Filter Pruning Without Damaging Networks Capacity

YUDING ZUO, BO CHEN, TE SHI, AND MENGFAN SUN

School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

Corresponding author: Bo Chen (bochen@uestc.edu.cn)

This work was supported in part by the Major Science and Technology Program of Sichuan Province under Grant 2018GZDZX0042, in part by Yibin Sci-Tech under Grant 2019GY001, and in part by the e-Commerce Key Lab of Jiangsu Province under Grant JSEB-201303.

ABSTRACT Due to its over-parameterized design, the deep convolutional neural networks lead to a huge amount of parameters and high computational cost, making it difficult to deploy on some devices with limited computational resources in reality. In this paper, we propose a method of filter pruning without damaging networks capacity to accelerate and compress deep convolutional neural networks. Differently from some existing filter pruning methods, we pay more attention to the damage by filter pruning to model capacity. In order to restore the original model capacity, we generate new feature maps on the basis of the remaining feature maps with lighter structure after pruning the redundant filters that are similar with the others. Experimental results on CIFAR10 and CIFAR100 benchmarks demonstrate the effectiveness of our method. Especially, our method reduces more than 49% FLOPs for VGGNet-16 on CIFAR10 with only 0.07% relative accuracy drop. The relative accuracy has even been increased by 0.13% with reducing more than 24% FLOPs. Moreover, our method accelerates ResNet-110 on CIFAR10 by 22.1% with 0.41% accuracy improvement, which exceeds the previous methods.

INDEX TERMS Convolutional neural networks, filter pruning, filter similarity, model capacity, networks compression and acceleration.

I. INTRODUCTION

The deeper and wider structure makes deep convolutional neural networks perform well in a variety of computer vision tasks, such as semantic segmentation [1], object detection [2] and image classification [3]–[5]. However, its over-parameterized design has led to a huge amount of parameters and expensive computational consumption. For example, VGGNet-16 [4] has 15M parameters and requires 313M float point operations (FLOPs) [6] to process a color image of size 32 × 32. It is difficult to deploy CNNs on resource-constrained devices, such as mobile devices. Therefore, the recent optimization trend of deep convolutional neural networks is to reduce its parameters and computational consumption, and at the same time ensure its performance, so that CNNs can be deployed on resource-constrained devices.

In recent years, many methods have been proposed to compress and accelerate CNNs. These methods can be roughly divided into networks pruning [6]–[15], low-bit quantification [16]–[20], knowledge distilling [21]–[23] and matrix decomposition [24]–[26]. Networks pruning is one of the most popular fields and has been widely studied. This paper also focuses on it. Reference [7] finds that there are some parameters that have little influence on the final accuracy and can be pruned. Reference [8] proposes a method of weight pruning based on a threshold. However, these methods of weight pruning may cause unstructured sparsities and require additional sparse matrix operation libraries or even specific hardware devices. Therefore, filter pruning is more widely studied. Reference [9] proposes a global greedy filter pruning method, which uses \( \ell_1 \)-norm to evaluate the importance of filter. After analyzing the sensitivity of each layer to pruning filters, the filters with smaller \( \ell_1 \)-norm will be pruned globally at a time. Reference [12] rethinks the norm-based criterion for filter pruning and proposes a filter pruning method via geometric median to prune redundant filters which are close to geometric median, rather than those less important filters.

As shown in Fig.1, there are some redundant feature maps in the convolutional layer, and these feature maps can be approximately generated from other feature maps. Among the above-mentioned pruning methods, only the filter selection method of [12] considers the similarity in feature maps. This method cannot actually maintain the model capacity,
although it still remains pruned filters which are set to zero in the networks for training. In this paper, we propose a filter pruning method without damaging model capacity. Our method can be described in the following steps. 1) The filters that are more replaceable are selected in each convolutional layer. The feature maps generated by these replaceable filters can often be approximately transformed by the remaining feature maps. 2) The selected filters are pruned, and to maintain the original networks capacity, the feature maps corresponding to the remaining filters are treated as the original feature maps to generate new feature maps with lighter group convolution [5]. 3) The pruned networks are retrained to restore its accuracy.

The contributions of this paper are as follows. 1) We pay more attention to the damage by filter pruning to the model capacity and propose a method of maintaining the integral capacity of the model when pruning filters. 2) We combine filter pruning with lightweight networks structure design to compress and accelerate deep convolutional neural networks for the first time. 3) Experiments on two benchmark datasets demonstrate the effectiveness of our method. Compared with previous methods, our method achieves the state-of-the-art results.

II. RELATED WORKS

In order to apply deep convolutional neural networks in actual production, many studies have focused on balancing the computational consumption and accuracy of the models. Reference [16]–[19] compress the original network by reducing the number of bits needed to represent each weight. Reference [17] proposes an incremental networks quantization method that can convert a full-precision float point neural networks model of any structure to a lossless low-bit binary model through three independent operations consisted of weight division, group quantization and retraining. To reduce the parameters and computational consumption, [24]–[26] utilize low-rank matrices to approximate the weight matrix in a neural networks. Reference [21]–[23] utilize large teacher networks to supervise small student networks for training to achieve networks compression. According to the similarity in feature maps in the convolutional layer, [27] builds the ghost module which generates virtual feature maps by cheap linear operations and build compact GhostNet model with ghost modules.

Reference [11]–[13], [28] utilize networks pruning to prune redundant weights and filters for compressing and accelerating CNN. Reference [28] treats pruning as an optimization problem to find weights that minimize the loss and satisfy a pruning cost condition. Reference [11] utilizes spectral clustering to classify the filters and prune the filters according to the importance of the categories. To identify insignificant channels, [13] applies $L_1$ regularization to the scaling factor of the batch normalization (BN) [29] layer. Reference [12] analyzes traditional norm-based criterion for evaluating the importance of filters and proposes a method to prunes the filters that can be replaced via geometric median, but not less important filters.

III. METHODOLOGY

A. PRELIMINARIES

We will introduce symbols and notations in this subsection. We denote $W_i \in \mathbb{R}^{N_i \times C_i \times K \times K}$ (1 $\leq i \leq L$), where $W_i$ represents the weight matrix connecting the $i_{th}$ and $i+1_{th}$ convolutional layers and $N_i$, $C_i$, $K$, $L$ represent the number of output channels, the number of input channels, the kernel size of filters and the number of network layers respectively. $F_{ij}(1 \leq j \leq N_i)$ represents the $j_{th}$ filter of the $i_{th}$ layer, and the dimension of filter $F_{ij}$ is $C_i \times K \times K$. We assume that input feature maps of $i_{th}$ layer is $X_i \in \mathbb{R}^{C_i \times H_i \times W_i}$, where $C_i$ is the number of input channels and $H_i$ and $W_i$ are the height and width of the input feature maps. $Y_i \in \mathbb{R}^{N_i \times H_i \times W_i'}$ is output feature maps of $i_{th}$ layer.

B. FILTER SELECTION

We can find from Fig.1 that some feature maps generated by the convolution layer are similar. If we prune some of these similar feature maps, the pruned feature maps can roughly recover from the remaining feature maps. Therefore, we select the most replaceable filters to prune, which is similar to [12].

We utilize the Euclidean distance to evaluate the similarity between the two filters. The smaller the distance is, the more similar the feature maps corresponding to the two filters are. We calculate the sum of Euclidean distances from each filter to all other filters in each layer as the evaluation criterion.

$$\text{Sum}_{i,j} = \sum_{j'=1}^{N_j} \| F_{ij} - F_{i,j'} \|_2, \quad s.t. j' \neq j. \quad (1)$$

$\text{Sum}_{i,j}$ in (1) represents the sum of Euclidean distance from $j_{th}$ filter to all other filters in $i_{th}$ layers. The filter $F_{i,j}$ with small
Algorithm 1: Algorithm of Filter Selection

Input: pretrained model $M$, pruning rate $P$

1. for $i = 1; i \leq L; i ++$
   2. for $j = 1; j \leq N_i; j ++$
      3. Calculate the sum of Euclidean distance from $F_{i,j}$ to other filters according to (1)
      4. Find $N_i \times P$ filters that satisfy (2)
   5. end
   6. end

Output: The mask matrix of filters

$Sum_{i,j}$ is selected to prune.

$$F_{i,x} = \arg\min(Sum_{i,x}), \quad s.t. 1 \leq x \leq N_i. \quad (2)$$

$F_{i,x}$ in (2) represents the selected filters in $i_{th}$ layer. The filter selection methods are summarized in Algorithm 1.

C. FILTER PRUNING AND RECONSTRUCTION

According to the filter selection method of III-B, we perform filter pruning globally on all convolutional layers at one time, and introduce group convolution in the pruned convolutional layer to generate new feature maps, as shown in Fig.2. We make the number of feature maps generated by pruned model in each layer same as that of the original model.

In Fig.2, “Conv” represents common convolution operation, “BN” represents batch normalization, “Relu” represents nonlinear activation function, “Identity” represents identity mapping [30], “Group Conv” represents group convolution, “Concatenate” represents dimensional concatenation. (a) is the original convolutional layer. Fig.2(b) and Fig.2(c) are the structure of reconstructing pruned feature maps. The difference between Fig.2(b) and Fig.2(c) is the order of Relu. Experiments in IV show that Fig.2(b) performs better than Fig.2(c) and the comparison of the performance of them at different pruning rates is shown in Table 1. In the case of maintaining the original model capacity, fine-tuning the pruned model can restore its accuracy easily, and even exceed the original accuracy. Our method is summarized in Fig.3.

In Fig.3, the blue matrix is pruned feature maps and the yellow matrix is residual feature maps. On one hand, the new feature maps are generated from residual feature maps and are shown as green matrix. On the other hand, the residual feature maps concatenate with the new feature maps to be the final output which has same channels with original output.

D. ANALYSIS ON THEORETICAL ACCELERATION AND COMPRESSION

We apply Algorithm 1 to select filters, and the method of Fig.3 is applied to prune filters and reconstruct feature maps. The group convolution shown in Fig.2 is applied to reconstruct pruned feature maps. According to the description of group convolution in [5], the parameters and FLOPs of the group convolution are $1/g$ of the common convolution, where $g$ is the number of groups. In fact, $g$ can influence acceleration rate and the bigger $g$ is, the bigger acceleration rate becomes. Moreover, $g$ must be divided by input and output channels of

| Pruning Ratio | Backbone | Baseline Acc(%) | Accelerated Acc(%) |
|---------------|----------|-----------------|-------------------|
| 0.25          | Fig.2(b) | 93.65           | 93.78             |
| 0.5           | Fig.2(c) | 93.65           | 93.58             |
| 0.75          | Fig.2(b) | 93.65           | 92.75             |
the group convolutional layer. We assume that pruning rate is \( P \), \( N_i \times P \) is the number of pruned filters and \( N_i \times (1 - P) \) is the number of remaining filters. In order to achieve the maximum acceleration rate, We set \( P \) to make one of \( N_i \times P \) and \( N_i \times (1 - P) \) can be divided by the other. Therefore, Maximum of \( g \) can be calculated as

\[
g = \min(N_i \times P, N_i \times (1 - P)).
\] (3)

Kernel size, stride and padding of the convolution operation are same with those of original convolution layer to ensure that the size of the feature map generated by the group convolution is the same as that of the original output. Inspired by [27], we set kernel size of group convolution as \( K' \), which is same as the kernel size in VGGNet and ResNet. In the \( i_{th} \) layer, the FLOPs of original model can be calculated as

\[
F_o = N_i \times H_i' \times W_i' \times C_i \times K^2.
\] (4)

The FLOPs of pruned model can be divided into two sections including primary convolution and group convolution. we can calculate the FLOPs of primary convolution as

\[
F_p = N_i \times (1 - P) \times H_i' \times W_i' \times C_i \times K^2.
\] (5)

The input and output channels of group convolution are \( N_i \times (1 - P) \) and \( N_i \times P \) respectively. The FLOPs of group convolution can be calculated as

\[
F_g = \frac{N_i \times P \times H_i' \times W_i' \times N_i \times (1 - P) \times K'^2}{g},
\] (6)

where \( g \) is demonstrated in (3), \( K' = K \) and \( P \ll N_i \). The theoretical speed-up ratio of pruned model can be calculated as

\[
r_o = \frac{F_o}{F_p + F_g} = \frac{C_i}{(1 - P) \times C_i + \frac{P \times (1 - P)}{\min(P, 1 - P)}} \approx \frac{1}{1 - P}
\] (7)

Similarly, the compression ratio can be calculated as

\[
r_c = \frac{W_o}{W_p + W_g},
\] (8)

where \( W_o, W_p \) and \( W_g \) represent original convolutional parameters, primary convolutional parameters and group convolutional parameters respectively. They can be calculated as

\[
W_o = N_i \times C_i \times K^2,
\] (9)

\[
W_p = N_i \times (1 - P) \times C_i \times K^2,
\] (10)

\[
W_g = \frac{N_i \times P \times N_i \times (1 - P) \times K'^2}{g}
\] (11)

Finally, the compression ratio is

\[
r_c = \frac{C_i}{(1 - P) \times C_i + \frac{P \times (1 - P)}{\min(P, 1 - P)}} \approx \frac{1}{1 - P}
\] (12)

E. HANDLING CROSS LAYER CONNECTIONS

The methods in III-B and III-C can be directly applied to plain CNN architectures such as VGGNet. However, some adaptations are required when it is applied to the networks with cross layer connections such as ResNet. For these networks, the structure of reconstruction is designed as Fig.4. In Fig.4 (b), group convolution generates new feature maps based on the remaining feature maps after pruning to ensure that number and size of the output feature maps are the same as those of the original output.

IV. EXPERIMENTS

In this section, we evaluate our proposed filter pruning method on CIFAR10 and CIFAR100 benchmarks with the single-branch VGGNet-16 and multi-branch ResNet-56 and ResNet-110 networks. We compare our method with the previous methods, such as [6], [9], [12], [13], [27], [31], [32].

A. DATASETS AND SETTING

1) CIFAR [33]

It is widely used in the field of image classification as a standard dataset. CIFAR dataset contains 60,000 \( 32 \times 32 \) colored images, with 50,000 images for training and 10,000 for testing. They are labeled for 10 and 100 classes in CIFAR-10 and CIFAR-100 respectively.

2) TRAINING SETTING

The strategies of VGGNet and ResNet baseline training are same as [12] and [34] respectively. All pruned models are...
retrained for 400 epochs with multi-step learning rate policy (0.1 for the first 200 epochs, 0.01 for the following 100 epochs, 0.001 for the next 75 epochs and 0.0001 for the rest epochs). Stochastic gradient descent (SGD) with momentum $0.9$ and weight decay $1e^{-4}$ is applied to retrain networks.

3) PRUNING SETTING
The first layer of VGGNet and ResNet networks has a small amount of parameters and low computational cost and it is sensitive to filter pruning, which is analyzed in [9]. Therefore, we do not prune the first layer. The hyperparameter $P$ is the pruning rate which is same in all convolutional layer. The groups of group convolution in Fig.2 must be divided by the number of input and output channels of group convolution and should satisfy (3). Therefore, we set $P \in \{0.125, 0.25, 0.5, 0.75, 0.875\}$ and $P \in \{0.25, 0.5, 0.75\}$ for VGGNet and ResNet respectively.

### B. RESULTS ON CIFAR10

We evaluate the different structure in Fig.2 with VGGNet-16 on CIFAR10 and the results are shown in Table 1. As we can see, the structure in Fig.2(b) performs better than that in Fig.2(c). We will apply the structure in Fig.2(b) to our following experiments.

1) **VGGNet-16 ON CIFAR10**

We test our method on VGGNet-16 with five different pruning rates: 0.125, 0.25, 0.5, 0.75 and 0.875 and compare with other methods. Table 2 shows the results. When pruning rate is 0.5, our method can achieve comparable performances comparing with previous methods with 49.5% FLOPs pruned. More importantly, our method can achieves 11.8% speed-up ratio with even 0.36% accuracy improvement, which achieves the state-of-the-art results.

2) **ResNet ON CIFAR10**

We test our method on ResNet-56 and ResNet-110 with three different pruning rates: 0.25, 0.5 and 0.75. As shown in Table 3, our method achieves the state-of-the-art performance. For example, our method reduces 22.2% FLOPs for ResNet-56 with even 0.19% relative accuracy improvement, which exceeds other methods. Reference [12] accelerates ResNet-110 by 52.3% speed-up ratio with 0.17% accuracy improvement, and our method achieves 48.2% speed-up ratio with 0.04% accuracy drop and achieves comparable performance. Compared with [6], our method accelerates ResNet-110 by 22.1% speed-up ratio with even 0.41% accuracy improvement. These results demonstrate that our method can accelerate and compress networks with comparable or better performances.

### C. RESULTS ON CIFAR100

Table 4 displays the comparison of different methods on CIFAR-100. For VGGNet-16, our method makes an outstanding contribution to the improvement on accuracy, which has improved by 1.04% with 11.8% FLOPs pruned.
In addition, our method can accelerate ResNet with relative accuracy improvement.

D. ANALYSIS ON RESULTS

From the Table 2–Table 4 we come to the conclusion that the model accuracy decreases as the pruning rate increases, which is in line with experimental expectations. Compared with other methods on different models, our method can achieve higher accuracy. In addition, we find that our method performs better on CIFAR10 than CIFAR100. Each category in CIFAR10 has more images for training than CIFAR100. In other words, networks on CIFAR10 can learn more information, to which our method can be better applied.

V. CONCLUSION

In this paper, we find that the previous filter pruning methods are facing the problem of damaging networks capacity. To solve the problem, we propose a method to prune the redundant filters that are similar with the others and generate new feature maps on the basis of the remaining feature maps with lighter structure to restore the original model capacity. The experiments show the effectiveness of proposed method. In addition, our method of restoring model capacity doesn’t conflict with previous filter pruning methods which can also be optimized by our method.

REFERENCES

[1] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1520–1528.
[2] R. Girshick, “Fast R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 1440–1448.
[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.
[4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, arXiv:1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556
[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.
[6] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 598–606.
[7] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” in Proc. Adv. Neural Inf. Process. Syst., 1990, pp. 598–605.
[8] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1135–1143.
[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. Peter Graf, “Pruning filters for efficient ConvNets,” 2016, arXiv:1608.08710. [Online]. Available: http://arxiv.org/abs/1608.08710
[10] Y. He, X. Zhang, and J. Sun, “Channel pruning for accelerating very deep neural networks,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 1389–1397.
[11] H. Zhuo, X. Qian, Y. Fu, H. Yang, and X. Xue, “SCSP: Spectral clustering filter pruning with soft self-adaptation manners,” 2018, arXiv:1806.05320. [Online]. Available: http://arxiv.org/abs/1806.05320

VOLUME 8, 2020

90929
