



A Pruning Method Based on Feature Abstraction Capability of Filters

Yi Tang, Xiang Zhang^(✉), and Ce Zhu

University of Electronic Science and Technology of China, Chengdu 611731, China
uestchero@uestc.edu.cn

Abstract. With the wide application of convolutional neural network, the optimization of CNN has received ever-increasing research focus. This paper proposes a new pruning strategy, which aims to accelerate and compress off-the-shelf CNN models. Firstly, we propose the pruning criteria for the feature abstraction capability of the filter, which is evaluated by combining the kernel sparsity of the filter with the dispersion of the feature maps activated by the filter. Then, the filter with weak Feature Abstraction Capability (FAC) is pruned to obtain a compact CNN model. Finally, fine-tuning is used to restore the generalization ability. And Compared with other pruning methods which use filters of the same layer for contrast, Our method normalizes each layer, the proposed criterion can be applied to the filters between cross-layer of CNN. Experiments on CAFAR-10 and CUB-200-2011 datasets verify the effectiveness of our method. The FAC-based method achieves better performance than previous filter importance evaluation criteria.

Keywords: Accelerate · Kernel sparsity · Dispersion · Feature Abstraction Capability

1 Introduction

In recent years, It is well known that convolutional neural networks have achieved great success in various computer vision tasks [1], including object detection [2–4], object classification [5,6] and semantic segmentation [7,8] and many others. CNNs have achieved state-of-the-art performance in these fields compared with traditional methods based on manually designed visual features [9]. However, with the deepening and widening of CNN convolution layer, higher computational overhead and larger memory are required, so it is difficult to deploy CNN model to resource-limited devices, such as mobile phones and embedded devices. As a result, the application of convolutional neural network in practical scenarios will be limited by various hardware equipment resources, such as storage space, computing power and battery power. For instance, AlexNet [10] network contains about 6×10^6 parameters, while some better networks like VGG [11] contain about 1.38×10^8 parameters. For less complex tasks, such as simple image recognition, the VGG network will require more than 500 MB memory

and 1.56×10^{10} Float Point Operations (FLOPs). The over-parameterized [12] of deep learning is a major obstacle to deployment on mobile devices.

Thus, network compression has drawn a significant amount of interests from both academia and industry. In recent years, numerous efficient compression methods have been proposed, including low-rank approximation [12, 13], parameter quantization [14, 15], and binarization [16]. Among them, network pruning [17–20] has excellent performance in reducing redundancy of CNNs, and it has better model deployment ability compared with parameter quantization. simultaneous pruning can be applied to different elements of CNNs, such as weights, filters, and layers. Early works in network pruning mainly resort to removing several unimportant weight connections from a well-trained network with negligible impact on network performance.

In this paper, a pruning strategy based on Feature Abstraction Capability (FAC) of filters is proposed. Zhou et al. [21] have shown that the features extracted by convolution kernels across layers or even the same layer have significant differences in the contribution of the final prediction results. The more information the feature map represents, the more important it is to the network. Therefore, we believe that the Feature Abstraction Capabilities of different filters are different, and use this criterion to guide the pruning of network elements.

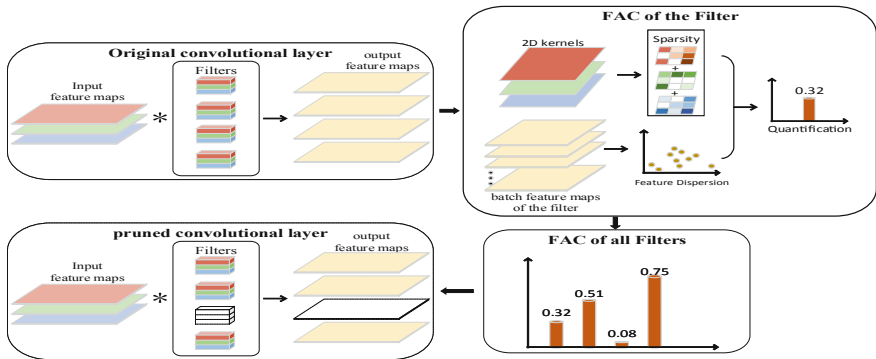


Fig. 1. The framework of our method, in the original convolution layer, the input feature graph is convoluted with the 3D filter to obtain the output feature map. In our FAC method, we first obtain the sparsity of the 3D filter and the dispersion of the batch feature maps by the L1-norm [24] and the data-driven method [18]. Finally, the quantized value of the FAC is obtained as the pruning index in combination with the sparsity and dispersion, and the filter with the lower FAC value in the volume layer is pruned to generate a more Compact network.

As shown in Fig. 1, the Feature Abstraction Capability (FAC) of the filter is obtained by evaluating the sparsity of the convolution kernel of the filter and the information richness contained in the feature map activated by the filter. Our main insight is that the feature map activated by the filter with lower Feature Abstraction Capability (FAC) is redundant. Pruning unimportant filters and

fine-tuning the network to restore its generalization capabilities. Finally, the CNN model accelerates and compresses during the training and testing phases, transforming the cumbersome network into a smaller model with a slight performance degradation. At the same time, we propose to normalize the quantized value of each filter's FAC, the proposed pruning strategy can be extended to all layers of the deep CNN, eliminating the need for threshold sensitivity analysis for each layer.

We evaluated our pruning framework by using two commonly used CNN models: VGG-16 [22] and Resnet-110 [23]. These two models are pruned on two benchmark datasets CIFAR10 and CUB_200_2011. These two data sets are representative. In the CIFAR10 dataset, our method still achieves $4.9\times$ compression and $1.77\times$ acceleration on VGG-16, with about 0.3% top-1 accuracy drop. Similarly, in the CUB_200_2011 dataset with 200 kinds of complex tasks of fine-grained classification, our method still had $4.2\times$ acceleration on VGG-16 with roughly 0.6% top-1 accuracy drop, which was better than most similar pruning algorithms.

2 Related Work

In this section, we will briefly introduce some popular network pruning methods in CNN compression, which can be divided into structured pruning and unstructured pruning.

Unstructured pruning is to zero the weight value below a certain threshold in the weights. Among them, what is impressive is that Han et al. [17] proposed to connect by pruning the weight of small magnitudes on AlexNet network and VGG network, and then retrain without affecting the overall accuracy, effectively reducing the number of parameters. However, this pruning operation generates an unstructured sparse model that requires sparse BLAS libraries or even specialized hardware to achieve acceleration.

Structured pruning reduces computational complexity and memory overhead by directly removing structured parts, such as kernels, filters, or layers, and is well supported by a variety of off-the-shelf deep learning platforms. For instance, One pruning criterion is sparsity activated by non-linear ReLU mappings. Hu et al. [18] proposed a data-driven neuron pruning approach to remove unimportant neurons. They argue that if most of these activated feature maps are zero, it is not important for neurons to have a high probability. The criterion measures the importance of neurons by calculating the average percentage of zeros (APoZ) in the activated feature map. However, the APoZ pruning criterion requires the introduction of different threshold parameters for each convolutional layer, which are difficult to accurately determine. Li et al. [24] proposed to remove unimportant filters based on the L1-norm. Molchanov et al. [19] calculated the influence of filters on network loss function based on Taylor expansion. According to the criterion, if the filter has little influence on the loss function, the filter can be safely removed. So they use Taylor expansion to approximate the change in loss. He et al. [23] proposes a channel selection method based on LASSO

regression, which uses least squares reconstruction to eliminate redundant filters. Similar to our study, Luo et al. [20] proposed a method to calculate entropy of filters to measure the information richness of the convolution kernel. However, only the information richness of the filter is considered in the method, and the strategy can only compare the entropy value of the same convolutional layer. Most of these methods need to accurately obtain the pruning threshold of each convolutional layer, but this is difficult to achieve. If fixed compression rate is used for pruning, it may lead to irreparable accuracy reduction.

In addition to the network pruning method, some other CNN compression methods are introduced, such as designing a more compact architecture. For example, it is known that most parameters of the CNN model exist in fully-connected layers, so the global average pooling is proposed to replace the full connection layer in the Network-In-Network [26]. Son et al. [27] reconstructed the network by unified representation of similar convolutions, so as to achieve effective compression of the network. However, this method has some limitations. It is only effective for 3×3 convolution kernels. Sandler et al. [28] proposed the use of depthwise separable convolution to build a lightweight network, which has also been widely used in mobile devices. It's important to note that our approach can be combined with these strategies to achieve a more compact and optimized network. As for ResNet-50, there exists less redundancy compared with classic CNN models. We can still bring $1.63\times$ acceleration and achieves $2.48\times$ reduction in FLOPs and parameters with 0.007 decrease in accuracy.

3 Pruning Method

In this section, we will describe in detail our pruning method based on the Feature Abstraction Capability of the filter. First, the general framework is given. Our main idea is to quantify the FAC of all convolutional layer filters, discard those filters with poor performance in each pruning, and restore their performance by fine-tuning. These implementation details will be released later. Finally, the training and pruning planning strategy we used in the experiment is introduced, which has less impact on the final prediction accuracy compared with other previous strategies.

3.1 Framework

Figure 1 illustrates the overall framework of our proposed FAC pruning method. We first obtain the weight values of all 2D kernels in the 3D filter, obtain the sparsity of the 3D filter from the sum of the L1-norm [24] of the 2D kernels, at the same time obtain the batch feature maps of the filter by the data-driven method [18], and then calculate the discreteness of batch feature maps. We use the discreteness of the filter batch feature maps to evaluate the richness of the information contained in the activated feature map, because if the difference in the feature map of the filter is small each time, we have enough reason to believe that the filter Feature Abstraction Capability is weak. Finally, we combine the

sparsity of the convolution kernel with the information dispersion of the activated feature map to make the estimated filter feature abstraction more accurate and robust.

Then, all weak filters are pruned from the original model to achieve a more optimized network architecture. Note that the corresponding input channels of filters in the next layer should be removed. Finally, the network is fine-tuned to restore its generalization performance.

3.2 Filter Sparsity

In the convolution layer i , the input tensor $I_i \in \mathbb{R}^{C \times H_{in} \times W_{in}}$ is convolved with a set of filter weights $W_i \in \mathbb{R}^{N \times C \times K_h \times K_w}$ to get the output tensor $Y_i \in \mathbb{R}^{N \times H_{out} \times W_{out}}$. Here, C is the number of the input feature maps, H_{in} and W_{in} are the height and width of the input feature maps, N is the number of the filters, H_{out} and W_{out} are the height and width of the output feature maps, K_h and K_w are the height and width of a filter. The convolution operation can be expressed by the following formula:

$$Y_n = I_i * W_n \tag{1}$$

Where denote the convolution operation, W_n is the weight of the n th filter in the convolutional layer I , $W_n \in \mathbb{R}^{C \times K_h \times K_w}$. Y_n is the feature map of n th filter, $Y_n \in \mathbb{R}^{H_{out} \times W_{out}}$.

We evaluate the sparsity of the filter by calculating the L1-norm of W_n , because it is known from Eq. (1) that if the absolute value of the weights value in W_n is mostly close to zero, the L1-norm will be small and the value in the output feature map of the filter will also be closer to zero. We think that such a feature map is approximately sparse, indicating that the filter’s Feature Abstraction Capability is also weaker. Therefore, for the n -th input feature map, we define its sparsity as:

$$S_n = \sum_{c=0}^C \sum_{i=0}^{K_h} \sum_{j=0}^{K_w} |W_{n,c,i,j}|, \quad n = 0, 1, 2, \dots, N \tag{2}$$

3.3 Discreteness of Feature Maps

In this paper, we propose a criterion based on the FAC of the filter to evaluate the importance of each filter. We believe that if the difference between the feature maps of each output of the filter is greater, the filter’s Feature Abstraction Capability will be stronger.

As shown in Fig. 2, we use global average pooling [26] for the activated feature maps output of layer i , in this way, a $N \times H_{out} \times W_{out}$ output tensor will be converted into a $1 \times N$ vector. At the same time, a corresponding score is obtained for each feature map output of the filter. In order to calculate the dispersion of each filter’s output feature map score, more output results are needed, so we calculate a score for each batch of the data set, and finally we will get a matrix

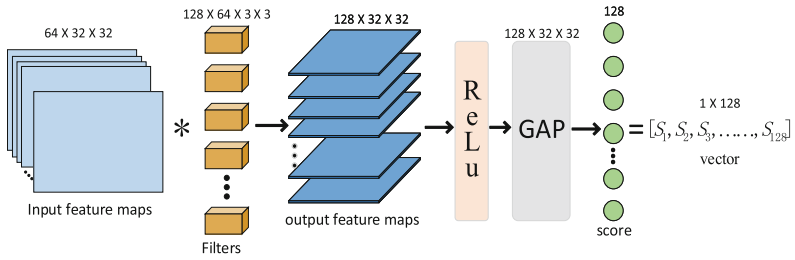


Fig. 2. How to use Global Average Pooling (GAP) to calculate the score of output feature map in convolutional layer i , it should be noted that feature maps are activated by ReLU function before GAP, because we think that the negative values in feature maps are filtered out in the network.

$M \in \mathbb{R}^{B \times N}$, where B refers to the number of batches in the training data set, and N is the output channel number.

Then, for $M_{:,j}$ in the matrix M represents the output scores vector of the j -th filter, let μ_j be the average of the scores of the j -th filter, the formula is as follows:

$$S_n = \sum_{i=0}^B M_{i,j}, \quad j = 0, 1, 2, \dots, N \tag{3}$$

Then, the feature maps dispersion of the j -th filter is:

$$D_j = \begin{cases} 0, & \mu_j = 0 \\ \sqrt{\frac{\sum_{i=0}^B (M_{i,j} - \mu_j)^2}{B}}, & otherwise \end{cases} \tag{4}$$

3.4 Definition of the FAC

From the above discussion, we know that the importance of the filter depends on two parts, the sparsity of the convolution kernel and the discreteness of the feature maps. Therefore, we combine the two parts to propose the Feature Abstraction Capability (FAC) to measure the importance of the filter by:

$$FAC = S_n * D_j \tag{5}$$

3.5 Normalization

In many papers, the pruning criterion is only applicable to the convolution kernel comparison between the same layers, and the scale inconsistency will occur when applied to the cross-layer. Therefore, in our method, we uses layer-wise L2-normalization to achieve reasonable rescaling:

$$\Theta^{(i)} = \frac{\Theta^{(i)}}{\sqrt{\sum_{j=0}^N (\Theta^{(i)})^2}} \tag{6}$$

Where, $\Theta^{(i)}$ refers to the set of all FAC filters in the layer i , which can be understood as a vector. $\Theta^{(i)}$ refers to the FAC of the j -th filter in the layer i .

3.6 Pruning and Fine-Tuning Strategy

There are two main types of network architectures: traditional convolution and fully-connected architectures, as well as some structural variants. VGG and Resnet are typical representatives, and we mainly introduce the pruning methods of these two networks. As shown in Table 1, we notice that more than 39% parameters exist in the fully-connected layers for VGG-16. Some papers [20] use global average pooling instead of the full connection layer, which can greatly reduce the number of parameters of the model, but also greatly reduce the convergence speed of the model, which may make it difficult to train the model back to the original accuracy. Therefore, we reduce the parameters of the full connection layer by pruning the filter of the last convolution layer to reduce the input channel of FC1 layer.

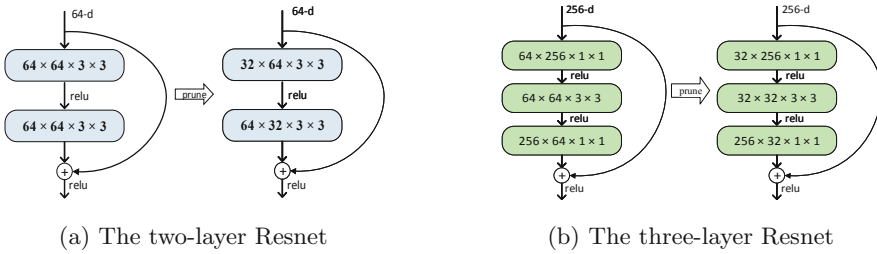


Fig. 3. Our pruning strategy for ResNet. For each residual block, the final convolutional layer filter cannot be pruned, reducing its input channel by pruning of the previous layer.

For ResNet, there are some restrictions in the pruning process due to the introduction of so-called “identity shortcut connection”. For example, the summation operation requires that the number of output channels per block in the same group needs to be consistent (see Fig. 3). In the ResNet network structure, two kinds of residual modules are mainly used, one is that two 3×3 convolutional networks are connected in series as one residual module, and the other is 1×1 , 3×3 , 1×1 of 3 convolutional networks are concatenated together as a residual module.

The final question is how to fine tune the entire network during the pruning process. Our strategy is prune and retrain iteratively. We found that most of the pruning method is to pruning filter of each layer at a fixed rate of pruning, and then fine-tuning the model a few times, but if the pruning rate is too high, the filter structure of the layer may be damaged. This problem will become more apparent in more complex task networks because too many filters are pruned at once in this layer.

So our method is to preset a compression ratio α for the whole network. We need to pruning $N \times (1 - \alpha)$ filters, and then we only pruned β filters each time, finally whole network is fine-tuned with few epochs to recover its performance

slightly. In this way, we only need to perform $N \times (1 - \alpha)/\beta$ pruning, and those β filters that are pruned each time are distributed across all convolutional layers of the network, not concentrated on one layer at a time. And the value of β in the experiment is 256. The fine-tuning method obtained better results in experiments than other methods.

Table 1. Overall performance of our approach to reduce FLOPs and parameters on the VGG-16 model. The experiment is based on the CIFAR10 dataset. Note that we will resize the 32×32 image size in CIFAR to 128×128 .

Layer	Feature map size	FLOPs			Parameters		
		Original	Pruned	Percentage	Original	Pruned	Percentage
Conv1-1	128×128	29.36M	21.56M	73.4%	1.79K	1.32K	73.5%
Conv1-2	128×128	605.02M	305.65M	50.5%	36.92K	18.65K	46.7%
Conv2-1	64×64	302.51M	37.4M	12.4%	73.85K	9.13K	12.4%
Conv2-2	64×64	604.50M	18.74M	3.1%	147.58K	4.58K	3.1%
Conv3-1	32×32	302.25M	5.91M	2.0%	295.16K	5.77K	1.9%
Conv3-2	32×32	604.24M	9.39M	1.5%	590.08K	9.17K	1.6%
Conv3-3	32×32	604.24M	16.83M	2.8%	590.08K	16.43K	2.8%
Conv4-1	16×16	302.12M	10.69M	3.5%	1.18M	41.74K	3.5%
Conv4-2	16×16	604.11M	24.84M	4.1%	2.36M	97.04K	4.1%
Conv4-3	16×16	604.11M	32.37M	5.3%	2.36M	126.44K	5.4%
Conv5-1	8×8	151.02M	9.63M	6.4%	2.36M	150.48K	6.4%
Conv5-2	8×8	151.02M	16.68M	6.9%	2.36M	260.69K	11.1%
Conv5-3	8×8	151.02M	20.62M	13.6%	2.36M	322.18K	13.7%
FC1	1	8.39M	2.92M	34.8%	8.39M	2.92M	34.8%
FC2	1	1.05M	1.05M	100.0%	1.05M	1.05M	100.0%
FC3	1	10.24K	10.24K	100.0%	10.24K	10.24K	100.0%
Total	–	5.02B	534.29M	10.6%	24.17M	5.03M	20.9%

4 Experiments

We used our pruning method to pruning two typical CNN models: VGG-16 and ResNet-50. We have implemented our approach using the deep learning framework Pytorch. The validity of the algorithm is verified on two datasets, CIFAR-10 and CUB_200_2011. The CIFAR-10 dataset consists of 60000 images, whose size is 32×32 , and the number of images in each category is 6000, with 10 categories. During training, images are converted to 128×128 , because if the image is too small, the FLOPs of the network itself will be very small, and the deceleration effect after pruning is not obvious. The Cub_200_2011 is a birds data set for fine-grained classification tasks, which contains 11788 images of 200 different bird species. It presents a significant challenge for pruning algorithms

to maximize model compression and acceleration without reducing accuracy too much. During training, all images of Cub_200_2011 are resize to 320×320 , After each pruning, we fine-tune the whole network in 8 epochs with learning rate varying from 10^{-3} to 10^{-5} . During the last pruning, the network is fine-tuned in 20 epochs with learning rate varying from 10^{-3} to 10^{-8} . All experiments are run on a computer equipped with Nvidia GTX 1080Ti GPU.

4.1 VGG-16 Pruning

The detailed distribution of FLOPs and parameters in each layer of VGG-16 is shown in Table 1. As we have seen, the 2nd - 12th layer convolutional layer contains 90% FLOPs. And we can see that our pruning method is also mainly aimed at layer 3–12. For the first two layers of convolutional layer, there is no large-scale pruning. We think that the first two layers of filters contain rich feature information, so they have stronger Feature Abstraction Capability than other filters in the layer. The side proves that our method has a certain degree of interpretability. The pruning rate we set is 80%, that is, 80% of the filters in the model are pruned off. Finally, we compare our method with following baselines on the VGG-16 model:

Taylor Expansion [19]: The effect of the filter on the network loss function is calculated based on the Taylor expansion method. According to this criterion, if the filter has little effect on the loss function, the filter can be safely removed.

APoZ [18]: The criterion measures the importance of filters by calculating the average percentage of zeros (APoZ) in the activated feature map.

Entropy [20]: The method to calculate entropy of filters to measure the information richness of the convolution kernel.

Table 2. Comparison of different model compression methods for VGG-16 network on CIFAR10.

Method	Top-1 Acc	FLOPs (%)	Params (%)	Speed up	Compression
Original	0.916	5.02B	24.17M	1.0×	1.0×
APoZ-50%	0.912	1.27B (25.3%)	8.39M (34.7%)	1.61×	2.7×
APoZ-80%	0.803	0.19B (3.8%)	3.34M (13.8%)	1.92×	5.4×
Taylor-50%	0.922	1.42B (28.2%)	6.94M (28.71%)	1.48×	3.8×
Taylor-80%	0.908	0.56B (11.2%)	1.03M (4.3%)	1.72×	6.1×
Entropy_GAP	0.868	1.56B (31.1%)	4.22M (17.5%)	1.49×	4.8×
FAC-50%	0.925	1.26B (25.1%)	11.09M (45.9%)	1.54×	2.0×
FAC-80%	0.913	0.53B (10.6%)	5.03M (20.8%)	1.77×	4.9×

Table 3. Comparison of different model compression methods for VGG-16 network on CUB_200_2011.

Method	Top-1 Acc	Top-5 Acc	FLOPs (%)	Params (%)	Speed up
Original	0.764	0.939	31.64B	304.1M	1.0×
APoZ-80%	0.556	0.833	1.40B (4.4%)	80.6M (26.5%)	5.6×
Taylor-50%	0.772	0.948	13.96B (44.1%)	91.8M (30.2%)	2.7×
Entropy-50%	0.728	0.932	9.42B (29.8%)	64.6M (21.2%)	3.5×
FAC-50%	0.786	0.950	11.62B (36.7%)	122.6M (40.3%)	3.1×
FAC-80%	0.758	0.937	6.74B (21.3%)	66.6M (21.9%)	4.2×

As shown in Tables 2 and 3, we used different algorithms for pruning VGG-16 networks in CIFAR10 and CUB.200.2011 datasets, among which the APOZ method pruned the filter of each layer with a fixed prune rate. We can see that when the pruning rate reaches 80%, the accuracy of the model drops very seriously. In the Entropy method, Luo et al. used GAP instead of the fully-connected layer, which greatly reduced the parameters of the model, but had a greater impact on the prediction accuracy of the model (which greatly increased the difficulty of convergence of the model). The Taylor method uses a pruning strategy similar to ours. It can be seen that the Taylor criterion has better performance for model size compression, but at the same pruning rate, our method has less precision loss and more excellent acceleration performance. As can be seen from the two tables, the larger the size of the input image, the greater the clipping acceleration. In the CUB.200.2011 dataset, the size of the input image is 320×320 . Our method can achieve about 4.7× reduction in FLOPs and parameters with 0.006 decrease in accuracy. When the pruning rate is 50%, the accuracy of the pruned model is even higher than the original model, and higher than other pruning methods.

By comparison, we can see that our FAC-based pruning method has better overall performance, and there is a better balance between model compression and model acceleration at the same pruning rate.

4.2 ResNet-110 Pruning

In the network structure of ResNet-110, it is divided into three hierarchies by the residual block, and the size of its corresponding feature maps are 32×32 , 16×16 , and 8×8 , respectively. According to the process of pruning for ResNets in Sect. 3.5, the pruned model for ResNet-110 was obtained on CIFAR-10. During the training process, the images are randomly cropped to 32×32 .

The overall performance of our method on pruning ResNet-110 is shown in Table 4, We prune this model with 2 different pruning rate (pruning 20%, 30% filters respectively). The best pruned model achieves 2.48× reduction in FLOPs and parameters with 0.007 decrease in accuracy. Unlike traditional CNN architectures, ResNet is more compact. There is less redundancy than the

Table 4. The pruned model for ResNet-110 on CIFAR-10 with different pruning rate.

Model	Top-1 Acc	Speed-up		Compression		Time
		#FLOPs	FLOPs%	#Para.	Para.%	
ResNet110	0.937	2.53×10^8	–	1.72×10^6	–	1.0×
FAC-20%	0.933	1.46×10^8	57.7%	1.02×10^6	59.3%	1.34×
FAC-30%	0.928	1.02×10^8	40.3%	0.72×10^6	41.9%	1.63×

VGG-16 model, so it seems more difficult to delete a large number of filters. However, when small pruning rate is adopted, our method can improve the performance of ResNet-110 to the maximum extent.

5 Conclusion

In this paper, we propose a pruning framework based on the Feature Abstraction Capabilities of filters to accelerate and compress the CNN model simultaneously in the training and inference phases. Compared with the previous pruning strategy, the pruning model has better performance. Our approach does not depend on any proprietary libraries, so it can be widely used in a variety of practical applications of current deep learning libraries.

In the future, we want to further explore the interpretability of model pruning, and then design pruning strategies that are more suitable for different visual tasks (such as semantic segmentation, target detection, image restoration, etc.). The pruned network will greatly accelerate these visual tasks.

References

1. Girshick R.: Fast R-CNN. In: The IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015)
2. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7263–7271 (2017)
3. Hu, H., Gu, J., Zhang, Z., et al.: Relation networks for object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018)
4. He, K., Gkioxari, G., Dollár, P., et al.: Mask R-CNN. In: The IEEE International Conference on Computer Vision (ICCV), pp. 2961–2969 (2017)
5. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
6. Huang, G., Liu, Z., Van Der Maaten, L., et al.: Densely connected convolutional networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
7. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)

8. Xu, Y.S., Fu, T.J., Yang, H.K., et al.: Dynamic video segmentation network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6556–6565 (2018)
9. Kuhn, M., Johnson, K.: An introduction to feature selection. In: Kuhn, M., Johnson, K. (eds.) *Applied Predictive Modeling*, pp. 487–519. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-6849-3_19
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: *International Conference on Neural Information Processing Systems (NIPS)*, pp. 1097–1105 (2012)
11. Simonyan, K., Zisserman, A.: ImageNet classification with deep convolutional neural networks. In: *International Conference on Learning Representations (ICLR)* (2015)
12. Lin, S., Ji, R., Chen, C., et al.: Holistic CNN compression via low-rank decomposition with knowledge transfer. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)* (2018)
13. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Coordinating filters for faster deep neural networks. In: *The IEEE International Conference on Computer Vision (ICCV)* (2017)
14. Wu, J., Leng, C., Wang, Y., et al.: Quantized convolutional neural networks for mobile devices. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4820–4828 (2016)
15. Jacob, B., Kligys, S., Chen, B., et al.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017)
16. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. In: *International Conference on Neural Information Processing Systems (NIPS)*, pp. 345–353 (2017)
17. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. In: *International Conference on Learning Representations (ICLR)* (2016)
18. Hu, H., Peng, R., Tai, Y.W., Tang, C.K.: Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint [arXiv:1607.03250](https://arxiv.org/abs/1607.03250)* (2016)
19. Molchanov, P., Tyree, S., Karras, T., et al.: Pruning convolutional neural networks for resource efficient inference. In: *International Conference on Learning Representations (ICLR)* (2017)
20. Luo, J.H., Wu, J.: An entropy-based pruning method for CNN compression. *arXiv preprint [arXiv:1706.05791](https://arxiv.org/abs/1706.05791)* (2017)
21. Zhou, B., Sun, Y., Bau, D., et al.: Revisiting the importance of individual units in CNNs via ablation. *arXiv preprint [arXiv: 1806.02891](https://arxiv.org/abs/1806.02891)* (2018)
22. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: *International Conference on Learning Representations (ICLR)* (2015)
23. He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778 (2016)
24. Li, H., Kadav, A., Durdanovic, I., et al.: Pruning filters for efficient convnets. In: *International Conference on Learning Representations (ICLR)* (2016)
25. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: *International Conference on Computer Vision (ICCV)* (2017)

26. Lin, M., Chen, Q., Yan, S.: Network in network. In arXiv preprint [arXiv:1312.4400](https://arxiv.org/abs/1312.4400) (2013)
27. Son, S., Nah, S., Lee, K.M.: Clustering convolutional kernels to compress deep neural networks. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11212, pp. 225–240. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01237-3_14
28. Sandler, M., Howard, A., Zhu, M., et al.: MobileNetV2: inverted residuals and linear bottlenecks. In: Computer Vision and Pattern Recognition (CVPR) (2018)