Wang et al. IPSJ Transactions on Computer Vision and Applications (2018) 10:11
https://doi.org/10.1186/s41074-018-0047-6

IPSJ Transactions on Computer Vision and Applications

RESEARCH PAPER Open Access

An multi-scale learning network with depthwise separable convolutions

Gaihua Wang[1,2], Guoliang Yuan[2*], Tao Li[2] and Meng Lv[2]

Abstract

We present a simple multi-scale learning network for image classification that is inspired by the MobileNet. The proposed method has two advantages: (1) It uses the multi-scale block with depthwise separable convolutions, which forms multiple sub-networks by increasing the width of the network while keeping the computational resources constant. (2) It combines the multi-scale block with residual connections and that accelerates the training of networks significantly. The experimental results show that the proposed method has strong performance compared to other popular models on different datasets.

Keywords: Multi-scale, MobileNet, Residual connections, Image classification

1 Introduction

Convolutional neural network (CNN) has been proposed since the late 1970s [1], and the first successful application is the LeNet [2]. In CNNs, weights in the network are shared, and pooling layers are spatial or temporal sampling using invariant function [3, 4]. In 2012, AlexNet [5] was proposed to use rectified linear units (ReLU) instead of the hyperbolic tangent as activation function while adding dropout network [6] to decrease the effect of overfitting. In subsequent years, further progress has been made by using deeper architectures [7–10].

For multiple-scale leaning network, in 2015, He et al. [11] proposed a ResNet architecture that consists of many stacked "residual units." Szegedy et al. [12] proposed an inception module by using a combination of all filter sizes 1×1, 3×3, and 5×5 into a single output vector. In 2017, Xie [13] proposed the ResNeXt network structure, which is a multiple-scale network by using "cardinality" as an essential factor. Moreover, the multiple-scale architectures have also been successfully employed in the detection [14] and feature selection [15]. The bigger size means a larger number of parameters, which makes the network prone to overfitting. In addition, larger network size can increase computational resources. Some efficient network architectures [16, 17] are proposed in order to build smaller, lower latency models. The MobileNet [18] is built primarily by depthwise separable convolutions for mobile and embedded vision applications. The ShuffleNet [19] utilizes pointwise group convolution and channel shuffle to greatly reduce the computation cost while maintaining the accuracy.

Motivated by the analysis above, in this paper, we use the multiple-scale network to construct a convolutional module. Different sizes can become more robust to scale the changes. Then, we use depthwise separable convolutions to modify the convolutional module. In addition, residual connections are combined into the network. The experimental results show that the proposed method has better performance and less parameter on different benchmark datasets for image classification.

The remaining of this paper is organized as follows: Section 2 reviews the related work. Section 3 details the architecture of the proposed method. Section 4 describes our experiment and discusses the results of our experiments. Section 5 concludes the paper.

2 The depthwise separable convolutions

Depthwise separable convolutions divide standard convolution into a depthwise convolution and a 1×1 pointwise convolution [18]. The depthwise convolution applies a single filter to each input channel, given the feature map is expressed by $D_F \times D_F \times M$. Depthwise convolution with one filter per input is as follows (Eq. (1)):

* Correspondence: guoliang_yuan@hotmail.com
[2]School of Electrical and Electronic Engineering, Hubei University of Technology, Wuhan 430068, China
Full list of author information is available at the end of the article

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \quad (1)$$

where $\hat{K}$ is the depthwise convolutional kernel of the size $D_k \times D_k \times M$. F is the feature map of the input. The m_{th} filter in $\hat{K}$ is applied to the m_{th} channel in F to produce output feature maps $\hat{G}$. i, j represent the pixel position of the convolutional kernel. k, l are the pixel positions of the feature map.

Pointwise convolution is a simple 1×1 convolution and used to create a linear combination of the depthwise layer. The architecture is shown in Fig. 1. The standard convolutional filters (Fig. 1a) are replaced by two layers: depthwise convolution (Fig. 1b) and pointwise convolution (Fig. 1c). The depthwise separable convolutions have the effect of drastically reducing the computation and model parameters.

3 The proposed method

Our method increases the multi-scale features of the network. It contains features at different scales of the input. In the merge of these features, we adopt two different merge methods: feature connection and feature addition. The two kinds of merge methods can be expressed by the following: Eq. (2) and Eq. (3) (our method 1 and method 2), respectively. And with the depthwise separable convolution, it reduces the complexity of calculations. G represents the feature map. h and w represent the width and height of the features map, respectively. G_n^m represents the m_{th} input feature maps at the n_{th} scale that need to be merged. i and j represent the corresponding pixel position.

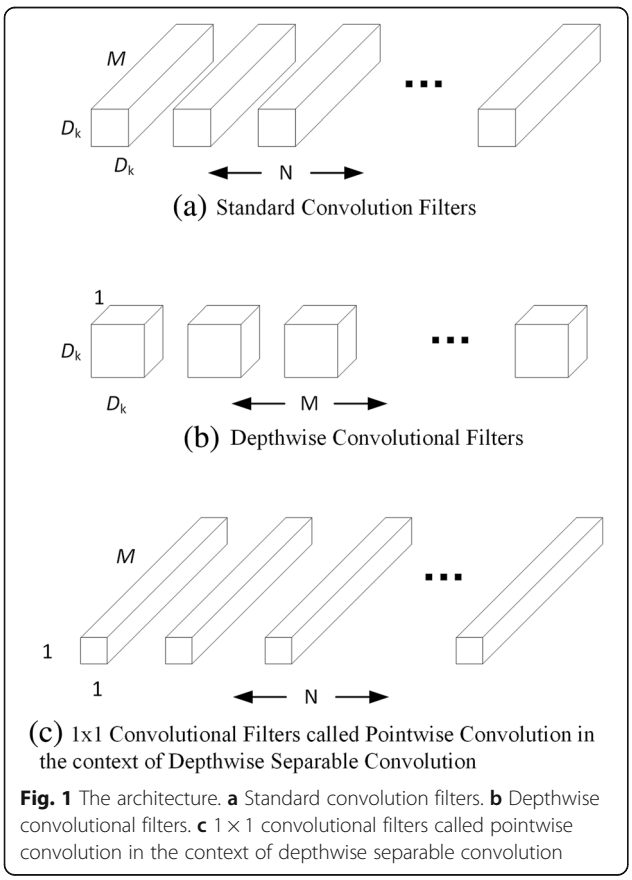

Fig. 1 The architecture. **a** Standard convolution filters. **b** Depthwise convolutional filters. **c** 1×1 convolutional filters called pointwise convolution in the context of depthwise separable convolution

$$G_{h \times w \times (m \times n)} = \text{concat}\left(G_1^1, \cdots, G_n^m\right) \quad (2)$$

$$G_{h \times w \times m} = \sum_{i,j \in h,w} \left(G_{1\,i \times j}^1 + \cdots + G_{n\,i \times j}^m\right) \quad (3)$$

In Eq. (2), concat is a connecting function. The feature is connected by different scales to a larger dimension. In Eq. (3), the feature is composed of the sum of the corresponding pixel of the different scale feature maps. The structures of the multi-scale separable convolutional unit are shown in Fig. 2. Figure 2a is a network unit of our method 1. Figure 2b is a network unit of our method 2. N represents the number of extended feature maps. The number of feature maps is increased by 1×1 pointwise convolution. From Fig. 2, multi-scale separable convolutions are constructed by using a 3×3 convolution layer and two 3×3 convolution layers. They are connected (Fig. 2a) or added (Fig. 2b) by different forms, which increase the width of the network. For smaller input datasets, we usually choose 2–3 units to construct the network. The network structure (input is 32×32) is described in detail (Table 1). It is composed of three parts: input layer, three network units, and two fully connected layers.

To compare the different network, we also split GoogLeNet, AlexNet, and MobileNet into different units respectively (Fig. 3). Figure 3a is a standard convolutional filter that is applied into AlexNet. Figure 3b is a depthwise separable convolutional filter that is applied into MobileNet. Figure 3c is the structure of GoogLeNet.

We give the calculation of the training parameters (Table 2). The training parameters of GoogLeNet are far more than MobileNet (almost 13 times). AlexNet is seven times more than MobileNet. And our methods are only two times MobileNet. But their performances are significantly higher than the MobileNet. The specific results can be seen in the experimental part (Section 4).

4 Experiments

In this section, the experiments on several public datasets are conducted to demonstrate the effectiveness of the proposed method. We also compare the proposed method with some existing methods including GoogLeNet, AlexNet, and MobileNet. Different layers are used for different datasets.

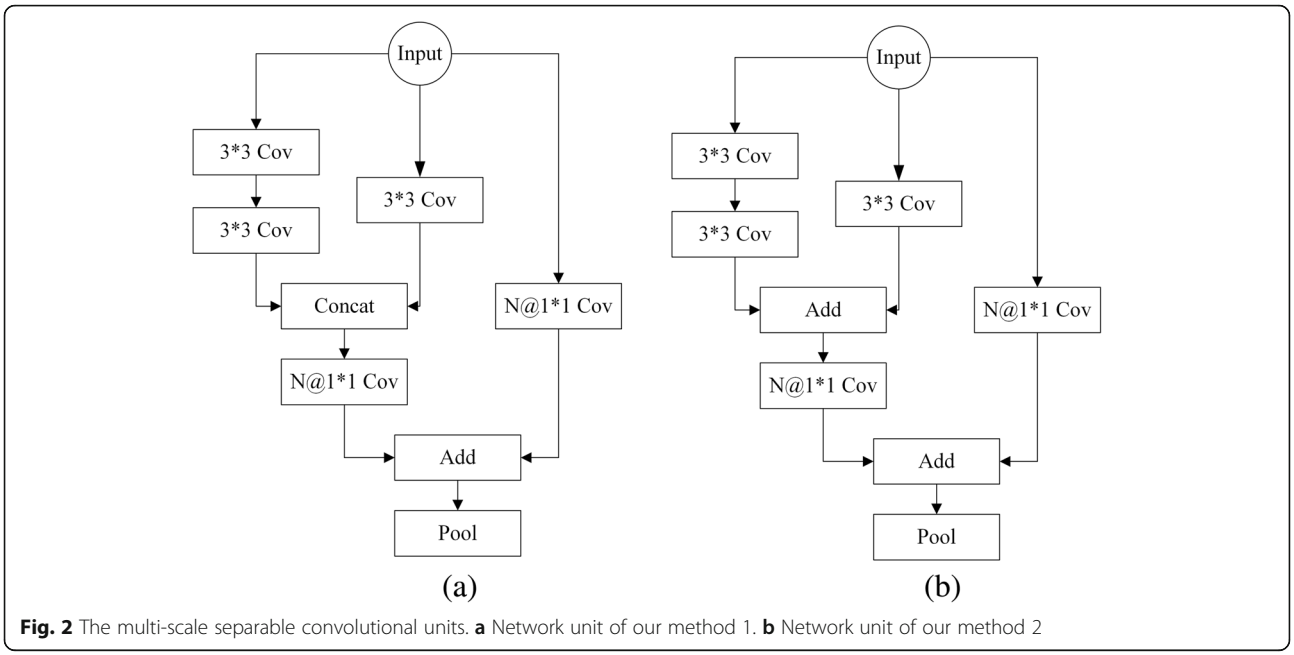

Fig. 2 The multi-scale separable convolutional units. **a** Network unit of our method 1. **b** Network unit of our method 2

4.1 Experiments on MNIST

The MNIST dataset contains 60,000 training and 10,000 test images of ten handwritten digits, which consist of 28×28 pixel gray images. We use two multi-scale separable convolutional units for our methods. Figure 4 is the visualization of tensorflow for our method 1. To compare the performance, we also choose two units for GoogLeNet, AlexNet, and MobileNet.

The experimental data can be shown in Table 3. We use 128 training images as a batch. Iteration of training is for 5000 batches. It can be seen that the accuracy of our method 2 is the highest, at 99.03%. The accuracy of our method 1 is 98.99%. MobileNet is only 94.59%, AlexNet is 97.80%, and GoogLeNet is 97.98%. According to Table 2, the parameters on different networks are compared. MobileNet has the least parameters. Our methods are twice more than MobileNet. GoogLeNet has the most parameters that need more computational resource.

Table 1 The proposed body architecture

Layers	Out size	K size	Stride	*N*	Out channels	
					Method 1	Method 2
Input	32×32					
Unit 1	16×16	3×3, 1×1	1	32	32	64
Unit 2	8×8	3×3, 1×1	1	64	64	128
Unit 3	4×4	3×3, 1×1	1	128	128	256
Fc1	384					
Fc2	192					

4.2 Experiments on CIFAR-10

The CIFAR-10 dataset is composed of ten classes. It splits into 50,000 train images and 10,000 test images. Each image is an RGB image of 32×32 pixels. Considering the larger parameters, GoogLeNet is only composed of two units. All other networks use three-layer units. We use 128 training images as a batch. Iteration of training is for 5000 batches.

The experimental results in Table 4 show that GoogLeNet's accuracy is the best for CIFAR-10. The accuracy of our method 1 is 72.4%. The accuracy of our method 2 is 71.1%. AlexNet is in between the two. MobileNet is only 65.6%. It also shows that AlexNet and GoogLeNet require the most training parameters. The training parameters of AlexNet are 13 times that of MobileNet. But its accuracy is only increased by 10%, the same with GoogLeNet that is more obvious. Our method 1 and our method 2 are up to 7% compared to MobileNet. The parameters of our methods are only twice more than that of MobileNet.

4.3 Experiments on the SVHN

SVHN is a dataset of Google's Street View House Numbers. It contains about 600,000 digit images, coming from a significantly harder real-world problem. It has two formats: full numbers and cropped digits. And we use the second one. All digits have been resized to a fixed resolution of 32×32 pixels. Considering the similarities between experiment 3 and experiment 2, such as the number of categories, the size of the image pixels, and so on, we adopt the network framework of

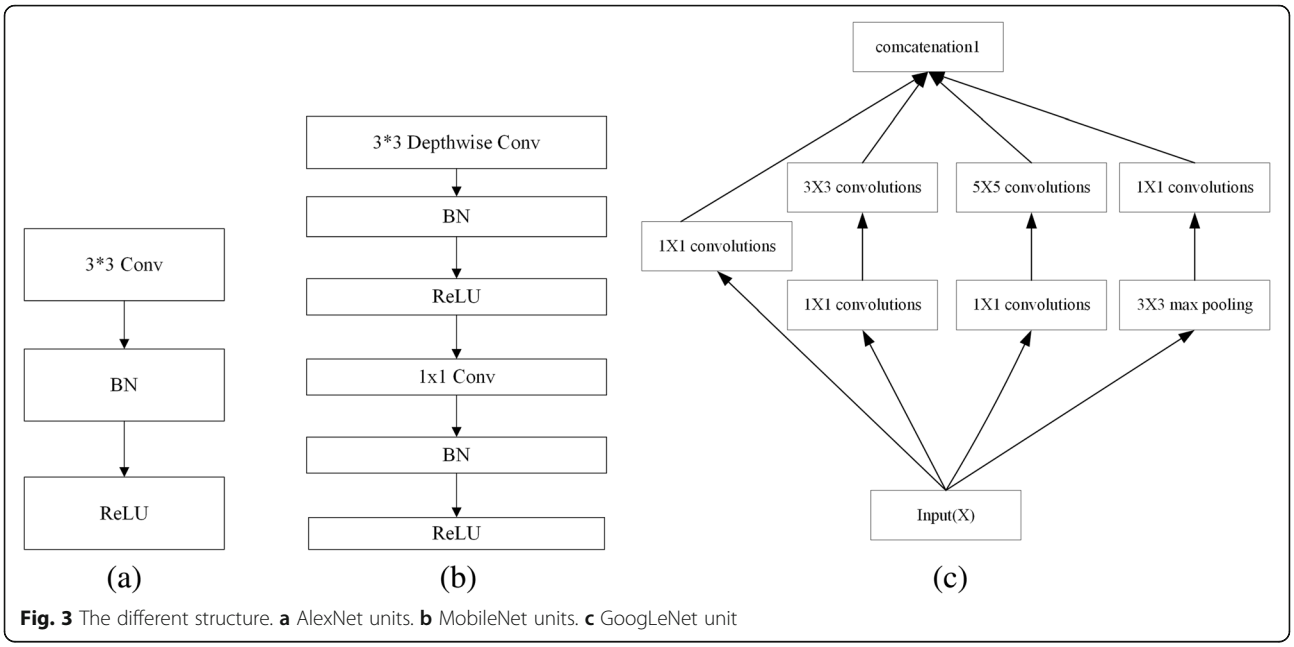

Fig. 3 The different structure. **a** AlexNet units. **b** MobileNet units. **c** GoogLeNet unit

experiment 2. GoogLeNet uses two network units, and others use three network units.

We use 128 training images as a batch. Iteration of training is for 10,000 batches. The experimental results are shown in Table 5. GoogLeNet and AlexNet's accuracy is very close (92.3 and 92.0%). The accuracy of our method 1 is 91.6%. The accuracy of our method 2 is 91.3%. MobileNet is 90.8%, but AlexNet and GoogLeNet require the most training parameters. Our method overcomes this shortcoming and improves its performance obviously. Meanwhile, it only needs two times the parameters of MobileNet.

4.4 Experiments on Tiny ImageNet

Tiny ImageNet is a subset of the ImageNet challenge (ILSVRC). It contains 200 different categories. Each class has 500 training images, 50 validation images, and 50 test images. In addition, the images are resized to 64 × 64 pixels (256 × 256 pixels in standard ImageNet).

Table 6 is the result of our experiment; the accuracy of GoogleNet (48.26) is the highest. AlexNet (44.25%) is the second highest. Their complexities are high and need more computational resource. Although MobileNet has lower computational complexity, its accuracy (34.53%) is very low. Our methods (method 1 and method 2) improve the testing accuracy while reducing the computational resource.

5 Discussion

To compare the performance, the curves of different networks are also analyzed. Figure 5 is the accuracy of the different network on MNIST. The maximum iterations of training are for 5000 batches. Testing is once per training 100 times. It can show that the performance of multi-scale networks is significantly best.

In Fig. 6, the accuracy of our methods and AlexNet is very close. GoogLeNet is relatively higher. MobileNet is relatively low. But our methods have a clear advantage over GoogLeNet and AlexNet in terms of training parameters.

In Fig. 7, it is the comparison of SVHN accuracy. The maximum iterations are for 10,000 batches. Testing is once per training 100 times. After 5000 iterations, we can find that the test accuracy is a clear upward trend. So we choose 10,000 as maximum iterations.

In Fig. 8, the test accuracy of different networks with different iterations is shown. Because of the complexity of the database, we set different parameters. The maximum iterations are for 42,000 batches. Testing is once per training 3000 times.

In contrast to Figs. 5, 6, 7, and 8, it can be found that different network structures show different test performances, and the same network also shows a big difference. On MNIST, our methods get the best accuracy. And they converge faster than GoogleNet, AlexNet, and

Table 2 Training parameters of different network unit

Network structure	GoogLeNet	AlexNet	MobileNet	Method 1	Method 2
Number of training parameters	N + N + N*9 + N + N*5*5 + N	N*3*3	N*1*1 + 3*3	3*3*3 + 2*N*1*1	
If $N = 32$	1216	288	41	91	

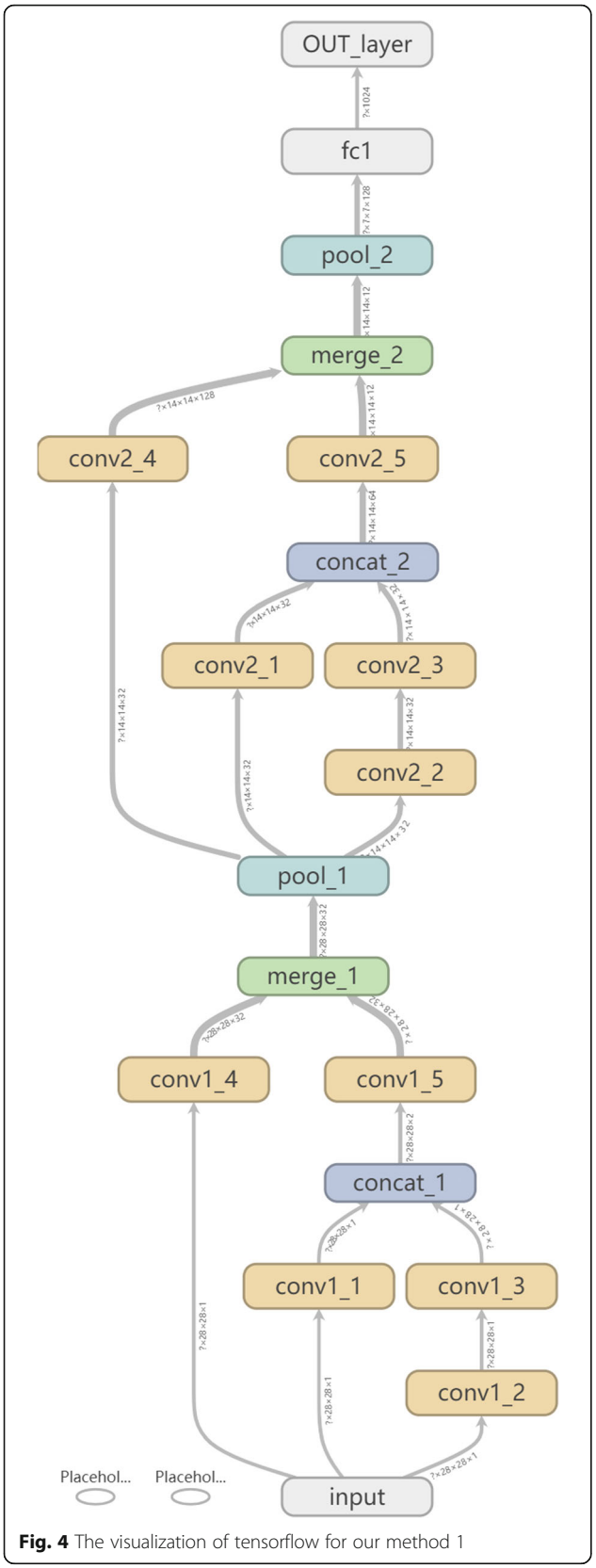

Fig. 4 The visualization of tensorflow for our method 1

Table 3 The performance of different network on MNIST

Methods	Iterations, 5000; batch_size, 128	
	Acc(%)	Number of training parameters
GoogLeNet	97.98	6640
AlexNet	97.80	864
MobileNet	94.59	114
Our method 1	98.99	246
Our method 2	99.03	246

Table 4 The performance of different network on CIFAR-10

Method	Iterations, 5000; batch_size, 128	
	Acc(%)	Number of training parameters
GoogLeNet	76.5	6640
AlexNet	75.7	2016
MobileNet	65.6	251
Our method 1	72.4	529
Our method 2	71.1	529

Table 5 The performance of different network on SVHN

Method	Iterations, 10000; batch_size, 128	
	Acc(%)	Number of training parameters
GoogLeNet	92.3	6640
AlexNet	92.0	2016
MobileNet	90.8	251
Our method 1	91.6	529
Our method 2	91.3	529

Table 6 The performance of different network on Tiny ImageNet

Method	Iterations, 42000; batch_size, 256	
	Acc(%)	Number of training parameters
GoogleNet	48.26	6640
AlexNet	44.25	2016
MobileNet	34.53	251
Our method 1	43.19	529
Our method 2	41.80	529

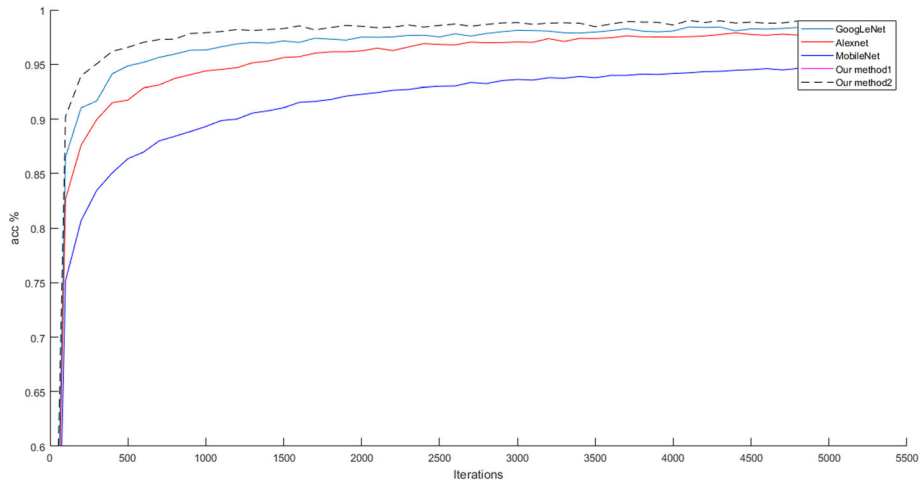

Fig. 5 The comparison of MNIST accuracy

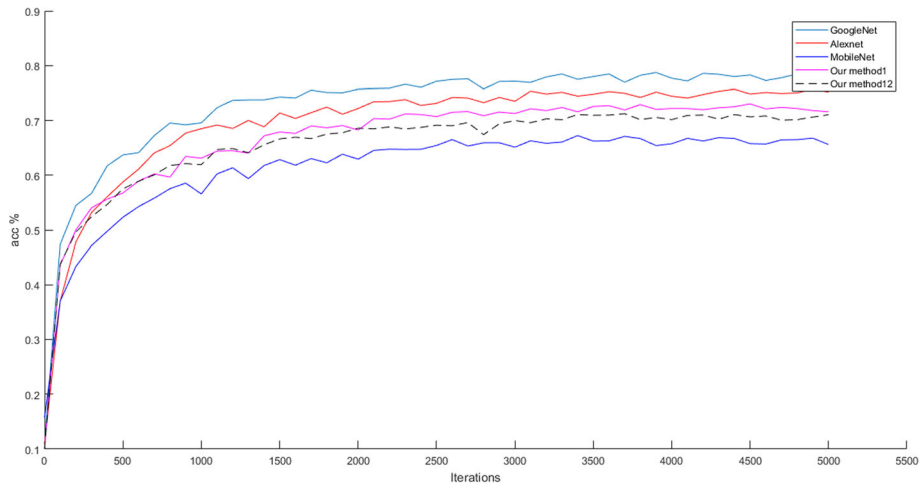

Fig. 6 The comparison of CIFAR-10 accuracy

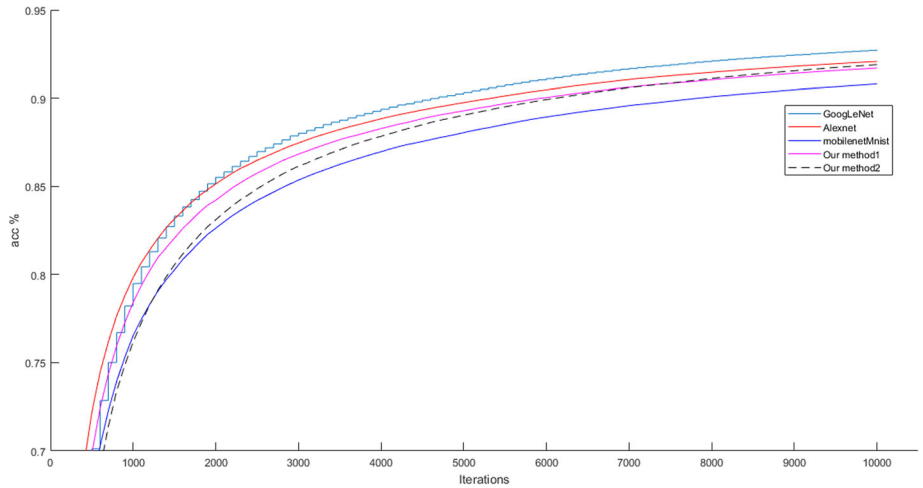

Fig. 7 The comparison of SVHN accuracy

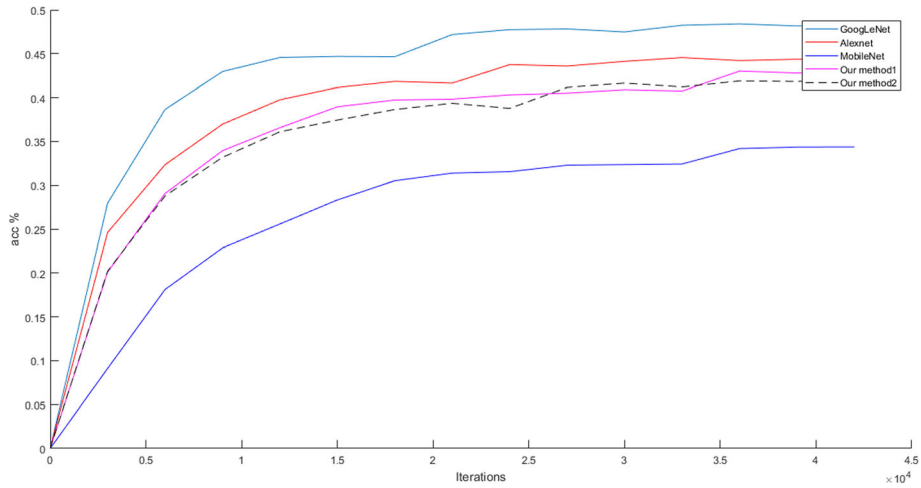

Fig. 8 The comparison of Tiny ImageNet accuracy

MobileNet. Training parameters of our methods are less than GoogleNet and AlexNet. On the Cifar-10, although AlexNet and GoogleNet have a slightly higher test accuracy, they require more training parameters. And the accuracy of our methods is very close to them. At the beginning of the training, our methods converge faster than others. On SVHN datasets, we use the same layers as that on CIFAR-10. The accuracy of our methods, AlexNet, and GoogleNet has the same trend. In our methods, the second method is slightly better than the first method. On Tiny ImageNet, this is a relatively complex dataset. Although the accuracy is not high, we can still make a relative comparison. The accuracy of GoogleNet is best. AlexNet is the second. Our methods are close to AlexNet while far superior to MobileNet.

6 Conclusions and future work

In this paper, we propose a multi-scale separable convolutional neural network, which overcomes the problem of multiple parameters and large computational resource. By comparing the accuracy and training parameters, the experimental results show that our methods are robust for improving the network accuracy and retaining less parameter. In the next work, we will further develop it to deeper networks and study its performance on larger datasets.

Acknowledgements

This work is supported by the Science and Technology Foundation of Hubei Provincial Department of Education under Grant No. Q20161405 and the National Nature Science Fund of China under Grant No. 61601176.

Availability of data and materials

The MNIST data, http://yann.lecun.com/exdb/mnist/
The CIFAR-10 data, http://www.cs.toronto.edu/~kriz/cifar.html
The SVHN data, http://ufldl.stanford.edu/housenumbers/
The Tiny ImageNet data, http://tiny-imagenet.herokuapp.com

Authors' contributions

GW provided the design methods and ideas. GY carried out the experiments, analysed the experimental results, and prepared this manuscript. ML collected relevant data. TL supervised the work and revised the paper. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Author details

[1]Hubei Collaborative Innovation Centre for High-efficiency Utilization of Solar Energy, Hubei University of Technology, Wuhan 430068, China. [2]School of Electrical and Electronic Engineering, Hubei University of Technology, Wuhan 430068, China.

Received: 20 March 2018 Accepted: 20 July 2018
Published online: 31 July 2018

References

1. Fukushima K (1980) A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 36:193–202.
2. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324.
3. Litjens G, Kooi T, Bejnordi BE, Setio AAA (2017) A survey on deep learning in medical image analysis. Med Image Anal 42:60–88.
4. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436.
5. Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1106–1114.
6. Srivastava N, Hinton G, Krizhevsky A (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958.
7. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. Computer Vision and Pattern Recognition 6. https://arxiv.org/abs/1409.1556v6.
8. Ren L, Lu J, Feng J, Zhou J (2017) Multi-modal uniform deep learning for RGB-D person re-identification. Pattern Recogn 72:446–457.
9. Liao Z, Carneiro G (2017) A deep convolutional neural network module that promotes competition of multiple-size filters. Pattern Recogn 71:94–105.
10. Afridi MJ, Ross A, Shapiro EM (2017) On automated source selection for transfer learning in convolutional neural networks. Pattern Recogn 000:1–11.
11. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition, pp 770–778.

12. Szegedy C, Liu W, Jia Y (2014) Going deeper with convolutions. arXiv 1409: 4842v1 1.
13. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. arXiv 1611:05431v2 2.
14. Ma Z, Chang X, Yang Y, Sebe N (2017) The many shades of negativity. IEEE Trans Multimedia 19:1558–1568.
15. Yang Y, Ma Z, Hauptmann GA (2013) Feature selection for multimedia analysis by sharing information among multiple tasks. IEEE Trans Multimedia 15:661–669.
16. Chollet F (2017) Xception: deep learning with depthwise separable convolutions. arXiv 1610:02357v3 3.
17. Wang M, Liu B, Foroosh H (2016) Factorized convolutional neural networks. arXiv 1608:04337v1 1.
18. Howard AG, Zhu M, Dmitry BC, Kalenichenko D (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv 1704: 04861v1 1.
19. Zhang X, Zhou X, Lin M, Sun J (2017) ShuffleNet: an extremely efficient convolutional neural network for mobile devices. arXiv 1707:01083v1 1.