0; w_1); w_2); \ldots; w_L) \quad (1)$$

where each f_l takes as input a datum x_{l-1} and a parameter vector w_l to produce as output a datum $x_l = f_l(x_{l-1}; w_l)$. The input x_0 of a CNN is an actual image, while the remaining parts $x_1, \ldots, x_L$ are the intermediate feature maps.

Specifically, each $x_l \in R^{H_l\times W_l\times C_l}$ is a 3D-array or tensor, where the first two dimensions denote the spatial dimension (i.e. Height × Width) and the third dimension denotes the number of feature channels. Note that a fourth dimension N_l can be used in x_l to denote a batch of $|N_l|$ samples. Training a CNN is to learn the parameter vector $w = (w_1, w_2, \ldots, w_L)$ and minimize a user specified loss function. Stochastic gradient descent (SGD) algorithm can be used to learn the parameter vector$w$. A back-propagation algorithm is used to compute the derivative of the loss with respect to the vector w in a memory efficient manner.

In this study, we used the mini-batch SGD with momentum algorithm [23], [24] to learn the CNN parameters and the code in [23] to compute the derivative by back-propagation. More specifically, the vector $w^s = (w_1^s, w_2^s, \ldots, w_L^s)$ is used to denote the learned parameters of a source CNN model. A target CNN model is learned on the target domain by incrementally transferring the weight parameters of a source CNN model from layer 1 to layer k,which can be formally

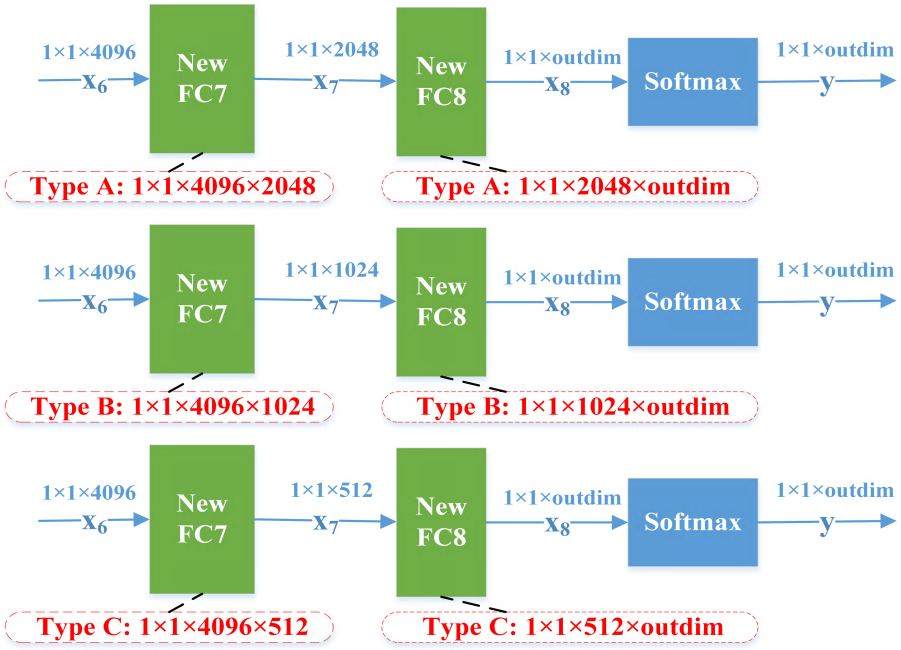


FIGURE 2. Three types of domain adaptation modules.

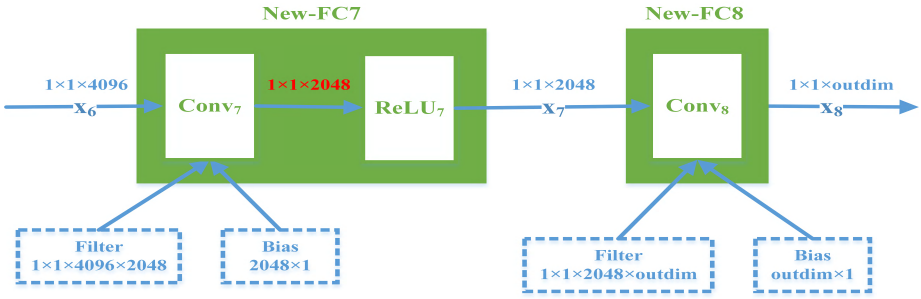


FIGURE 3. Illustration of internal units of new-FC7 and new-FC8 layers.

represented as follows:

$$h_k(x) = g(x_0; w_1^S, \ldots, w_k^S, w_{k+1}^T, \ldots, w_L^T) \tag{2}$$

where the parameters from the layer 1 to the layer k of the source CNN model are locked when continuing the back-propagation on the target training data. In other words, the learning rates of these locked layers are set to zero when being optimized by the SGD with momentum algorithm.

Since the last FC7 and FC8 layers have been removed, the transferrable parameters of the source CNN model lie in the C1-C6 and FC6 layers. Considering that more generic features are learned in the lower layer of a CNN, we skip the C1 and C2 layers and incrementally fine-tune the weight parameters starting from the C3 layer. The initial weight parameters of the New-FC6 and New-FC7 layers are randomly initialized before fine-tuning a CNN model.

C. CNN MODEL ENSEMBLE

As shown in Table 1, we build a matrix to represent a CNN model factory. Each cell in Table 1 denotes a target CNN model learned on the target domain by transferring weight parameters from some layers of a source CNN model. Taking M23 as an example, it is a fine-tuned CNN model trained by CNN-S-1024 and transfers parameters from C1 to C4 layers of the source CNN. CNN-S-1024 indicates that the selected CNN model is CNN-S and we removed the FC7 and FC8 layers from CNN-S and added two New-FC7 and New-FC8 layers of type B (see Fig. 2).

Given the factory of the fine-tuned CNN models, three different CNNs ensemble manners are considered:

1) HORIZONTAL ENSEMBLE

Five cells at each row of the CNN model in Table 1 are combined to generate a horizontal ensemble. For a selected

TABLE 1. A factory of fine-tuned CNN models.

	C1→C2	C1→C3	C1→C4	C1→C5	C1→FC6
CNN-S-512	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}
CNN-S-1024	M_{21}	M_{22}	M_{23}	M_{24}	M_{25}
CNN-S-2048	M_{31}	M_{32}	M_{33}	M_{34}	M_{35}
CNN-M-512	M_{41}	M_{42}	M_{43}	M_{44}	M_{45}
CNN-M-1024	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}
CNN-M-2048	M_{61}	M_{62}	M_{63}	M_{64}	M_{65}
CNN-F-512	M_{71}	M_{72}	M_{73}	M_{74}	M_{75}
CNN-F-1024	M_{81}	M_{82}	M_{83}	M_{84}	M_{85}
CNN-F-2048	M_{91}	M_{92}	M_{93}	M_{94}	M_{95}

CNN architecture, the ensemble is composed of five CNN models learned by incremental parameter transfer. For example, we can combine the outputs of M_{31}, M_{32}, M_{33}, M_{34} and M_{35} to make the ensemble prediction. The ensemble diversity is introduced by incremental parameter transfer.

2) VERTICAL ENSEMBLE

Nine cells at each column of the CNN model in Table 1 are combined to generate a vertical ensemble. For each selected parameter transfer manner, the ensemble is composed of nine CNN models learned by nine CNN architectures. For example, we can combine the outputs of M_{11}, M_{21}, M_{31}, M_{41}, M_{51}, M_{61}, M_{71}, M_{81} and M_{91} to make the final ensemble prediction. The ensemble diversity is introduced by various CNN architectures.

3) SELECTIVE ENSEMBLE

For selective ensemble, a novel Clustering-based Selective Ensemble (CSE) algorithm is proposed to pick a user specified number of models from the CNN model factory, as described in Algorithm 1. The 0/1 matrix Q is generated as follows: suppose we have $|T|$ CNN models $\{h_t\}_{t=1}^{T}$ and a validation data set $V = \{x_i, y_i\}_{i=1}^{m}$, where $x_i \in R^n$ and $y_i \in \{1, \ldots, C\}$. If we used $h_t^j \in \{1, \ldots, C\}$ to denote the predicted labels of the model h_t on a validation example (x_j, y_j), then a 0/1 matrix $Q^{m\times T}$ can be generated on the validation data set V, where the element on the j-th row and t-th column of Q is set to 1 when h_t^j is equal to y_j and set to 0 otherwise.

Algorithm 1 CSE algorithm

Input: 0/1 Matrix $Q_{m\times T}$ generate on the validation data set, k for k medoids clustering, distFunc is the distance function used in k-medoids.

Procedure:

1: Generate k clusters by k medoids(Q, k, distFunc);
2: for each cluster C_i
3: Sort the classifiers in C_i according to their accuracy on the validation data set in descend order;
4: return the most accuracy models in C_i;
5: end for

Output: the index of k selected CNN models

Selective ensemble can be used when we have a validation data set to create the matrix Q. Therefore, it requires an additional validation data set. Note that each complete data set is spitted into three parts, namely training data, validation data and testing data in this work.

Inspire by the work in [25], five pair-wise ensemble diversity measures were used to define five different distance functions for the k medoid algorithm, which can generate different model clusters. Note that the sigmoid function is used to transform the values of some diversity measure to the range [0, 1].

D. ENSEMBLE PREDICTION

Suppose we have selected $|S|$ ($|S| << |T|$) models $\{h_t\}_{t=1}^{S}$ from the CNN model factory and want to make a prediction by the ensemble of $|S|$ models on a test instance x. The ground-truth label of x is in the set $\{1, 2, \ldots, C\}$, the predicted outputs of the model h_t on the instance x is a C-dimensional vector $h_t(x)$, where the i-th element of the vector $h_t(x)$ is an estimation of posterior probability $P(i|x)$. Note that the last layer of the proposed CNN architecture is a softmax layer.

A $C\times S$ dimensional matrix R can be used to represent the output of the ensemble of $|S|$ CNN models on instance x, where the element $R_{i,j}$ in the i-th row and j-th column of matrix R is the output of the j-th CNN model for class i. In order to assign a proper ensemble prediction label $H(x)$ on x, six different voting strategies have been considered:

1) Majority voting: $H(x) = fv(max_idx(R, 1))$;
2) MaxMax voting: $H(x) = max_idx(max(R, 2), 1)$;
3) SumMax voting: $H(x) = max_idx(sum(R, 2), 1)$;
4) MinMax voting: $H(x) = max_idx(\min(R, 2), 1)$;
5) AvgMax voting: $H(x) = max_idx(mean(R, 2), 1)$;
6) ProdMax voting: $H(x) = max_idx(prod(R, 2), 1)$.

In the above formulations, *max_idx(x, d)* returns the index of the maximum value in a d-dimensional array x along the d-th dimension; *fv(x)* returns the most frequent value in a vector x; *max(X, d)* computes the maximum value of a matrix X along its d-th dimension; *sum(X, d)* compute the sum values of matrix X along its d-th dimension; *min(X, d)* computes the minimum value of matrix X along its d-th dimension; *prod(X, d)* computes the product along the d-th dimension of matrix X. For an $m\times n$ matrix X, computing the max/min/sum/mean/prod value alone its first dimension returns a $1\times n$ vector, but returns an $m\times 1$ vector when computing along its second dimension.

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

1) DATASETS

For the source task, a large-scale ImageNet data set was used to train the source CNN models. For the target task,

two public data sets *Caltech*-101 [26] and Berkeley *Office* [27], [28] were used to train target CNN models. There are 101 object classes (ignoring the background class) in the *Caltech-101* data set and 31 object classes in the *Office* data set. The images of the two data sets were categories into four domains: Caltech, Amazon, DSLR and Webcam.

The target task is to identify the object category when given an image from the target domain. Although data augmentation method can be used to enlarge the target training data, we did not use it in our experiments. Hence, the improvement of image classification accuracy is come from CNN weight parameter transfer and model combination.

2) TRAINING SETUP

In the following experiments, we directly used the VGG-S, M, F models which can be downloaded from the website http://www.vlfeat.org/matconvnet/pretrained/. These models are trained by MatConvNet [23] on the ImageNet data set with the network architecture proposed in [6]. Note that we only used these pre-trained source models and target images rather than the source images when fine-tuning a CNN model for the target task.

When training a CNN model for the target task, a mini-batch SGD with momentum algorithm was used. The mini-batch size was set to 65 and the moment rate was set to 0.9. The training epoch was set to 10. In order to transfer parameters from some layers of a source CNN model, the learning rates of these layers were set to 0 before fine-tuning. The learning rates of the other CNN layers were set to a relatively smaller value 0.001 for each epoch.

In addition, five ensemble diversity measures [25] were used in the distance function of the k medoid clustering algorithm for selective ensemble (i.e. Disagreement, Q-statistic, Double-fault, Correlation Coefficient and Kappa-statistic). Note that we used the sigmoid function to transform the values of some ensemble diversity measure to the range [0], [1]. As the setting of initial clustering centroid can affect the results of clustering, we repeated 200 times and return the one with the lowest total sum of distances, as well as overall replicates.

3) MODEL EVALUATION

For evaluating the performance of a target CNN model, we use the error rate (number of misclassified test images divided by the total number of test images) averaged over 3 randomly selected target testing image sets. For each target image set, we use 40% images for training, 30% images for validation and 30% images for testing. The validation images are only used by the proposed MSE algorithm.

B. EMPIRICAL STUDY

1) FINE-TUNED CNN MODELS WITHOUT USING COMBINATION

In the first experiment, a model factory of fine-tuned CNNs (see Table 1) was learned with the proposed CNN architectures. We computed the average error rate of each fine-tuned CNN model without using combination. As shown in Table 2, we presented the average error rate on nine CNN architectures and five weight parameter transfer manners. Note that the notation "$C_i \rightarrow C_j$(or F_j)" denotes the weight parameters from C_i to C_j (or Fj) layers of a source CNN model are locked when training a target CNN model.

For Caltech domain, transferring the weight parameters in the "C1→C2" layers of the source CNN models are better than other layers in most cases. For the Amazon domain, transferring the weight parameters in the "C1→C5" layers of the source CNN models are better than other layers in most cases. For DSLR and Webcam domains, the CNN lays which containing the valuable weight parameters are varied when using different CNN architectures. Moreover, setting the fourth and the third dimension of the convolution filters in New-FC7 and New-FC8 lays to be 1024 or 2048 is better than 512 mostly.

The experimental results indicate that the beneficial weight parameters in the source CNN model are different when using different CNN architectures and fine-tuning on different target domains. There are several factors to affect the results of weight parameter transfer when we have fewer training data, including the number of training data, the number of object categories and the feature distribution difference between the source and target domains etc. In sum, transferring the entire weight parameters of a source CNN model is not a good choice when we have few training data. In our view, the more training data we have, the less weigh parameters in a source CNN should be transferred.

2) HORIZONTAL ENSEMBLE OF FINE-TUNED CNN MODELS

In the second experiment, we combined the fine-tuned CNN models in each row of Table 1 and utilized the six combination methods (see section III-D) to make an ensemble prediction. Table 3 shows the best average error rates over three randomly selected test image sets of the four target domains.

As can be seen from Table 1 and Table 2, the best average error rates were decreased from 0.2752 to 0.2308 for the Caltech domain, from 0.2578 to 0.2255 for the Amazon domain, from 0.2800 to 0.2191 for the DSLR domain, and from 0.1949 to 0.1475 for the webcam domain. In addition, setting the fourth and the third dimension of the convolution filters in New-FC7 and New-FC8 lays to be 2048 is better than 512 and 1024.

The experimental results show that horizontally combining the fine-tuned CNN models in each row of Table 1 to make an ensemble prediction is better than making an isolate prediction by each fine-tuned CNN model when we have inadequate target training data.

3) VERTICAL ENSEMBLE OF FINE-TUNED CNN MODELS

In the third experiment, we combined the fine-tuned CNN models in each column of Table 1 and used the six combination methods (see section III-D) to make the ensemble

TABLE 2. Average error rate of incrementally fine-tuned cnn models without using model combination four domains, three random image set splitting (40% for training, 30% for validation and 30% for testing).

Caltech	C1→C2	C1→C3	C1→C4	C1→C5	C1→FC6	DSLR	C1→C2	C1→C3	C1→C4	C1→C5	C1→FC6
CNN-S-512	**0.3503**	0.3586	0.3890	0.4011	0.4234	CNN-S-512	**0.3638**	0.4305	0.4229	0.4457	0.5809
CNN-S-1024	**0.3431**	0.3471	0.3685	0.3779	0.3534	CNN-S-1024	**0.3524**	0.3638	0.3714	0.4114	0.3867
CNN-S-2048	0.3189	0.3090	0.3308	0.3413	**0.2997**	CNN-S-2048	0.3257	0.3428	0.3448	0.3581	**0.2838**
CNN-M-512	**0.3141**	0.3165	0.3490	0.3651	0.4054	CNN-M-512	0.3409	**0.3390**	0.3467	0.3581	0.5143
CNN-M-1024	**0.3072**	0.3116	0.3360	0.3409	0.3493	CNN-M-1024	**0.2800**	0.3467	0.3714	0.4000	0.3733
CNN-M-2048	0.2815	**0.2812**	0.3055	0.3076	0.3156	CNN-M-2048	0.3143	0.3162	0.3162	0.3714	**0.2838**
CNN-F-512	**0.3302**	0.3319	0.3495	0.3538	0.4422	CNN-F-512	0.3638	**0.3581**	0.3886	0.3619	0.5085
CNN-F-1024	**0.3013**	0.3017	0.3184	0.3272	0.3706	CNN-F-1024	**0.3295**	0.3429	0.3448	0.3352	0.3524
CNN-F-2048	**0.2752**	0.2826	0.2908	0.2991	0.3401	CNN-F-2048	0.3124	0.3143	0.3238	0.3143	**0.2819**
Amazon	**C1→C2**	**C1→C3**	**C1→C4**	**C1→C5**	**C1→FC6**	**Webcam**	**C1→C2**	**C1→C3**	**C1→C4**	**C1→C5**	**C1→FC6**
CNN-S-512	**0.3068**	0.3169	0.3287	0.3088	0.3840	CNN-S-512	**0.2744**	0.3205	0.3577	0.3385	0.4500
CNN-S-1024	0.3046	0.3076	0.3107	**0.2892**	0.3437	CNN-S-1024	0.3103	0.2923	0.2821	**0.2731**	0.3154
CNN-S-2048	0.3019	0.2927	0.2869	**0.2678**	0.2950	CNN-S-2048	0.2513	0.2551	0.2448	0.2487	**0.2295**
CNN-M-512	0.2927	0.2976	0.2942	**0.2869**	0.3824	CNN-M-512	**0.2179**	0.2603	0.3090	0.2962	0.3872
CNN-M-1024	0.2819	0.2769	0.2827	**0.2696**	0.3287	CNN-M-1024	0.2295	**0.2231**	0.2667	0.2718	0.2821
CNN-M-2048	0.2708	0.2609	0.2731	**0.2578**	0.2835	CNN-M-2048	0.2154	0.2064	0.2295	0.2410	**0.1949**
CNN-F-512	0.3084	0.3061	0.3061	**0.2869**	0.3982	CNN-F-512	0.2615	**0.2449**	0.2513	0.2654	0.4013
CNN-F-1024	0.2938	0.2873	0.3046	**0.2804**	0.3398	CNN-F-1024	0.2474	**0.2205**	0.2603	0.2243	0.2872
CNN-F-2048	0.2892	0.2716	0.2739	**0.2704**	0.2946	CNN-F-2048	0.2231	**0.2180**	0.2308	0.2359	0.2321

TABLE 3. The best average error rate of horizontal ensemble when using six combination methods. four domains, three random image set splitting (40% for training, 30% for validation and 30% for testing).

	Caltech	Amazon	DSLR	Webcam
CNN-S-512	0.2880	0.2923	0.3733	0.2718
CNN-S-1024	0.2647	0.2701	0.3124	0.2218
CNN-S-2048	0.2682	0.2601	0.2571	0.1564
CNN-M-512	0.2806	0.2566	0.2876	0.2205
CNN-M-1024	0.2639	0.2467	0.2495	0.1679
CNN-M-2048	**0.2308**	**0.2255**	0.2305	0.1500
CNN-F-512	0.2922	0.2654	0.3124	0.1833
CNN-F-1024	0.2699	0.2589	0.2514	0.1654
CNN-F-2048	0.2507	0.2447	**0.2191**	**0.1475**

TABLE 4. The best average error rate of vertical ensemble when using six combination methods. four domains, three random image set splitting (40% for training, 30% for validation and 30% for testing).

	C1->C2	C1->C3	C1->C4	C1->C5	C1->FC6
Caltech	0.1674	**0.1547**	0.1570	0.1560	0.1560
Amazon	0.2182	0.2048	0.2068	**0.2037**	0.2079
DSLR	0.1238	**0.1105**	0.1295	0.1409	0.1257
Webcam	0.0820	0.0821	0.0910	**0.0795**	0.0859

prediction. Table 4 shows the best average error rate over 3 randomly selected test image sets for the four target domains. As can be seen from Table 3 and Table 4, the best average error rate was decreased from 0.2308 to 0.1547 for the Caltech domain, from 0.2255 to 0.2037 for the Amazon domain, from 0.2191 to 0.1105 for the DSLR domain and from 0.1475 to 0.0795 for the webcam domain.

The experimental results show that vertically combining the fine-tuned CNN models in each column of Table 1 to make an ensemble prediction is better than horizontally combining the fine-tuned CNN models in each row of Table 1 when we have inadequate training data. It was also observed that less target training data lead to more model overfitting. Therefore, combining the fine-tuned CNN models which are trained by different CNN architectures can greatly improve the ensemble diversity and effectively solve the model overfitting problem.

4) SELECTIVE ENSEMBLE OF FINE-TUNED CNN MODELS

In the fourth experiment, the proposed CSE algorithm is used to automatically select a user specified number of models from the fine-tuned CNN model factory. Considering there are nine base CNN models in vertical ensemble, we selected nine CNN models with CSE algorithm. We separately used five ensemble diversity measures in the distance function

of CSE algorithm and only reported the best results on six combination prediction methods.

As shown in Table 5, the average error rate was further decreased from 0.1547 to 0.1485 for the Caltech domain and from 0.2037 to 0.2033 for the Amazon domain. However, the average error rate was increased from 0.1105 to 0.1181 for the DSLR domain and unchanged for the Webcam domain. As there are less validation data in the DSLR domain, the generated matrix Q of the CSE algorithm over-fitted the validation data of DSLR domain. The experimental results show that the proposed CSE algorithm can slightly improve the accuracy of object recognition when we have a validation data set.

TABLE 5. The best average error rate of selective ensemble when using six combination methods. Three random image set splitting (40% for training, 30% for validation and 30% for testing).

	Disagree Dist.	D-Fault Dist.	Kappa Dist.	Q-Stat. Dist.	Corr. Dist.
Caltech	0.1526	**0.1485**	0.1525	0.1509	0.1525
Amazon	**0.2033**	0.2106	**0.2033**	**0.2033**	**0.2033**
DSLR	0.1238	**0.1181**	0.1219	0.1295	0.1276
Webcam	0.0833	0.0808	**0.0795**	**0.0795**	**0.0795**

C. COMPARATIVE EXPERIMENTS AND ANALYSIS

In the comparative experiment, we compared the proposed method with the general CNN fine-tuning method and a CNN parameter transfer method which was proposed by Oquab *et al.* [10]. More specifically, we compared our method against two baselines, which are described as follows:

GCNNF: General CNN Fine-tune method [29], [30]. For the GCNNF, the weight parameters of the Conv. layers (i.e. C1-C5), FC6 and FC7 layers were locked, and the last FC8 layer of the source CNN-X (X=S, M, F) model was reinitialized and further trained on the target data. The size of FC8 layer is set to be the same as the object categories in the target task.

TMLIR-D: Transfer Mid-level Image Representation using CNNs [10]. If we use the same notation as [10], then the FC6 and FC7 have equal size 4096, FCa has size D (D=512, 1024 and 2048), and FCb has a size equal to the number of object classes. The FCa and FCb were reinitialized and further trained on the target data. In other words, the weight parameters in the Conv. layers (i.e. C1-C5), FC6 and FC7 layers of a source CNN model were transferred to the target task.

As shown in Table 6, the average error rate of GCNNF baseline is better than TMLIR baseline in most case. Moreover, the source CNN-F model is the worst one for CNN parameter transfer when using GCNNF baseline. The GCNNF is much better than TMLIR for Caltech and Amazon domains. For the DSLR domain, GCNNF is not better than TMLIR. We can infer that when we have a relative large target training data set, it is better to choose GCNNF rather than TMLIR. Considering the inadequate training images in the DSLR domain, we can infer that TMLIR-D may be more suitable for the object recognition task when we have inadequate target training data. When comparing the experimental results of Table 5 and 6, we can conclude that our method is consistently better than the GCNNF and TMLIR baselines.

TABLE 6. Average error rate of two baseline algorithms four domains, three random image set splitting (40% for training, 30% for validation and 30% for testing).

Caltech	GCNNF	TMLIR-512	TMLIR-1024	TMLIR-2048
CNN-S	**0.1862**	0.3918	0.3520	0.3060
CNN-M	0.1988	0.3959	0.3581	**0.2951**
CNN-F	0.2290	0.4484	0.3917	0.3337
Amazon	**GCNNF**	**TMLIR-512**	**TMLIR-1024**	**TMLIR-2048**
CNN-S	**0.2275**	0.3487	0.2930	**0.2624**
CNN-M	0.2493	0.3422	0.2992	0.2654
CNN-F	0.2624	0.3655	0.3176	0.2746
DSLR	**GCNNF**	**TMLIR-512**	**TMLIR-1024**	**TMLIR-2048**
CNN-S	0.2743	0.3733	0.2819	0.2667
CNN-M	**0.2705**	0.3543	0.2971	**0.2362**
CNN-F	0.3010	0.4247	0.3048	0.2438
Webcam	**GCNNF**	**TMLIR-512**	**TMLIR-1024**	**TMLIR-2048**
CNN-S	0.2026	0.3141	0.2423	0.2026
CNN-M	**0.1782**	0.2949	0.2116	**0.1885**
CNN-F	0.2167	0.3077	0.2436	0.2064

D. COMPUTATIONAL EFFICIENCY ANALYSIS EXPERIMENTS

In the computational efficiency analysis experiments, two experiments were conducted to analyze the space and time cost of the proposed methods.

1) SPACE COST

In this experiment, we considered the DSLR target domain and record the parameter size of each incrementally fine-tuned CNNs. The parameter size of the pre-trained CNN-S, CNN-M and CNN-F models, which were trained on the Imagenet dataset, is 393M, 393M and 232M. As shown in Table 7, the parameter size of CNN models is unchanged in each row of Table 7, which means that the parameter size does not change for any fixed CNN architecture when using the incrementally fine-tune technique. However, when the internal structure of a CNN architecture changed, the parameter size of incrementally fine-tuned CNN models is changed. The experimental results also indicate that the parameter size of incrementally fine-tuned CNN model in DSLR domain is less than the pre-trained source CNN model. The reason may be that the number of object category in the DSLR domain is less than the Imagenet source domain and the New-FC7 and New-FC8 domain adaptation layer (See Figure 2) require fewer parameters.

Since the models of each row in Table 7 were trained by an incremental manner, their parameter value should

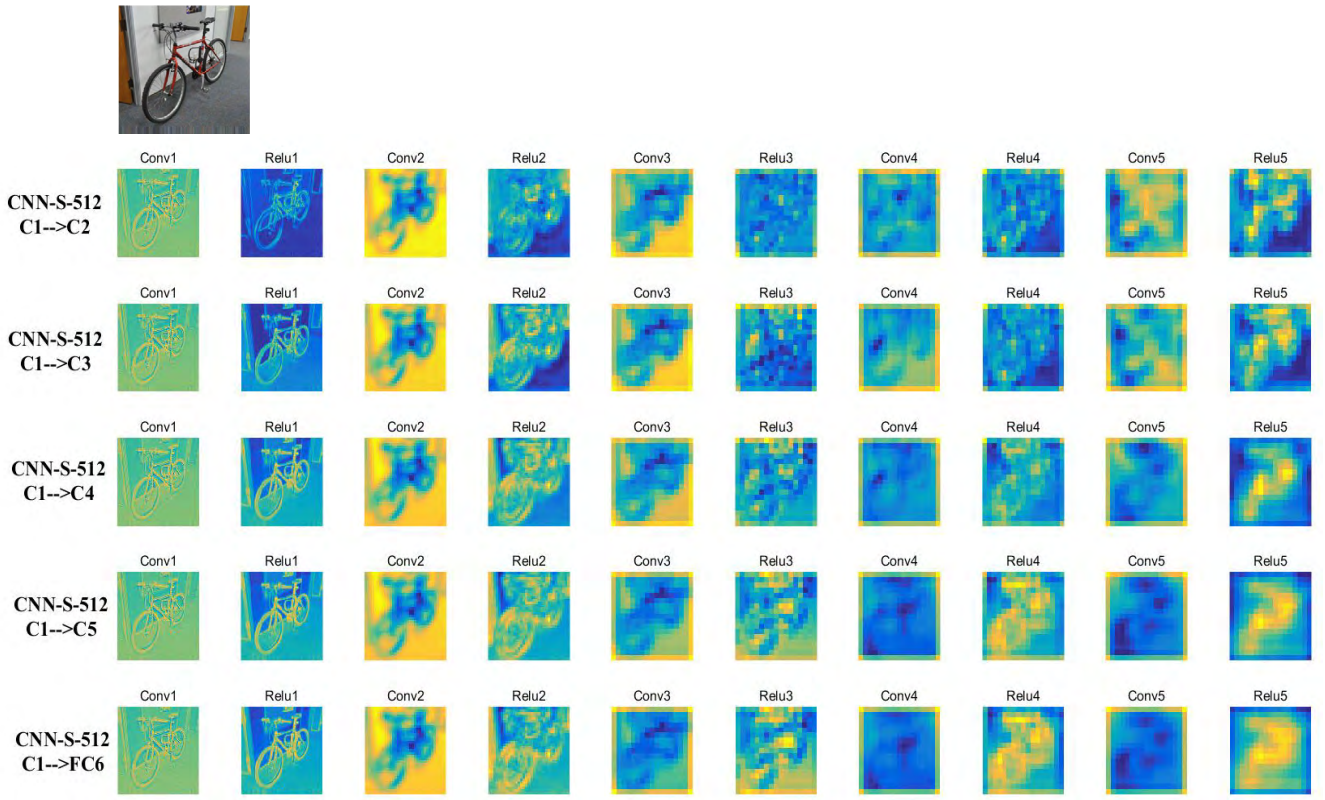


FIGURE 4. Feature representation of a bicycle image based on 5 incrementally fine-tuned CNN-S-512 models.

TABLE 7. The parameter size of incrementally fine-tuned cnn models (target domain: dslr, unit: mega byte).

	C1→C2	C1→C3	C1→C4	C1→C5	C1→FC6
CNN-S-512	321	321	321	321	321
CNN-S-1024	329	329	329	329	329
CNN-S-2048	345	345	345	345	345
CNN-M-512	321	321	321	321	321
CNN-M-1024	329	329	329	329	329
CNN-M-2048	345	345	345	345	345
CNN-F-512	161	161	161	161	161
CNN-F-1024	169	169	169	169	169
CNN-F-2048	185	185	185	185	185

be different. To better understand it, we visualize the feature representations learned at the hidden layers of 5 incrementally fine-tuned CNN models based on the CNN-S-512 architecture for a bicycle image picked from the test image set, as shown in Figure 4.

More specifically, suppose we have selected N models from the factory of fine-tuned CNNs (see Table 1) and the parameter size of each CNN model is $\{S_t\}_{t=1}^{N}$, the parameter size of the CNNs model ensemble is $\sum_{t=1}^{N} S_t$.

2) TIME COST

In this experiment, we compared the time cost for predicting the class labels of a given set of testing images. The hardware setting is: 2 cores Intel®Xeon®CPU E5-1620 v3, @3.50G Hz and 3.50GHz, 16G memory. The DSLR, Amazon and Webcam domains were considered. We computed the average time cost over 3 randomly selected target testing image sets. The number of testing images for DSLR, Amazon and Webcam is 175, 869 and 260, respectively. The experimental results are shown in Table 8-Table 10. As shown in Table 8, the prediction time cost of CNN-F based horizontal ensemble methods is less than the other methods. When the dimensions of New-FC7 and New-FC8 layers are varied, the prediction time has been less affected. Considering the space cost of CNN-F architecture and the experimental results of Table 3, CNN-F architecture is the best choice for horizontal ensemble of incrementally fine-tuned CNN models. When comparing the experimental results of Table 9 and Table 10, it takes close prediction time for both vertical ensemble and selective ensemble models. However, it takes extra time for selective ensemble method to select models from the fine-tuned CNN model factory by MSE algorithm on a validation dataset. Considering it includes nine fine-tuned CNN models in vertical ensemble and selective ensemble, which is more than five fine-tuned CNN models in horizontal ensemble. It is normal to take less prediction time for horizontal ensemble.

It must be mentioned that it take much more training time to build the fine-tuned CNN model factory (see Table 1). However, each base model in the CNN model factory is independent and can be trained in parallel.

TABLE 8. Average time cost for horizontal ensemble CNNs prediction on dlsr, amazon and webcam domains (unit: second).

Horizontal Ensemble	DSLR (#IMG=175)	AMAZON (#IMG=869)	WEBCAM (#IMG=260)
CNN-S-512	67.9624	318.8252	91.7203
CNN-S-1024	69.3167	321.3569	88.6893
CNN-S-2048	69.6439	323.6182	87.8191
CNN-M-512	61.4820	281.2047	77.9951
CNN-M-1024	61.3449	280.8938	82.0057
CNN-M-2048	63.0249	287.5488	81.9153
CNN-F-512	32.1930	133.3015	40.4167
CNN-F-1024	32.9536	139.0984	42.5547
CNN-F-2048	33.2313	318.8252	41.0260

TABLE 9. Average time cost for vertical ensemble CNNs prediction on dlsr, amazon and webcam domains (unit: second).

Vertical Ensemble	DSLR (#IMG=175)	AMAZON (#IMG=869)	WEBCAM (#IMG=260)
C1->C2	97.1622	452.7380	125.6858
C1->C3	98.3003	446.6054	127.6232
C1->C4	98.8181	442.1716	131.2683
C1->C5	99.5252	447.6860	131.1923
C1->FC6	99.0460	447.7943	128.1802

TABLE 10. Average time cost for selective ensemble CNNs prediction on dlsr, amazon and webcam domains (unit: second).

Selective Ensemble	DSLR (#IMG=175)	AMAZON (#IMG=869)	WEBCAM (#IMG=260)
Disagree Dist.	98.7449	434.7557	125.6922
D-Fault Dist.	98.8143	489.5441	132.2442
Kappa Dist.	96.6370	448.3695	125.3093
Q-Stat. Dist.	98.2294	450.2125	126.1517
Corr. Dist.	96.2889	451.3072	129.1698

V. CONCLUSIONS AND FUTURE WORK

In this paper, the problem of combining an ensemble of fine-tuned CNN models for cross-domain and multiclass object recognition task is empirically studied. A novel method is proposed to combine a group of incrementally fine-tuned CNN models for studying the horizontal/vertical/selective ensemble manners on two data sets. The experimental results demonstrate the effectiveness and good performance of the proposed methods. In particular, a Matlab software package is provided to reproduce the experimental results of this paper and enable other researchers to conduct further study.

The major weakness of the proposed methods is the space cost of an ensemble of fine-tuned CNN models is high. Hence, it is better to compress the CNN models without loss of model diversity when performing incrementally fine-tune on the target domain. In recent years, several CNN compression methods have been proposed, such as [31]–[33]. The future study should focus on improving the model diversity when compressing a set of fine-tuned CNN models. In addition, the selective ensemble requires an extra validation data set. It is better to design some metrics to evaluate the diversity of CNN models and guide the user to select proper models from a fine-tuned CNNs model factory without using extra validation data set.

In the future, we will study how to significantly compress the size of each individual CNN model, as well as other selective ensemble algorithms. In addition, we plan to re-implement the Matlab source codes with C++ on our service robot platform. Except for object category recognition, the proposed methods can also be used in other fields with inadequate training data, such as brain-computer interface [34]–[36], natural language processing [37], [38] and remote sensing [39], [40] etc. Hence, using the proposed methods in other scientific research fields is also a meaningful work.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[2] J. Li, Z. Zhang, and H. He, "Hierarchical convolutional neural networks for EEG-based emotion recognition," *Cogn. Comput.*, vol. 10, no. 2, pp. 368–380, 2018. doi: 10.1007/s12559-017-9533-x.

[3] G. Zhong, S. Yan, K. Huang, Y. Cai, and J. Dong, "Reducing and stretching deep convolutional activation features for accurate image classification," *Cogn. Comput.*, vol. 10, no. 1, pp. 179–186, 2018. doi: 10.1007/s12559-017-9534-9S.

[4] X. Liu and Z. Deng, "Segmentation of drivable road using deep fully convolutional residual network with pyramid pooling," *Cogn. Comput.*, vol. 10, no. 2, pp. 272–281, 2018. doi: 10.1007/s12559-017-9524-y.

[5] N. Tajbakhsh *et al.*, "Convolutional neural networks for medical image analysis: Full training or fine tuning?" *IEEE Trans. Med. Imag.*, vol. 35, no. 5, pp. 1299–1312, May 2016.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional networks," in *Proc. 25th Brit. Mach. Vis. Conf.*, Nottingham, U.K., 2014, pp. 1–12.

[7] J. Ding, B. Chen, H. Liu, and M. Huang, "Convolutional neural network with data augmentation for SAR target recognition," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 3, pp. 364–368, Mar. 2016.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[9] D. P. Helmbold and P. M. Long. (2016). "Fundamental differences between dropout and weight decay in deep networks." [Online]. Available: https://arxiv.org/abs/1602.04484v2

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

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Neural Inf. Process. Syst. (NIPS)*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, San Diego, CA, USA, 2015, pp. 1–14.

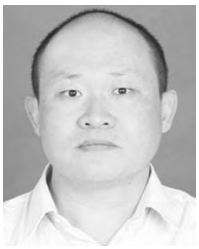
[13] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[15] J. Hoffman, S. Gupta, J. Leong, S. Guadarrama, and T. Darrell, "Cross-modal adaptation for RGB-D detection," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, Stockholm, Sweden, May 2016, pp. 5032–5039.

[16] A. Kumar, J. Kim, D. Lyndon, M. Fulham, and D. Feng, "An ensemble of fine-tuned convolutional neural networks for medical image classification," *IEEE J. Biomed. Health Inform.*, vol. 21, no. 1, pp. 31–40, Jan. 2017.

[17] G. Wen, Z. Hou, H. Li, D. Li, L. and J. Xun, "Ensemble of deep neural networks with probability-based fusion for facial expression recognition," *Cognit. Comput.*, vol. 9, no. 5, pp. 597–610, Oct. 2017.
[18] L. Zhao, J. Wang, X. Li, Z. Tu, and W. Zeng. (2016). "On the connection of deep fusion to ensembling." [Online]. Available: https://arxiv.org/abs/1611.07718v1
[19] Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang, "Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 7340–7351.
[20] M. A. Wani and S. Afzal, "A new framework for fine tuning of deep networks," in *Proc. IEEE 16th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Cancun, Mexico, Dec. 2017, pp. 359–363.
[21] T. Tan *et al.*, "Optimize transfer learning for lung diseases in bronchoscopy using a new concept: Sequential fine-tuning," *IEEE J. Transl. Eng. Health Med.*, vol. 6, 2018, Art no. 1800808.
[22] G. Wang *et al.*, "Interactive medical image segmentation using deep learning with image-specific fine tuning," *IEEE Trans. Med. Imag.*, vol. 37, no. 7, pp. 1562–1573, Jul. 2018.
[23] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 689–692.
[24] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 1139–1147.
[25] G. D. C. Cavalcanti, L. S. Oliveira, T. J. M. Moura, and G. V. Carvalho, "Combining diversity measures for ensemble pruning," *Pattern Recognit. Lett.*, vol. 74, no. 15, pp. 38–45, Apr. 2016.
[26] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental Bayesian approach tested on 101 object categories," *Comput. Vis. Image Understand.*, vol. 106, no. 1, pp. 59–70, Jan. 2007.
[27] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proc. 11th Eur. Conf. Comput. Vis.*, Heraklion, Crete, Greece, Sep. 2010, pp. 213–226.
[28] J. Hoffman, E. Rodner, J. Donahue, B. Kulis, and K. Saenko, "Asymmetric and category invariant feature transformations for domain adaptation," *Int. J. Comput. Vis.*, vol. 109, no. 1, pp. 28–41, 2014.
[29] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. 31th Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 647–655.
[30] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Columbus, OH, USA, Jun. 2014, pp. 806–813.
[31] Y. Shen *et al.*, "CS-CNN: Enabling robust and efficient convolutional neural networks inference for Internet-of-Things applications," *IEEE Access*, vol. 6, pp. 13439–13448, 2018.
[32] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized CNN: A unified approach to accelerate and compress convolutional networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4730–4743, Oct. 2018.
[33] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic CNN compression via low-rank decomposition with knowledge transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published. doi: 10.1109/TPAMI.2018.2873305.
[34] Y. Jiao, Y. Zhang, Y. Wang, B. Wang, J. Jin, and X. Wang, "A novel multilayer correlation maximization model for improving CCA-based frequency recognition in SSVEP brain–computer interface," *Int. J. Neural Syst.*, vol. 28, no. 8, pp. 1750039-1–1750039-14, 2017.
[35] Z. Jin, G. Zhou, D. Gao, and Y. Zhang, "EEG classification using sparse Bayesian extreme learning machine for brain–computer interface," in *Neural Computing and Applications*. London, U.K.: Springer, 2018, pp. 1–9. doi: 10.1007/s00521-018-3735-3.
[36] Y. Zhang *et al.*, "Strength and similarity guided group-level brain functional network construction for MCI diagnosis," *Pattern Recognit.*, vol. 88, pp. 421–430, Apr. 2019.
[37] H. Zhuang, C. Wang, C. Li, Y. Li, Q. Wang, and X. Zhou, "Chinese language processing based on stroke representation and multidimensional representation," *IEEE Access*, vol. 6, pp. 41928–41941, 2018.
[38] S. L. Marie-Sainte, N. Alalyani, S. Alotaibi, S. Ghouzali, and I. Abunadi, "Arabic natural language processing and machine learning-based systems," *IEEE Access*, vol. 7, pp. 7011–7020, 2019.
[39] Q. Zhang, Q. Yuan, C. Zeng, X. Li, and Y. Wei, "Missing data reconstruction in remote sensing image with a unified spatial-temporal-spectral deep convolutional neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 8, pp. 4274–4288, Mar. 2018.
[40] X. Xu, W. Li, Q. Ran, Q. Du, L. Gao, and B. Zhang, "Multisource remote sensing data classification based on convolutional neural network," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 2, pp. 937–949, Feb. 2018.



XUESONG ZHANG received the bachelor's and master's degrees in computer application and technology from Liaoning Shihua University, China, in 2003 and 2006, respectively, and the Ph.D. degree in control theory and engineering from the Dalian University of Technology, Dalian, China, in 2017. He joined the Software Technology Institute, Dalian Jiaotong University, Dalian, as a Teaching Assistant (2006), and became a Lecturer (2008). His current research interests include robot vision, indoor scene understanding, object recognition and detection, and machine learning.

FEI YAN received the bachelor's and Ph.D. degrees in control theory and engineering from the Dalian University of Technology, Dalian, China, in 2011. In 2013, he joined the Dalian University of Technology, where he held a postdoctoral position, and became a Lecturer, in 2015. He is currently an Associate Professor with the School of Control Science and Engineering, Dalian University of Technology. His research interests include mobile robot 2D and 3D mapping, autonomous navigation, 3D scene recognition, and reconstruction.

YAN ZHUANG received the bachelor's and master's degrees in control theory and engineering from Northeastern University, China, in 1997 and 2000, respectively, and the Ph.D. degrees in control theory and engineering from the Dalian University of Technology, China, in 2004. In 2005, he joined the Dalian University of Technology, as a Lecturer, and became an Associate Professor, in 2007, where he is currently a Professor with the School of Control Science and Engineering. His research interests include mobile robot 3D mapping, outdoor scene understanding, 3D-laser-based object recognition, 3D scene recognition, and reconstruction.

HUOSHENG HU (M'94–SM'01) received the M.Sc. degree in industrial automation from the Central South University, Changsha, China, in 1982, and the Ph.D. degree in robotics from the University of Oxford, Oxford, U.K., in 1993. He is currently a Professor with the School of Computer Science and Electronic Engineering, University of Essex, U.K., leading the Robotics Research Group. His research interests include behaviour-based robotics, human–robot interaction, service robots, embedded systems, data fusion, learning algorithms, mechatronics, and pervasive computing. He has published around 450 papers in journals, books, and conferences in these areas, and has received a number of Best Paper Awards. He is a Founding Member of the IEEE Robotics and Automation Society Technical Committee on Networked Robots, a Fellow of the IET and InstMC, and a Senior Member of the ACM. He has been a Program Chair or a member of the Advisory/Organising Committee for many international conferences such as the IEEE ICRA, IROS, ICMA, ROBIO, ICIA, ICAL, and IASTED RA, CA, and CI conferences. He currently serves as the Editor-in-Chief for the *International Journal of Automation and Computing*, the Editor-in-Chief for the *Online Robotics Journal*, and the Executive Editor for the *International Journal of Mechatronics and Automation*.

CHUNGUANG BU received the bachelor's degree in mechatronic engineering from the Harbin University of Science and Technology, China, in 1996, and the master's degree in mechatronic engineering from Northeastern University, China, in 1999. He joined the State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences, in 1999. His research interests include dynamics and control of mobile robot locomotion, control of manipulators, and specialized robot systems.