IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 18, 2020, accepted October 13, 2020, date of publication October 19, 2020, date of current version February 17, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3031908

Improving the Performance of VGG Through Different Granularity Feature Combinations

YUEPENG ZHOU[1], HUIYOU CHANG[1], YONGHE LU[2], XILI LU[3], AND RUQI ZHOU[1]
[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China
[2]School of Information Management, Sun Yat-sen University, Guangzhou 510006, China
[3]School of Information and Engineering, Shaoguan University, Shaoguan 512005, China

Corresponding authors: Huiyou Chang (changhuiyou@qq.com) and Yonghe Lu (luyonghe@mail.sysu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 71373291, in part by the Basic and Applied Basic Research Fund of Guangdong Province under Grant 2019B1515120085, in part by the Science and Technology Planning Project of Guangdong Province under Grant 2016B030303003, in part by the Scientific Research Project of Guangdong Education Department under Grant 2016KQNCX153, and in part by the Special projects in key areas of universities in Guangdong Province under Grant 2020ZDZX1023.

ABSTRACT Convolutional neural networks have achieved amazing success in many areas in recent years, and VGG is a widely used convolutional neural network model. However, it has some limitations, such as a large number of parameters, which take up significant memory and thus restrict its application in resource-constrained scenarios such as mobile devices and embedded systems. In convolutional neural network models, the different number of convolutional layers can extract different granularity features, which represent the different levels of importance in the image recognition process. Here, we propose a new VGG architecture with different granularity feature combinations that combine different granularity features from block1, block2, block3, block4, and block5 in VGG. Each block is followed by a local fully connected layer to reduce the dimensionality of the coarse and fine features, and five different granularity features are combined as the input of the first global fully connected layer. By combining the features of different blocks, it can increase information flow from a lower layer directly to a fully connected layer and increase feature reuse without adding too many connections. The addition of five local fully connected layers means an increase in parameters, so we reduce the neuron number in two global fully connected layers to reduce the number of parameters. The well-known datasets CIFAR-10 and MNIST were used to evaluate the network's classification performance. The experimental results show that the proposed model achieves better training and testing performance than traditional VGGs and reduces the number of parameters.

INDEX TERMS Combination, convergence speed, different granularity, feature, performance.

I. INTRODUCTION

In recent years, convolutional neural networks (CNNs) have achieved great success in image classification tasks through supervised learning. CNNs are effective for learning better feature representations in the field of computer vision [1]–[7]. We have witnessed a tremendous improvement in image classification tasks due to the use of supervised learning combined with the powerful model of CNNs [8].

CNNs have become an important tool for machine learning and many related fields. The superior performance of deep CNNs usually comes from their deeper and wider architectures [3]; however, the networks have suffered from an increasing memory burden, which has become a bottleneck for many applications [9].

Several works have shown that significant information of the input images is lost with depth in the successful ImageNet classification CNNs [10], and the deeper structure does not always guarantee a better result, as validated by the experiments of He and Sun [11].

Visual recognition requires rich representations that span levels from low to high and scales from small to large, with resolutions from fine to coarse; the skip connections from the shallower to deeper stages can merge scales and resolutions [12], and the different scales and resolutions can better express the features.

VGG is a widely used CNN model proposed by Karen Simonyan and Andrew Zisserman. VGG is the abbreviation

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

of the Oxford Visual Geometry Group. The VGG16 and VGG19 [8] models are stacks of convolutional layers, followed by three fully connected (FC) layers; specifically, the first two have 4096 neurons each, and the third performs a 10-way CIFAR-10 [13] and MNIST [14] classification. The final layer is the softmax layer. The configuration of the FC layers is the same in all networks [8]. Table 1 illustrates the architecture and parameters of models VGG16 and VGG19.

TABLE 1. The VGG Model [2].

	VGG16	VGG19
block1	conv3-64	conv3-64
	conv3-64	conv3-64
	maxpool	
block2	conv3-128	conv3-128
	conv3-128	conv3-128
	maxpool	
block3	conv3-256	conv3-256
	conv3-256	conv3-256
	conv3-256	conv3-256
		conv3-256
	maxpool	
block4	conv3-512	conv3-512
	conv3-512	conv3-512
	conv3-512	conv3-512
		conv3-512
	maxpool	
block5	conv3-512	conv3-512
	conv3-512	conv3-512
	conv3-512	conv3-512
		conv3-512
	maxpool	
	FC-4096	
	FC-4096	
	FC-1000	
	softmax	

In Table 1, the ''3'' after ''conv'' represents the filter size, and the number after ''conv3-'' represents the number of filters. We use VGG16 and VGG19 as our baselines, with stack 3×3 convolutions and 2×2 max-pooling operations to build deeper networks.

Some excellent studies [15]–[18] have shown that there is considerable redundancy in a traditional VGG model, and layer-by-layer connections make the network replicate features of the front layer in the whole network. Hosny *et al.* [19], [20] performed several experiments using different models, such as Alex-net, ResNet, VGG, and GoogleNet. Their results showed that the VGG model required substantial memory and high-configuration hardware. Hammad and El-Sankary [21] improved the performance of VGG in terms of power, area, and speed by replacing the exact multipliers with approximate multipliers. In VGG, the FC layers with weight matrices of sizes 25088×4096, 4096×4096 and 4096×1000 [22] are shown in Table 1. This suggests that the large models may be redundantly designed for the task; VGG16 has more than 130M parameters [23].

ResNet is time consuming in the training process, requiring 8 GPUs to train for 2 to 3 weeks, which restricts its application in resource-constrained scenarios such as mobile devices and embedded systems [23]. DenseNet connects each layer to every other layer in a feedforward fashion, so it has L(L+1)/2 connections, whereas traditional networks have only L connections [24]. Huang *et al.* [25] proposed a CondenseNet model to prune the redundant connections between layers [25]; compared to DenseNet, CondenseNet used only 1/10 of the computation at comparable accuracy levels.

Filter pruning enables a model with a structured sparsity and more efficient memory usage than weight pruning and thus achieves a more realistic acceleration [3]. Emily Denton's experiments showed that deep neural networks are heavily overparametrized [26], which leads to high computational cost and a high memory footprint. Liu *et al.* [27] used pruning algorithms to reduce the parameters to 30%-60% of the traditional model while maintaining or even improving the accuracy. Therefore, filter pruning is highly advocated for accelerating networks. Many works adopted filter pruning [28]–[32] in their models.

These approaches are effective because deep networks often have a substantial number of redundant weights that can be pruned or quantized without sacrificing (and sometimes even improving) accuracy. For convolutional networks, different pruning techniques may lead to different levels of granularity [33].

In this article, we present a different granularity combination VGG (DGC-VGG) architecture that takes the widely used VGG model [8] as an example. We use two benchmark datasets, CIFAR-10 and MNIST, to investigate the effects of the loss value, accuracy, convergence speed, and the number of parameters.

The contributions in this work can be summarized as follows:

1) By combining the features of different blocks, the model can increase information flow from a lower layer directly to an FC layer and increase feature reuse without adding too many connections. The accuracy can be improved, and the loss value and the number of parameters can be reduced.

2) In the DGC-VGG model, as the number of neurons increases in the FC layer, all four indicators of the model are improved.

3) In the proposed model, no preprocessing is used. The DGC-VGGs exhibit faster convergence and better robustness, and they are more stable as the epoch numbers increase.

The remainder of this article is organized as follows. Section II presents the related works of CNNs. In Section III, we describe the experimental setup and provide details regarding the compared methods, datasets and models used

for evaluation. In Section IV, we present the results from the experiments and compare the methods. Finally, Section V concludes the paper.

II. RELATED WORKS

We review the recent progress in CNNs. The related methods are categorized and introduced as follows.

A. SKIP CONNECTION

Some neural network layers comprise a deep convolutional neural network [7]. Deeper layers can extract more semantic and global features, but these signs do not prove that the last layer is the ultimate representation for any task [12].

In fact, skip connections have proven to be effective for classification and regression [24]. He *et al*. [4] used a skip connection of identity mapping to improve convergence; the vanishing gradient problem was also greatly relieved by introducing the skip connections into the network. To deepen the networks, skip connections or shortcut paths have been widely adopted to ease network training [34]. In [12], it is shown that a skip connection structure is an efficient way to make the top layers accessible to the information from the bottom layers, as successfully adopted in ResNet [4] and Highway Network [35].

Skip connections can alleviate the vanishing gradient problem and strengthen feature propagation while encouraging feature reuse and reducing the number of parameters [24].

B. FEATURE COMBINATION

Additionally, feature combination is a critical dimension of the architecture to further increase the information flow from a lower layer directly to a higher layer. For the operation of a join layer, some choices seem reasonable, including concatenation and addition [24].

Concatenating feature maps, which are learned from different layers, can increase variation in the input of subsequent layers and improve efficiency. Huang *et al*. [24] replaced the identity mapping in the residual block by concatenating operations so that new feature learning can be reinforced and old feature reuse can be maintained; they never combine features through summation before they are passed into a layer. DenseNet concatenates several layers densely; it can reduce parameters while maintaining better accuracy.

Larsson *et al*. [36] instantiated each joint to compute the elementwise mean of its inputs, and ResNet adopts a residual sum operation in each stage, which has proven to be a simple and efficient way to build a deeper neural network [24].

C. DIFFERENT GRANULARITY

The different numbers of convolutional layers can extract the different granularity features. Li *et al*. [37] used two branches for local and global feature extraction in their SMGN model, the SMGN extracts multiple granularity features; these features are concatenated together as multiple granularity features of person images to enhance their discriminating ability. Du *et al*. [38] proposed a progressive training strategy that effectively fuses features from different granularities for fine-grained visual classification. Sun *et al*. [39] conducted multigranularity emotional block partitioning on network texts and performed sentiment analysis comparisons with different granularities.

III. DGC-VGG

Inspired by the above research work, we add skip connection to our model and use connection and addition methods to combine the different granularity features.

In our model, different granularity features, obtained by different blocks, are transferred into a one-dimensional character by an FC operation. After five convolutional blocks, all features are combined by using concatenation and summation operations. We reduce the neuron number in the FC layers to prune the redundant neurons. The DGC-VGG model is shown in Figure 1.

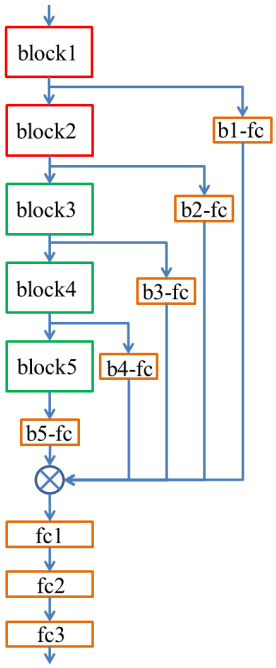

FIGURE 1. The DGC-VGG model.

In DGC-VGG, the different numbers of convolutional blocks can extract the different granularity features. DGC-VGG combines the different granularity features of different convolution blocks and contains shorter connections between the lower block and the first global FC layer, which can better represent the original image. We modify the neuron number in FC layers to improve the performance and reduce the parameters of the model.

We take the widely used VGG16 and VGG19 models as the baseline, both of which have five blocks. The first two blocks contain two layers, and the last three blocks contain three layers in VGG16 and four layers in VGG19.

Each layer contains three consecutive operations: a 3×3 convolution, batch normalization (BN) [40], and a rectified linear unit (ReLU) [41] activation function. Clevert *et al*. [42]

found that BN can improve ReLU networks. ReLU keeps positive inputs unchanged and outputs zero for the negative inputs; therefore, it can avoid the vanishing gradient problem, enabling the training of much deeper supervised neural networks [43]. Currently, ReLU is the most widely used activation function. Although there are various alternatives to ReLU, none have managed to replace it due to inconsistent gains [44]. Each block contains one 2×2 max-pooling layer at the end of the last layer:

$$f(x_1, \ldots, x_n) = max\left(0, \sigma(BN(\sum \omega_i * x_i + b))\right) \quad (1)$$

where σ is the ReLU activation, BN is batch normalization, and ω_i and b are the weights and bias in the convolution, respectively.

After each block, a local FC layer is used to extract the different granularity features and reduce the dimensionality of the outputs, and then the five outputs are combined into a single tensor as the input of the first global FC layer. The details are shown in Figure 1.

In Figure 1, block1 and block2 have the same number (two) of convolution layers and one max-pooling layer; we mark the two blocks in red. Block3, block4 and block5 have the same number (three or four) of convolution layers and one max-pooling layer; these blocks are marked in green. All FC layers (local and global FC layers) are marked in brown. The symbol $\otimes$ represents the combination of different granularity features. We use two combined methods in DGC-VGG: concatenation and summation.

$$O = Concat(o_{b1}, o_{b2}, o_{b3}, o_{b4}, o_{b5}) \quad (2)$$

Concat denotes the concatenation operator, and o_{b1}, o_{b2}, o_{b3}, o_{b4}, and o_{b5} refer to the feature maps produced by all blocks:

$$O = Sum(o_{b1}, o_{b2}, o_{b3}, o_{b4}, o_{b5})/5 \quad (3)$$

where Sum denotes the summation. The outputs after five local FC layers are averaged as the input of the first global FC layer.

The additional FC layer not only saves the original features but also increases the number of parameters in the training and testing processes. We try to reduce the number of neurons to reduce the parameters. In VGG16 and VGG19, the first two FC layers have 4096 neurons each; we set the number of neurons to 1024, 512, 256, 128, and 64, which will improve the performance without increasing the training and testing time.

IV. EXPERIMENTS

A Lenovo computer equipped with a dual-core Intel Core i7 processor and 96 GB of DDRAM was used to perform the experiments. We implemented the VGGs and DGC-VGGs using the TensorFlow library [45].

A. DATASETS

In this section, we present the experimental results for two benchmark datasets, MNIST [14] and CIFAR-10 [13]. MNIST is considered a simple and solved problem that involves digit image classification [46], which consists of 60,000 training images and 10,000 testing images of handwritten numbers with a size of 28×28. There are 10 classes corresponding to digits from 0 to 9. CIFAR-10 is a significantly more complex image classification problem than MNIST; it consists of 50,000 training images and 10,000 test images with 10 classes of natural objects.

B. PARAMETER SETTINGS ON MNIST AND CIFAR-10

In each block, 3×3 convolutional kernels with stride 1 are used in convolutional layers to analyze images, and 1-padding is adopted to ensure that the scale of images does not change after convolution. The filter number of each block is set as 64, 128, 256, 512 and 512. Max-pooling layers are utilized in the encoder to $2\times$ downsample the images. In the max-pooling layers, 2×2 filters with stride 2 are used, and the number of filters is set as that of the convolutional layer. Each block is followed by a local FC layer, which contains the same number of neurons as that of the first two FC layers. The neuron numbers are set as 64, 128, 256, 512, and 1024.

We train for 20 epochs on MNIST, use the Adam optimization strategy with a learning rate of 0.0001 and use he_normal initialization for the parameters. The batch size is 128.

The structure and parameters used in CIFAR-10 are similar to those used for MNIST. We train CIFAR-10 for 164 epochs, and the batch size is 250. We adopt a segmented learning rate; if the epoch number is less than 81, then the learning rate is 0.1; if the epoch number is greater than 121, then the learning rate is 0.001; in other cases, the learning rate is 0.01.

C. RESULTS AND DISCUSSION ON CIFAR-10

We evaluate the DGC-VGG networks on the traditional VGG16 and VGG19 models by concatenation and summation operations, respectively. All models are trained from scratch, and all other hyperparameters are kept fixed.

We change the neuron number in the FC layer of DGC-VGG16 with two combination methods and evaluate the model on CIFAR-10 with four measure indexes (training loss, training accuracy, testing loss, testing accuracy). The experimental results of DGC-VGG16 (Figure 2) with the concatenation (DGC-VGG16 (Concat)) operation on CIFAR10 are presented in the appendix. The number after Concat in Figure 2 represents the neuron number in b1-fc, b2-fc, b3-fc, b4-fc, b5-fc, fc1, and fc2 in Figure 1.

We modify the combination method in Figure 1 to combine the feature by the summing operation. After each FC layer that follows each block, the granularity of features is from coarse to fine, and we adopt a summation method to combine the different granularity features and then calculate their average. The experimental results of DGC-VGG16 with the summation operation (DGC-VGG16(Sum)) are shown in Figure 3 in the appendix. The number after Sum in Figure 3 also represents the neuron number in seven FC layers.

Figures 2 and 3 show that as the number of neurons increases in the FC layer, all four indicators of the model

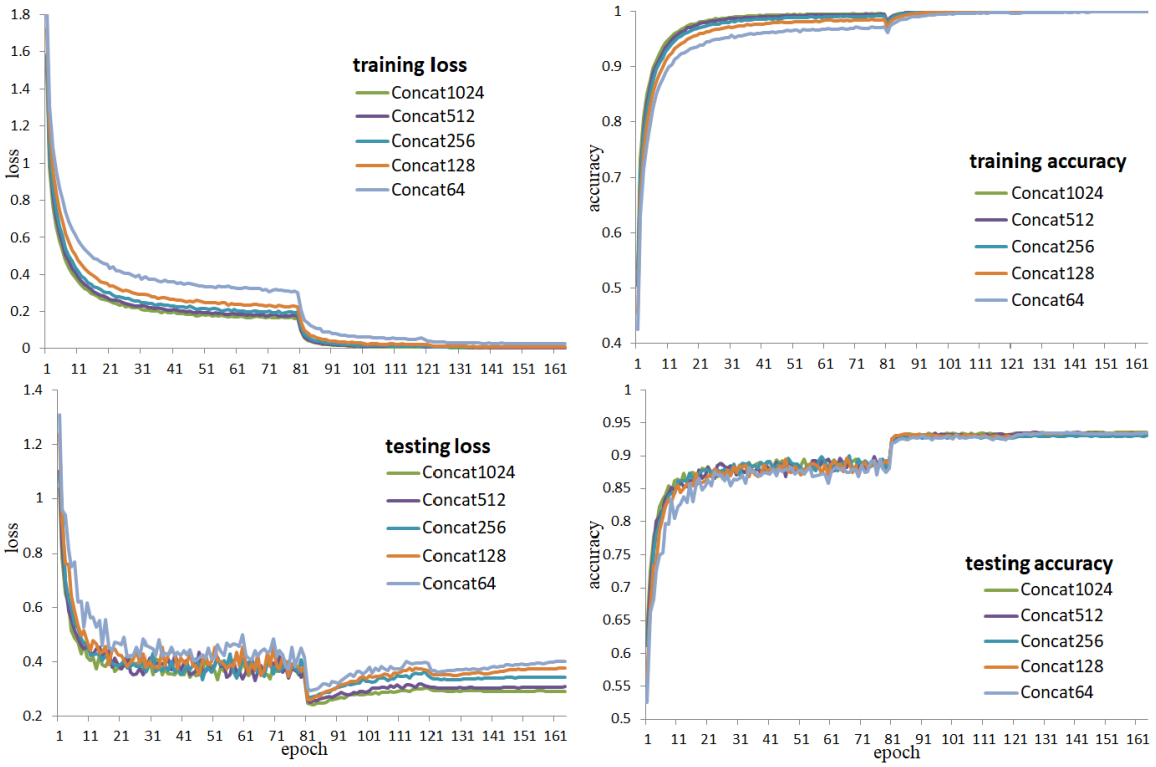

FIGURE 2. Results of DGC-VGG16 (Concat) on CIFAR-10.

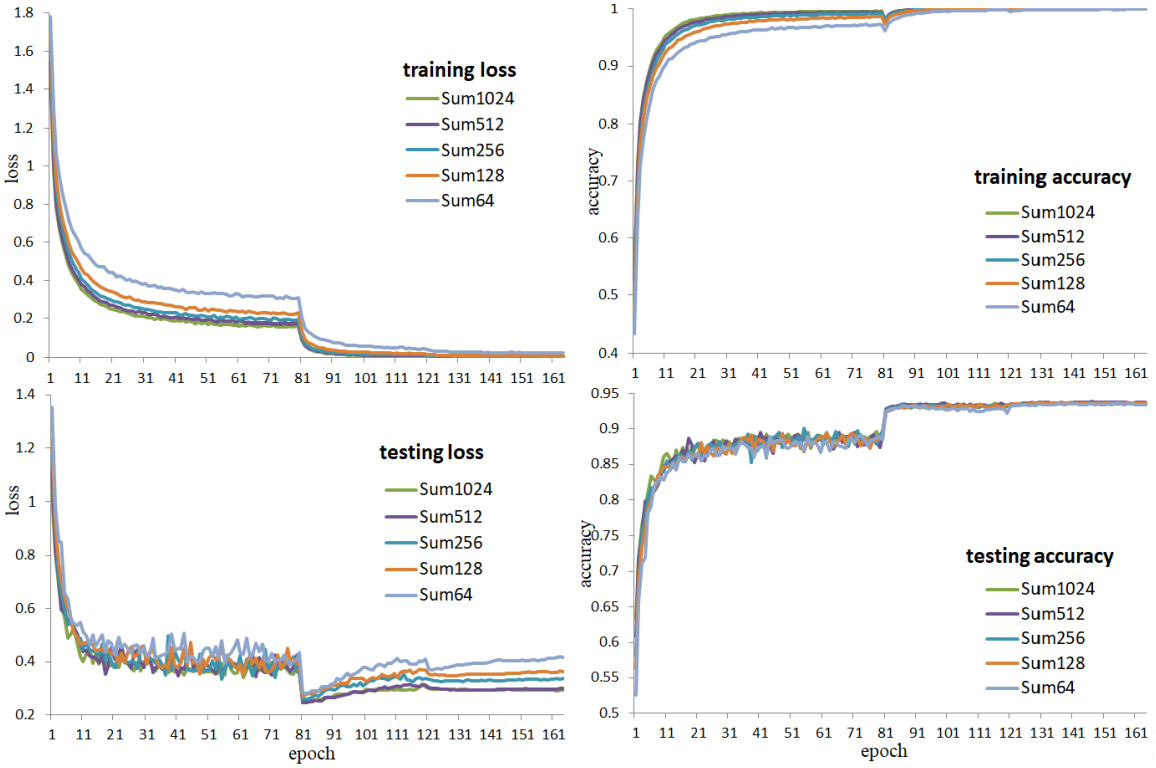

FIGURE 3. Results of DGC-VGG16(Sum) on CIFAR-10.

are better. The experimental results of VGG16, DGC-VGG16 (Concat), and DGC-VGG16(Sum) are shown in Figure 4.

Figure 4 reveals that in the training stage, the loss value and accuracy of the three models are very close (the convergence speed of DGC-VGG16s is faster), but in the testing

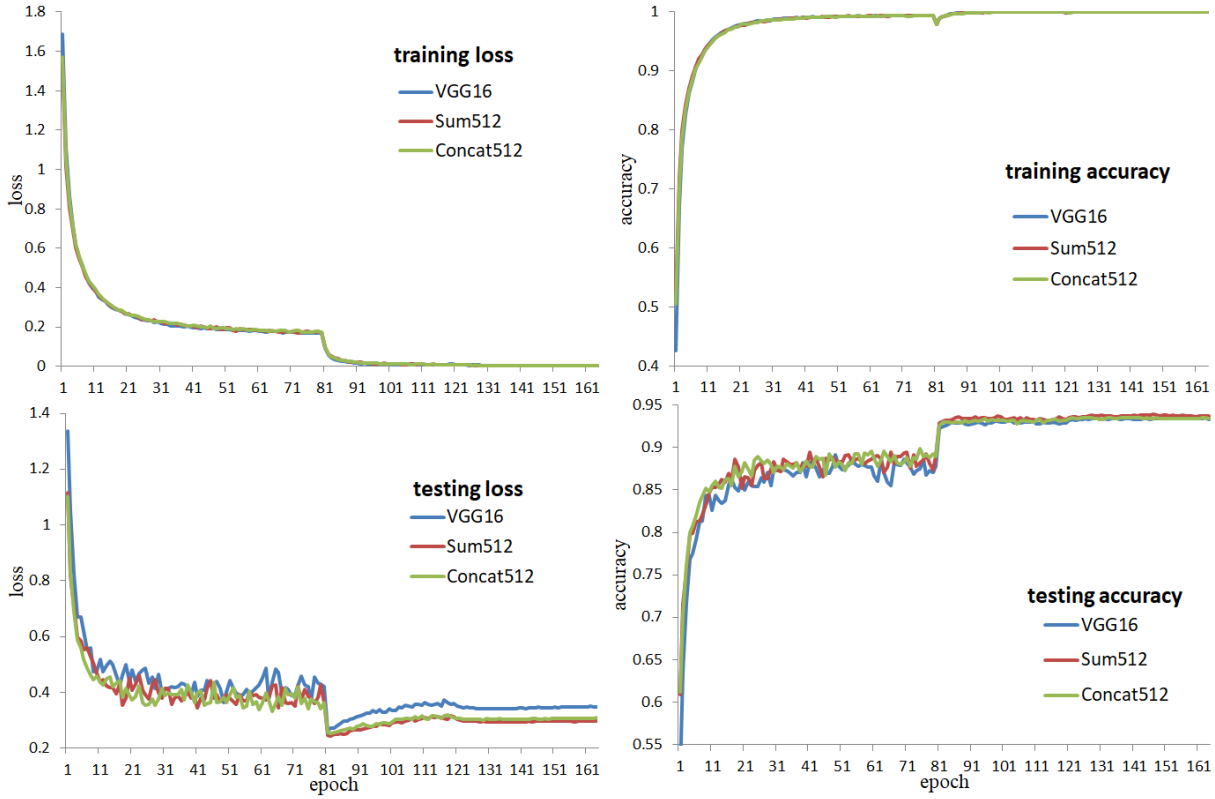

FIGURE 4. Results of VGG16 and DGC-VGG16 on CIFAR-10.

TABLE 2. Parameters of VGG16 and DGC-VGG16S on CIFAR-10.

Model	VGG16	Concat512	Sum512
Parameters (M)	38.08	30.79	29.79

stage, the two measure indexes of DGC-VGG16 have great advantages. According to Table 1 and Figure 1 and formula 4, the parameters of DGC-VGG16s are fewer than those of VGG16, as shown in Table 2:

$$P = fn_p * C^2 * fn_n + fn_n \tag{4}$$

where P denotes the parameters; fn_p represents the filter number of the previous layer;C is the filter size, and we use 3×3 receptive fields throughout the whole net, so C is 3; and fn_n represents the filter number of the next layer.

DGC-VGG16(Sum) and DGC-VGG16(Concat) improve the average test accuracy by 0.92% and 0.89% and reduce the average loss by 11.51% and 11.77%, respectively, on Cifar10 compared to VGG16, as shown in Table 3.

We perform the same experiments with DGC-VGG19 on CIFAR-10, and the experimental results of DGC-VGG19 and DGC-VGG16 are similar; as the number of neurons increases in the FC layer, the four indicators of the model DGC-VGG19 are better. The experimental results of DGC-VGG19 (Concat) are shown in Figure 5, and the results of DGC-VGG19 (Sum) are shown in Figure 6 in the appendix.

From Figures 5 and 6, we can obtain the same conclusion as that reached from Figures 2 and 3; that is, with an increasing neuron number, each measure index of the model DGC-VGG19 is better. Therefore, we choose DGC-VGG with 512 neurons in the FC layer to evaluate DGC-VGG16, DGC-VGG19, VGG16, and VGG19 on CIFAR-10. The experimental results of VGG19, DGC-VGG19 (Concat), and DGC-VGG19 (Sum) are shown in Figure 7.

DGC-VGG19(Sum) and DGC-VGG19(Concat) improve the average test accuracy by 2.1% and 1.9% and reduce the average loss by 17.26% and 16.93%, respectively, on Cifar10 compared to VGG19, as shown in Table 3.

The experimental results between VGG19 and DGC-VGG19s and VGG16 and DGC-VGG16s are slightly different. In the training and testing stages, all four measure indexes of DGC-VGG19s are better than those of VGG19, and the parameters of DGC-VGG19s are less than those of VGG19, as shown in Figure 7 and Table 4.

From Figures 4 and 7 and Table 2 and Table 4, we can conclude that the DGC-VGGs are better than the traditional VGG model on the CIFAR10 dataset, with respect to not only the four measure indexes in training and testing but also the number of parameters and convergence speed.

D. RESULTS AND DISCUSSION ON MNIST

We use the DGC-VGGs to repeat the experiments on MNIST by modifying several parameters: the training epoch, learning rate, batch size, and so on, as mentioned in Section IV.B.

TABLE 3. The Performance and Parameters of DGC-VGG Compared with VGG. the Symbol ↓ Indicates Reduction, and ↑ Indicates improvement.

ID	Dataset	Network	Parameters	Testing loss	Testing accuracy
1	CIFAR10	DGC-VGG16(Sum)	21.77%↓	11.51%↓	0.92%↑
2	CIFAR10	DGC-VGG16(Concat)	19.14%↓	11.77%↓	0.89%↑
3	CIFAR10	DGC-VGG19(Sum)	19.21%↓	17.26%↓	2.1%↑
4	CIFAR10	DGC-VGG19(Concat)	16.89%↓	16.93%↓	1.9%↑
5	MNIST	DGC-VGG16(Sum)	21.77%↓	--	11.92%↑
6	MNIST	DGC-VGG16(Concat)	19.14%↓	--	12.03%↑
7	MNIST	DGC-VGG19(Sum)	19.21%↓	--	0.95%↑
8	MNIST	DGC-VGG19(Concat)	16.89%↓	--	0.84%↑

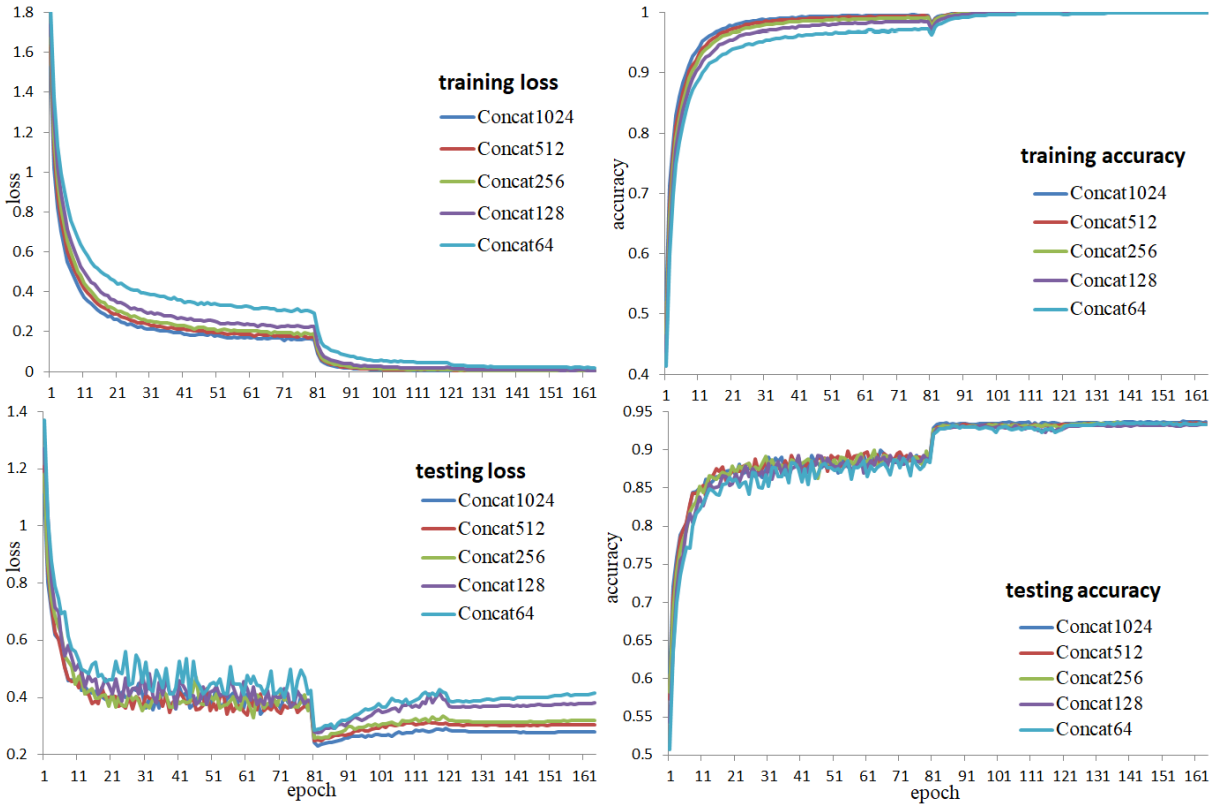

FIGURE 5. Results of DGC-VGG19 (Concat) on CIFAR-10.

TABLE 4. Parameters of VGG19 and DGC-VGG19s on CIFAR-10.

Model	VGG19	Concat512	Sum512
Parameters (M)	43.14	35.86	34.86

Because MNIST is considered a simple and solved problem that involves digit image classification, CIFAR-10 is a significantly more complex image classification problem than MNIST [46], and we evaluate the model on MNIST with three measure indexes (training loss, training accuracy, and testing accuracy). The results of DGC-VGG16(Concat) on MNIST are shown in Figure 8 in the appendix, and the number after Concat represents the neuron number in seven FC layers.

As seen in Figure 8, the difference between loss and accuracy during training is not obvious; the accuracy does not increase with increasing number of neurons, unlike those on CIFAR10. The accuracies of DGC-VGG16(Concat) with different neuron numbers in the FC layers are shown in Table 5 in the appendix.

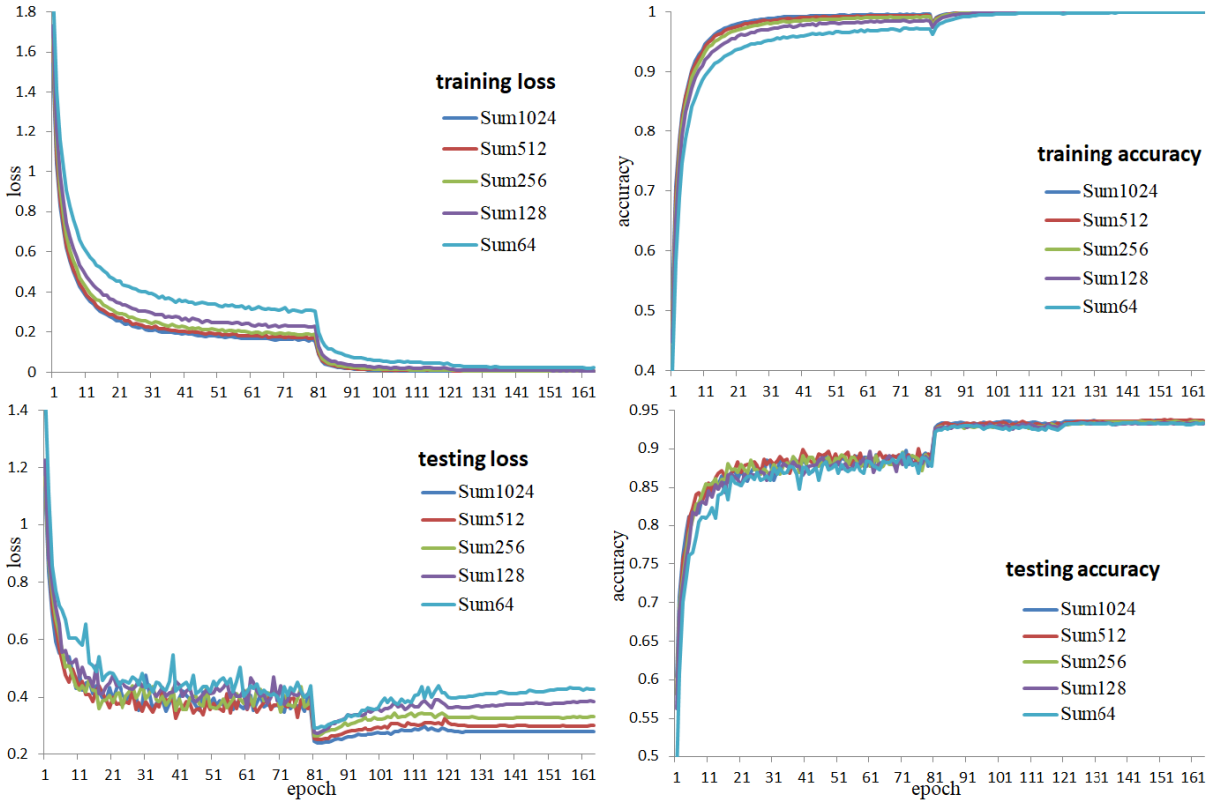

FIGURE 6. Results of DGC-VGG19(Sum) on CIFAR-10.

TABLE 5. The Testing Accuracy of DGC-VGG16(Concat) on MNIST.

Model	Concat1024	Concat512	Concat256	Concat128	Concat64
Accuracy	0.9895	0.9914	0.9903	0.9874	0.9906

TABLE 6. The Testing Accuracy of DGC-VGG16(Sum) on MNIST.

Model	Sum1024	Sum512	Sum256	Sum128	Sum64
Accuracy	0.9895	0.9914	0.9903	0.9874	0.9906

We modified the method to combine the features by a summing operation; the experimental results of DGC-VGG16(Sum) are shown in Figure 9 in the appendix. The accuracies of DGC-VGG16(Sum) with different neuron numbers in the FC layers are shown in Table 6 in the appendix.

DGC-VGG16(Sum) and DGC-VGG16(Concat) improve the average test accuracy by 11.92% and 12.03%, respectively, on MNIST compared to VGG16, as shown in Table 3.

When the model is changed on MNIST, the experimental results of DGC-VGG19s are the same as those of DGC-VGG16s. The training loss and accuracy are shown in Figure 10 in the appendix, and the accuracies of DGC-VGG19(Concat) are shown in Table 7 in the appendix. The experimental results of DGC-VGG19(Sum) are the same as

TABLE 7. The Testing Accuracy of DGC-VGG19(Concat) on MNIST.

Model	Concat1024	Concat512	Concat256	Concat128	Concat64
Accuracy	0.9931	0.9852	0.9917	0.9877	0.9927

TABLE 8. The Testing Accuracy of DGC-VGG19(Sum) on MNIST.

Model	Sum1024	Sum512	Sum256	Sum128	Sum64
Accuracy	0.9928	0.9863	0.9918	0.9935	0.9896

TABLE 9. The Testing Accuracy and Parameters of VGG16 and DGC-VGG16s on MNIST.

Model	VGG16	Concat512	Sum512
Accuracy	0.8858	0.9924	0.9914
Parameters (M)	38.08	30.79	29.79

those of DGC-VGG19(Concat), as shown in Figure 11 and Table 8 in the appendix.

DGC-VGG19(Sum) and DGC-VGG19(Concat) improve the average test accuracy by 0.95% and 0.84%, respectively, on MNIST compared to VGG19, as shown in Table 3.

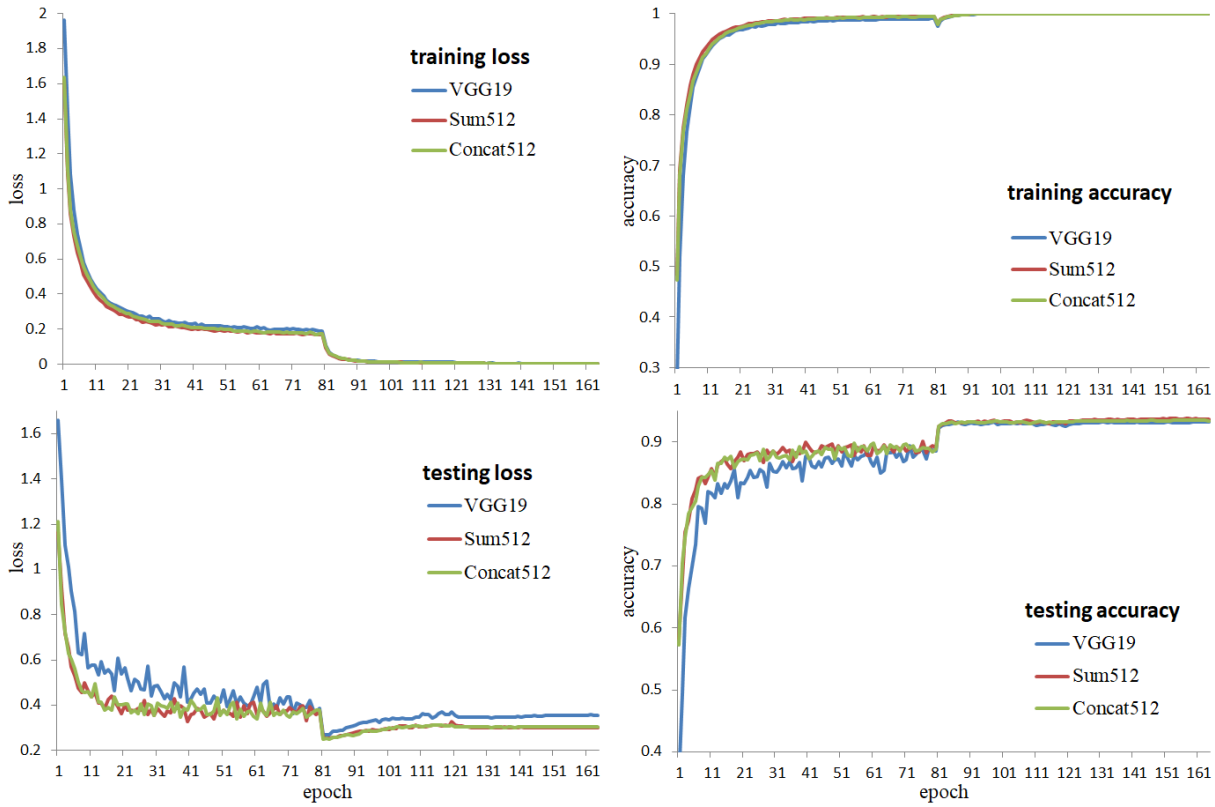

FIGURE 7. Results of VGG19 and DGC-VGG19s on CIFAR-10.

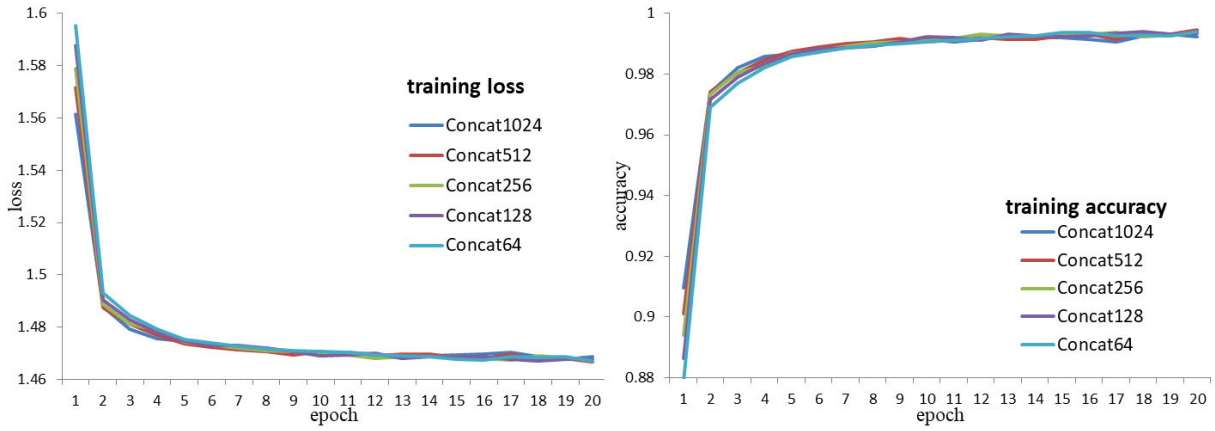

FIGURE 8. Training loss and accuracy of DGC-VGG16(Concat) on MNIST.

TABLE 10. The Testing Accuracy and Parameters of VGG19 and DGC-VGG19s on MNIST.

Model	VGG19	Concat512	Sum512
Accuracy	0.977	0.9852	0.9863
Parameters (M)	43.14	35.85	34.85

From Figures 10 and 11, we can obtain the same conclusion as that from Figures 8 and 9. With an increase in the neuron number, the difference between the loss and accuracy during training is not obvious after epoch 5. Therefore, we choose DGC-VGG with 512 neurons in the FC layer to evaluate DGC-VGG, VGG16, and VGG19 on MNIST. The experimental results of DGC-VGG16 with two methods of the feature combination and VGG16 are shown in Figure 12. The experimental results of DGC-VGG19 and VGG19 are shown in Figure 13. The accuracies and parameters of the three models are shown in Tables 9 and 10, respectively.

Figures 12 and 13 show that the performance of DGC-VGG is better than that of VGG, while DGC-VGGs

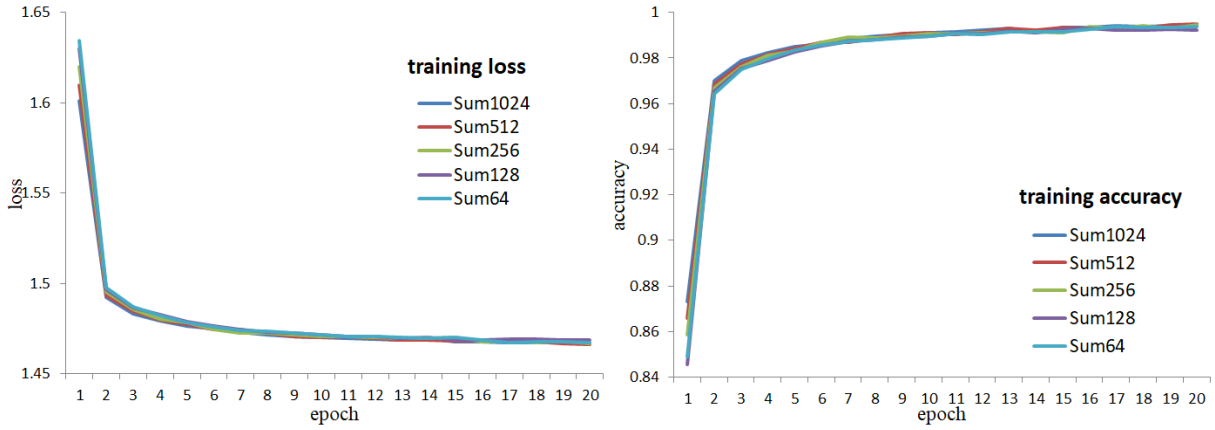

FIGURE 9. Training loss and accuracy of DGC-VGG16(Sum) on MNIST.

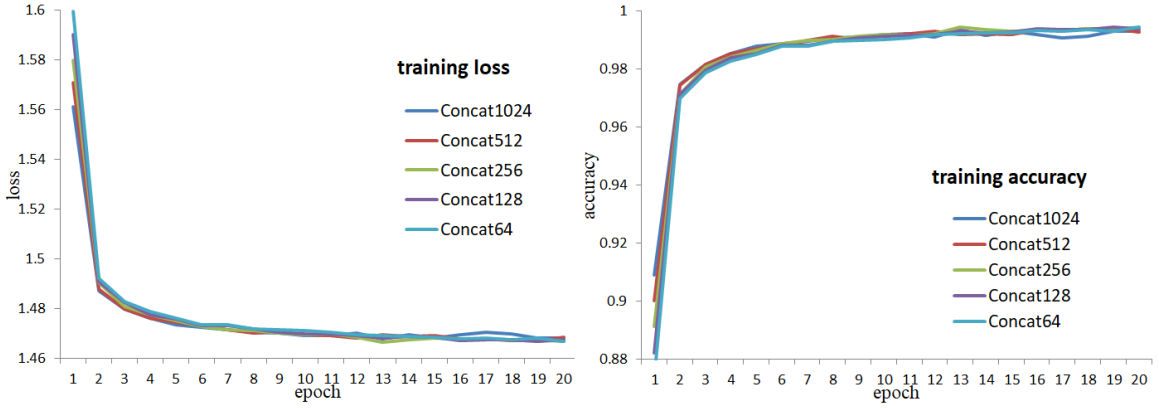

FIGURE 10. Training loss and accuracy of DGC-VGG19(Concat) on MNIST.

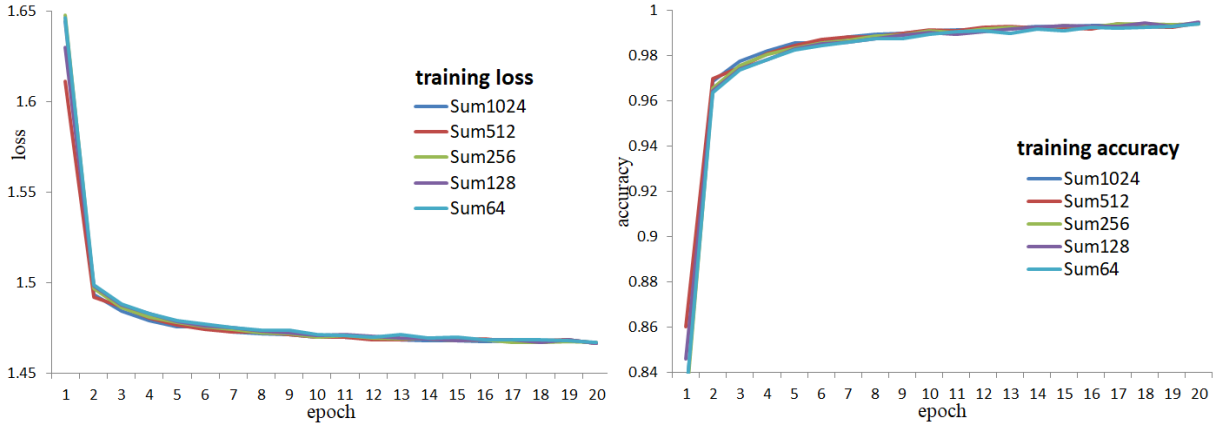

FIGURE 11. Training loss and accuracy of DGC-VGG19(Sum) on MNIST.

tend to be more stable than VGG with increasing epoch number. Different numbers of convolutional layers can extract different granularities of the features. DGC-VGGs combine the features from shallower and deeper convolution blocks, and they contain shorter connections from shallower convolutional block to the FC layer, which can increase information flow from a lower layer directly to a higher layer and can increase feature reuse without adding too many connections, so the accuracies are better.

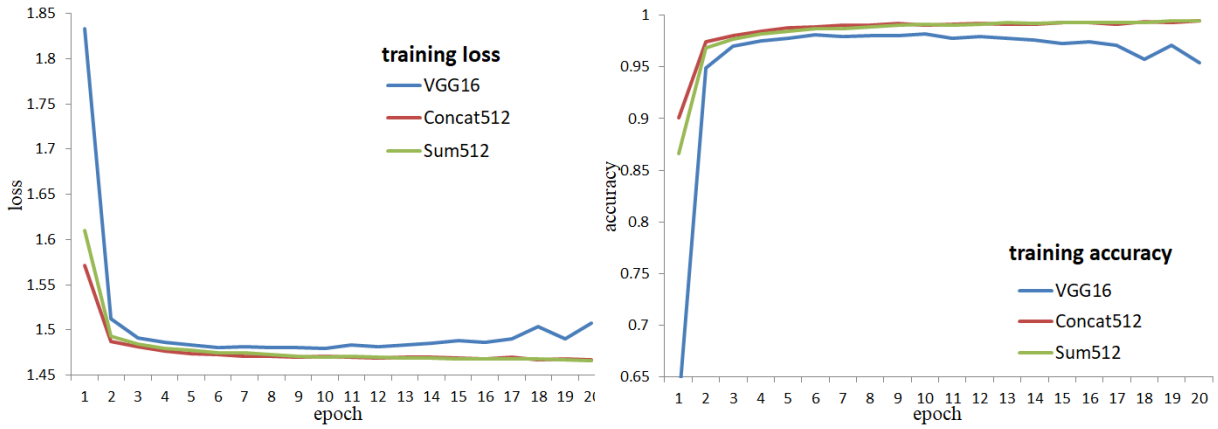

FIGURE 12. Results of VGG16 and DGC-VGG16s on MNIST.

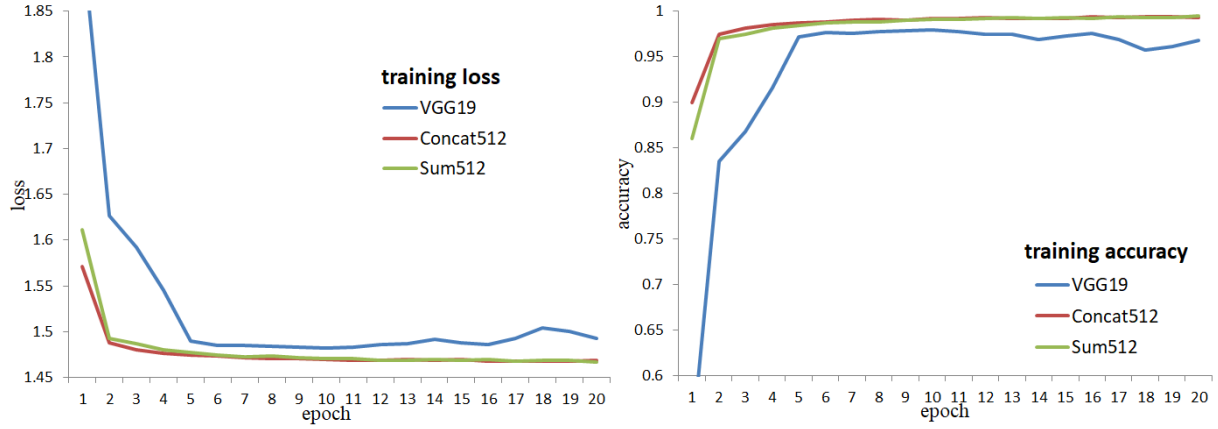

FIGURE 13. Results of VGG19 and DGC-VGG19s on MNIST.

V. CONCLUSION

In this study, we propose a new convolutional neural network architecture in which we add five local FC layers after each block in VGG and combine the features that are output from five local FC layers as the input of the first global FC layer. Different numbers of convolutional layers could extract different granularities of features, and DGC-VGG combines the different granularity features of different convolution layers, which could better represent the original image. Experiments on the image classification task demonstrate that DGC-VGG achieves higher accuracy, lower loss value, faster convergence speed, and fewer parameters. DGC-VGG improves the average test accuracy by 0.89% to 2.1% and reduces average test loss by 6.03% to 17.03%, respectively, on Cifar10, and improves the average test accuracy by 0.84% to 0.95% on MNIST.

APPENDIX

See Figs. 2, 3, 5, 6, 8–11, and Tables 5–8.

ACKNOWLEDGMENT

The authors gratefully acknowledge the Editor, Associate Editor and the anonymous referees for their invaluable suggestions and comments.

REFERENCES

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

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 1–9.

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

[5] H. Zhou, W. Ouyang, J. Cheng, X. Wang, and H. Li, "Deep continuous conditional random fields with asymmetric inter-object constraints for online multi-object tracking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 4, pp. 1011–1022, Apr. 2019, doi: 10.1109/TCSVT.2018.2825679.

[6] S. Sun, Z. Kuang, L. Sheng, W. Ouyang, and W. Zhang, "Optical flow guided feature: A fast and robust motion representation for video action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1390–1399.

[7] K. M. Hosny, M. A. Kassem, and M. M. Foaud, "Skin cancer classification using deep learning and transfer learning," in *Proc. 9th Cairo Int. Biomed. Eng. Conf. (CIBEC)*, Cairo, Egypt, Dec. 2018, pp. 90–93.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[9] A. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2214–2224.
[10] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 4829–4837.
[11] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Boston, MA, USA, Jun. 2015, pp. 5353–5360.
[12] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2403–2412.
[13] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.
[15] B. O. Ayinde, T. Inanc, and J. M. Zurada, "Redundant feature pruning for accelerated inference in deep neural networks," *Neural Netw.*, vol. 118, pp. 148–158, Oct. 2019, doi: 10.1016/j.neunet.2019.04.021.
[16] B. O. Ayinde and J. M. Zurada, "Building efficient convnets using redundant feature pruning," in *Proc. ICLR Workshop Submission*, 2018, pp. 1–9.
[17] J. Zou, T. Rui, Y. Zhou, C. Yang, and S. Zhang, "Convolutional neural network simplification via feature map pruning," *Comput. Electr. Eng.*, vol. 70, pp. 950–958, Aug. 2018, doi: 10.1016/j.compeleceng.2018.01.036.
[18] H. Zhou, J. M. Alvarez, and F. Porikli, "Less is more: Towards compact CNNs," in *Computer Vision—ECCV*. Cham, Switzerland: Springer, 2016, pp. 662–677.
[19] K. M. Hosny, M. A. Kassem, and M. M. Foaud, "Classification of skin lesions using transfer learning and augmentation with alex-net," *PLoS ONE*, vol. 14, no. 5, May 2019, Art. no. e0217293, doi: 10.1371/journal.pone.0217293.
[20] K. M. Hosny, M. A. Kassem, and M. M. Fouad, "Skin melanoma classification using deep convolutional neural networks," in *Deep Learning for Computer Vision: Theories and Applications*, M. Hassaballah and A. Awad, Eds. Boca Raton, FL, USA: CRC Press, 2020, pp. 1–27.
[21] I. Hammad and K. El-Sankary, "Impact of approximate multipliers on VGG deep learning network," *IEEE Access*, vol. 6, pp. 60438–60444, Oct. 2018, doi: 10.1109/ACCESS.2018.2875376.
[22] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 442–450.
[23] L. J. Dong, W. X. Yi, C. Yi, and Y. H. Peng, "Structure optimization of convolutional neural networks: A survey," *Acta Automatica Sinica*, vol. 46, no. 1, pp. 24–37, Jun. 2020.
[24] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 2261–2269.
[25] G. Huang, S. Liu, L. V. D. Maaten, and K. Q. Weinberger, "CondenseNet: An efficient DenseNet using learned group convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 2752–2761.
[26] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2014, pp. 1269–1277.
[27] Z. Liu, M. Sun, and T. Zhou, "Rethinking the value of network pruning," in *Proc. ICLR*, 2019, pp. 1–29.
[28] C. Yang, Z. Yang, A. M. Khattak, L. Yang, W. Zhang, W. Gao, and M. Wang, "Structured pruning of convolutional neural networks via l1 regularization," *IEEE Access*, vol. 7, pp. 106385–106394, Aug. 2019, doi: 10.1109/ACCESS.2019.2933032.
[29] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 1398–1406.
[30] L. Liang, L. Deng, Y. Zeng, X. Hu, Y. Ji, X. Ma, G. Li, and Y. Xie, "Crossbar-aware neural network pruning," *IEEE Access*, vol. 6, pp. 58324–58337, Oct. 2018, doi: 10.1109/ACCESS.2018.2874823.
[31] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, Jun. 2018, pp. 9194–9203.
[32] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montréal, QC, Canada, 2018, pp. 875–886.
[33] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," 2017, *arXiv:1705.08922*. [Online]. Available: http://arxiv.org/abs/1705.08922
[34] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, and Q. Liao, "LCSCNet: Linear compressing-based skip-connecting network for image super-resolution," *IEEE Trans. Image Process.*, vol. 29, pp. 1450–1464, Sep. 2020, doi: 10.1109/TIP.2019.2940679.
[35] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," 2015, *arXiv:1505.00387*. [Online]. Available: http://arxiv.org/abs/1505.00387
[36] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–11.
[37] D.-X. Li, G.-Y. Fei, and S.-W. Teng, "Learning large margin multiple granularity features with an improved siamese network for person re-identification," *Symmetry*, vol. 12, no. 1, p. 92, Jan. 2020, doi: 10.3390/sym12010092.
[38] R. Du, D. Chang, J. Xie, Z. Ma, Y.-Z. Song, J. Guo, and A. K. Bhunia, "Fine-grained visual classification via progressive multi-granularity training of jigsaw patches," 2020, *arXiv:2003.03836*. [Online]. Available: https://arxiv.org/abs/arXiv:2003.03836
[39] H. Sun, G. Wang, and S. Xia, "Text tendency analysis based on multi-granularity emotional chunks and integrated learning," *Neural Comput. Appl.*, Apr. 2020, doi: 10.1007/s00521-020-04901-y.
[40] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, Lille, France, 2015, pp. 448–456.
[41] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat. (AISTATS)*, Palermo, Italy, 2011, pp. 315–323.
[42] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*. [Online]. Available: http://arxiv.org/abs/1511.07289
[43] Y. Li, C. Fan, Y. Li, Q. Wu, and Y. Ming, "Improving deep neural network with multiple parametric exponential linear units," *Neurocomputing*, vol. 301, pp. 11–24, Aug. 2018, doi: 10.1016/j.neucom.2018.01.084.
[44] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," 2017, *arXiv:1710.05941*. [Online]. Available: http://arxiv.org/abs/1710.05941
[45] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and M. Kudlur, "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Conf. Oper. Syst. Des. Implement.*, Savannah, GA, USA, 2016, pp. 265–283.
[46] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Netw.*, vol. 106, pp. 249–259, Oct. 2018, doi: 10.1016/j.neunet.2018.07.011.

YUEPENG ZHOU received the master's degree in computer science from the Liaoning University of Technology, in 2009. He is currently pursuing the Ph.D. degree with Sun Yat-sen University, China. His research interests include deep neural networks, computer vision, and database.

HUIYOU CHANG is currently a Professor with the School of Data and Computer Science, Sun Yat-sen University. He is mainly engaged in embedded systems, database, workflow, operating systems, large-scale manufacturing management information systems, and computer integrated manufacturing system. He is currently an Expert with the National Machinery Industry, the Guangdong Manufacturing Informatization Expert Group, Department of Science and Technology, and the Guangdong Enterprise Informatization Expert Group (Economic and Trade Commission), a member of ICMLC Program of Famous International Conference, and the Chairman of the Guangzhou Forum of Young Computer Science and Technology of China Computer Society.

YONGHE LU is currently a Professor with the School of Information Management, Sun Yat-sen University. He is mainly engaged in semantic analysis and service of information resources, automatic text classification and clustering, text topic extraction, and intelligent information processing. He is also a Peer-Review Expert of the National Natural Fund Committee and a Science and Technology Consulting Expert of the Science and Technology Department of Guangdong Province.

XILI LU received the master's degree in computer science from the Liaoning University of Technology, in 2010. Her research interests include deep neural networks, data warehouse, and data mining.

RUQI ZHOU was born in 1971. He received the M.S. degree in computer technology from Guangxi Normal University. He is currently pursuing the Ph.D. degree in computer science and technology with Sun Yat-sen University. He is also a Professor with the Guangdong University of Education. His current research interests include machine learning, intelligent information processing, petri net theory and application, and granular computing. He is a Senior Member of the China Computer Federation and a member of the Chinese Association for Artificial Intelligence.