# Multiscale Multipath Ensemble Convolutional Neural Network

Xuesong Wang, *Member, IEEE*, Achun Bao, Enhui Lv, and Yuhu Cheng, *Member, IEEE*

***Abstract*—Most convolutional neural network (CNN) models require large parameter quantity, high computational consumption, and deeper layers to achieve better performance, which limit their further applications. For improvement, a lightweight multiscale multipath ensemble CNN (MSME-CNN) is proposed. First, the shallow-layer and deep-layer convolution output features are directly concatenated to form a multipath ensemble mode, which can not only avoid gradient vanishing but also fuse the multilayer features. In addition, each path of the network is composed of multiscale low-rank convolution kernels in parallel. This structure can extract multiscale features of the input and improve the feature extraction ability of the network. Meanwhile, the convolution kernel low-rank approximation can effectively compress the model complexity. Furthermore, the proposed sparse connection mechanism of the convolution kernel helps to reduce the complexity, so that higher classification accuracy can be obtained with less parameters and computational load. Finally, the linear sparse bottleneck structure is used to fuse multiscale features and compress the convolution channel, which further improves the network performance. Experiments of four commonly used image recognition datasets verify the superiority of MSME-CNN to several baseline models.**

***Index Terms*—Computational load, convolutional neural network (CNN), ensemble, multipath, multiscale, parameter quantity.**

## I. Introduction

COMPARED with traditional feature learning methods, convolutional neural networks (CNNs) [1], [2] have achieved better performance. However, most CNNs suffer from large parameter quantity and high computational consumption, which limit their applications in light devices and mobile terminals. Therefore, how to reduce model redundancy of CNNs to achieve both lightweight and high classification accuracy has become the key to its large-scale application [3], [4]. The key component of input feature extraction in CNN is the convolution layer. The more efficient and the less redundant the extracted features are, the less storage space and computation resources the network requires [5]. In addition, the operation of convolution layer is essentially dense matrix or vector multiplication. The sparser the multiplication operation is, the lower the model complexity would be, which helps to achieve higher efficiency. Based on the above points, existing compression methods of CNN can be generally divided into two categories: 1) low-rank decomposition [6]–[8], pruning [9], [10], knowledge distillation [11], [12], and weight coding quantization [13]–[15]. However, most of the above-mentioned methods can only solve a certain problem. Besides, it still needs specifically designed software/hardware accelerators to accelerate training; and 2) novel lightweight network structure. For example, large convolution kernel decomposition [16], small convolution kernel [17], and shrinkage the number of convolution kernels and sparse connections [18] have been reported to compress CNN. Compared with the compression methods that based on existing models, a novel efficient network will not only improve computational consumption and classification performance but also show high portability.

In 2012, AlexNet [18] achieved a milestone on the ImageNet dataset [19], leading the rise of deep learning. However, there are more than 60 million parameters in AlexNet, which occupy more than 230 MB (BVLC caffemodel), limiting its application on light equipments. In order to reduce the model storage space, some improvements have been achieved based on AlexNet. SqueezeNet [20] introduced a multiscale convolution structure to build a fire module, where the channel is first compressed through a $1 \times 1$ convolution kernel, and then amplified through a multiscale convolution kernel of $1 \times 1$ and $3 \times 3$ in parallel. In comparison with AlexNet, SqueezeNet only occupies 2% storage space without any loss of classification accuracy. Besides, with low bit quantization [21], SqueezeNet achieves approximately 510 times compression of storage space. Although SqueezeNet implements efficient model compression, it only focuses on compression rather than accuracy improvement. In order to decrease model complexity and improve computational efficiency simultaneously, GoogLeNet [22] adopts the Inception module, which is composed by parallel multiscale convolution kernels, to extract input features. The classification accuracy is dramatically improved by GoogLeNet, which only

Manuscript received January 6, 2020; accepted January 30, 2020. This work was supported by the National Natural Science Foundation of China under Grant 61976215 and Grant 61772532. This article was recommended by Associate Editor Z. Liu. *(Corresponding author: Yuhu Cheng.)*

Xuesong Wang, Achun Bao, and Yuhu Cheng are with the Engineering Research Center of Intelligent Control for Underground Space, Ministry of Education, China University of Mining and Technology, Xuzhou 221116, China, also with the Xuzhou Key Laboratory of Artificial Intelligence and Big Data, China University of Mining and Technology, Xuzhou 221116, China, and also with the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China (e-mail: wangxuesongcumt@163.com; 962571062@qq.com; chengyuhu@163.com).

Enhui Lv is with the School of Medical Information and Engineering, Xuzhou Medical University, Xuzhou 221004, China (e-mail: lehdyb@126.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2020.2972695

requires 6.8 M for parameter storage, showing a shrinkage of nearly ten times comparing with AlexNet. However, the computational load of GoogLeNet is relatively high due to the deeper network structure and dense convolution calculation. Therefore, Inception V2 model [16] uses multiple small convolution kernels to replace large convolution kernels following VGG network to decrease the computational load [17]. Based on Inception V2, Inception V3 model [23] decomposes the $3 \times 3$ square convolution kernel into two long stripe convolution kernels with size $3 \times 1$ and $1 \times 3$ for further compression. Xception [24] replaces the densely connected $3 \times 3$ convolution kernel with depthwise convolution kernel and increases the number of channels for more efficient recognition. For original convolution operation, each convolution kernel is convolved with all input channels, then the convolved features of each channel are summarized pixel by pixel to output a feature map. The depthwise convolution essentially limits the convolution operation of each convolution kernel in their group, that means, each convolution kernel is only convolved with the input of its corresponding input channels, and the inputs of all convolution kernels are different, thereby the output feature redundancy is reduced. Although Inception V2, Inception V3, and Xpcetion achieve high classification accuracy and parameter efficiency, they still have the problem of large computational complexity, which still limits their applications to scenarios of strict computational requirements.

Compared with the above methods, some studies pay more attention to the compression of model parameters and computational load, thus achieve better tradeoff between network performance and computational consumption. To this end, Howard *et al.* [25] proposed a lightweight neural network called MobileNet V1, which achieved higher classification accuracy with less parameters and computational load. The main innovation of MobileNet V1 is using a depthwise separable convolution rather than original convolution, as well as a width factor and a resolution factor for model complexity control. The depthwise separable convolution consists of a depthwise convolution and a pointwise convolution. To be more specific, the depthwise is a completely sparse connection convolution, and the pointwise is a $1\times1$ convolution. Thus, the computational load and parameter amount can be reduced by about 9 times and 7 times, respectively, for a $3\times3$ convolution kernel, somehow with slight decrease in classification accuracy. Although the depthwise separable convolution has strong feature extraction ability, the subsequent pointwise convolution consumes a large amount of calculation and parameters due to large number of depthwise convolution channels. For this point, ShuffleNet [26] uses the channel shuffle method to improve pointwise convolution in depthwise separable convolution. In addition, ShuffleNet achieves higher classification accuracy and computational efficiency by combining the improved ResNet [27]. The starting point of the channel shuffle is feature correlation reduction between channels after the group convolution, which may result in poor information circulation. By shuffling channel orders, the input channel of the same convolution kernel is changed, thereby achieving feature fusion. Based on MobileNet V1, MobileNet V2 [28] imported new bottleneck structures and residual connections for more efficient compression. In addition, IGCV1 [29] introduces multiple group convolutions and reorders the channels so that the output of each module is the convolution result of all input features. The interleaved group convolution not only enables the better fusion of features but also improves the computational efficiency. On this basis, IGCV2 [30] replaces the pointwise convolution in MobileNet V1 with two interleaved group convolutions, which further improves the computational efficiency. Based on IGCV2, IGCV3 [31] uses sparse low-rank convolution kernels and limits the number of times of super-channel appearing in different paths to improve efficiency. However, the improvement is relatively limited and the performance on small datasets is still not satisfactory. Based on the consideration that the ensemble of shallow networks can build deep networks [32], Wang *et al.* [33] proposed the multipath ensemble CNN (ME-CNN) by directly concatenating the low-level with high-level output features. ME-CNN divides the network into several ensemble blocks according to the size of feature maps, and each ensemble block is a stack of multiple convolutional layers. The output of each convolutional layer is concatenated through the shortcut connection structure as the ensemble block output.

Starting from the construction of high-efficiency, low-rank, and less redundant structure, we design a neural network computing module. It mainly includes three aspects: 1) multiscale low-rank convolution kernel approximation; 2) convolution kernel sparse connection; and 3) linear sparse bottleneck structure. We replace the basic convolution module in ME-CNN [33] with the above-mentioned computing module to obtain a lightweight multiscale ME-CNN (MSME-CNN).

The remainder of this article is organized as follows. The details of the proposed MSME-CNN model are described in Section II. The experimental results on several image recognition datasets are reported in Section III, followed by a conclusion in Section IV.

## II. Proposed MSME-CNN

Since the CIFAR and tiny ImageNet are commonly used benchmark datasets for performance evaluation of deep networks, we take them as an example here. The structure of MSME-CNN for CIFAR [34] and tiny ImageNet datasets is mainly composed of three ensemble blocks, three weighted transform layers and three pooling layers, as shown in Fig. 1. Inside the ensemble block, the batch normalization (BN) layer [16] and ReLU [35] activation function are considered as a composite nonlinear map $H(\cdot)$. Suppose the input image is $x$, after the $l$th convolution layer, BN layer, and ReLU activation function, the nonlinear mapping of $x$ is $H_l(x)$

$$H_l(x) = H_l\big(w_{(\lambda)} * x + b\big) \tag{1}$$

where $w_{(\lambda)}$ represents convolution weight, $b$ represents bias, and $*$ represents convolution operation, $\lambda$ represents size of convolution kernel, which can take 1 or 3.

For convenience, we can rewrite (1) as

$$H_l(x) = H_l^{(\lambda)}(x). \tag{2}$$

Since each convolution layer is composed of parallel multiscale convolution kernels of $1 \times 1$ and $3 \times 3$, the convolution layer can be expressed as

$$H_l(x) = \left[H_l^{(1)}(x), H_l^{(3)}(x)\right] \tag{3}$$

where [ ] indicates concatenation operation. The output of the $n$th ensemble block $f_n(x)$ is the concatenation of output feature maps of $l$ convolution layers, which can be expressed as

$$f_n(x) = \left[\left[H_{m+1}^{(1)}(x), H_{m+1}^{(3)}(x)\right], \left[H_{m+2}^{(1)}(x), H_{m+2}^{(3)}(x)\right] \\ \ldots, \left[H_{m+l}^{(1)}(x), H_{m+l}^{(3)}(x)\right]\right] \tag{4}$$

where $m$ represents the serial number of the last convolution layer in the $(n-1)$th ensemble block. The output of the $l$th convolution layer is convolved by multiscale low-rank convolution kernels, then obtain feature maps after the nonlinear mapping of $H(\cdot)$. Then feature maps are concatenated to obtain the output, which will be input into next layer

$$H_{l+1}(x) = \left[H^{(1)}(H_l(x)), H^{(3)}(H_l(x))\right]. \tag{5}$$

According to (4) and (5), the output of an ensemble block $f_n(x)$ can be represented as

$$f_n(x) = \left[\left[H_{m+1}^{(1)}(x), H_{m+1}^{(3)}(x)\right] \\ \left[H_{m+2}^{(1)}\left(\left[H_{m+1}^{(1)}(x), H_{m+1}^{(3)}(x)\right]\right) \\ H_{m+2}^{(3)}\left(\left[H_{m+1}^{(1)}(x), H_{m+1}^{(3)}(x)\right]\right)\right], \cdots\right]. \tag{6}$$

According to (6), the input information of the ensemble block is convolved by the progressively increasing convolution layer inside the ensemble block, which can not only avoid gradient vanishing problem effectively but also extract multiscale features step by step through multiscale convolution. In this way, the network achieves stronger learning ability.

## *A. Low-Rank Approximation*

For the popular Caffe [36] framework, the convolution process can be summarized as follows. First, the input features are split into small matrices. Then, the small matrices and convolution kernels are stretched into vectors, respectively, and the vectors are combined into a large matrix. Finally, the convolution operation is implemented by the matrix multiplication. According to the above process, the convolution is essentially a dense matrix operation of input features and convolution kernels. In particular, the larger the convolution kernel size is, the larger the matrix dimension of the convolution kernel stretch combination will be, and the higher the rank will be. The matrix multiplication will be denser, resulting in larger computational consumption. Therefore, from the perspective of computational consumption, convolution kernels with smaller size are preferred to reduce model complexity. However, the capabilities of feature extraction and nonlinear expression of $1 \times 1$ convolution kernel with the smallest size are relatively limited. If the network only contains $1 \times 1$ convolution kernels, the network performance will be greatly degraded, and the low-rank approximation information will be lost. Most convolution kernels used in ME-CNN [33] ensemble block are $3 \times 3$. Although ME-CNN gets better tradeoff between computational consumption and network performance, the large amount of computation required for $3 \times 3$ convolution kernels is still unsatisfied in case of limited computing power. Thus, MSME-CNN uses multiscale low-rank convolution kernels to approximate high-rank convolution kernels.

Let $C_i$ and $C_o$ be the numbers of convolutional input channels and output channels, respectively, $P$ be the parameter quantity, and $F$ be the computational load. Assuming that the convolution operation does not change the feature map size $S$, the parameter quantity and computational load of the $3 \times 3$ convolution kernel can be expressed as

$$P = C_i \times C_o \times 9 \tag{7}$$
$$F = C_i \times C_o \times S^2 \times 9. \tag{8}$$

If $3 \times 3$ convolution kernel is decomposed with $3 \times 1$ and $1 \times 3$ strip convolution kernels, the parameter quantity and computational load will then be

$$P = C_i \times C_o \times 6 \tag{9}$$
$$F = C_i \times C_o \times S^2 \times 6. \tag{10}$$

After that $3 \times 3$ convolution kernel is approximated with $3 \times 3$ and $1 \times 1$ convolution kernels, the parameter quantity and computational load can be expressed as

$$P = C_i \times \frac{C_o}{2} \times 9 + C_i \times \frac{C_o}{2} \times 1 = C_i \times C_o \times 5 \tag{11}$$
$$F = C_i \times C_o \times S^2 \times 5. \tag{12}$$

According to (7)–(12), for a convolution layer with the same number of inputs and outputs, the low-rank decomposition and low-rank approximation of convolution kernel can significantly reduce computational load and parameter quantity of the network. In addition, it can be seen that the compression ratio of low-rank approximation is 1.8, which is higher than that of the low-rank decomposition (1.5). This is because the number of output channels of multiscale convolution kernels is half of the original single-scale convolution kernel, and thus the amount of computation and parameter savings are brought about by replacing half the number of $3 \times 3$ with $1 \times 1$ convolution kernels. Furthermore, the low-rank decomposition makes a convolution layer into two layers, which increases the depth of the network and brings about the risk of gradient vanishing. Low-rank approximation does not change the depth of network, but can extract multiscale features, so it has a greater advantage.

## *B. Sparse Connection*

In the conventional convolution operation, a set of convolution kernels are densely connected to a set of input feature maps. That is, each convolution kernel is convolved with all input feature maps. Such a dense connection structure between the feature map and convolution kernel also results in a large computational load, especially, when the number of input feature maps or convolution kernels is large. Therefore, MSME-CNN further reduces computational load and parameter quantity through the sparse connection mechanism, which is achieved by group convolution.

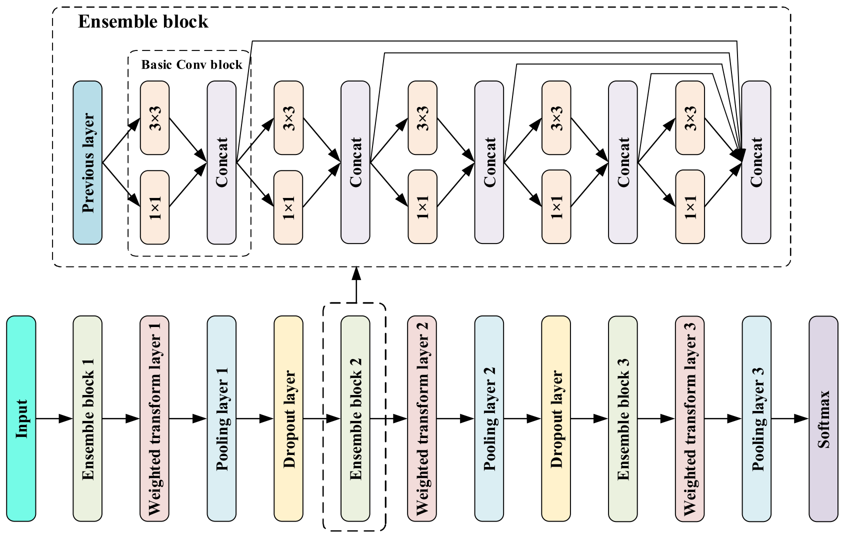


Fig. 1. Proposed MSME-CNN model.

Group convolution was first applied to AlexNet [18], followed by cases that MobileNet used a tighter depthwise separable convolution. Although the depthwise separable convolution can reduce the computational load of convolutional operation, the depthwise convolution only accepts a single feature map as input. At this time, the input information of each convolution kernel is too small, which easily leads to unstable training and even entry to the dead zone by most neurons. In addition, the pointwise convolution occupies the most computational load of the depthwise separable convolution, making convolution compression inefficient. Therefore, only group convolution is used in the convolution block of MSME-CNN.

The computational load and parameter quantity of the conventional $3 \times 3$ convolution kernel are as shown in (7) and (8), while that of the depthwise separable convolution can be expressed as

$$P = 9C_i + C_i \times C_o \tag{13}$$
$$F = (9C_i + C_i \times C_o) \times S^2. \tag{14}$$

For group convolution with $g$ groups, the computational load and parameter quantity can be expressed as

$$P = \frac{C_i \times C_o \times 9}{g} \tag{15}$$
$$F = \frac{C_i \times C_o \times 9}{g} \times S^2. \tag{16}$$

It can be seen from (13)–(16) that when $g \geq 8$, the computational load and parameter quantity of group convolution are smaller than that of depthwise separable convolution, and $g < C_i$ can avoid the problem of depthwise separable convolution.

## C. Linear Sparse Bottleneck Structure

The basic convolution block of MSME-CNN has two forms. The first one only contains multiscale low-rank sparse connection convolution kernels, as shown in Fig. 1. Based on the above structure and SqueezeNet [20], we designed another more efficient basic convolution block with a linear sparse bottleneck structure, as shown in Fig. 2.

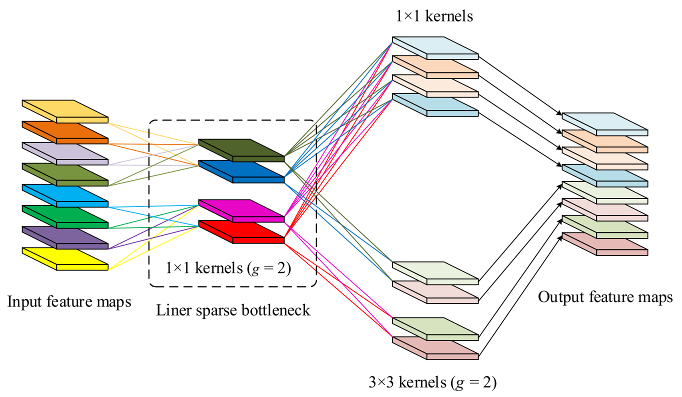


Fig. 2. Basic Conv block with linear sparse bottleneck structure.

In the basic convolution block shown in Fig. 1, the output feature maps are directly concatenated. The number of output feature channels of convolution block is equal to the sum of all low-rank convolution kernel channels, and a large number of output channels increase the computational load for the next basic convolution block. Therefore, introducing a bottleneck structure of $1 \times 1$ convolution before the low-rank convolution kernels can help reduce the computational load, and the number of channels is set to be half of the number of input channels. That is, the number of multiscale low-rank convolution kernel input channels $C_i$ is 0.5 times of the original. According to (11) and (12), the computational load and parameter quantity of multiscale low-rank convolution part are also halved, but the newly added bottleneck structure brings about additional computation and parameters. To this end, we also introduced the sparse connection mechanism based on group convolution to the bottleneck structure, which reduces computational load and fuses different scales of features simultaneously. For example, if the number of group convolutions in the bottleneck structure is 2, the two groups of convolution kernels will convolve two-scale features of the input. So different scale features interact on their respective scales, and then multiscale features are extracted with multiscale convolution kernels, respectively. The sparse bottleneck structure is realized by the 1 × 1 group convolution, thus its computational consumption is relatively low. But all negative activation values are filtered by the ReLU activation function, which weakens its structure and spoils feature extraction ability. Therefore, the sparse bottleneck structure abandoned ReLU activation, it only weights the input features linearly, followed by multiscale low-rank convolution portion directly. The experimental results show that the linear sparse bottleneck structure reaches higher classification accuracy, indicating the importance of the linear mechanism.

## III. Experiments

### A. Datasets

The performance of the proposed MSME-CNN model is measured on CIFAR and ImageNet datasets. The CIFAR-10 and CIFAR-100 datasets are composed of 60 000 32 × 32 pixel color images, divided into 50 000 training images and 10 000 testing images. There are 10 categories in CIFAR-10 while 100 categories in CIFAR-100, thus CIFAR-100 is more detailed and is more difficult to classify. For the CIFAR dataset, data augmentation scheme is adopted [27], [37]–[41]: padding four pixels on each side, and randomly cropping 32 × 32 pixel images which are sampled from the padded image or its horizontal flip. For testing, no data augmentation scheme is used.

The tiny ImageNet dataset has 200 categories, consisting of 120 000 64 × 64 pixel color images, which are divided into three parts: 1) training set; 2) validation set; and 3) testing set. The training set has 100 000 images, and the validation and testing sets have 10 000 images, respectively. Since images in the testing set are unlabeled, the prediction results cannot be verified, the validation set is used to verify the network performance. For the tiny ImageNet dataset, the data augmentation scheme is adopted [30]: the training images are randomly resized to [64, 80], and randomly cropped 64 × 64 pixel images which are sampled from the padded image or its horizontal flip. For testing, no data augmentation scheme is used.

ImageNet [19] contains approximately 1.28 million images of color objects of different sizes, with a total of 1000 categories. Compared with CIFAR and tiny ImageNet datasets, the ImageNet dataset is more difficult to classify. For the training set, the data augmentation method is that: scaling the short side of images to 256 pixels while maintaining the original aspect ratio, and randomly cropping 224 × 224 pixel images which are sampled from the padded image or its horizontal flip. For the training set, the data augmentation method is that: scaling the short side of images to 256 pixels while maintaining the original aspect ratio, and centrally cropping 224 × 224 pixel images, and only testing the center crop accuracy.

TABLE I
Network Structure of MSME-CNN for CIFAR Dataset

| layer | output dimension | output channel | group | MSME-CNN |
|---|---|---|---|---|
| Conv1_x | 32×32 | 64$k$ | $g_1$ | [1×1, 3×3] × 6 |
| Weighted1 | 32×32 | 128$k$ | $g_2$ | 1×1 |
| Pooling1 | 16×16 | 128$k$ | -- | 2×2 avg. |
| Conv2_x | 16×16 | 128$k$ | $g_1$ | [1×1, 3×3] × 6 |
| Weighted2 | 16×16 | 256$k$ | $g_3$ | 1×1 |
| Pooling2 | 8×8 | 256$k$ | -- | 2×2 avg. |
| Conv3_x | 8×8 | 256$k$ | $g_1$ | [1×1, 3×3] × 6 |
| Weighted3 | 8×8 | 10/100 | -- | 1×1 |
| Pooling3 | 1×1 | 10/100 | -- | 8×8 glo. avg. |
| Classification | -- | -- | -- | softmax |

TABLE II
Hyper-Parameter Settings of MSME-CNN With Different Parameter Quantities

| model | #params (M) | $k$ | $g_1$ | $g_2$ | $g_3$ | Dropout ratio |
|---|---|---|---|---|---|---|
| MSME-CNN 0.3× | 0.66 | 1 | 16 | 1 | 1 | 0.1 |
| MSME-CNN 0.4× | 0.83 | 1 | 8 | 1 | 1 | 0.1 |
| MSME-CNN 0.5× | 1.2 | 1 | 4 | 1 | 1 | 0.1 |
| MSME-CNN 0.7× | 1.7 | 2 | 32 | 2 | 4 | 0.3 |
| MSME-CNN 1.0× | 2.3 | 2 | 32 | 1 | 1 | 0.3 |

TABLE III
Comparison of MSME-CNN Classification Error Rates With Different Parameter Quantities

| model | #params (M) | CIFAR-10 (%) | CIFAR-100 (%) | tiny ImageNet (%) |
|---|---|---|---|---|
| MSME-CNN 0.3× | 0.66 | 5.35 | 23.4 | 38.48 |
| MSME-CNN 0.4× | 0.83 | 5.12 | 23.0 | 38.0 |
| MSME-CNN 0.5× | 1.2 | 4.95 | 22.57 | 37.3 |
| MSME-CNN 0.7× | 1.7 | 4.75 | 21.96 | 36.8 |
| MSME-CNN 1.0× | 2.3 | **4.65** | **21.7** | **36.4** |

TABLE IV
Comparison of Classification Error Rates Between MSME-CNN and ME-CNN

| model | | #params (M) | CIFAR-10 (%) |
|---|---|---|---|
| ME-CNN [33] | 21-1 | 0.33 | 7.33 |
| | 21-2 | 1.26 | 5.67 |
| | 21-4 | 5.00 | 5.10 |
| | 42-1 | 0.68 | 6.85 |
| | 42-2 | 2.70 | 5.62 |
| | 42-4 | 10.7 | 4.68 |
| MSME-CNN | 21-0.3× | 0.66 | 5.35 |
| | 21-0.4× | 0.83 | 5.12 |
| | 21-0.5× | 1.2 | 4.95 |
| | 21-0.7× | 1.7 | 4.75 |
| | 21-1.0× | 2.3 | **4.65** |

### B. Network Hyper-Parameter Settings

Caffe [36] is used for accelerating training. Our experimental environment is a desktop PC equipped with Intel core i7-6850K, 64-G memory and a 4-channel

TABLE V
COMPARISON OF CLASSIFICATION ERROR RATES BETWEEN MSME-CNN AND IGCV2

| model | depth | #params (M) | FLOPs (M) | CIFAR-10 (%) | CIFAR-100 (%) | tiny ImageNet (%) |
|---|---|---|---|---|---|---|
| IGCV2 [30] | 20 | 0.65 | 152 | 5.49 | **22.95** | 38.81 |
| MSME-CNN | 21 | 0.66 | 158 | **5.35** | 23.40 | **38.48** |

NVIDIA GTX1080Ti graphic card. The network hyper-parameters on CIFAR dataset are as follows.

1) The initial learning rate follows a multistep decay from 0.1 to 0.0001. The rate drops down to 0.01 in 100 epochs, 0.001 in 150 epochs, and 0.0001 in 200 epochs. Stop training when the loss no longer drops.
2) Training the network with SGD plus Nesterov [42] momentum of 0.9.
3) Adopting the MSRA [43] weight initialization method.
4) Batch size is set to 128 for all training, 32 batch per GPU.
5) Weight decay is set to 0.0005.

Table I shows the architecture of MSME-CNN for CIFAR dataset, where $k$ denotes the widening factor, $g_1$, $g_2$, and $g_3$ are the group numbers of group convolutions, "$[1\times1, 3\times3] \times 6$" means there are six multiscale parallel convolution layers, "avg." denotes average pooling, and "glo. avg." denotes global average pooling.

Based on the configuration of Table I, in order to construct MSME-CNN model with different parameter quantities, the above-mentioned four parameters, $k$, $g_1$, $g_2$, and $g_3$ have different settings, as shown in Table II. In addition, when $k$ is 1, the model is thin, and it is less prone to over-fitting, so the Dropout [44] ratio is small. In Table II, "#params" means the number of network parameters, and "Network $s\times$" means reducing the network parameters of "Network 1.0×" by $s$ times.

The hyper-parameter settings of MSME-CNN on the tiny ImageNet dataset are the same as that on the CIFAR dataset. In addition, in order to simplify the experiment, the network structure of all MSME-CNN on the tiny ImageNet dataset is the same as that on the CIFAR dataset, expect that the stride of the first convolution layer is set as 2, which is the same as in [30].

The hyper-parameter settings of MSME-CNN on the ImageNet dataset are as follows: 1) the initial learning rate is set to 0.1, then scaled by 0.95 by each epoch, and the total training epoch is 120; 2) training the network with SGD plus Nesterov momentum of 0.9; 3) adopting the MSRA weight initialization method; 4) batch size is set to 128 for training, 32 batch per GPU; and 5) weight decay is set to 0.00004.

## C. Experimental Results and Analysis

In order to validate the superiority of MSME-CNN, we carried out following experiments according to two aspects. First, the performance of MSME-CNN with different parameter quantities was evaluated, such as classification error rate and running time. Second, the performance of MSME-CNN was compared with state-of-the-art models, such as IGCV1 [30], IGCV2 [30], IGCV3 [31], MobileNetV2 [28], Swapout [45], DFN [46], ResNet [27], DFN-MR1 [49], etc.

TABLE VI
COMPARISON OF LIGHTWEIGHT MODEL CLASSIFICATION ERROR RATES WITH SIMILAR PARAMETER QUANTITIES

| model | #params (M) | CIFAR-10 (%) | CIFAR-100 (%) |
|---|---|---|---|
| IGCV1 [30] | 2.3 | 8.23 | 29.93 |
| IGCV2 [30] | 2.3 | 5.24 | 22.55 |
| IGCV3 [31] | 2.3 | 5.04 | 22.05 |
| MobileNetV2 [28] | 2.3 | 5.44 | 22.91 |
| MSME-CNN 1.0× | 2.3 | **4.65** | **21.70** |

TABLE VII
COMPARISON OF CLASSIFICATION ERROR RATES BETWEEN MSME-CNN AND IGCV3 WITH SIMILAR PARAMETER QUANTITIES

| model | #params (M) | CIFAR-10 (%) | CIFAR-100 (%) |
|---|---|---|---|
| IGCV3 0.5× [31] | 1.2 | 5.27 | 22.71 |
| MSME-CNN 0.5× | 1.2 | **4.95** | **22.57** |
| IGCV3 0.7× [31] | 1.7 | 5.08 | 22.17 |
| MSME-CNN 0.7× | 1.7 | **4.75** | **21.96** |
| IGCV3 1.0× [31] | 2.4 | 5.04 | 22.05 |
| MSME-CNN 1.0× | 2.3 | **4.65** | **21.70** |

The classification error rates of MSME-CNN models with different parameter quantities on CIFAR and tiny ImageNet datasets are shown in Table III. Since the purpose of building the MSME-CNN model is to implement a lightweight network, the parameter settings for all MSME-CNN models are similar to IGCV2 and IGCV3 for better comparison. It should be noted that all of the network models are trained with the augmented training set.

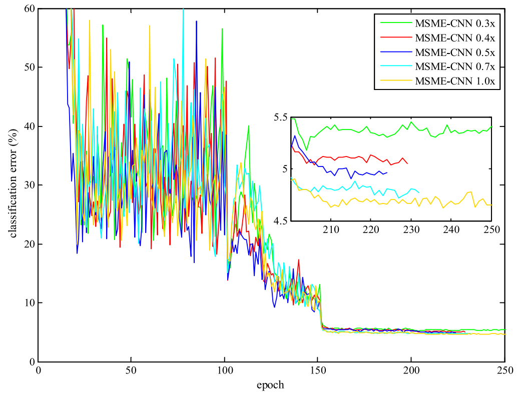
All the models in Table III have different parameter quantities by changing parameters $k$, $g_1$, $g_2$, and $g_3$. All MSME-CNN models in Table III have a depth of 21 layers and a small amount of parameters, and the smallest MSME-CNN model has a low classification error rate with only 0.66 M parameters, indicating that MSMS-CNN model has strong feature extraction ability. In addition, with the amount of model parameters increases, the model capacity increases gradually, so the classification error rates on three datasets also gradually decrease, and reach the lowest when the parameter amount reaches 2.3 M.

Fig. 3 shows the classification error rate curves of MSME-CNN with different parameter quantities on CIFAR and tiny ImageNet datasets. It can be seen from Fig. 3 that the classification error rates on three datasets decrease with the increase of parameter quantity.

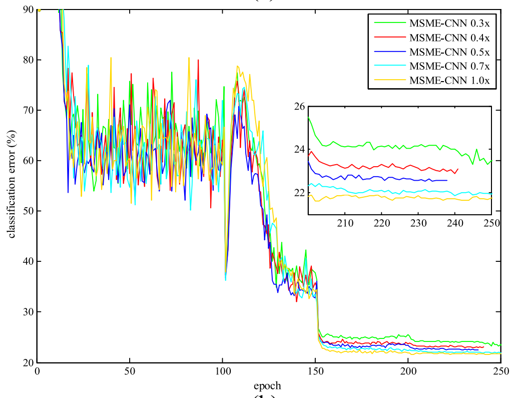
Table IV shows the comparison of classification error rates between MSME-CNN and ME-CNN on the augmented

TABLE VIII
RUNNING TIME COMPARISON OF MSME-CNN WITH DIFFERENT PARAMETER QUANTITIES ON CIFAR-10

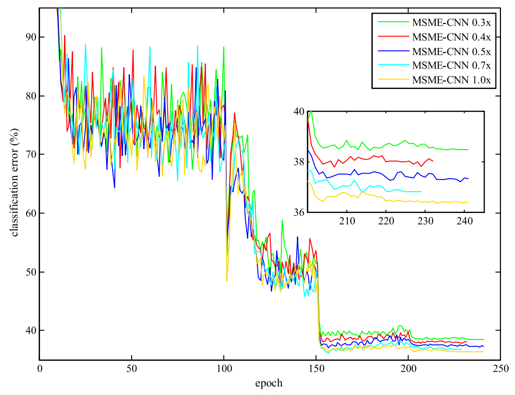
| model | #params (M) | forward propagation time (ms) | | backward propagation time (ms) | | total time (ms) | |
|---|---|---|---|---|---|---|---|
| | | CPU | GPU | CPU | GPU | CPU | GPU |
| MSME-CNN 0.3× | 0.66 | 36.4 | 9.2 | 33.2 | 9.5 | 70.2 | 18.8 |
| MSME-CNN 0.4× | 0.83 | 41.3 | 8.1 | 34.0 | 8.0 | 75.9 | 16.4 |
| MSME-CNN 0.5× | 1.2 | 44.5 | 7.5 | 36.2 | 7.1 | 81.3 | 14.7 |
| MSME-CNN 0.7× | 1.7 | 71.5 | 12.2 | 50.7 | 13.4 | 122.8 | 26.5 |
| MSME-CNN 1.0× | 2.3 | 73.9 | 12.4 | 52.2 | 13.4 | 126.8 | 26.1 |



(a)



(b)



(c)

Fig. 3. Classification error rate curves of MSME-CNN. (a) Classification error rate curves on CIFAR-10 dataset. (b) Classification error rate curves on CIFAR-100 dataset. (c) Classification error rate curves on tiny ImageNet dataset.

CIFAR-10 dataset, in which each model is represented by "depth-width." For example, the 21-1 model means 21-layer network with the widening factor of 1. It can be seen that the parameter efficiency of MSME-CNN is higher than that of ME-CNN. For example, the parameter quantities of "ME-CNN 21-2" and "MSME-CNN 21-0.5×" are 1.26 M and 1.2 M, respectively, and the classification error rates of the two models are 5.67% and 4.95%, respectively. The depths of these two models are the same and the parameter quantities are very close, but the classification error rate of MSME-CNN 21-0.5× is 0.72% lower than that of ME-CNN 21-2, indicating that MSME-CNN has stronger generalization ability and higher parameter efficiency. In addition, the parameter quantities of "ME-CNN 42-1" and "MSME-CNN 21-0.3×" are also very close, but the classification error rate of MSME-CNN 21-0.3× is much lower than that of ME-CNN 42-1, and its depth is only half of ME-CNN 42-1, which also shows that the proposed MSME-CNN model has higher parameter efficiency.

TABLE IX
IMPACT OF DIFFERENT HYPER-PARAMETERS ON NETWORK PERFORMANCE

| model | #params (M) | $k$ | $g_1$ | $g_2$ | $g_3$ | CIFAR-10 (%) |
|---|---|---|---|---|---|---|
| MSME-CNN 0.3×* | 0.66 | 0.5 | 1 | 2 | 4 | 5.56 |
| MSME-CNN 0.3× | 0.66 | 1 | 16 | 1 | 1 | **5.35** |
| MSME-CNN 1.0×* | 2.3 | 1 | 1 | 2 | 4 | 4.88 |
| MSME-CNN 1.0× | 2.3 | 2 | 32 | 1 | 1 | **4.65** |

Table V shows the comparison of classification error rates between MSME-CNN and IGCV2 on three datasets, in which the number of float-point multiplication-adds (FLOPs) is used to evaluate the network computational complexity. It can be seen from Table V that the depth, parameter quantity and FLOPs of MSME-CNN and IGCV2 are very close, but the classification error rates of MSME-CNN on CIFAR-10 and tiny ImageNet datasets are relatively low, indicating the parameter efficiency of MSME-CNN is high.

Table VI shows the classification error rates of MSME-CNN, three IGC models, and MobileNetV2 on the CIFAR dataset. The classification error rates of IGC and MobileNetV2 are obtained by [30]. It can be seen that the classification error rate of MSME-CNN on the CIFAR dataset is significantly lower than that of IGC and MobileNetV2, which indicates that the multipath ensemble mode, multiscale convolution kernel low-rank approximation and sparse connection enable MSME-CNN stronger feature extraction ability, generalization ability, and higher parameter efficiency.

Table VII compares classification error rates of MSME-CNN and IGCV3 with similar parameter quantities on the CIFAR dataset. It can be seen that in the case of similar parameter quantity, MSME-CNN has a lower rate than

TABLE X
COMPARISON OF CLASSIFICATION ERROR RATES OF DIFFERENT LIGHTWEIGHT NETWORK MODELS

| model | depth | #params (M) | CIFAR-10 (%) | CIFAR-100 (%) | tiny ImageNet (%) |
|---|---|---|---|---|---|
| Swapout [45] | 20 | 1.1 | 6.58 | 25.86 | -- |
| DFN [46] | 50 | 3.7 | 6.40 | 27.61 | -- |
| DFN [46] | 50 | 3.9 | 6.24 | 27.52 | -- |
| ResNet [27] | 110 | 1.7 | 6.41 | 27.76 | -- |
| ResNet(pre-act) [47] | 164 | 1.7 | 5.46 | 24.33 | -- |
| ResNet [48] | 110 | 1.7 | 5.52 | 28.02 | 46.5 |
| DFN-MR1 [49] | 56 | 1.7 | 4.94 | 24.46 | -- |
| RiR [50] | 18 | 10.3 | 5.01 | 22.90 | -- |
| ResNet34 [51] | 34 | 21.4 | -- | -- | 46.9 |
| ResNet18-2× [51] | 18 | 25.7 | -- | -- | 44.6 |
| WRN-32-4 [48] | 32 | 7.4 | 5.43 | 23.55 | 39.63 |
| DenseNet($k$ = 12) [48] | 40 | 1.0 | -- | -- | 39.09 |
| DenseNet($k$ = 12) [37] | 40 | 1.0 | 5.24 | 24.42 | -- |
| DenseNet-BC($k$ = 12) [37] | 100 | 0.8 | **4.51** | 22.27 | -- |
| IGCV2 [30] | 20 | 0.65 | 5.49 | 22.95 | 38.81 |
| VGG-16-pruned [52] | 16 | 5.40 | 6.60 | 25.28 | -- |
| VGG-19-pruned [53] | 19 | 2.30 | 6.20 | -- | -- |
| ResNet-164-B-pruned [53] | 164 | 1.21 | 5.27 | 23.91 | -- |
| DenseNet-40-pruned [53] | 40 | 0.66 | 5.19 | 25.28 | |
| MSME-CNN 0.3× | 21 | 0.66 | 5.35 | 23.4 | 38.48 |
| MSME-CNN 1.0× | 21 | 2.3 | 4.65 | **21.7** | **36.4** |

IGCV3 on both CIFAR-10 and CIFAR-100 datasets. In particular, the parameter quantity of MSME-CNN 0.5× is only half of IGCV3 1.0×, but its classification error rate is lower, so the parameter efficiency of MEME-CNN 0.5× is more than double of IGCV3 1.0×. In addition, the depth of IGCV3 0.5× is 50 layers, while the depth of MSME-CNN is only 21 layers. In summary, MSME-CNN shows more advantages in terms of parameter efficiency and model depth than IGCV3.

We researched the computer consumed the time of MSME-CNN with different parameter quantities when the size of the input image of CIFAR-10 is 32 × 32 pixel and the batch size is 1. We recorded the experimental results of 50 iterations and take the average. Table VIII shows the average values of forward propagation time, backward propagation time, and total training time. It can be seen that: 1) as the parameter quantity increases, the model becomes more complex, resulting in a gradual increase in the propagation time; 2) from MSME-CNN 0.3× to MSME-CNN 0.7×, as the parameter quantity increases, the propagation time first decreases and then rises. This is because when the parameter quantity is small, the model width $k$ is small and the number of group convolutions $g$ is relatively large. However, the Caffe framework does not specifically optimize the group convolution. Therefore, the model propagates faster as $g$ becomes smaller. From MSME-CNN 0.7×, $k$ becomes larger and $g$ becomes smaller, so the propagation time is also gradually reduced. 3) The GPU running time of all models is much smaller than the CPU running time, indicating that GPU is more suitable for network training and parallel processing; and 4) Theoretically, the total time should be equal to the sum of the forward and backward propagation time. However, since the computer time here is in milliseconds and the operations, such as average value calculation and screen display are involved, the total time is not strictly equal to the sum of forward and backward propagation time.

Table IX shows the effect of four hyper-parameters $k$, $g_1$, $g_2$, and $g_3$ on network performance under the same parameter

TABLE XI
NETWORK STRUCTURE OF MSME-CNN FOR IMAGENET DATASET

| layer | output dimension | output channel | group | MSME-CNN |
|---|---|---|---|---|
| Conv | 112×112 | 24 | -- | 3×3 |
| Pooling1 | 56×56 | 24 | -- | 3×3 max. |
| Conv1_x | 56×56 | 32 | 2 | [1×1, 3×3] × 3 |
| Pooling2 | 28×28 | 96 | -- | 2×2 avg. |
| Conv2_x | 28×28 | 192 | $\begin{bmatrix}2\\16\end{bmatrix}$ | $\begin{bmatrix}1\times1\\1\times1,\ 3\times3\end{bmatrix}\times5$ |
| Weighted1 | 28×28 | 192 | 32 | 1×1 |
| Pooling3 | 14×14 | 192 | -- | 2×2 avg. |
| Conv3_x | 14×14 | 384 | $\begin{bmatrix}2\\16\end{bmatrix}$ | $\begin{bmatrix}1\times1\\1\times1,\ 3\times3\end{bmatrix}\times7$ |
| Weighted2 | 14×14 | 384 | 32 | 1×1 |
| Pooling4 | 7×7 | 384 | -- | 2×2 avg. |
| Conv4_x | 7×7 | 768 | $\begin{bmatrix}2\\16\end{bmatrix}$ | $\begin{bmatrix}1\times1\\1\times1,\ 3\times3\end{bmatrix}\times4$ |
| Weighted3 | 7×7 | 1024 | 32 | 1×1 |
| Pooling5 | 1×1 | 1000 | -- | 7×7 glo. avg. |
| Fc | -- | 1000 | -- | -- |
| Classification | -- | -- | -- | softmax |

quantity. The model with "*" is a new model generated by changing the four hyper-parameters. With the same number of parameters, the larger the $k$ and $g_1$ are, the lower error the model gets, indicating that a large number of convolution channels is better in the case of the same model complexity. In general, the larger the number of convolution channels is, the more adequate the extracted features are, and the larger $g_1$ is, the smaller the interchannel redundancy, so network performance can be improved without changing the model complexity.

Table X shows the comparison of classification error rates of MSME-CNN and different lightweight models on the augmented CIFAR-10, CIFAR-100, and tiny ImageNet

TABLE XII
COMPARISON OF CLASSIFICATION ACCURACY BETWEEN MSME-CNN AND OTHER MODELS ON IMAGENET DATASET

| model | #params (M) | FLOPs (M) | Top-1 (%) | Top-5 (%) |
|---|---|---|---|---|
| AlexNet [18] | 60 | 720 | 57.2 | 80.3 |
| GoogLeNet [22] | 6.8 | 1550 | 69.8 | -- |
| VGG16 [17] | 138 | 15300 | 71.5 | 90.1 |
| SqueezeNet [20] | 1.25 | 833 | 57.5 | 80.3 |
| MobileNetV1 [25] | 4.2 | 569 | 70.6 | 89.5 |
| MSME-CNN$^{\dagger}$ | 4.1 | 1461 | **71.9** | **90.5** |
| MSME-CNN$^{\dagger\dagger}$ | 3.3 | 339 | 68.1 | 88.2 |
| MSME-CNN | 3.3 | 339 | 69.3 | 89.1 |

datasets. It can be seen that the proposed MSME-CNN model achieves a lower classification error rate with only a few layers and parameter quantity. The classification error rates of MSME-CNN with only 21 layers and 0.66 M parameter quantity on the three datasets are lower than most lightweight models, such as Swapout [44] and ResNet [27]. MSME-CNN with the maximum parameter quantity of 2.3 M gets the lowest error rates on CIFAR-100 and tiny ImageNet datasets, which indicates that the proposed MSME-CNN model has strong generalization ability.

Table XI shows the architecture of MSME-CNN for ImageNet dataset. Table XII compares MSME-CNN with other models on the ImageNet dataset, where "†" means MSME-CNN without linear sparse bottleneck structure, and "††" means MSME-CNN with nonlinear sparse bottleneck structure. All the used baseline models, including AlexNet [18], GoogLeNet [22], VGG16 [17], SqueezeNet [20], and MobileNetV1 [25] adopted two indexes, Top-1 and Top-5 classification accuracies, to measure the network performance. Therefore, we show the Top-1 and Top-5 results of our MSME-CNN on the ImageNet dataset for the ease of comparison. The Top-1 classification accuracy compares the ground truth against the first predicted class, and the Top-5 compares the ground truth against the first five predicted classes: an image is deemed correctly classified if the ground truth is among the Top-5, regardless of its rank in them [22]. It can be seen from Table XI that the proposed MSME-CNN without linear sparse bottleneck structure has achieved good results, and the computational load and parameter quantity are smaller than those of GoogLeNet and VGG16. The MSME-CNN model with linear sparse bottleneck structure achieved a Top-1 classification accuracy of 69.3%, and the computational complexity is relatively low, achieving a good tradeoff between computational consumption and classification accuracy. In addition, the computational loads and parameter quantities of nonlinear and linear sparse bottleneck structures are exactly the same, but the linear sparse bottleneck structure has higher classification accuracy, indicating that the linear structure loses less information.

It can be observed from experimental results that: 1) using the multiscale low-rank convolution kernel to approximate the high-rank convolution kernel can effectively reduce the computational load and parameter quantity of the network, thus improve the computational efficiency; 2) the convolution kernel sparse connection mechanism further reduces the computational load; 3) the linear sparse bottleneck structure compresses the network channel by $1 \times 1$ group convolution without activation function, and fuses the multiscale features, respectively, with linear structure retaining input information to the greatest extent; and 4) benefit from the above three structures, the proposed MSME-CNN has achieved high computational efficiency, and the performances on CIFAR and ImageNet datasets are superior to the discussed models.

## IV. CONCLUSION

CNNs have the issue of a large amount of parameters and high computational load. On the basis of ME-CNN, an improved MSME-CNN is proposed. By comparing with start-of-the-art models, MSME-CNN is characterized by: 1) using multipath ensemble structure as the basic network can solve gradient vanishing problem and fuse multilayer features; 2) replacing high-rank convolution kernel with multiscale low-rank convolution kernel can effectively reduce computational load and extract more abundant features; 3) introducing the group convolution sparse connection mechanism to extract different input features with different convolution kernels, which helps to remove convolution redundancy; and 4) adding a linear sparse bottleneck structure before multiscale low-rank convolution can reduce the number of input channels and fuse different scale features, respectively. The linear mechanism avoids information loss by bottleneck structure and further improves network performance. The experimental results on CIFAR and ImageNet datasets show that MSME-CNN can achieve higher classification accuracy and computational efficiency.

## REFERENCES

[1] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989.
[2] M. L. Wang, H.-X. Li, X. Chen, and Y. Chen, "Deep learning-based model reduction for distributed parameter systems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 12, pp. 1664–1674, Dec. 2016.
[3] K. Muhammad, J. Ahmad, Z. H. Lv, P. Bellavista, P. Yang, and S. W. Baik, "Efficient deep CNN-based fire detection and localization in video surveillance applications," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 46, no. 7, pp. 1419–1434, Jul. 2019, doi: 10.1109/TSMC.2018.2830099.
[4] A. Kamel, B. Sheng, B. Yang, P. Li, R. Shen, and P. Yang, "Deep convolutional neural networks for human action recognition using depth maps and postures," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 9, pp. 1806–1819, Sep. 2019, doi: 10.1109/TSMC.2018.2850149.
[5] T. Zhang, G. X. Su, C. M. Qing, X. M. Xu, B. L. Cai, and X. F. Xing, "Hierarchical lifelong learning by sharing representations and integrating hypothesis," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, doi: 10.1109/TSMC.2018.2884996.
[6] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. J. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *Proc. Conf. Learn. Represent.*, San Juan, Puerto Rico, 2016, pp. 1–16.
[7] X. Y. Zhang, J. H. Zou, K. M. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016.
[8] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 1269–1277.
[9] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1379–1387.

[10] Z. W. Zhuang *et al.*, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 883–894.

[11] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," in *Proc. Empiri. Methods Natural Lang. Process.*, Austin, TX, USA, 2016, pp. 1317–1327.

[12] G. B. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 742–751.

[13] J. X. Wu, C. Leng, Y. H. Wang, Q. H. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 4820–4828.

[14] I. Hubara, M. Courbariaux, D. Soudry, R. Ei-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, Jan. 2017.

[15] F. Juefei-Xu, V. N. Boddeti, and M. Savvides, "Local binary convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 19–28.

[16] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, Lille, France, 2015, pp. 448–456.

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

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[19] 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.* Miami, FL, USA, 2009, pp. 248–255.

[20] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size," 2016. [Online]. Available: arXiv:1602.07360.

[21] Y. C. Gong, L. Liu, M. Yang, and L. Bourdey, "Compressing deep convolutional networks using vector quantization," 2014. [Online]. Available: arXiv:1412.6115.

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

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Woina, "Rethinking the inception architecture for computer vision" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 2818–2826.

[24] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 1251–1258.

[25] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.

[26] X. Y. Zhang, X. Y. Zhou, M. X. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 6848–6856.

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

[28] M. Sandler, A. Howard, M. L. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 4510–4520.

[29] T. Zhang, G.-J. Qi, B. Xiao, and J. D. Wang, "Interleaved group convolutions," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 4373–4382.

[30] G. T. Xie, J. D. Wang, T. Zhang, and J. H. Lai, "IGCV2: Interleaved structured sparse convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 8847–8856.

[31] K. Sun, M. J. Li, D. Liu, and J. D. Wang, "IGCV3: Interleaved low-rank group convolutions for efficient deep neural networks," in *Proc. British Mach. Vis. Conf.*, Newcastle, U.K., 2018, pp. 1–13.

[32] A. Veit, M. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 550–558.

[33] X. Wang, A. Bao, Y. Cheng, and Q. Yu, "Multipath ensemble convolutional neural network," *IEEE Trans. Emerg. Topics Comput. Intell.*, early access, doi: 10.1109/TETCI.2018.2877154.

[34] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Rep. TR-2009, 2009.

[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, 2010, pp. 807–814.

[36] Y. Q. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Conf. Multimedia*, Orlando, FL, USA, 2014, pp. 675–678.

[37] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 4700–4708.

[38] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proc. Conf. Learn. Represent.*, Banff, AB, Canada, 2014, pp. 1–10.

[39] C.-Y. Lee, S. N. Xie, P. Gallagher, Z. Y. Zhang, and Z. W. Tu, "Deeply-supervised nets," in *Proc. Conf. Artif. Intell. Stat.*, San Diego, CA, USA, 2015, pp. 562–570.

[40] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 2377–2385.

[41] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *Proc. Europe Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 646–661.

[42] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 1139–1147.

[43] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. Int. Conf. Comput. Vis.*, Santiago, Chile, 2015, pp. 1026–1034.

[44] 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, Jan. 2014.

[45] S. Singh, D. Hoiem, and D. Forsyth, "Swapout: Learning an ensemble of deep architectures," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 28–36.

[46] J. D. Wang, Z. Wei, T. Zhang, and W. J. Zeng, "Deeply-fused nets," 2016. [Online]. Available: arXiv:1605.07716.

[47] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Europe Conf. Comput. Vis.*, Amsterdam, The Netherlands, 2016, pp. 630–645.

[48] G. Huang, Y. X. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," 2017. [Online]. Available: arXiv:1704.00109.

[49] L. M. Zhao, J. D. Wang, X. Li, Z. W. Tu, and W. J. Zeng, "On the connection of deep fusion to ensembling," 2016. [Online]. Available: arXiv:1611.07718.

[50] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," in *Proc. Conf. Learn. Represent.*, San Juan, Puerto Rico, 2016, pp. 1–7.

[51] L. Cordeiro, "Wide residual network for the tiny imagenet challenge," Stanford Univ., Stanford, CA, USA, Rep. 2017.927, 2017.

[52] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016. [Online]. Available: arXiv:1608.08710.

[53] Z. Liu, J. G. Li, Z. Q. Shen, G. Huang, S. M. Yan, and C. S. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. Int. Conf. Comput. Vis.*, Venice, Italy, 2017, pp. 2736–2744.

**Xuesong Wang** (Member, IEEE) received the Ph.D. degree in control science and technology from the China University of Mining and Technology, Xuzhou, China, in 2002.

She is currently the Dean of the Engineering Research Center of Intelligent Control for Underground Space, Ministry of Education, and the Xuzhou Key Laboratory of Artificial Intelligence and Big Data, and a Professor with the School of Information and Control Engineering, China University of Mining and Technology. Her main research interests include machine learning, bioinformatics, and artificial intelligence.

Prof. Wang is currently an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS and *International Journal of Machine Learning and Cybernetics*.

**Achun Bao** received the B.S. degree in control engineering from Shandong University, Jinan, China, in 2016. He is currently pursuing the master's degree in control science and engineering with the China University of Mining and Technology, Xuzhou, China.

His research direction is machine learning and deep learning.

**Enhui Lv** received the Ph.D. degree in control science and engineering from the China University of Mining and Technology, Xuzhou, China, in 2019.

He is currently a Lecturer with the School of Medical Information and Engineering, Xuzhou Medical University, Xuzhou. His main research interests include image classification and deep learning.

**Yuhu Cheng** (Member, IEEE) received the Ph.D. degree in control theory and control technology from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2005.

He is currently a Professor with the Engineering Research Center of Intelligent Control for Underground Space, Ministry of Education, the Xuzhou Key Laboratory of Artificial Intelligence and Big Data, and the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China. His main research interests include machine learning and intelligent system.