IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 19, 2018, accepted July 21, 2018, date of publication July 31, 2018, date of current version August 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2860785

Deep Convolutional Network Based on Pyramid Architecture

ENHUI LV[1], XUESONG WANG[1], (Member, IEEE), YUHU CHENG[1], (Member, IEEE), AND QIANG YU[2]

[1]School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China
[2]School of Electrical and Power Engineering, China University of Mining and Technology, Xuzhou 221116, China

Corresponding author: Yuhu Cheng (chengyuhu@163.com)

This work was supported by the National Natural Science Foundation of China under Grants 61772532 and 61472424.

ABSTRACT Deep convolutional network demonstrates that the classification accuracy can be remarkably improved by increasing the number of network layers, however, increases the accuracy by 1% of costs nearly doubling the number of layers. Meanwhile, gradient dispersion will occur in the training process, which leads to performance degradation. In order to solve the problem of training difficulty with the increased number of layers, we focus on network architecture and propose a deep convolutional network based on the pyramid structure. In the network architecture, as the number of layers increased, the feature map dimensions (i.e., the number of channels) are gradually increased at each layer to distribute the burden concentrated at locations of structural units affected by downsampling, such that all units are equally distributed. By exploring the sequence between the stacked elements inside the structural unit, we design a pyramidal building block, as its shape gradually widens from the top downwards, which is called the deep pyramid convolutional network (DPCNet). Experimental results on CIFAR-10 and CIFAR-100 datasets have shown that DPCNet has the superior generalization capability and can effectively improve the image classification accuracy.

INDEX TERMS Deep convolution network, pyramid architecture, feature map dimension, gradient dispersion.

I. INTRODUCTION

In recent years, the Deep Convolution Network [1] (DCN) has made great progress in solving complex tasks [2]–[5] in computer vision with significantly improved performance. Deep network architectures are known for its superior performance, because these network architectures commonly have deeply stacked convolutional filters with nonlinearity. Recent research [6], [7] also shows that the network depth takes an extremely important role in the network architecture. For example, object detection and image classification in computer vision task [8]–[13] all adopt deep networks, and the depth ranges from 8 [8] to 1202 [13].

In general, existing methods [14], [15] decrease the size of feature maps by increasing the stride of the filter or pooling, while stacking several convolution filters to increase the feature dimension. However, with the network depth increasing, accuracy gets saturated and then degrades rapidly [16]–[18]. Unexpectedly, such degradation is not caused by overfitting, and adding more layers to a suitably deep model leads to higher training error [13]. Moreover, training very deep networks has a problem of diminishing feature reuse, which results in very slow training. In order to make the ultra-deep DCN effectively trained, methods have been proposed, including well-designed initialization strategies [7], [17], shortcut connection [13], [19]–[22] and layer-by-layer training [23]–[26]. For DCN, the stochastic initialization strategy is simple and direct, but its disadvantages are also obvious. It not only affects the speed of network learning, but also leads to the learning process falling into local optimum [27]. Therefore, Glorot and Bengio [17] proposed an initialization method Xavier. It not only satisfies the necessary conditions of initialization, the activation value of each layer is unsaturated and remains not zero, but the variances of the activation value and state gradient of each layer are consistent with the variance during propagation. However, since the activation value of the sigmoid is symmetrical about zero, the variance of activation value and parametric gradient after Xavier initialization will gradually decrease, causing data instability

due to variance change after the update. Thereafter, the highway network proposed by Srivastava *et al.* [28] trains DCN by using a shortcut of shortcut connection within residual unit for residual learning, compared to other network architectures shows better generalization capability, which means effective utilization of deep learning features and fast convergence of the training progress. Note that there is an essential difference between the deep residual network and the highway network. The residual link of the latter is gate-controlled and the weights of these gates can be learned. Meanwhile, several variants are developed by modifying the ResNet architecture, such as Pre-activation ResNet [19], Stochastic Depth [21], has achieved very good results. Thus, the residual structure is considered very important for building an ultra-deep network. However, the follow-up study found that the depth residual network is formed by stacking exponential shallow networks, thus bypassing the DCN training problem. Larsson *et al.* [26] pointed out that shallow network output response is more rapid and the deep one is more accurate. By combining multiple shallow and deep ones, a novel FractalNet fractal network structure [26] is proposed. A ''student-teacher'' learning mechanism plays a role in the formation of internal network supervision depth through repeated combination of a plurality of different depth of network, which is help to construct an ultra-deep FractalNet. However, most modular building blocks of this kind of network design are formed by the fractal network embedding sub-structure. After network training by gradient descent method, the usage of this structure is prone to generate too much redundant information, resulting in overfitting phenomenon. As we all know, the training of DCN usually adopts the layer-by-layer training mechanism based on the gradient descent method. The network is trained from bottom to top layer by layer, and the previous layer output is used as the input of the next one. The disadvantage of this learning mechanism is that with the deepening of the network layers, the image pixels in the first layer are discarded and the connection between the higher layer and the input becomes more and more sparse, causing the error correction signal smaller from the top layer down. The parameters are updated very slow and cannot learn effectively. Therefore, Zagoruyko and Komodakis [22] proposed a new network architecture Wide ResNet based on residual module structure by reducing network depth and increasing the width, which also achieved good performance. Although this method is not widely used, it provides a new way of thinking. It is worth noting that the above-mentioned deep network architectures are stacks consisting of a large number of convolutional layers, and they perform down-sampling along the spatial dimension via pooling. Meanwhile, the feature map dimension is sharply increased at down-sampling locations to extract the diversified high-level attributes, which is very effective for improving model generalization. This is also the widely adopted method of controlling the size of feature maps. However, this can increase the burden concentrated at locations of structural unit affected by down-sampling. Therefore, there should be an overall smooth down-sampling in the overall architecture, which should be combined with numerical increase of channels. Based on above analysis, we designed a pyramid network architecture. In this network, instead of sharply increasing the feature map dimension at units that perform down-sampling, feature map dimension at all units is steadily increased. The pyramid network architecture, structural units, together with several related aspects such as ReLU and Dropout, are discussed with regard to how to affect the performance. In our study, we refer to this network architecture as a deep pyramid convolution network (DPCNet) and a pyramidal building block. That is, the number of feature maps gradually increases as a function of the depth, which resembles a pyramid structure that the shape gradually widens from the top downwards. The advantages of DPCNet are as follows: 1) By gradually increasing the dimension of feature maps at each layer, the burden of structural units concentrated at the down-sampling can be effectively distributed across all units; 2) The concatenation operation used in DPCNet can not only maximize the information back propagation, but also improve the reuse rate of feature maps; 3) The use of CReLU [29] as the activation function in the front layer of DPCNet not only preserves the positive and negative phase information extracted by the filter, but also reduces the filter redundancy caused by ReLU nonlinearity, so that the training parameters can be more effectively utilized; 4) The network parameters can be trained end-to-end by using Stochastic Gradient Descent with a mini-batch [30] (MB-SGD) to further avoid gradient dispersion.

II. DEEP PYRAMID CONVOLUTIONAL NETWORK

A. FEATURE MAP DIMENSION CONFIGURATION

Generally, ResNet uses a couple of convolution layers as a building block and the feature map dimension in the same building block remains consistent. The connected next unit multiplies the feature map dimension, and the input layer is followed by a convolutional channel to reduce image dimension. As is shown in Fig. 1, the feature map dimension from the input layer building block remains unchanged, and the feature map dimension in the down-sampling is doubled. In the case of CIFAR-10 and CIFAR-100 datasets [31], the feature map dimension D can be described as follows:

$$D = \begin{cases} 16, & if\ m = 1 \\ 16 \cdot 2^{m-2}, & if\ m \geq 2 \end{cases} \tag{1}$$

where $m \in \{1, 2, 3, 4, 5\}$ denotes the building block group. The building block that belongs to the same group has an equal feature map dimension. In the first group, there is only one convolution layer used to extract the RGB image features. For the m-th group, after the building block features are extracted, the feature map size is down-sampled by half and the number of dimensions is doubled.

As is shown in Fig. 2, DPCNet feature map dimension shows a trend of gradually linear growth. Taking CIFAR-10 and CIFAR-100 datasets as an example, the feature map

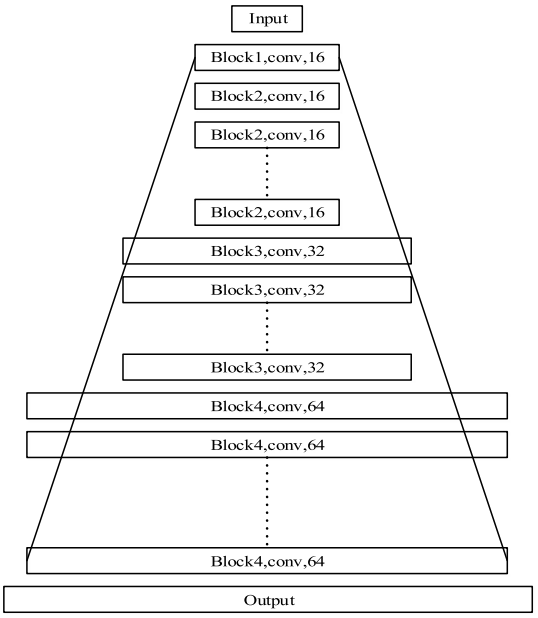


FIGURE 1. Diagram of feature map dimension of ResNet.

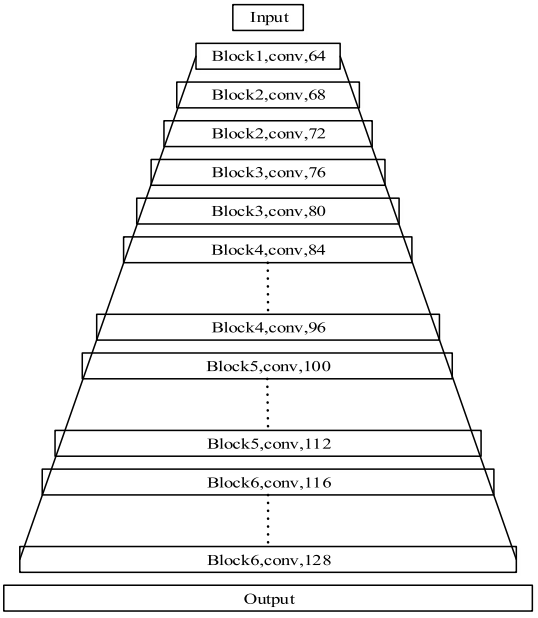


FIGURE 2. Diagram of feature map dimension of DPCNet.

dimension D can be described as follows:

$$D = \begin{cases} 64, & if\ N = 1 \\ 64 + \alpha \cdot N, & if\ N \geq 2 \end{cases} \quad (2)$$

where N denotes the network layer, α denotes the step factor. Table 1 shows the structure of DPCNet for the CIFAR-10 and CIFAR-100 datasets.

B. NETWORK ARCHITECTURE

To maximize the network generalization capability, it is natural to ask the following question: ''Can we design a better building block by altering the stacked elements inside the building block in more principled way?''. Therefore, the design idea of pyramid building block is described in detail as follows.

TABLE 1. Structure of DCPNet for CIFAR-10 and CIFAR-100 datasets.

Layer	Output Size	Layer Design(size, #filter)
Block1_conv1	32×32	3×3 conv, 64, stride 1
Block2_conv2	32×32	[3×3 conv, $64+\alpha \cdot N$]
Block2_pool1	16×16	Max 2×2, stride 2(“same”)
Block3_conv3	16×16	[3×3 conv, $64+\alpha \cdot N$]
Block3_pool2	8×8	Ave 2×2, stride 2(“same”)
Block4_conv4	8×8	[3×3 conv, $64+\alpha \cdot N$]
Block4_pool3	4×4	Ave 2×2, stride 2(“same”)
Block5_conv5	4×4	[3×3 conv, $64+\alpha \cdot N$]
Block5_pool4	2×2	Ave 2×2, stride 2(“same”)
Block6_conv6	2×2	[3×3 conv, $64+\alpha \cdot N$]
Block6_pool5	1×1	Ave 2×2, stride 2(“same”)
Classification layer	1×1	10/100

1) RECEPTIVE FIELD TREATMENT

With wide application of DCN in the computer vision field, many researchers have tried to improve the network architecture to fully reflect the network performance. For example, using smaller accepted fields or steps in the first convolutional layer. Different from the network architecture of Krizhevsky *et al.* [8] and Szegedy *et al.* [6], DPCNet does not use a relatively larger receptive field in the convolution layer of the first layer (For example, Krizhevsky *et al.* [8] used the 11 × 11 receptive field with stride 4; Szegedy *et al.* [6] used the 7 × 7 receptive field with stride 2). Instead, we use the 3 × 3 receptive field with stride 1 throughout the whole network, which is similar to the VGG [32] network architecture. Each pixel in the input can be convolved to get fuller image information. By introducing a plurality of relatively smaller nonlinear correction layer stacks to form a building block, the decision function is more distinguishable while the number of parameter can be reduced.

2) A NEW BUILDING BLOCK

The convolution layer that contains Rectified Linear Unit (ReLU) [33] layer and Batch Normalization (BN) [9] layer can form the basic building block of DPCNet. As in the building block the residual unit types proposed by He *et al.* [13] in ResNet, although network performance is increased by empirically improving the building block, further improvements have not yet to be studied. Therefore, the pyramid structure unit can be optimized by modifying the

stacked elements inside the building block, which leaves the room for performance improvement of DPCNet.

a: SHORTCUT DIRECT CONNECTION

We consider a single image x_0 that is passed through a convolutional network. The network comprises $(l+1)$ layers and each implements a nonlinear transformation $H_{l+1}(\cdot)$, whereas the nonlinear transformation $H_{l+1}(\cdot)$ serves as a composite function that contains BN, ReLU, pooling, or convolution operation. To further maximize the information flow between layers to improve the backpropagation gradient, a shortcut direct connection method that is different from the shortcut connection of the residual network is adopted. The difference is that concatenation [34] is used for shortcut direct connection when receiving the front-layer feature map, while the residual uses summation method. Shortcut direct connection is characterized by the access to any previous feature maps for a given layer, and any one layer of input can be spread to followers. Therefore, the formula for the $(l+1)$-th layer feature map is:

$$x_{l+1} = H_{l+1}([x_0, x_1, \cdots, x_{l+1}]) \tag{3}$$

where $[x_0, x_1, \cdots x_{l+1}]$ denotes the feature map of mutual concatenation. However, when the size of the feature map changes after the network performs a pooling operation, the concatenation operation of the feature map in (3) is not feasible. Therefore, a pooling layer is used as a transitional layer, forms a building block between the two pooling layers, and performs a feature concatenation operation in the building block. Considering the increase of layers, DPCNet's feature map dimension is gradually increasing, whereas concatenation operation feature dimension will get higher, causing slower network training. Therefore, concatenation operation is only used in Block 1, Block 2, and Block 3 in Table 4, which can not only maximize the information backpropagation but also solve the training difficulty due to excessive dimension of the feature map.

b: CReLU

It is found that the pre-convolution layer of DCN can extract the positive and negative phase information of the input signal through filter, and the parameter distribution has stronger negative correlation. With the deepening of the network layer, this correlation is gradually weakened, which is called pairing phenomenon [29]. As an important activation function in deep learning, ReLU is as follows: $\mathrm{ReLU}(x) = max(0, x)$. Due to the nonlinearity essence and thus the capability of non-negative element filtering, application in the previous network will generate a large amount of redundant information. In order to effectively eliminate the pairing phenomenon and redundancy information generated by ReLU, a new Concatenated Rectified Linear Unit (CReLU) is introduced, which is:

$$\mathrm{CReLU}(x) = [\mathrm{ReLU}(x), \mathrm{ReLU}(-x)] \tag{4}$$

This activation function can not only preserve the positive and negative phase information extracted by the filter, but also reduce the redundant information of the filter caused by ReLU nonlinearity, so that the trainable parameters are more effectively utilized. How it works: convolution of the eigenvalues is reversed and concatenated with the original eigenvalues passed to ReLU, and finally corresponding activation output value is obtained. As the network layers deepen, the negative correlation is gradually weakened. Therefore, in this paper it is only used in the building blocks 2-3, as shown in Table 4.

c: ReLUs

Including ReLUs in the building blocks of pyramidal units is essential for nonlinearity. However, the performance can empirically vary depending on the locations and the number of ReLUs. He *et al.* [19] and Shen *et al.* [35] also pointed out that ReLUs inside the residual unit seem to have the function of filtering non-negative elements, which affects the overall network performance, and the heavy use of ReLUs in each residual unit can have a negative impact on network performance. Therefore, the position of ReLUs in the building block of pyramidal unit is of great significance to the deep pyramid convolutional network performance. First, delete the first ReLU in the building block of pyramidal unit, as is shown in Fig. 3(a). The first ReLU is removed and the blocks are changed to conv-BN-conv-BN-ReLU. Then, deleting the second ReLU in the building block of pyramidal unit, as is shown in Fig. 3(b). Removing the second ReLU and change the building block to conv-BN-ReLU-conv-BN. The experimental results in Table 2 show that the performance of DPCNet decreased after any layer ReLU in building block of pyramidal unit is deleted. It is obviously shown that: 1) The pyramid building block between the two convolutions without ReLU weakens the architecture ability; 2) The second convolution layer in the pyramidal building block cannot guarantee the nonlinearity of the structure unit without ReLU. Therefore, providing an appropriate number of ReLUs in building block of pyramidal unit can ensure feature space diversity and improve network performance.

TABLE 2. Testing error of DPCNet with different Building Blocks on CIFAR-10 dataset.

Building Block	Testing Error
(a)Removing the first ReLU	6.64%
(b)Removing the second ReLU	6.66%
(c)DPNet with ReLUs	5.92%
(d)Dropout between convs	5.79%
(e)Dropout after the final conv	5.43%

d: DROPOUT IN PYRAMID BLOCKS

Dropout was first proposed in [36] and then applied on top layers in the network architecture [8] with a large number of parameters to prevent inter-feature coadaptation

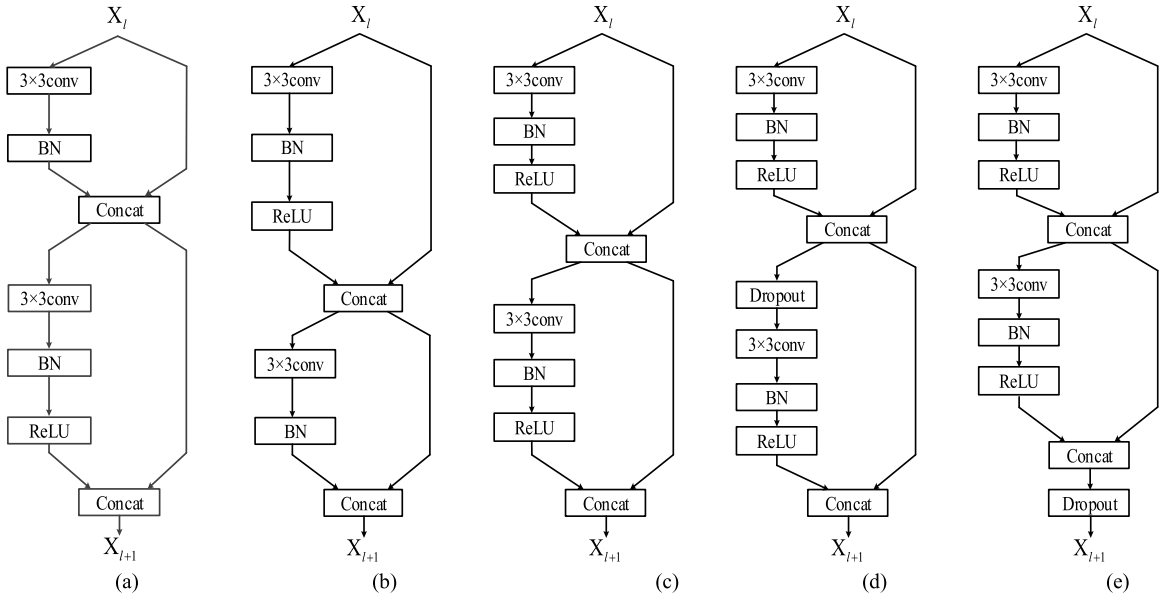


FIGURE 3. Various types of building blocks. (a) Removing the first ReLU. (b) Removing the second ReLU (c) DPNet with ReLUs. (d) Dropout between convs. (e) Dropout after the final conv.

and overfitting. Then dropout is replaced by BN [9] to solve the problem of internal covariance shift caused by the change of data distribution. However, as a regularization technique and experimental studies have shown that a network with BN achieves better classification accuracy. As the DPCNet deepens with the number of layer increases, the feature map dimension gradually increases, and does the corresponding network parameter as well. The BN layer was introduced into the network architecture to regularize the data distribution. To further strengthen the network training to improve model generalization capability, the dropout layer is added in the building block of pyramidal unit to regularize network parameters. Overfitting can be avoided by enhancing the degree of feature repetition. Dropout has been introduced into the identity mapping block in the residual network [19]. However, the experimental results show that the network performance has not been improved. To introduce the dropout more reasonably in the building block of pyramidal unit, the following work was done. First, add a dropout layer between two convolution layers in the building block and place it behind the ReLU and BN layers to prevent overfitting, as is shown in Fig. 3(d). Then add the dropout layer to the two convolutions in the building block as shown in Fig. 3(e). The experimental results in Table 2 show that adding dropout layer in the building block effectively prevents overfitting of the network. Dropout layer is added at the bottom to maximize network performance and improve the image classification accuracy.

III. EXPERIMENTAL RESULTS AND ANALYSIS

A. EXPERIMENTAL DATASETS

In view of the limited computational resources, we evaluate and compare the performance of DPCNet with that of the existing networks [13], [22], [37], [38] using typical benchmark datasets: CIFAR-10 and CIFAR-100 [39]. CIFAR-10 contains 10 classes, each with 5000 training images and 1000 test images. There are 60,000 RGB three-channel images with a size of 32 × 32. CIFAR-100 is similar to CIFAR-10 with the same size and number of samples, but CIFAR-100 contains 100 classes, and each with 500 training images and 100 test images, which makes the classification task difficulty increased significantly. Using the relatively popularity ResNet [13] architecture as a benchmark contrast method, most of the same settings are retained when preprocessing dataset image. The standard data augmentation scheme is widely used for these datasets. First, zero-pad the images with 4 pixels on each side to form an image with a size of 40 × 40. Then, randomly crop to produce 32 × 32-size images, followed by horizontally mirroring the images. The images preprocess by normalizing the images using the channel means and standard deviations.

B. PARAMETER ANALYSIS

DPCNet was modeled using the Caffe learning framework, running on a single GPU model for Tesla K40c. To improve the efficiency of GPU computing, the learning framework is optimized with CUDA and CUDNN. For DPCNet, the step factor α and the number of layers N directly affect the image classification performance. When the step factor α tries to increase, the total number of layers must be decreased. To find an optimal ratio, the number of layers is fixed $N = 18$ and the α value influence on the image classification performance is discussed. Take the building mode of dropout after the final conv as an example. The building block is constructed based on DPCNet as a benchmark, CIFAR-10 dataset and the results are discussed in Table 3. It can be observed

TABLE 3. Comparison of testing error under different values of step factor on CIFAR-10 dataset.

Depth	α	Input Feat. Dim.	Output Feat. Dim.	#of Params	Testing Error
18	4	64	128	1.38M	6.80%
18	12	64	256	4.13M	6.15%
18	28	64	512	14.32M	5.67%
18	8	128	256	5.51M	5.80%
18	24	128	512	16.50M	5.13%
18	16	256	512	22.02M	5.05%
18	12	320	512	25.37M	5.21%

that: 1) When the input feature map dimension is fixed, the number of parameters increases with increase of the output feature map dimension, and the testing error decreases; 2) When the output feature map dimension is fixed, the smaller the step factor, the lower the testing error; 3) When the number of parameters is 25.37M, classification performance decreases. This is because of overfitting phenomenon; 4) $N = 18$ and $\alpha = 16$ turned out to be the optimal combination, which therefore is further applied in the follow-up experiment.

C. EXPERIMENTAL STUDY

DPCNet is trained using backpropagation by SGD with Nesterov momentum for 250 epochs, and a mini-batch size of 128. The initial learning rate is set as 0.1. The learning rate is set by heuristic and annealing methods, specifically as follows: when the classification accuracy no longer increases, the current learning rate dropped to 1/10 of the previous learning rate; After 4 annealing strides, the final learning rate is set as 1/10000 of the initial. The filter parameters are initialized by Glorot and Bengio [17], and we use a weight decay of 0.0005, a momentum of 0.9. Specific network parameter configuration is shown in Table 4. And ''conv_x'' denotes conv-BN-ReLU. A dropout layer is added into each block after the down-sampling unit position.

1) PERFORMANCE EVALUATION

Based on the above observations and analysis, we compared the performance of DCPNet and traditional DCN. First, we build DPCNet and traditional DCN with the approximation number of parameters and the same number of layers. The testing error is used to evaluate network performance, and the CIFAR-10 dataset is used. The testing errors are shown in Table 5, where ''Training Time'' denotes the average training time consumption per 100 iterations. It can be observed that DCPNet has similar time consumption compared to DCN, but the testing error was decreased by nearly 40%. This shows that the proposed network architecture is an effective means of improving generalization capability. The main index of time performance is the time complexity. The time complexity of DCN is analyzed according to the feature map dimension. Assume the input feature map dimension is n, the overall time complexity of DCN is:

$$\text{Time} \sim O\left(\sum_{l=1}^{P} M_l^2 \cdot K_l^2 \cdot C_{l-1} \cdot C_l\right)$$

where P is the network depth, l is the l-th convolutional layer, M is the size of output feature maps, K is the size of convolution kernels, C_{l-1} is the feature map dimension of the previous layer, and C_l is the feature map dimension of the next layer. Since the traditional DCN and DPCNet constructed have the same convolution kernel and feature map size, the time complexity formula can be simplified as follows:

$$\text{Time} \sim O\left(\sum_{l=1}^{P} C_{l-1} \cdot C_l\right)$$

Take an 18-layer network as an example, the entire time complexity of DCPNet is $o\left(18n^2 + (258\alpha + 3)n + 1360\right)$, where α denotes the step factor. The traditional DCN time complexity is $O(1184n^2+3n)$. Therefore, the time complexity of DPCNet and traditional DCN belongs to the same order, i.e., $o(n^2)$.

2) EFFECT OF CONCATENATION OPERATION

The concatenation operation used in DPCNet can not only maximize the information back propagation, but also improve the reuse rate of feature maps. However, as the number of layer increases, the dimension of feature map is gradually increasing. After the concatenation operation, the dimension of the feature map will become larger, which will easily slow the network training. The relationship between the network performance and the number of blocks covered by the concatenation operation is further studied, which is shown in Table 6. In Table 6, ''DPCNet-C0'' denotes that the network architecture has no concatenation operation, ''DPCNet-C1-3'' denotes that the concatenation operation is only used in Blocks 1-3 and ''DPCNet-C6'' denotes that the entire network adopted concatenation operations. From the experimental results, it can be noted that as concatenation operation covers more blocks, the amount of network parameters and the training time gradually increase, while the testing error consistently decreases, i.e., the generalization capability is improved.

TABLE 4. Network parameter configuration.

Layer Name	Output Size	Layer Design(size, #filter)	Activation Function
Block1_conv0	32×32	3×3 conv, 256, stride 1	
Block2_conv2_1	32×32	3×3 conv, 272, stride 1	CReLU
Block2_conv2_2	32×32	3×3 conv, 288, stride 1	ReLU
Block2_pool1	16×16	Max 2×2, stride 2("same"), dropout 0.1	
Block3_conv3_1	16×16	3×3 conv, 304, stride 1	CReLU
Block3_conv3_2	16×16	3×3 conv, 320, stride 1	ReLU
Block3_pool2	8×8	Ave 2×2, stride 2("same"), dropout 0.2	
Block4_conv4_1	8×8	3×3 conv, 336, stride 1	ReLU
Block4_conv4_2	8×8	3×3 conv, 352, stride 1	ReLU
Block4_conv4_3	8×8	3×3 conv, 368, stride 1	ReLU
Block4_conv4_4	8×8	3×3 conv, 384, stride 1	ReLU
Block4_pool3	4×4	Ave 2×2, stride 2("same"), dropout 0.3	
Block5_conv5_1	4×4	3×3 conv, 400, stride 1	ReLU
Block5_conv5_2	4×4	3×3 conv, 416, stride 1	ReLU
Block5_conv5_3	4×4	3×3 conv, 432, stride 1	ReLU
Block5_conv5_4	4×4	3×3 conv, 448, stride 1	ReLU
Block5_pool4	2×2	Ave 2×2, stride 2("same"), dropout 0.4	
Block6_conv6_1	2×2	3×3 conv, 464, stride 1	ReLU
Block6_conv6_2	2×2	3×3 conv, 480, stride 1	ReLU
Block6_conv6_3	2×2	3×3 conv, 496, stride 1	ReLU
Block6_conv6_4	2×2	3×3 conv, 512, stride 1	ReLU
Block6_pool5	1×1	Ave 2×2, stride 2("same"), dropout 0.4	
Classification layer	1×1	10/100	SoftmaxLoss

TABLE 5. Performance comparison of DCPNet and DCN on CIFAR-10 dataset.

Network	Depth	#of Params	Training Time	Input Feat. Dim.	Output Feat. Dim	Testing Error
DCN	18	0.97M	7.6s	8	128	9.8%
DPCNet	18	0.98M	7.8s	16	112	6.7%

TABLE 6. Performance comparison of DPCNet covering different blocks in concatenation operation on CIFAR-10 dataset.

Network	Depth	#of Params	Training Time	Input Feat. Dim.	Output Feat. Dim.	Testing Error
DPCNet-C0	18	21.3M	22.8s	256	512	4.8%
DPCNet-C1-3	18	22.02M	24.6s	256	512	4.6%
DPCNet-C6	18	30.4M	26.8s	256	512	3.6%

3) VARIATION OF NETWORK DEPTH

We also compare the performance of pyramid network with different depths to explore how performance changes as the number of network layers increasing. In this research, the entire network adopted concatenation operations. First, we fix the initial input feature dimension as 256 and vary the depth among {18, 24, 30, 36}. The experimental results are shown in Table 7 on the CIFAR-10 dataset, where "Testing Time" denotes the average testing time consumption per 100 iterations. It can be observed that with

TABLE 7. Performance comparison of DPCNet with different depths on CIFAR-10 dataset.

Network	Depth	#of Params	Training Time	Testing Time	Input Feat. Dim.	Output Feat. Dim.	Testing Error
DPCNet-C6	18	30.4M	26.8s	1.3s	256	512	3.6%
DPCNet-C6	24	55.4M	35.9s	2.0s	256	608	4.5%
DPCNet-C6	30	93.0M	46.3s	2.6s	256	704	4.7%
DPCNet-C6	36	138.2M	60.1s	3.4s	256	800	4.8%

TABLE 8. Performance comparison of time on CIFAR-10 Dataset.

Network	Depth	#of Params	Input Feat. Dim.	Output Feat. Dim.	Training Time	Testing Time
DPCNet-C6	18	30.4M	256	512	26.8s	1.3s
DPCNet-C6	24	55.4M	256	608	35.9s	2.0s
DPCNet-C6	30	93.0M	256	704	46.3s	2.6s
DPCNet-C6	36	138.2M	256	800	60.1s	3.4s
ResNet	110	1.7M	16	64	36.6s	6.9s
Wide ResNet	28	36.5M	160	640	72.6s	16.1s

the increase of network depth, all the indexes including the parameter amount, training time, testing time and testing error become poor. This shows that the optimization of pyramid network suffers from the increased depth.

4) PERFORMANCE COMPARISON OF COMPUTATION TIME

We compare DPCNet with ResNet and Wide ResNet. In our study, the entire DPCNet adopted concatenation operations. The observation from Table 8 is that DPCNet-C6 training/testing time exhibits consistent improvement when the number of parameter increases. For example, the training time of 18-layer DPCNet-C6 with 30.4M parameter is 26.8s, while ResNet reaches 38.6s with 1.7M parameters, and Wide ResNet reaches 72.6s with 36.5M. The training/testing time result of 110-layer ResNet is worse than that of DPCNet-C6, although ResNet has less parameters. This is because of the over-deep network. Meanwhile, the results shown in Table 8 verify that DPCNet-C6 with the linear growth in parameters is able to achieve better performance.

D. CLASSIFICATION RESULTS

1) ACCURACY

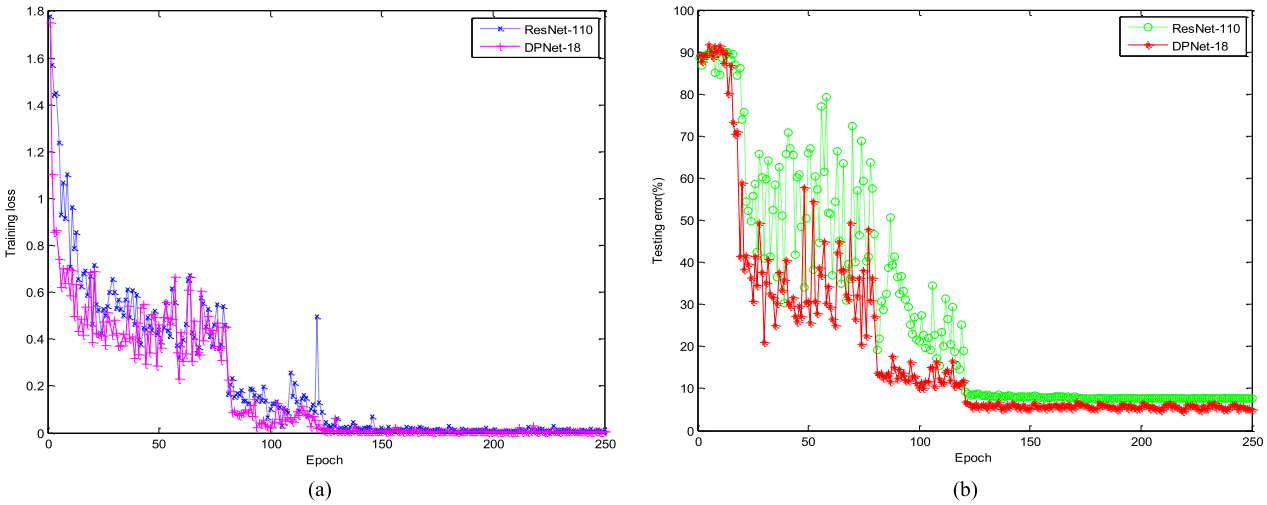
We compare our DPCNet with several deep models including NIN [24], All-DCN [25], DSN [20], FitNet [38], Highway [28], ResNet [13], Pre-activation ResNet [19], Stochastic Depth [21], Wide ResNet [22], Weighted ResNet [35], FractalNet [26], and ReNeXt-29 [39]. The following observations can be obtained from Table 9: 1) Except ReNeXt, DPCNet-C6 achieves the lowest testing errors on both CIFAR-10 and CIFAR-100 datasets; 2) Although the testing errors of DPCNet are slightly higher than that of ReNeXt, our DPCNet is relatively shallow; 3) DPCNet has reduced by 50% testing error compared with the most widely used ResNet [13]; 4) Compared with FractalNet [26] and Wide ResNet [22], the parameter utilization of DPCNet is higher when the same testing error is achieved. In this case, the FractalNet parameter quantity is 38.6M and the DPCNet parameter quantity is 30.4M. The symbol "-" in Table 9 denotes that there is no related description and experiment in corresponding literature.

2) REUSE RATE OF FEATURE MAPS

The residual block with identity mapping that allows to train very deep networks is at the same time a weakness for ResNet [22]. As gradient flows through the network, there is nothing to force it to go through residual block weights and it can avoid learning anything during training. Therefore, it is possible that only a few blocks learn useful representations. This problem is formulated as a diminishing feature reuse. The observation is that DPCNet indeed resolves this phenomenon, by attempting to gradually increase the feature map dimension at all units and to evenly distribute the burden of increasing the feature maps. However, the concatenation operation used in DPCNet can not only maximize the information back propagation, but also improve the reuse rate of feature maps. We further analyze the effect of DPCNet by comparing it with ResNet, and following experimental results are shown. We compare the training loss and testing error curves of DPCNet with those of ResNet in Fig. 4. For DPCNet, a depth of 18 layers with a step factor $\alpha = 16$ is used, and the standard ResNet with 110 layers is used for comparison. The experimental results show that: 1) Compared with ResNet, DPCNet has a lower training loss and testing error, indicating that the difference between the

TABLE 9. Testing error comparisons of different deep network models on CIFAR-10 and CIFAR-100 datasets.

Network	#of Params	Output Feat. Dim.	Depth	Testing Error	
				CIFAR-10	CIFAR-100
NIN[24]	-	-	-	8.81%	35.68%
All-DCN[25]	-	-	-	7.25%	33.71%
DSN[20]	-	-	-	7.97%	34.57%
FitNet[38]	-	-	-	8.39%	35.04%
Highway[28]	-	-	-	7.72%	32.39%
ResNet[13]	1.7M	64	110	6.43%	25.16%
ResNet[13]	10.2M	64	1001	-	27.82%
ResNet[13]	19.4M	64	1202	7.93%	-
Pre-activation ResNet[19]	1.7M	64	164	5.46%	24.33%
Pre-activation ResNet[19]	10.2M	64	1001	4.62%	22.71%
Stochastic Depth[21]	1.7M	64	110	5.23%	24.58%
Stochastic Depth[21]	10.2M	64	1202	4.91%	-
Wide ResNet(width×4)[22]	8.7M	256	40	4.97%	22.89%
Wide ResNet(width×10)[22]	36.5M	640	28	4.17%	20.50%
Weighted ResNet[35]	19.1M	64	1192	5.10%	-
FractalNet[26]	38.6M	1024	21	5.22%	23.30%
FractalNet+Dropout/Drop-path[26]	38.6M	1024	21	4.60%	23.73%
ReNeXt-29(8×64d)[39]	34.4M	-	29	3.65%	17.77%
ReNeXt-29(16×64d)[39]	68.1M	-	29	3.58%	17.31%
DPCNet-C1-3(N=18, α=16)	22.02M	512	18	**4.20%**	**22.69%**
DPCNet-C6(N=18, α=16)	30.4M	512	18	**3.60%**	**19.21%**



(a) (b)

FIGURE 4. Performance comparison between ResNet and DPCNet (CIFAR-10 dataset). (a) Training loss versus epoch plots. (b) Testing error versus epoch plots.

predicted and real-value is lower. It also means that DPCNet is easier to be optimized, features are much effectively utilized and network gets converged faster; 2) DPCNet has superior testing accuracy, thereby confirming the advantage of generalization capability compared with ResNet.

IV. CONCLUSION

In this paper, we focus on network architecture design and proposed a deep convolutional network based on pyramid structure. With respect to feature map dimension, instead of sharply increasing the feature map dimension at units that perform down-sampling, the feature map dimension gradually increases at all units to involve as many locations as possible. Furthermore, we also developed a novel pyramidal unit, which includes a new building block for a pyramidal unit with concatenation operation, CReLU, ReLU, and Dropout. This design leads to further improved generalization ability. Experiments on CIFAR-10 and CIFAR-100 datasets shows that DPCNet is easier to be optimized, features are used more effectively and the network converges faster than hierarchically stacked DCN networks.

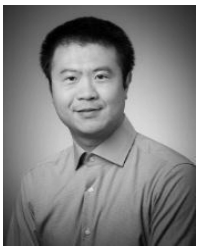
REFERENCES

[1] H. M. Bui, M. Lech, E. Cheng, K. Neville, and I. S. Burnett, "Object recognition using deep convolutional features transformed by a recursive network structure," *IEEE Access*, vol. 4, pp. 10059–10066, 2016.
[2] W. H. Deng, Y. K. Fang, and Z. Q. Xu, "Facial landmark localization by enhanced convolutional neural network," *Neurocomputing*, vol. 273, pp. 222–229, Jan. 2018.
[3] C. Deng, Z. J. Chen, X. L. Liu, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3893–3903, Aug. 2018.
[4] Y. Su, Y. Lu, M. Chen and A.-A. Liu, "Spatiotemporal joint mitosis detection using CNN-LSTM network in time-lapse phase contrast microscopy images," *IEEE Access*, vol. 5, pp. 18033–18041, 2017.
[5] H. Y. Zhu *et al.*, "YoTube: Searching action proposal via recurrent and static regression networks," *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 2609–2622, Jun. 2018.
[6] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[7] K. He, X. Zhang, S. Q. Ren, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, Feb. 2015, pp. 1026–1034.
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 84–90, Jun. 2012.
[9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, 2015, pp. 1–11.
[10] W. Cao, J. Yuan, Z. He, Z. Zhang, and Z. He, "Fast deep neural networks with knowledge guided training and predicted regions of interests for real-time video object detection," *IEEE Access*, vol. 6, pp. 8990–8999, 2018.
[11] R. Girshick. (2015). "Fast R-CNN." [Online]. Available: https://arxiv.org/abs/1504.08083
[12] S. Ren, K. He, R. Girshick, and J. Sun. (2016). "Faster R-CNN: Towards real-time object detection with region proposal networks." [Online]. Available: https://arxiv.org/abs/1506.01497
[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[14] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 8, pp. 1553–1565, Aug. 2014.
[15] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang. (2017). "Interleaved group convolutions for deep neural networks." [Online]. Available: https://arxiv.org/abs/1707.02725
[16] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
[17] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.
[18] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 5353–5360.
[19] K. He, X. Zhang, S. Ren, and J. Sun. (2016). "Identity mappings in deep residual networks." [Online]. Available: https://arxiv.org/abs/1603.05027
[20] C. Y. Lee, S. Xie, and P. W. Gallagher, "Deeply-supervised nets," *J. Mach. Learn. Res.*, vol. 38, pp. 562–570, Feb. 2015.
[21] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Weinberger. (2016). "Deep networks with stochastic depth." [Online]. Available: https://arxiv.org/abs/1603.09382
[22] S. Zagoruyko and N. Komodakis. (2016). "Wide residual networks." [Online]. Available: https://arxiv.org/abs/1605.07146
[23] J. Schmidhuber, "Learning complex, extended sequences using the principle of history compression," *Neural Comput.*, vol. 4, no. 2, pp. 234–242, Mar. 1992.
[24] M. Lin, Q. Chen, and S. Yan. (2014). "Network in network." [Online]. Available: https://arxiv.org/abs/1312.4400
[25] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. (2014). "Striving for simplicity: The all convolutional net." [Online]. Available: https://arxiv.org/abs/1412.6806
[26] G. Larsson, M. Maire, and G. Shakhnarovich. (2016). "FractalNet: Ultra-deep neural networks without residuals." [Online]. Available: https://arxiv.org/abs/1605.07648
[27] J. Masci *et al.*, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *Proc. Int. Conf. Artif. Neural Netw. Mach. Learn.* in Lecture Notes in Computer Science and Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 6791. Springer-Verlag, 2011, pp. 52–59.
[28] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1–9.
[29] W. L. Shang, K. Sohn, D. Almeida, and H. Lee, "Understanding and improving convolutional neural networks via concatenated rectified linear units," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2217–2225.
[30] Q. Qian, R. Jin, J. F. Yi, L. Zhang, and S. Zhu, "Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (SGD)," *Mach. Learn.*, vol. 99, no. 3, pp. 353–372, 2014.
[31] M. Z. Alom, M. Hasan, C. Yakopcic, and T. M. Taha. (2017). "Inception recurrent convolutional neural network for object recognition." [Online]. Available: https://arxiv.org/abs/1704.07709
[32] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556
[33] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[34] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 2261–2269.
[35] F. Shen and G. Zeng, "Weighted residuals for very deep networks," in *Proc. Int. Conf. Syst. Inf.*, 2016, pp. 936–941.
[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
[37] B. Q. Li and Y. Y. He, "An improved ResNet based on the adjustable short," *IEEE Access*, vol. 6, pp. 18967–18974, 2018.
[38] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. (2015). "FitNets: Hints for thin deep nets." [Online]. Available: https://arxiv.org/abs/1412.6550
[39] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 1492–1500.

ENHUI LV received the master's degree from Shenyang Ligong University in 2014. He is currently pursuing the Ph.D. degree with the China University of Mining and Technology. His main research interests include image classification and deep learning.

XUESONG WANG received the Ph.D. degree from the China University of Mining and Technology in 2002. She is currently 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. She is currently the Associate Editor of the IEEE Transactions on Systems, Man, and Cybernetics: Systems and the *International Journal of Machine Learning and Cybernetics*.

YUHU CHENG received the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, in 2005. He is currently a Professor with the School of Information and Electrical Engineering, China University of Mining and Technology. His main research interests include machine learning, transfer learning, and intelligent system.



QIANG YU received the Ph.D. degree from the University of Bundeswehr Muenchen, Munich, Germany, in 2012. He is currently an Associate Professor with the School of Electrical and Power Engineering, China University of Mining and Technology. His main research interests include intelligent learning and systems.