Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8196906, 15 pages
https://doi.org/10.1155/2018/8196906

Research Article

Multiactivation Pooling Method in Convolutional Neural Networks for Image Recognition

Qi Zhao, Shuchang Lyu, Boxue Zhang, and Wenquan Feng

School of Electronics and Information Engineering, Beihang University, Beijing 100191, China

Correspondence should be addressed to Qi Zhao; zhaoqi@buaa.edu.cn

Received 21 February 2018; Accepted 8 May 2018; Published 26 June 2018

Academic Editor: Xuyun Zhang

Convolutional neural networks (CNNs) are becoming more and more popular today. CNNs now have become a popular feature extractor applying to image processing, big data processing, fog computing, etc. CNNs usually consist of several basic units like convolutional unit, pooling unit, activation unit, and so on. In CNNs, conventional pooling methods refer to 2×2 max-pooling and average-pooling, which are applied after the convolutional or ReLU layers. In this paper, we propose a Multiactivation Pooling (MAP) Method to make the CNNs more accurate on classification tasks without increasing depth and trainable parameters. We add more convolutional layers before one pooling layer and expand the pooling region to 4×4, 8×8, 16×16, and even larger. When doing large-scale subsampling, we pick top-k activation, sum up them, and constrain them by a hyperparameter σ. We pick VGG, ALL-CNN, and DenseNets as our baseline models and evaluate our proposed MAP method on benchmark datasets: CIFAR-10, CIFAR-100, SVHN, and ImageNet. The classification results are competitive.

1. Introduction

Convolutional neural networks (CNNs) have excellent performance on image classification and many other visual tasks [1–10] in recent years since AlexNet [11] achieved great success in ImageNet Challenge. With the rapid development of hardware capacity, wider and deeper networks can be trained smoothly. Nowadays, increasing the depth of networks is the main trend of improving networks' overall performance. The first proposed convolutional neural network, LeNet5 [12], has 5 layers. AlexNet contains 8 layers which consist of 5 convolutional layers and 3 fully connected layers. VGG [13] networks are designed even deeper. The deepest number of layers reaches 19, while GoogLeNet [14] achieves 22. Residual Networks (ResNets) [16, 17] and Dense Convolutional Networks (DenseNets) [15] which have been proposed in the last two years start to use shortcuts structure to allow the networks to surpass 100, even 1000-layer barrier.

When networks are deep, gradient vanishing is a serious problem during training process. ResNets [16, 17] and DenseNets [15] explicitly take advantage of the shortcuts structure between each two blocks. Highway Networks [18] integrate nonlinear transform and transform gates to create shortcuts implicitly. FractalNets [19, 20] apply drop paths, which can achieve the similar effect as ResNets. All of the methods can effectively solve the gradient vanishing.

Research [21] shows that classification accuracy will degrade rapidly when applying backpropagation [22] to deeper plain networks. Normalization method [23], to some degree, can help ease the problem. Despite lacking the potential to become deeper, plain networks still have many advantages, especially when dealing with some embedded vision tasks. The ALL Convolution Net (ALL-CNN) [24] uses only convolutional layers with small amount of parameters instead of alternating convolutional and max-pooling layers, which has outstanding performance. Based on plain network structure, many small networks perform better on limited resource. Similarity Networks (SimNets) [25] use the similarity operator and global MEX pooling method to improve the capacity of small-size networks as much as possible. SqueezeNet [26] reduces the channels of 3×3 filters and partially substitutes them by 1×1 in order to simplify the networks without hurting network capability. MobileNets [27] use depth-wise separable convolutions to build lightweight networks and work well on embedded devices. Moreover, some notable researches such as ShuffleNet [28] and SEP-Nets [29] have already

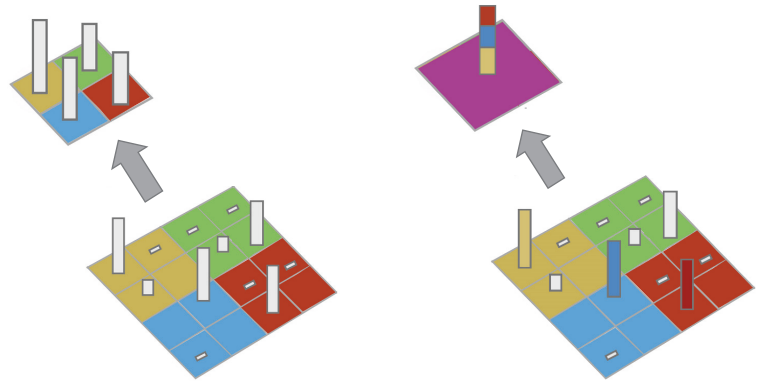

Figure 1: Comparison between 2×2 max-pooling (left) [36] and 4×4 top-3 Multiactivation Pooling (right) where σ is set to 0.5.

applied residual structure on mobile device and can reduce computational complexity effectively. With concise structure and less parameters, many researches [10] apply different kinds of CNNs as part of their model to process the big data, which improves the whole performance.

In this paper, we propose a new pooling method that picks top-k activation in every pooling region (Figure 1) to make sure that the maximum information can pass through subsampling gates. To enable this method suitable for plain networks, we do some necessary adjustment on networks architecture. **First,** we use small 3×3 receptive fields (stride = 1) throughout the whole net and add more convolutional layers before certain pooling layer. Nonlinear transform unit: ReLU [30] is followed after each convolutional layer. As input feature maps pass through more convolutional and ReLU layers, output feature maps become sparse. Therefore, expanding the pooling region and picking the max activation surely have effect on reducing noise even if some information might be ignored. **Second,** max-pooling method shows its strong performance in sparse features representation, but more key features will certainly be dropped when pooling regions are larger. To further improve the representation capability, we pick top-k activation, sum them up, and constrain the summation by a constant σ, which ranges from zero to one. If σ is set equivalent to $1/k$, the output value of pooling layer is the average of the picked top-k activation. Generally, we make σ a little bigger than $1/k$ to prevent activation from weakening too much when there are only few features active.

In plain networks, such as VGG and ALL-CNN, our MAP method is competitive in achieving higher accuracy on classification tasks with depth and trainable parameters not increased. When reducing the number of layers, the cost of graphics memories or internal memories will decrease a lot during training process. Therefore, networks with MAP method have huge advantage against both traditional networks and deep networks with shortcuts structure. Figure 2 shows the plain networks used different pooling strategies. In Networks with shortcuts structure, like DenseNets, we proposed plain structure (Figure 3) with MAP method into the Transition layers to extract features. Although a little more trainable parameters are added, the classification results improve.

We evaluate MAP method on 4 notable benchmarks datasets (CIFAR-10, CIFAR-100 [31], SVHN, and ImageNet). We mainly choose VGG [13] as our baseline models and it turns out that our enhanced models outperform them. We also apply our method to ALL-CNN [24] and DenseNets [15] and also get better results.

2. Related Works

In visual task, most of the networks insert pooling layers between convolutional layers for features abstraction and dimension reduction. The idea of pooling originates in Hubel and Wiesel's seminars [32] on complex cells in the visual cortex. In 1982, Fukuhshima and Miyake used feature pooling in neocognitron [33]. Pooling method has been widely used since the concept of convolutional neural networks structure was first put forward. LeNet-5 [12] has two convolutional layers and a pooling layer closely follows each of them. At that time, max-pooling and average-pooling both performed well in LeNet-5. Although the development of CNN slowed down for a period, the researches of pooling method never stop. In many famous traditional algorithms like SIFT [34] and HOG [35], pooling method plays an important role.

Two most popular pooling methods, max-pooling and average-pooling, have already been deeply researched in theory [37]. When choosing pooling strategies, there is always a trade-off between preserving more information which may include noise and decreasing more noise as well as useful information. Through research, it is found that max-pooling is more suitable for sparse coding than average-pooling [38].

Recently, CNNs have achieved great success on various visual tasks. Pooling layers have already become a fixed part in almost every network. Moreover, many new pooling methods are proposed one after another, which inspire us a lot. Different from conventional deterministic pooling operations, Stochastic Pooling [39] method picks activation randomly from each pooling region according to multinomial distribution. This notable research highly improves the generalization of networks. When applying Dropout to max-pooling layers [40] on training datasets, a new pooling method called scaled max-pooling [41] is designed to be applied on testing datasets. This idea of adopting different pooling methods on training and testing datasets is novel

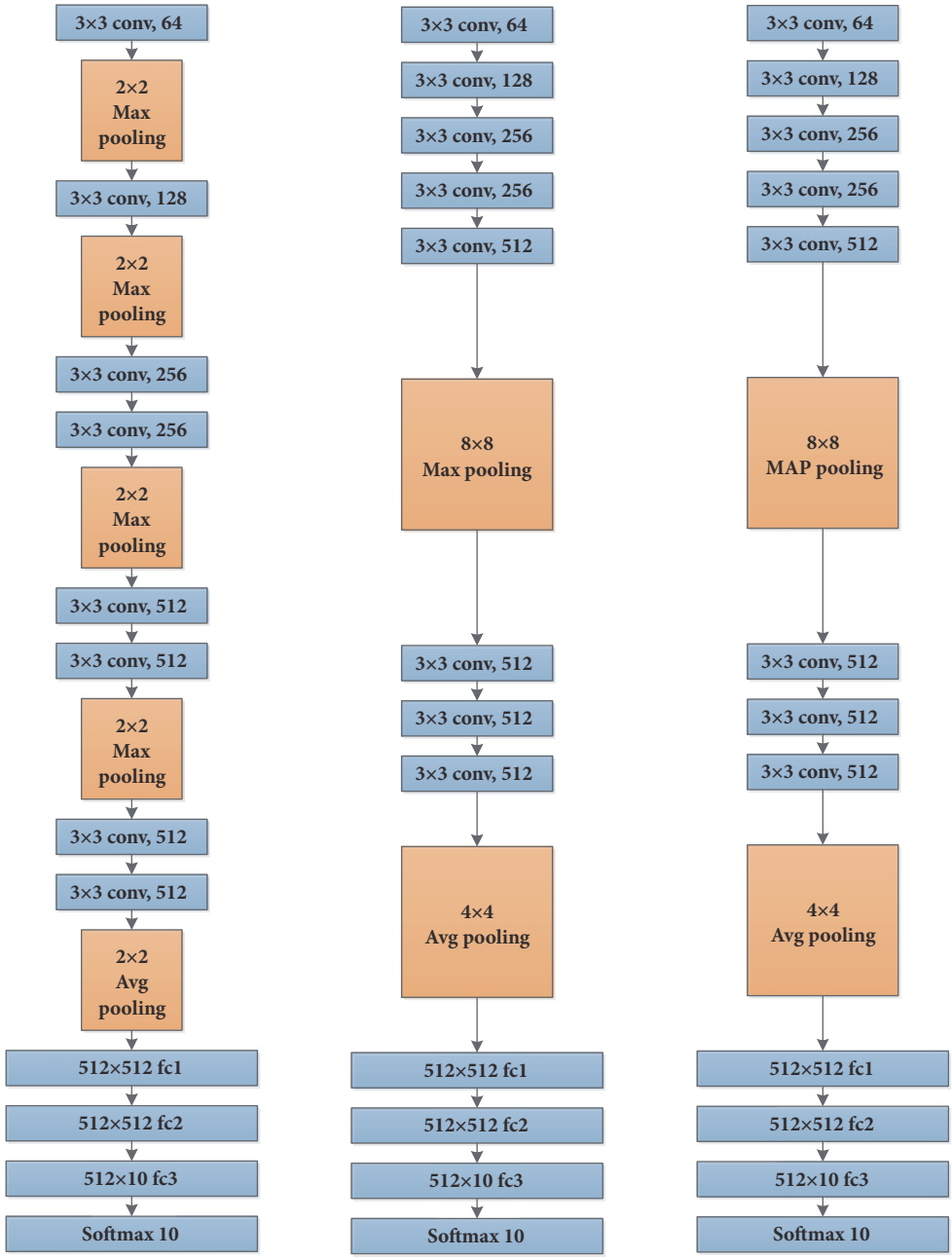

Figure 2: Plain networks for CIFAR-10. **Left:** applying VGG-11 [13] model, which is slightly different from the model designed for ImageNet. **Middle:** applying large-scale pooling (using 8×8 max-pooling to extract lower level features and 4×4 average-pooling for high-level features [36]) in VGG-11 model without increasing its depth and parameters. **Right:** applying MAP method in the network.

and effective. Combined with Dropout, probability weighted pooling [39] method also performs well. Another creative method, Spatial Pyramid Pooling [42] method, uses different size of pooling windows before the fully connected layers to solve the problem of feature loss caused by arbitrary input image size. This method has huge advantage on extracting different active features, which is similar to our idea. In addition, the idea of expanding pooling regions has already been implemented in some networks such as Value Iteration Network (VIN) [43]. VIN uses the large-scale channel-wise max-pooling method to improve the generalization of networks. The idea of mixing max-pooling and average-pooling method [44] has been proven effective. It is similar to our proposed method.

At the bottom of networks, almost every high-level feature in feature maps is useful. Global average-pooling (GAP) [45] method enables all of them to contribute to the final representation. Global max-pooling [46], which focuses on the most active feature in every feature map, is a universal method in object detection tasks. These two existing global pooling methods share a key characteristic: they both concentrate on large-scale pooling size.

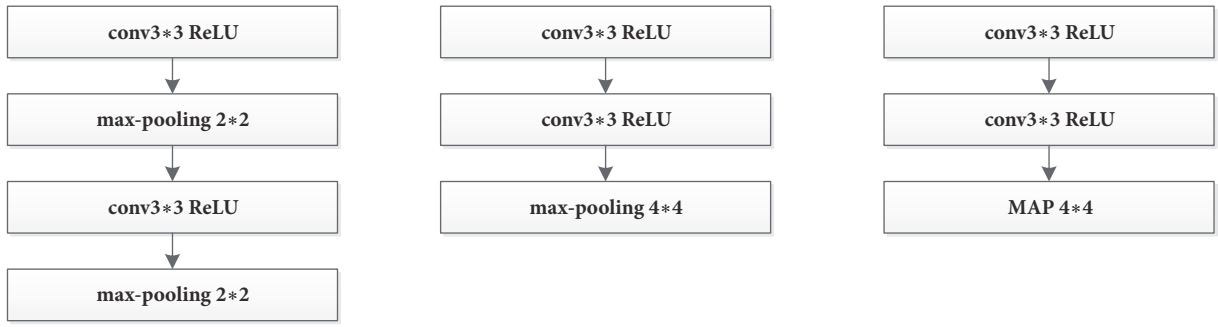

Figure 3: Plain structures used in Transition layers of DenseNets to better extract features. **Left:** using 2×2 max-pooling and downsample twice. **Middle:** using 4×4 max-pooling. **Right:** using 4×4 MAP method.

3. Multiactivation Pooling

3.1. Conventional Pooling Methods. Formally, we denote a single input feature map of a pooling layer as an assemblage $\mathbf{X}$. Before entering the pooling gate, $\mathbf{X}$ can be regarded as a combination of several small local regions [39], $\mathbf{X_0}, \ldots, \mathbf{X_n}$, where n is constrained by both the size of input feature maps and the size of pooling regions.

Consider an arbitrary local region $\mathbf{X_i}$ where $\boldsymbol{i}$ is an index between 0 and n. We denote the size of pooling region as M and each element as x. Here, we define

$$\mathbf{X_i} = (x_1, x_2, \ldots, x_{M\times M}) \tag{1}$$

The conventional pooling methods, max-pooling and average-pooling, make the use of different strategies dealing with the elements in each pooling region. We denote the output after pooling as act. Equation (2) describes max-pooling:

$$act_i = \max_{1\le j\le M\times M} \left(x_j\right) \tag{2}$$

Equation (3) describes average-pooling:

$$act_i = \frac{\mathbf{1}}{M\times M} \sum_{j=1}^{M\times M} x_j \tag{3}$$

Max-pooling method only picks the most active feature in a pooling region. On the contrary, average-pooling method takes all of the features into consideration. Thus, max-pooling method detects more texture information, while average-pooling method preserves more background information.

3.2. Multiactivation Pooling Method. In most cases, 2×2 pooling method without overlapping is frequently used. This means M in (1) is always set to 2. Large-scale pooling means M is set bigger such as 4, 8, and 16, which depends on the input image size. Figure 2 (**middle**) shows the 8×8 max-pooling and 4×4 average-pooling in VGG-11 architecture.

Based on the large pooling region, we propose Multi-activation Pooling method which allows top-k activation to pass through the pooling gate, where k indicates the total number of the picked activation. If $k = M\times M$, it means every activation will contribute to the final output; if $k = 1$, only the max activation can pass through the gate. To avoid mixing too much useless features, k is set small and constrained by the size of pooling region; e.g., k is set to 4, while pooling region is 8×8.

In an arbitrary pooling region $\mathbf{X_i}$, as defined in (1), we denote the lth-picked activation as act_l, where $l \in [1, k]$. Here we have

$$act_l = \max\left(\mathbf{X_i} \ominus \sum_{j=1}^{l-1} act_j\right) \tag{4}$$

where the symbol $\ominus$ means removing elements from an assemblage. The summation symbol in (4) indicates a small set of elements, which contains top1~ $(l-1)$ activation but not adding up all the activation numerically.

After picking out the top-k activation, we do not simply add them together as an output or just compute the average of them. We propose a hyperparameter σ as a constraint factor to multiply the sum of the top-k activation.

The final output refers to

$$\text{output} = \sigma * \sum_{j=1}^{k} act_j \tag{5}$$

Here, the summation symbol means sum operation, where $\sigma \in (0, 1)$. Particularly, if $\sigma = 1/k$, the output is the average value.

The constraint factor, σ, is used to adjust the output. When few features remain active after ReLU gate, i.e., few positive values can move forward, if σ is set too small, positive values will be heavily weakened. On the contrary, when more features remain active, if σ is set more close to 1, output values will be big and easy to distort. In practice, the value of σ is influenced by the value of k. We carry out a lot of experiments and finally get a group of proper combination of σ and k shown in Table 2.

3.3. Network Architectures. We investigate two different plain network architectures in popularity: VGG and ALL-CNN series. We mainly do research on VGG-11 and VGG-13 architectures shown in Table 1. We also pick one model from ALL-CNN series as our baseline model, which is shown in

Table 1: Baseline architectures. Models A and E are two baseline architectures [13] with slightly different number of filters in some layers. Others are architectures with large-scale pooling layers or MAP layers. Conv3 indicates that the filter kernel sizes are 3×3. The ReLU gate is not shown for simplicity. The structures between small-size datasets (CIFAR, SVHN) and large-scale datasets (ImageNet) are different in input layers and fully connected layers. Parameters on the right of the slash belong to large-scale datasets.

Networks architectures based on VGG							
VGG-11					VGG-13		
A	B	C	D	E	F	G	H
Small-size datasets: Input (32×32 RGB image)/Large-scale datasets: Input (224×224 RGB image)							
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
2×2 maxpool	conv3-128	conv3-128	conv3-128	conv3-64	conv3-64	conv3-64	conv3-64
	conv3-128	conv3-128	conv3-128	**2×2 maxpool**	conv3-128	conv3-128	conv3-128
	4×4 maxpool/MAP	conv3-128	conv3-128		conv3-128	conv3-128	conv3-128
conv3-128		conv3-256	conv3-256	conv3-128	conv3-256	conv3-256	conv3-256
2×2 maxpool		**8×8 maxpool/MAP**	conv3-256	conv3-128	**4×4 maxpool/MAP**	conv3-256	conv3-256
			16×16 maxpool/MAP	**2×2 maxpool**		conv3-256	conv3-256
						8×8 maxpool/MAP	conv3-512
conv3-256				conv3-256			**16×16 maxpool/MAP**
conv3-256				conv3-256			
2×2 maxpool				**2×2 maxpool**			
	conv3-128				conv3-256		
	conv3-256				conv3-256		
	conv3-256				conv3-512		
conv3-512	**4×4 maxpool/MAP**			conv3-512	**4×4 maxpool/MAP**		
conv3-512				conv3-512			
2×2 maxpool		conv3-256		**2×2 maxpool**		conv3-512	
		conv3-512				conv3-512	
		conv3-512				conv3-512	
conv3-512	conv3-512	**4×4 avgpool**	conv3-512	conv3-512	conv3-512	**4×4 avgpool**	conv3-512
conv3-512	conv3-512		conv3-512	conv3-512	conv3-512		conv3-512
2×2 avgpool	**2×2 avgpool**		**2×2 avgpool**	**2×2 avgpool**	**2×2 avgpool**		**2×2 avgpool**
fc1-512×512/25088×4096							
fc2-512×512/4096×4096							
fc3-512×10/4096×1000							
Softmax-10/1000							

Table 2: The optimal combinations of σ and k under different pooling size.

pooling size	4×4	8×8	16×16
(k, σ)	(3, 0.5)	(4, 0.5)	(8, 0.25)

Table 5. For network with shortcuts structure, we pick one model from DenseNets series as our baseline model, which is shown in Table 7.

VGG. The series of VGG architectures first adopt small size (3×3) receptive field and use large number of filters (512) with much more trainable parameters in each convolutional layers. VGG architectures also put more convolutional layers together before pooling layers, which better extract features.

ALL-CNN. ALL-CNN first uses convolutional layers to subsample the feature maps. Compared to VGG, ALL-CNN are designed with less trainable parameters.

DenseNets. DenseNets are typical deep convolutional neural networks with shortcuts structure. The architecture alternatively uses Dense Blocks and Transition layers to extract features [15]. In every Transition layer, structures such as "BN-ReLU-conv1-avgpool (2×2)" are adopted to subsample the features.

3.4. Implementation. On VGG nets, all convolutional layers have equal kernel size 3×3 together with the same padding strategy, zero-padded by one pixel. On ALL-CNN, 1×1 convolutional layers are also included. When applying large-scale pooling method or MAP to certain architectures, total number of layers and trainable parameters are kept the same as baseline models. Before the fully connected layers and softmax classifier, average-pooling method is adopted in every model. When dealing with two key hyperparameters, σ and k, we use different combinations according to different pooling size. Table 2 shows the optimal combinations in experiment.

4. Experimental Results

4.1. Datasets

CIFAR. The two benchmark datasets, CIFAR-10 and CIFAR-100 [31], are consisted of tiny RGB images with 32×32 pixels. Both of the two datasets contain 50000 training images and 10000 testing images. Before training, we adopt a standard data normalization method [23] using the channel means and standard deviations. For training data augmentation, we randomly apply horizontal flip and crop on 4 pixels padded images. Finally, we evaluate the classification result by the test error rate.

SVHN. The Street View House Number datasets is a real-world image dataset obtained from house numbers in Google Street View images, which contains 10 classes, one for each digit. The dataset contains 73257 digits for training, 26032 digits for testing, and 531131 additional, somewhat less difficult samples, to use as extra training data. The images in SVHN are in two formats: original images and MNIST-like images with 32×32 pixels, and we choose the latter in our experiments.

ImageNet. The ILSVRC 2012 classification dataset [47, 48] consists of 1.2 million images for training and 50,000 for validation, 1000 classes in total. We adopt the same data augmentation scheme for training images as in [16, 17] and apply a single-crop or 10-crop with size 224×224 at test. Our experiment is only based on a few models due to the limit of computing speed because of the lack of GPU device.

4.2. Training Details. For every network, we use stochastic gradient descent (SGD) with momentum of 0.9 and weight decay of 0.0005 in backpropagation process. The initial learning rate is set to 0.05. The batch size is set to 128 and the images in every batch are shuffled every epoch. We do not use BN [23] and LRN [11] in every layer of our networks in order to keep the sparsity of feature maps. In fully connected layers, 50% dropout rate is adopted. For small-size datasets (CIFAR and SVHN), the total number of epochs is set to 150 and learning rate will be divided by 10 every 45 epochs.

For ImageNet, we train models for 100 epochs with the batch size of 128. The learning rate is set to 0.1 initially and is lowered by 10 times at epoch 40 and 80. We only experiment this large-scale datasets on a few model of VGG-11 series (Table 1). Due to different baseline models, some training details will be slightly adjusted.

4.3. Classification on VGG. To evaluate the efficiency of our MAP method, we train the networks shown in Table 1 on VGG architectures. We evaluate top-1 error rates and show the result in Table 3.

Large-Scale Max-Pooling Method. When the depth of layers and the number of parameters keep unchanged, piling up more convolutional layers together with large-scale pooling area extract features better. Based on this theory, we first reconstruct the two VGG architectures and then replace the original 2×2 pooling layers with larger-scale pooling layers. As is shown in Table 1, model A and model E are two baseline models. Model B and model F use 4×4 pooling layers. Model C and Model G use 8×8 pooling layers, while model D and model H apply even larger pooling size 16×16. Increasing the size of pooling layers leads the number of pooling layers to decrease. The result in second row of Table 3 shows that large-scale max-pooling method decreases error rate.

MAP Method. Table 3 shows that large-scale max-pooling method helps improve the classification accuracy. Then, we keep using the large pooling size, but instead of using the max-pooling method, we replace it with MAP method. The results from the last row of Table 3 (both **top** and **bottom**) are the most noticeable. First, with MAP method, the lowest error rate based on VGG-11 architecture is 6.94%, dropping around 20% compared to the original error rate of baseline model.

Table 3: Error rates in the test dataset of CIFAR-10. **Top:** error rates based on VGG-11 model and its extension models. **Bottom:** error rates based on VGG-13 model and its extension models.

		VGG-11 error rate (%)	
Baseline model(model A)		8.72	
With large-scale pooling method	4×4 maxpool (model B)	8×8 maxpool (model C)	16×16 maxpool (model D)
	8.24	7.84	8.19
With MAP method	4×4 MAP (model B)	8×8 MAP (model C)	16×16 MAP (model D)
	8.07	**6.94**	7.76
		VGG-13 error rate (%)	
Baseline model(model E)		7.50	
With large-scale pooling method	4×4 maxpool (model F)	8×8 maxpool (model G)	16×16 maxpool (model H)
	7.08	7.22	7.35
With MAP method	4×4 MAP (model F)	8×8 MAP (model G)	16×16 MAP (model H)
	6.96	**6.85**	7.07

Table 4: Error rates in the test dataset of CIFAR-100, SVHN, and ImageNet on VGG models.

Models	CIFAR-100 (%)	SVHN (%)	ImageNet top-5 (%)
Baseline model(A)	30.96	3.04	10.86
With 4×4 large-scale pooling method(B)	30.37	2.87	-
With 4×4 MAP method(B)	30.04	2.44	-
With 8×8 large-scale pooling method(C)	29.45	2.36	10.54
With 8×8 MAP method(C)	**28.37**	**2.03**	**10.19**
Baseline model(E)	29.05	2.27	-
With 4×4 large-scale pooling method(F)	28.62	2.14	-
With 4×4 MAP method(F)	27.92	1.98	-
With 8×8 large-scale pooling method(G)	28.03	2.05	-
With 8×8 MAP method(G)	**27.4**	**1.89**	-

We get similar result when the baseline model is VGG-13. The error rate also drops when integrating the MAP method into models. Second, if we analyze the result by column, it is clear that models with MAP layers perform better than models with max-pooling layers. We also try average-pooling method. Most of the results are even poorer than the results of baseline model.

Noticeably, the relationship between pooling size and error rate is not monotonic. Through experiment, we find that blindly increasing the pooling size is of no use.

On CIFAR-100 datasets, models with MAP method work much better than baseline models. For the reason that 8×8 MAP method is the strongest (Table 3), we pick 2 models with this method as comparison. Results are shown in Table 4. When integrating MAP method in VGG-11, the error rate decreases around 2.5%. In VGG-13, the error rate comes to 1.5%. With MAP method, model C performs even better than model E that has two more layers and more trainable parameters. On SVHN datasets, model with 8×8 MAP method is also the strongest as Table 3 shows. The classification results prove that MAP method with large pooling kernel size can still work well, while the number of classes are larger and the training images of each class are fewer.

4.4. Classification on ALL-CNN. Through changing the stride from 1 to 2, convolutional layers are able to subsample as well as pooling layers. Based on ALL-CNN, we first rearranged the convolutional layers and then integrated large-scale max-pooling and MAP method into the model (Table 5). We only experimented on CIFAR-10 datasets and got exciting classification results in Table 6. It shows that the architecture with 8×8 MAP method lowers the error rate to 7.11%, which outperforms the result (8.07%) of our baseline model and even a little better than the result in their paper (7.25%).

4.5. Classification on DenseNets. To integrate our MAP method into DenseNets, we change the original structure into "BN-ReLU-[conv3-ReLU] ×3-MAP (4×4)". As is shown in Table 7, DenseNet-40_MAP is the model with MAP method Based on DenseNet-40 with L = 40, k = 12, $\theta = 1$ [15].

Table 5: The baseline ALL-CNN architecture [24] and our modified conv-maxpool/MAP architectures.

conv-maxpool/MAP		All-CNN
Input (32×32 RGB image)		
conv3-96	conv3-96	conv3-96
conv3-96	conv3-96	conv3-96
conv3-96	conv3-96	
maxpool/MAP 4×4	conv3-192	conv3-96 stride=2
conv3-192	conv3-192	conv3-192
conv3-192		conv3-192
maxpool/MAP 4×4	**maxpool/MAP 8×8**	conv3-192 stride=2
conv3-192	conv3-192	conv3-192
conv3-192	conv3-192	conv1-192
		conv1-10
avgpool 2×2	avgpool 4×4	avgpool 6×6
fc1 192×192	fc1 192×192	Softmax-10
fc2 192×10	fc2 192×10	
Softmax-10	Softmax-10	

Table 6: Error rates of the baseline model ALL-CNN and its derived models on CIFAR-10 datasets.

Method	Error rate (%)	Params
ALL-CNN [24]	7.25	1.35M
ALL-CNN_ours	8.17	1.35M
conv-maxpool 4×4	8.79	1.35M
conv-maxpool 8×8	7.91	1.35M
conv-MAP 4×4	8.03	1.35M
conv-MAP 8×8	**7.11**	1.35M

Table 7: The baseline DenseNet-40 architecture [15] and our DenseNet-40_MAP architectures. Two models are both designed for CIFAR-10 datasets.

Layers	Densenet-40	Densenet-40_MAP
Convolution	conv3 stride 1 padding 1	
Dense Block(1)	[BN-ReLU-conv3] ×12	
Transition Layer(1)	[BN-ReLU] conv1 avgpool 2×2	[BN-ReLU] **[conv3-ReLU] ×3** **MAP 4×4**
Dense Block(2)	[BN-ReLU-conv3] × 12	
Transition Layer(2)	[BN-ReLU] conv1 avgpool 2×2	[BN-ReLU] **[conv3-ReLU] ×3** **MAP 4×4**
Dense Block(3)	[BN-ReLU-conv3] × 12	
Classification Layer	avgpool 8×8 10D fully-connected, softmax	avgpool 2×2

We used the code implemented on Pytorch at https://github.com/liuzhuang13/DenseNet. When training the two models, the initial learning rate is set to 0.1 and is divided by 10 at 50% and 75% of total 200 epochs [15]. The batch size is 64. The classification results are shown in Table 8. The model with plain structure (Figure 3, **right**) achieves lower error rate

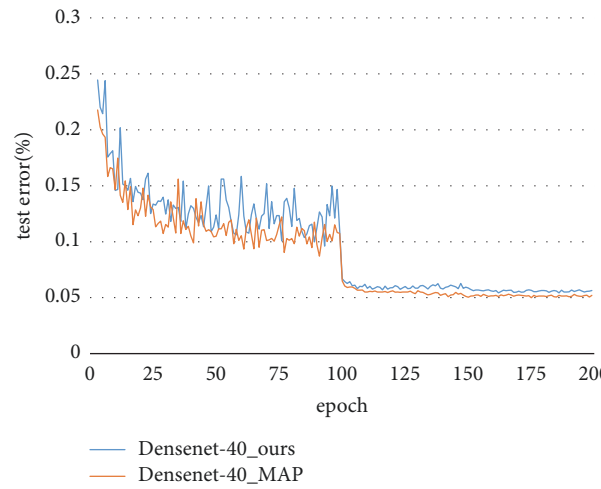

Figure 4: Testing error rate curves of DenseNet-40 and DenseNet-40_MAP on CIFAR-10.

on every small-size datasets (4.99% for CIFAR-10, 24.03% for CIFAR-100, 1.68% for SVHN) than baseline model. CIFAR-10 testing error rate curves show the strong performance of DenseNet-40_MAP (Figure 4) more clearly.

In plain networks, the basic structure like "conv3-ReLU-MAP" (Figure 3) is proved very effective on extracting features. In DenseNets, we integrate the structure into Transition layers to deal with output feature maps from Dense Blocks and classification results are encouraging.

4.6. Classification on ImageNet. We only used this large-scale datasets on model A and model C in our experiment. The results are shown in Table 4. Compared to VGG-11 baseline model, VGG-11 model with 8×8 MAP method lowers the top-5 error rate from 10.86% to 10.19%. Although limited results are listed, our results on large-scale datasets are still encouraging.

Table 8: Error rates in the test dataset of CIFAR and SVHN. The first row shows the error rate of the baseline model DenseNet-40 from paper [15]. The second row shows the error rate of our implement on the baseline model DenseNet-40. The third row shows the error rate of our DenseNet-40_MAP model in Table 7.

Method	CIFAR-10 (%)	CIFAR-100 (%)	SVHN (%)	Params
Densenet-40 [15]	5.24	24.42	1.79	1.05M
Densenet-40_ours	5.43	24.91	1.92	1.05M
Densenet-40_MAP	**4.99**	**24.03**	**1.68**	4.32M

Table 9: The four architectures are designed with fewer convolutional layers for CIFAR-10 datasets. Model A0 uses 2×2 max-pooling layer and 2×2 average-pooling layer. Model B0 adopts 8×8 MAP method. Model C0 uses 1×1 convolutional layer to replace the 3×3 convolutional layer right after MAP layer. Model D0 reduces one fully layer based on model C0. ReLU gates after each convolutional layer are not shown for simplicity.

VGG-9			
A0	B0	C0	D0
Input (32×32 RGB image)			
conv3-64 maxpool 2×2	conv3-64 conv3-128	conv3-64 conv3-128	conv3-64 conv3-128
conv3-128 maxpool 2×2	conv3-128 conv3-256	conv3-128 conv3-256	conv3-128 conv3-256
conv3-128 maxpool 2×2	**MAP 8×8**	**MAP 8×8**	**MAP 8×8**
conv3-256 maxpool 2×2	conv3-256 conv3-512	**conv1-256** conv3-256	**conv1-256** conv3-256
conv3-256 conv3-512 avgpool 2×2	avgpool 4×4	avgpool 4×4	avgpool 4×4
fc1 512×512		fc1 256×256	fc1 256×256
fc2 512×512		fc2 256×256	fc2 256×10
fc3 512×10		fc3 256×10	-
Softmax-10			

4.7. Classification on Compact Models

Model Complexity Analysis. We evaluate the model complexity mainly in several perspectives. First, we compare the number of parameters among VGG models. From Table 10, it can be seen that the two series model B-D and model F-H both have fewer trainable parameters than their baseline models. Compared to baseline model A, model B-D series cut trainable parameters from 9.8M to 5.4M. To achieve that, we simply reduce the number of filters in some convolutional layers. It is obvious that models with large-scale max-pooling method or MAP method show strong performance even when the trainable parameters are fewer. Second, each of the three models such as model B-D has the same number of trainable parameters and depth. MAP method is better than large-scale max-pooling method under the same condition.

All of our test models are simply composed of only three kinds of different layers: convolutional layers, pooling layers, and fully connected layers without adding any additional layers like BN [23] and LRN [11]. In addition, all of our searched networks are not that deep. It saves a lot of memory during training and it is easy to implement.

Compact Model Classification. Based on models with same number of trainable parameters and depth, we have proven our proposed method more effective. For further research, we keep reducing the number of layers. Based on model A, we drop two more convolutional layers and name the model series VGG-9 which is shown in Table 9. By comparing the classification results of model A, A0, and B on CIFAR-10 datasets (Figure 5), we conclude that the test error rate of model B0 that has MAP layers achieves 8.59%, even lower than the result of model A which has more layers and trainable parameters.

Table 10: The number of trainable parameters and depth in different VGG models are shown in Table 1. "Conv-D" indicates the number of convolutional layers and "FC-D" indicates the number of fully connected layers.

Models	Params	Depth	Conv-D	FC-D
A	9.8M	11	8	3
B-D	5.4M	11	8	3
E	9.9M	13	10	3
F-H	8.2M	13	10	3
A0	2.9M	9	6	3
B0	2.9M	9	6	3
C0	1.3M	9	6	3
C0+shortcuts	1.3M	9	6	3
D0	1.2M	8	6	2

Compared to model A, model B0 cuts down nearly 70% of the trainable parameters from 9.8M to 2.9M. However, 2.9M is still a large number. Therefore, more compact models with smaller amount of parameters are proposed in Table 10. Inspired by bottleneck structure [15], we choose to use 1×1 filters instead of 3×3 on the first convolutional layer locating after MAP layer in model C0. Compared to model B0, model C0 achieves better result (Figure 5) with more than 50% decline of trainable parameter (Table 10). Based on this encouraging result, we continue reducing one fully connected

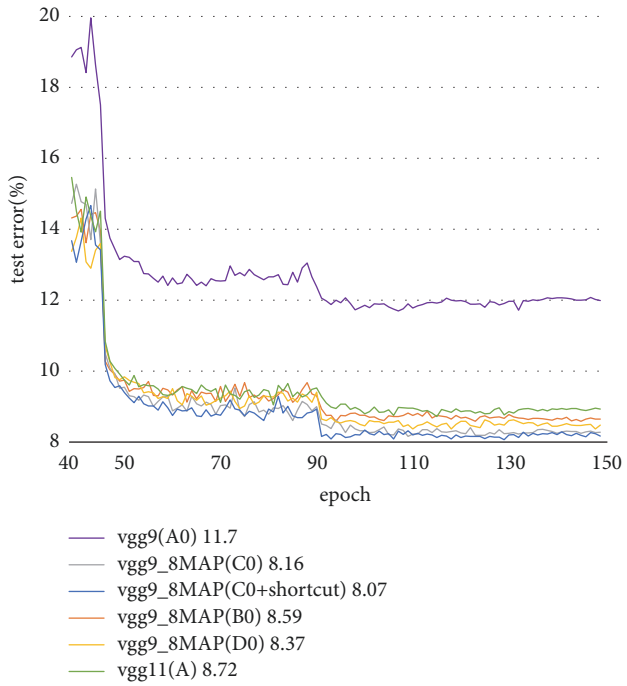

Figure 5: Testing error rate curves on CIFAR-10 of model A0 (vgg9), model B0 (vgg9_8MAP), model A (vgg11), model C0, model D0, and model C0 with shortcut structure. Values after each model name represent the lowest error rate (%).

layer and achieve close result to model C0. Both model C0 and D0 perform better than baseline model A with only 12% of the trainable parameters.

However, using the MAP method without increasing the number of parameters and layers cannot avoid increasing the computational complexity during training process, because more filters are forced to slide on bigger features maps.

5. Discussion

5.1. MAP Method Analysis

Sparsity Analysis. In CNNs, smaller receptive field size is proved more effective in most cases. Researches also show that piling up more convolutional layers extract features better. If basic blocks like "conv3-ReLU" are put together, the networks will be able to do multilevel template matching where negative values are set to zero by ReLU. Output feature maps are very sparse and activation left is quite important.

Figure 6 shows the output feature maps in different convolutional layers. Model A-C adopts different pooling size and the arrangement of the convolutional layers is different either. From left to right, the feature maps become more and more sparse. Theoretically, average-pooling is not suitable in sparse feature maps because useful activation will be heavily diluted. From top to bottom, we find that activation distributes densely in small local parts. So simply using max-pooling method may cause the loss of some good activation. Our MAP method finds a way to balance feature lost and feature dilution, which performs very well.

Visualization with Deconvolution. Deconvolution can be thought as a model that uses the same components (filtering, pooling) but in reverse. Here, deconvolution is used on our pretrained CNN model C and we use this method to map multiactivation back to the input pixel space, showing what input pattern eventually cause given activation in the feature maps. [36].

In this paper, we use basic structure (Figure 3 right) in our CNN models to thinning the feature maps. Every feature map in different convolutional layers can be regarded as a kind of representation of the original image. In other words, every feature map reflects characteristics of an original image. Through deconvolution, the reflection maps are visualized. In Figure 7, the deconvolution results on CIFAR-10 are shown. We use pretrain model C (Table 1), which contains an 8×8 pooling layers. There are 5 convolutional layers piling up before the large kernel-size-pooling layers. The channel numbers of each layer are 64, 128, 128, 128, and 256. As is shown in Figure 7, features maps become more and more sparse from the first convolutional layer to the fifth convolutional layer. Sparse feature maps often reflect the remarkable local characteristics of an image. In this way, our models can focus on different local parts specifically owned by different class of images, which represent raw images better.

5.2. Model Structure Analysis.

On plain networks and networks with shortcuts structure, MAP layers are always combined with plain structure shown in Figure 3. Through the analysis of feature maps sparsity and deconvolution, we believe that the plain structure can provide good activation and MAP method can make good use of this activation. In addition, the size of feature maps after 8×8 MAP layer becomes small (4×4 on CIFAR-10; 28×28 on ImageNet). We think that each activated feature (positive value in feature maps) on small feature maps is relatively independent. Therefore, trying to find the relationship between activated features in a 3×3 area may be unnecessary. Through experiment, we prove that 1×1 convolution can better represent the features picking out from MAP layer. At the same time, the trainable parameters are reduced a lot.

Since the MAP method can provide such good activation, we find a good way to take better advantage of this activation. Through mixing the plain structure and shortcuts structure, the classification results improve again! In Figure 8, the shortcuts structure is used to sum the output from MAP layer and the output from the 1×1 convolutional layer. We apply this structure to model C0 (Table 9) and name the new model "C0+shortcuts". All the parameters are keeping the same. The lowest error rate decreases from 8.16% to 8.07%. Noticeably, in Figure 5, the error rate curve of the new model lies beneath all the other curves, indicating that the overall capability of the new model is superior when fully trained.

5.3. Comparison between MAP and Mixing Pooling.

Mixing pooling method [44] mainly contributes to doing a trade-off between max-pooling and average-pooling. Both our method and theirs are based on a hypothesis that only using max-pooling method or average-pooling method cannot

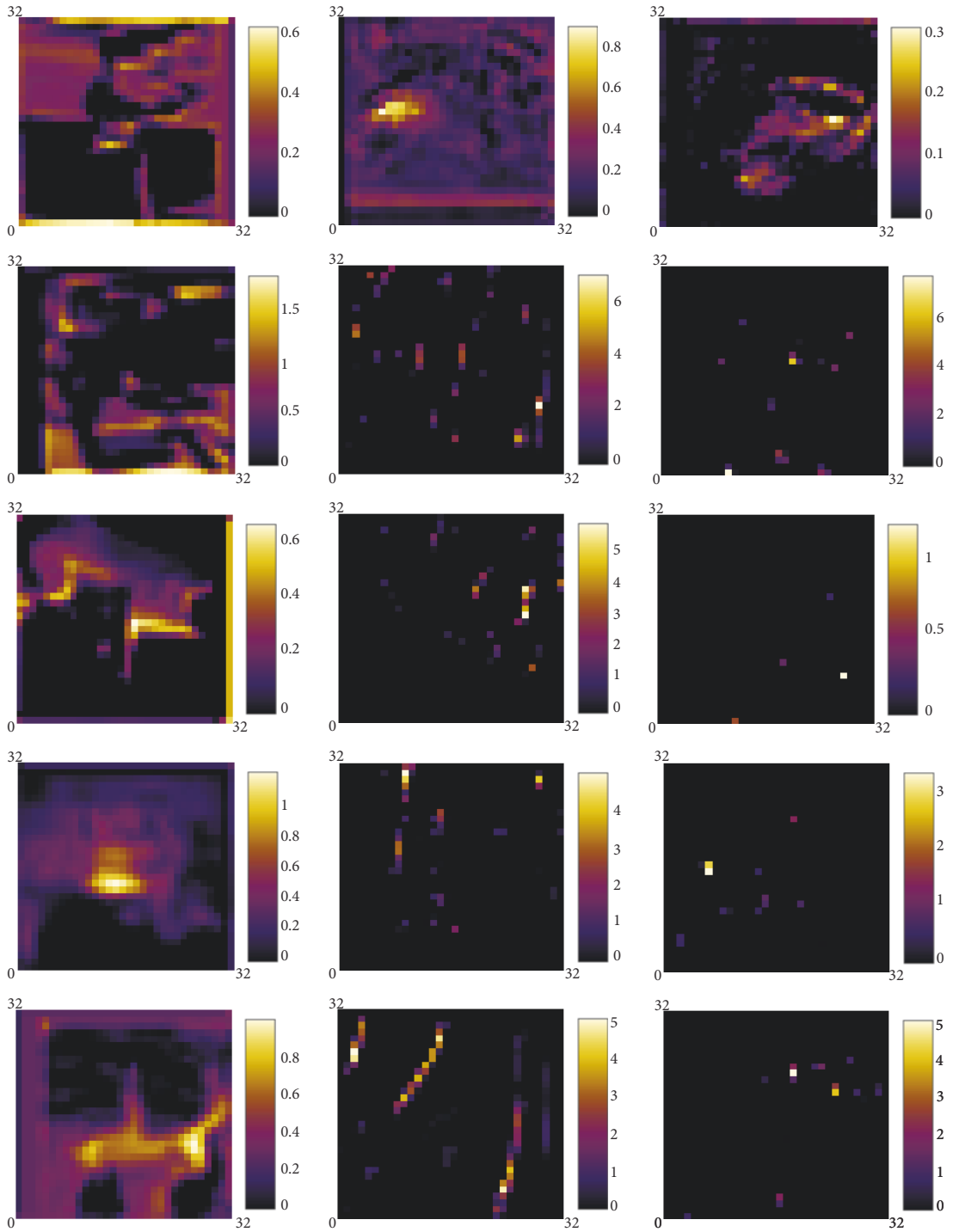

FIGURE 6: Each image indicates a certain convolutional layer's output feature map. Each column consists of 5 feature maps, which are extracted during training process of 3 different VGG-11 models. From top to bottom, the feature maps are extracted after training 1, 30, 60, 90, and 120 epochs, respectively. From left to right, the output feature maps are randomly picked from different convolutional layers, which locate right before the first pooling layer in each model (**left column:** the first convolutional layers in model A; **middle column:** the third convolutional layer in model B; and **right column:** the fifth convolutional layer in model C). Therefore, the kernel size of all maps is 32×32. The colour bar on the right of each map indicates the pixel intensity; i.e., the brighter the pixel colour, the bigger the pixel value. In addition, black corresponds to value 0.

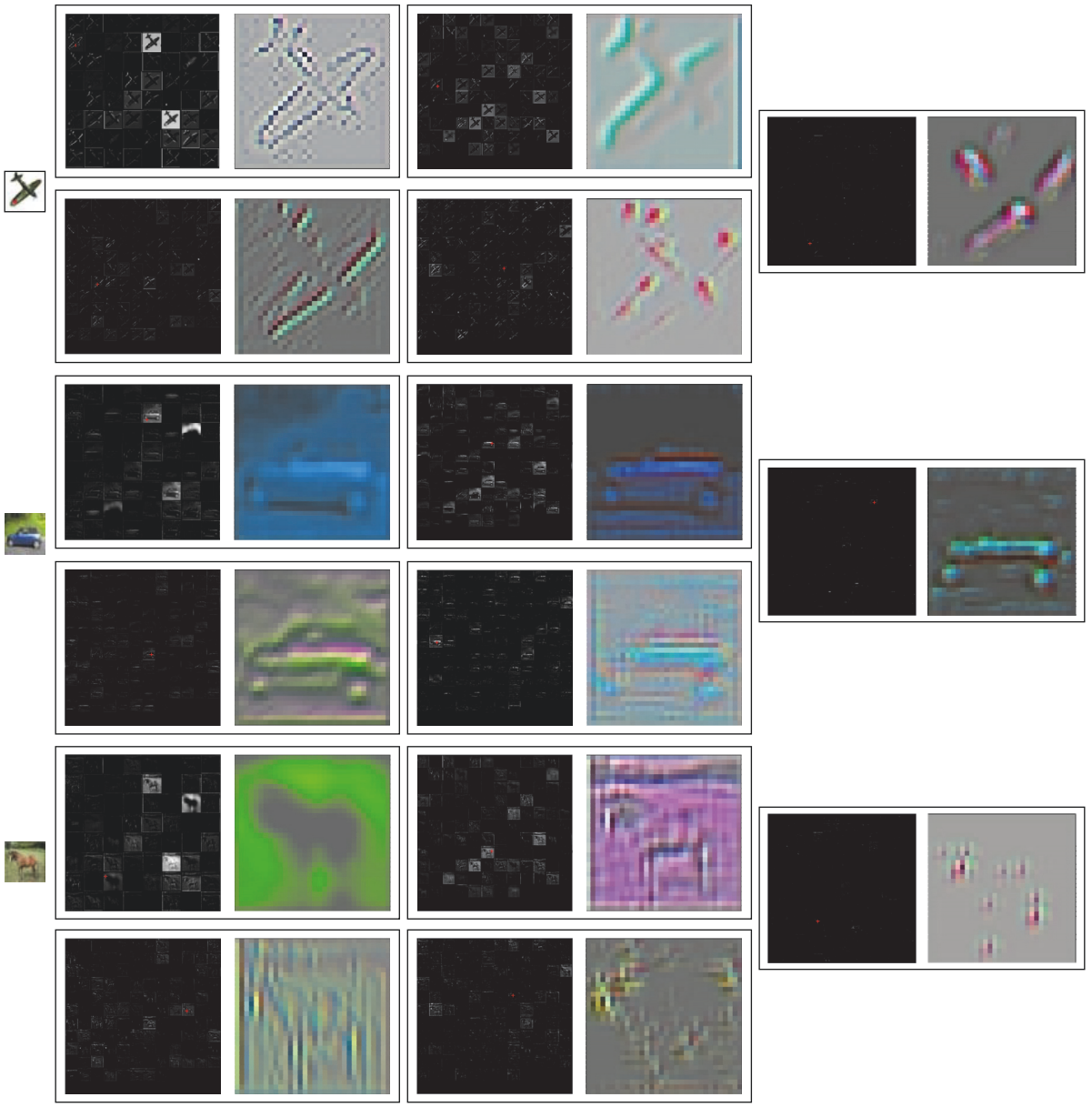

Figure 7: The left most three images are raw images from CIFAR-10 datasets. The first image is from class 0 (plane). The second image is from class 1 (car) and the third image is from class 7 (horse). Each raw image has 5 pairs of images on the right. The left image in each pair contains all channels' output feature maps from its corresponding convolutional layer. The right image in each pair is the reflection map of one feature map casually selected from all channels. The upper-left corner pairs are extracted from the first convolutional layer. The upper-right corner, lower-left, lower right, and the very right are extracted from the second, third, fourth, and fifth convolutional layer, respectively. In feature maps, black corresponds to 0 pixel value.

produce optimal results. However, MAP and mix pooling are quite different in the following aspects.

First, the novelty of mixing pooling method is bringing learning into the pooling operation. Two main expressions are

$$output = \alpha * \max(\boldsymbol{x}) + (1 - \alpha) * avg(\boldsymbol{x}) \tag{6}$$

$$output = \sigma\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}\right) * \max(\boldsymbol{x}) + \left(1 - \sigma\left(\boldsymbol{w}^{T}\boldsymbol{x}\right)\right) * avg(\boldsymbol{x}) \tag{7}$$

where $\boldsymbol{x}$ denotes the values in the region being pooled; $\alpha \in [0, 1]$ is a scalar specifying a certain combination of max and average; $\boldsymbol{w}$ denotes the values in a "gating mask". $\sigma(\boldsymbol{w}^{T}\boldsymbol{x}) = 1/(1 + \exp(-\boldsymbol{w}^{T}\boldsymbol{x}))\epsilon[0, 1]$ is a sigmoid function. α and $\boldsymbol{w}$ are trainable.

Obviously, mixing pooling method combines max-pooling and average-pooling and constrains them with trainable factors. However, our MAP method only chooses top-k activation in a pooling region, which means most of the features are ignored.

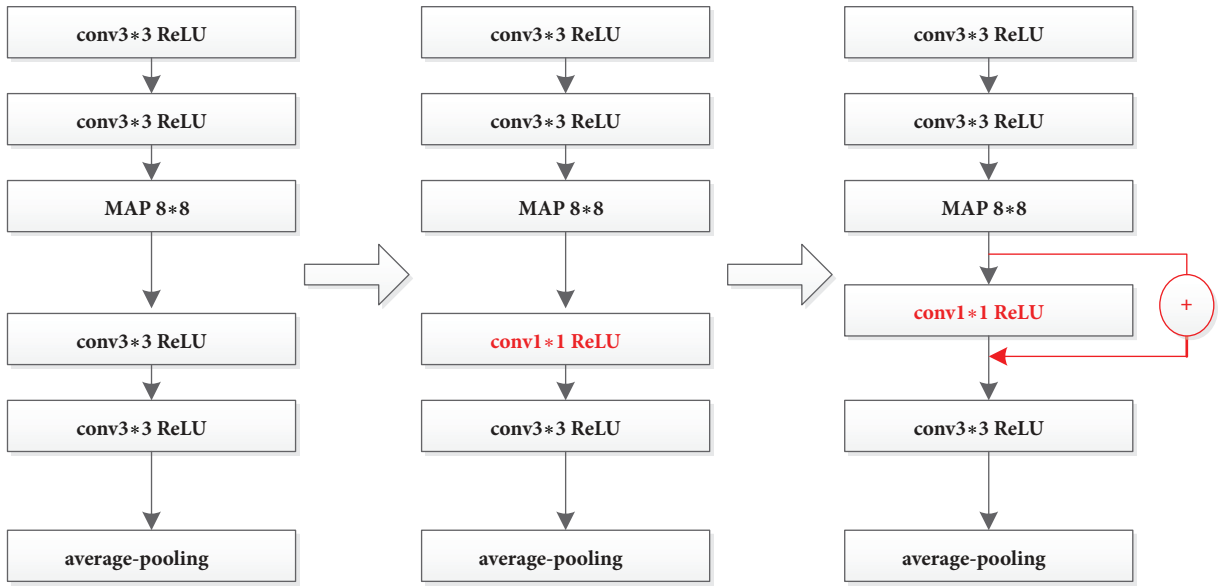

Figure 8: Plain structure and plain structure mixing with shortcuts structure.

Table 11: Different CIFAR-10 classification results with different combinations of k and σ based on VGG-11. The left two column results are based on the modelusing 4×4 MAP method. The right two column results are based on the model using 8×8 MAP method.

(k, σ)	result (%)	(k, σ)	result (%)
(1, 1)	8.24	(1,1)	7.84
(3, 0.5)	**8.07**	(4, 0.5)	**6.94**
(3, 0.75)	8.21	(4, 0.75)	7.14
(3, 0.33)	8.16	(4, 0.25)	7.07
(4, 0.5)	8.25	(6, 0.5)	7.15
(4, 0.25)	8.31	(6, 0.17)	7.33
(6, 0.5)	8.40	(8, 0.5)	7.57
(8, 0.5)	8.51	(8, 0.125)	7.63

Second, our MAP method is working on large pooling kernel size and takes good use of the sparsity of feature maps, while mixing pooling method does not take feature maps pattern and pooling kernel size into consideration.

Third, MAP method does not need more layers and trainable parameters. It can even achieve higher performance with fewer trainable parameters. On the contrary, mixing pooling method introduces trainable parameters into pooling layers to help improve model capacity.

5.4. Hyperparameter Analysis. In Table 2, the optimal combinations of σ and k under different pooling size are listed. Instead of arbitrarily choosing these two hyperparameters, we picked the best combination of σ and k through a great amount of experiments. The CIFAR-10 classification results with different hyperparameters based on VGG-11 are shown in Table 11. The first row of Table 11 corresponds to the max-pooling result and the fourth row corresponds to the average-pooling result. On large sparse feature maps, only choosing one or two features ignores quite big number of active features. On the contrary, choosing too much features adds a lot of useless closed features, which leads to feature dilution.

6. Conclusion

In this paper, we propose a new pooling method, which refers to Multiactivation Pooling (MAP) method. It provides a new subsampling idea to better pick features. We apply our method on convolutional neural networks. In our experiment, MAP method can highly improve the classification accuracy in CNNs. Actually, the plain structure "conv-ReLU-MAP" is a better feature-extractor in CNNs. Based on this structure, we also prove the feasibility of applying MAP method on DenseNets. Moreover, we carry out additional experiments such as sparsity analysis and deconvolution to illustrate that the plain structure can provide "good" activation.

We hope to study further and try to explore the potential of the basic layers. MAP method still has huge space for development. It may perform better by carefully fine-tuning hyperparameters, σ and k. In the future, we plan to study further on MAP method and test it on different datasets and different networks.

Data Availability

All the datasets in this paper are available on the Internet. Previously reported CIFAR (CIFAR-10 and CIFAR-100) data were used to support this study and are available at http://www.cs.toronto.edu/~kriz/cifar.html. Previously reported SVHN data were used to support this study and are available at http://ufldl.stanford.edu/housenumbers/. The ImageNet (ILSVRC2012) data used to support the findings of this study were supplied by http://image-net.org/download-images

under license and so cannot be made freely available. Signing up at http://image-net.org/download-images should be made for data availability.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

All the authors are supported by the National Natural Science Foundation of China (Grant 61772052).

References

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[2] L. Zhao and K. Jia, "Multiscale CNNs for Brain Tumor Segmentation and Diagnosis," *Computational and Mathematical Methods in Medicine*, vol. 2016, Article ID 8356294, 2016.

[3] Y. Xu, G. Yu, Y. Wang, X. Wu, and Y. Ma, "Car detection from low-altitude UAV imagery with the faster R-CNN," *Journal of Advanced Transportation*, vol. 2017, Article ID 2823617, 11 pages, 2017.

[4] J. Shotton, J. Winn, C. Rother, and A. Criminisi, "TextonBoost for image understanding: multi-class object recognition and segmentation by jointly modeling texture, layout, and context," *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–23, 2009.

[5] P. Sermanet, D. Eigen, X. Zhang et al., *Overfeat: Integrated recognition, localization and detection using convolutional networks*, 2013, arXiv:1312.6229.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 779–788, July 2016.

[7] T. N. Sainath, B. Kingsbury, G. Saon et al., "Deep Convolutional Neural Networks for Large-scale Speech Tasks," *Neural Networks*, vol. 64, pp. 39–48, 2015.

[8] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[9] E. Ribeiro, A. Uhl, G. Wimmer, and M. Häfner, "Exploring deep learning and transfer learning for colonic polyp classification," *Computational and Mathematical Methods in Medicine*, vol. 2016, Article ID 6584725, 16 pages, 2016.

[10] Z. Lv, H. Song, P. Basanta-Val, A. Steed, and M. Jo, "Next-Generation Big Data Analytics: State of the Art, Challenges, and Future Research Topics," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1891–1899, 2017.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[12] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[13] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014, https://arxiv.org/abs/1409.1556.

[14] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[15] G. Huang, Z. Liu, L. v. Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, Honolulu, HI, July 2017.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 770–778, July 2016.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the European Conference on Computer Vision*, pp. 630–645, 2016.

[18] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Proceedings of the 29th Annual Conference on Neural Information Processing Systems, NIPS 2015*, pp. 2377–2385, can, December 2015.

[19] G. Larsson, M. Maire, and G. Shakhnarovich, *Fractalnet: Ultra-deep neural networks without residuals*, 2016, https://arxiv.org/abs/1605.07648.

[20] Y. Yang, H. Li, Y. Han, and H. Gu, "High resolution remote sensing image segmentation based on graph theory and fractal net evolution approach," in *Proceedings of the 2015 ISPRS International Workshop on Image and Data Fusion, IWIDF 2015*, pp. 197–201, usa, July 2015.

[21] H. Lei, T. Han, F. Zhou et al., "A deeply supervised residual network for HEp-2 cell classification via cross-modal transfer learning," *Pattern Recognition*, vol. 79, pp. 290–302, 2018.

[22] Y. LeCun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML '15)*, pp. 448–456, July 2015.

[24] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, *Striving for simplicity: The all convolutional net*, 2014.

[25] N. Cohen, O. Sharir, and A. Shashua, "Deep SimNets," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 4782–4791, usa, July 2016.

[26] F. N. Iandola, M. W. Moskewicz, K. Ashraf et al., *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 MB model size*, 2016, arXiv:1602.07360.

[27] A. G. Howard, M. Zhu, B. Chen et al., *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017, arXiv:1704.04861.

[28] X. Zhang, X. Zhou, M. Lin, and J. Sun, *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*, 2017, arXiv:1707.01083.

[29] Z. Li, X. Wang, X. Lv, and T. Yang, *SEP-Nets: Small and Effective Pattern Networks*, 2017, 1706.03912.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve Restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML '10)*, pp. 807–814, Haifa, Israel, June 2010.

[31] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[32] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex," *The Journal of Physiology*, vol. 160, pp. 106–154, 1962.

[33] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[34] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[35] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, vol. 1, pp. 886–893, June 2005.

[36] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part I*, vol. 8689 of *Lecture Notes in Computer Science*, pp. 818–833, Springer, 2014.

[37] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 2559–2566, San Francisco, Calif, USA, June 2010.

[38] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 1794–1801, June 2009.

[39] M. D. Zeiler and R. Fergus, *Stochastic pooling for regularization of deep convolutional neural networks*, 2013, arXiv:1301.3557.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol. 71, pp. 1–10, 2015.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[43] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," in *Proceedings of the 30th Annual Conference on Neural Information Processing Systems, NIPS 2016*, pp. 2154–2162, esp, December 2016.

[44] C. Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Artificial Intelligence and Statistics*, pp. 464–472, 2016.

[45] M. Lin, Q. Chen, and S. Yan, *Network in network*, 2013, arXiv:1312.4400.

[46] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*, pp. 1717–1724, IEEE, Columbus, Ohio, USA, June 2014.

[47] J. Deng, W. Dong, and R. Socher, "ImageNet: a large-scale hierarchical image database," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, Miami, Fla, USA, June 2009.

[48] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.