PAPER • OPEN ACCESS

Pruning the deep neural network by similar function

Pruning the deep neural network by similar function

Hanqing Liu[1], **Bo Xin**[1], **Senlin Mu**[2], **Zhangqing Zhu**[1]

[1]School of Management & Engineering, Nanjing University, 210093, China

[2]Nanjing Research Institute for Agricultural Mechanization of National Ministry of Agriculture, 210014, China

xinbo@nju.edu.cn

Abstract. Recent deep neural networks become deeper and deeper, while the demand for low computational cost model will be higher and higher. The exists pruning algorithm usually focus on pruning the network layer by layer, or using the weight sum as important score. However, these methods do not work very well. In this paper, we propose a unified framework to accelerate and compress cumbersome CNN models. We put it into an optimization problem to find a subset of the model which can produce the most comparable outputs. We concentrate on filter level pruning. Experiment shows that our method has surpassed the exists filter level pruning algorithm. Taking the network as a whole is better than pruning it layer by layer. We also have an experiment on the large scale ImageNet dataset. The result shows that we can accelerate the VGG-16 by 3.18× without accuracy drop.

1. Introduction

Deep convolution neural networks (DCNNs) have achieved a great success, especially in areas such as computer vision, e.g., image classification [2, 3, 4], semantic segmentation [8], object detection [9, 10], face recognition [11] and generative models [24, 25]. Modern state-of-the-art neural network architecture becomes more and more deep, and could gain more accuracy on the ImageNet Classification Challenge [1], e.g., AlexNet [2], VGGNet [3], GoogleNet [4], ResNets[5, 6] and DenseNets [7]. However, due to its extreme depth, these neural networks are also suffering a huge amount of calculation burden and usually require powerful computing devices such as GPUs. Since deep learning becomes more and more popular, people become more and more longing to deploy the large DCNN into small devices, such as mobile phones.

Many recent works have been proposed to compress cumbersome large CNNs in many different ways, including but not limited to neural network pruning [12, 14], weight quantization [13], knowledge distillation [15, 16], efficient architecture design [17, 18]. Since recent state-of-the-art network architecture always contains dozens of convolution layers which have a major computation cost, our method is focused on the direction of pruning convolution neural networks' filters by evaluating each filter's important score.

This paper introduces a new approach to pruning neural networks into a slimmer one and can reduce computation greatly. Our method towards to evaluate every filter's importance, and prune out the least important filters. Experiments shows that although pruned a few filters, the networks could regain its accuracy quickly by retrain them as long as the pruned filters are not important. On the contrary, if we prune out the important filters, the networks can hardly regain a high accuracy.

Contemporary researchers usually focused on filters themselves. For example, [14] uses the weight sum to evaluate filters' important score. However, our experiments showed that despite the really small filters (close to 0) which contribute little, the rest normal filters do not usually follow this rule, which means that relatively small filters can perform vital roles. Another direction of evaluating filters' importance by considering the difference between feature maps which are produced before and after pruning [12]. However, such criterion is too strict, because even if the feature maps are different from the original networks, the final output could be the same. Moreover, this method only concentrates a single layer which only contains limited information. Different layer has different jobs [26]. Our experiments will show that different layers might have different importance. Thus, focusing on single layers will ignore a lot of information. Unlike [12], our method takes a consideration of the entire network, and the result shows that particular layers could be more important than other layers. And [19] evaluate each neuron by detecting the drop of the accuracy when this neuron is pruned away. However, this criterion is relative relaxed because [15] tells us that the neural networks not only produce the most confident prediction, but also output dark knowledge, which means that a neuron contains a lot of dark knowledge may be seem not important by this criterion, because it might have no influence on the most confident output. To overcome these drawbacks, we proposed a new approach to evaluate the importance of each filter, and we prune out the least important filters. The idea hidden in this method is that we want the network to perform a similar function with the full model.

In a summary, we are looking for a way to find a subset of the original network to perform a similar function with the original full model, and the more similar, the better. We are not checking every intermediate feature maps, instead, we evaluate the final outputs.

In this paper, we introduce a weighting filters' importance method which has a perspective of the entire network, not part of it or filter itself. Our goal is to find a smaller network which can perform a similar function of the original network. As to one particular filter, we weigh its importance by measuring the difference between the functions which are implemented by pruned network and original network. If this difference is large which means that the pruned network's function is very different from the original one leads to a conclusion that this filter is relative important, while if we prune a filter generate a new network whose outputs are similar or even same with the original network's outputs means that this filter is less important. Once we weighed all the filters' importance by using this difference, we could prune out the least important ones. In a summary, we want to find a small convolution neural network which is pruned from the original neural network to perform the most similar function as the original one. To do this, we use the KL divergence to evaluate this difference and use randomly chosen data as input.

Experiments on CIFAR-10 benchmark datasets show that by using our method, we can still compress the VGG-16 [3] by the ratio about 90% with little drop of the accuracy. Compared with the method in [12, 14], our pruned network has better performance.

2. Related Work

In this section, we review some related work in several directions.

Neural Network Architecture Design. Since AlexNet [2] Ignited the interest of deep learning, people began to design neural networks by experiment. From VGGNet [3] to ResNets [5], neural networks became deeper and deeper and had more convolution layer. Recent work began to pay attention to small network architecture in order to run it on mobile devices. By using the idea of sparse connected channels (grouped channels), the state-of-the-art light model such as MobileNets [17], ShuffleNets [18] and MobileNetV2 [20] have much less computation cost compared with cumbersome models but still could maintain a good performance on ImageNet classification benchmark. In our experiment, we found that light models still have redundancy. Our goal is to compress such light model to a more efficient model.

Weight Quantization. Neural networks trained on X86 CPUs or GPUs usually have float32 or float64 weight. Weight quantization always wants to reduce the computation of a network by turning its float weight to more storage and computational efficient type, say, fixed point or even binary numbers. These lower bit numbers such as fixed point numbers can replace the original multiplication by other efficient operation such as XNOR. INQ [13] proposed a method that using a lower bit number than float could produce a comparable accuracy with the original networks. XNOR-Net [21] and Binary-Weighted-Nets [22] forces the weight in neural networks to be binary numbers which means they are restricted to {-1, 1}. Compared with the original networks, the quantized networks are relatively small and more efficient. Moreover, quantized network could be implemented on devices like FPGA. However, these approach usually facing a degradation of accuracy.

Neuron/Filter-level Pruning. A cumbersome network usually requires larger computing resources due to its large number of channels. There are several works aimed to reduce the number of channels in convolutional networks. This idea can trace back to [23]. [23] tries to weigh the importance of weight by measuring the salience of the output with or without these weight, which usually is the cross entropy loss, and uses the Taylor expansion to approximate. [12] measures the salience by detecting the difference of feature maps. [14] uses the so-called magnitude to measure the importance of parameters. [28] also uses the Taylor expansion, however, with first derivative.

Knowledge Distillation. Knowledge distillation tends to transfer knowledge from a teacher model to a student model, and, hopefully, it can bring improvement of precision of the student model. [15] uses a pre-trained cumbersome teacher network to produce so-called soft targets as input for the student network. [15] thinks that a pre-trained teacher model can provide more information than one-hot ground truth label. By taking the advantages of the teacher models, student models can achieve more accuracy. [16] provide a method that training the student network by mimicking the teacher network, and gain the impressive improvement. This method tends to use attention map to complete knowledge transfer.

3. Method

In this section, we will introduce our approach in detail. First we will present our idea behind this method which we called *similar function*. Next we will introduce our framework and algorithm.

3.1 Similar function

Pruning is a popular way to reduce complexity of the cumbersome networks. Our method focuses on filter/neuron-level pruning which means that we prune an entire filter away once we believe that they are unimportant. And our method requires pre-trained models.

We use randomly chosen data, denoted as X, in the dataset which the pre-trained model was trained on as input. Our target is to find a subset of the whole parameters $w \in \theta$, to produce the most similar outputs of the original networks. We use the Kullback–Leibler divergence to calculate this similarity. Thus we also use softmax function to turn the logits which outputted by the network into probability distribution:

$$D(X,\theta) = \phi[f(X,\theta)] \tag{1}$$

Where $\phi(*)$ denotes the softmax function, and $f(*)$ denotes the neural network. And θ represents the corresponding parameters of the network.

And we use $D(X,\theta)$ to calculate the KL divergence:

$$L = KL[D(X,w) \,||\, D(X,\theta)] \tag{2}$$

Thus, we want to find subset of parameters $w \in \theta$ to generate the smallest L:

$$\begin{aligned} &\underset{w \subset \theta}{\arg\min}\ L = KL[D(X,w) \,||\, D(X,\theta)] \\ &s.t.\ |w| = r \times |\theta|,\ r \in (0,1) \end{aligned} \tag{3}$$

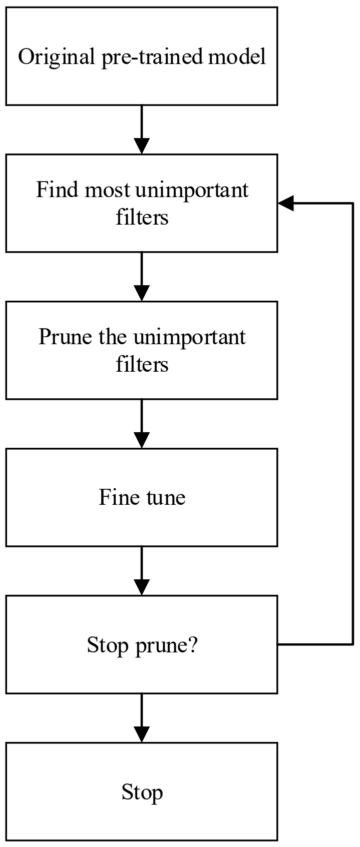
Where $r \in (0,1)$ is the pruning ratio which we treat it as a hyperparameter.

Unfortunately, this optimization problem (2) is NP hard, which means that we cannot solve it directly. Thus, we use a greedy algorithm to find an approximate solution.

3.2 Framework

Although we cannot directly find an optimal subset of θ to produce the minimum L, we can find an optimal subset $\hat{\theta}$ which excludes one filter. Thus, it provides an opportunity to solve Eq.3 greedily. After we traverse all the filters in the network by deleting them, we can find the most unimportant ones. After pruning them, the network will usually lose its accuracy. Thus, we are required to retrain the pruned network to regain the accuracy. And we do it again. As illustrated in figure 1, we conclude our framework to this:

1) Each filter's important score is evaluated;
2) The most unimportant filters are pruned;
3) Fine Tuning.


Original pre-trained model
Find most unimportant filters
Prune the unimportant filters
Fine tune
Stop prune?
Stop
Figure 1. Frame work of pruning algorithm.

3.3 Algorithm

We present a greedy algorithm to find an approximate solution of the optimization problem of eq. 2. We use a triplet $< X, w_i, L_i >$ to denote each filter's process, where $X \in \mathbb{R}^{C\times H\times W}$ is the input tensor which has height H, width W and C channels. And $w_i \in \mathbb{R}^{N_1\times K\times K}$ is a set of filters in the pre-trained model which has N_1 input channels and with $K\times K$ kernel size. L_i is a scalar, and it represents the difference between outputs generated by the pre-trained model with or without w_i by feeding data X. We use the Kullback–Leibler divergence to calculate L_i:

$$L_i = KL[D(X,\theta) \,||\, D(X,\hat{\theta})] \tag{4}$$

Where $w_i \in \theta, w_i \notin \hat{\theta}$ denotes the parameters of the networks. And the optimal subset:

$$w_{w\subset\theta} = \arg\min \frac{1}{n}\sum_{j=1}^{n} KL[D(X_j, w) \,||\, D(X_j, \theta)] \tag{5}$$

Once we traverse all the filters, we will get the set L_i, $i \in \{1,2,...,S\}$. We can find the least important filters. The pseudo code can be found in Algorithm 1.

4. Experiment

We study our algorithm's performance in this section. We will compare with some other several filter level pruning algorithm and show that the performance of our algorithm is far better than others'. Then we will show our result on CIFAR-10 dataset with VGG-16 [3].

Algorithm 1. A greedy algorithm to optimize Eq. 3

```
Input: Data $X$; Pre-trained model $M$; Termination condition $r$; Pruning filter quantity in one turn $a$; Training set $\{x, y\}$
Output: Pruned model $M'$;
while not $r$:
    $S \leftarrow \varnothing$
    for each convolutional layer $\Lambda$ in $M$:
        $S \leftarrow S \cup \Lambda.filters$
    $L \leftarrow \varnothing$
    for each $i$ in $S$:
        $L_i = \frac{1}{n}\sum_{j=1}^{n} KL[D(X_j, w_i) \,||\, D(X_j, \theta)]$
        $L \leftarrow L \cup L_i$
    $\hat{L} = topa(L)$
    $M' \leftarrow$ Prune $\hat{L}$
    Fine tune $M'$ by $\{x, y\}$
    $M \leftarrow M'$
```

4.1 Performance comparison

We focus our comparison on the different filter level pruning algorithm as follows:

(1) Random. This is our base line. Because random pruning is the simplest way to prune the network. Without priori knowledge, one could only prune the network randomly. However, we have the pre-trained parameters, which means that we cannot prune the network worse than randomly pruning.

(2) Magnitude [14]. This criterion considers the less important filters are those with smaller parameters. Because those filters produce poorer activation than others'. So each filter's important score is $s_i = \sum |W(i,:,:,:)|$.

(3) Thinet [12]. This criterion is similar to ours. It considers the Euclidean distance between the feature map with and without particular filters as the important scores. And similarly, Thinet solve the optimization problem greedily. The difference between our algorithm and Thinet is that we evaluate every filter in the entire network while Thinet prune the network layer by layer.

To compare these methods, we use the CIFAR-10 benchmark. CIFAR-10 contains 10 classes. Each class has 6000 32×32 pictures. To avoid overfitting, we removed the last fully connected layers in VGG-16 and replaced with a global average pooling layer along with a fully connected layer to classify. Each pruning strategy requires a pre-trained model. We use the same pre-trained model. After pruning, one fine tune epoch is employed. Except the difference of pruning strategy, everything else is same. The pre-trained VGG-16 has 6.7% of top-1 error. The result is presented in figure 2.

Figure 2 shows the result of each pruning strategy's performance. We evaluate four different strategies. In the experiment, we found that the accuracy once we pruned the network without fine tuning could also reveal the performance. And our similar function method could achieve the best accuracy without fine tune, which means that our method can retain the most of information. While random pruning and weight sum strategy is the worst. It seems that magnitude does not possess a strong correlation with weight's importance. A small parameter could have a large impact on the outputs. Thus, the result of magnitude strategy is as poor as the base line is reasonable.

As to the comparison of Thinet and our similar function method, the reason why our method is slightly better is probably that our method takes every filter into account, and prune the network in a whole perspective instead of pruning it layer by layer. The remaining networks show that our method could reveal not only the importance of each layer, but also each filter's. And the result shows that the last layer of each group (conv1-2, conv2-2, conv3-3) is more important than others. The least important layers will remain fewer filters than those important ones. Thus, our method shows better and more robust result.

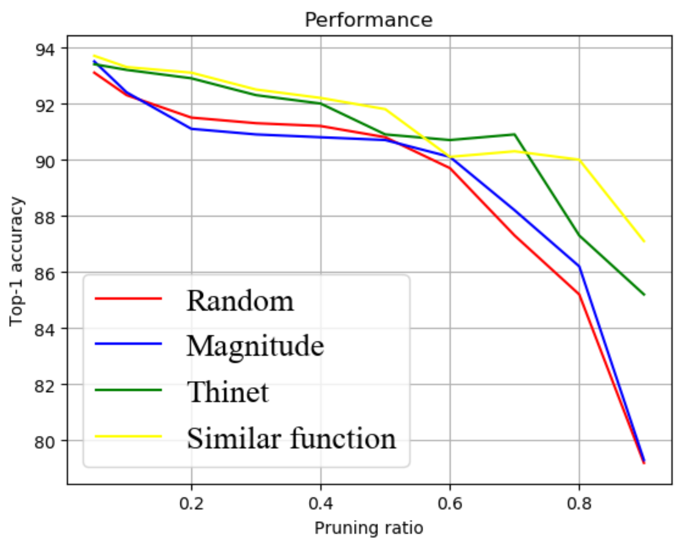


Figure 2. The result of performance comparison with different pruning strategy. Based on benchmark CIFAR-10.

4.2 VGG-16 on ImageNet

In this section, we will evaluate the performance of our similar function method on the large scale ImageNet dataset [1]. ILSCVR-12 is the benchmark of the state-of-the-art networks. It contains about one million images, and these images form 1000 classes. We use 1000 images (each class has one image) to find the importance of each filter via algorithm 1. Also, we use a pre-trained model whose top-1 accuracy is 68.34%. We use the standard training set and validation set to train and validate our models.

During fine tuning, we use SGD algorithm to optimize the model with learning rate from 10^{-3} reduces to 10^{-4}. The mini-batch size is 256. Also, data augmentation is employed. The augmentation contains random crop from 256×256 images and random flip. During the validation, only random crop is used. The pruning ratio is 0.5. We only prune the convolution layers, while the other parameter is same as the original VGG-16 [3]. The result on ImageNet is showed in table 1. Our method pruned fewer filters in the last layer of each group, so it will have more parameters than Thinet. The interesting thing is that after pruning, the network gains a lifting of accuracy. Probably due to discarding the least important filters, the network could somehow avoid the overfitting.

Table 1 Pruning VGG-16 on ImageNet.

Model	Top-1 accuracy(%)	Top-5 accuracy(%)	#parameters(M)	FLOPs(B)
Original VGG	68.34	88.44	138.34	30.94
Similar function	69.62	89.62	131.86	9.73
Thinet	69.11	89.32	131.44	9.58
Training from scratch	66.91	87.32	131.86	9.73

We also train a new model whose architecture is same as the pruned one from scratch. It shows that this network has a lower accuracy than the pruned one, which means that pruning algorithm is better than training scratch. It reveals the fact that although two networks have same architecture, they could have different ability of learning. The only difference of the networks is initialization of parameters. One has part of pre-trained parameters while the other only has randomly initialized parameters. This only difference could bring huge accuracy gap.

We also calculate the FLOPs of each model. When combine with the number of parameters, it shows that convolution layers often have fewer parameters but have huge computation cost. By focusing on pruning convolution layers, one can reduce the inference time dramatically. On the other hand, fully connected layers usually have massive parameter size. One could replace it with global average pooling to reduce the amount of parameters. It can also help alleviating overfitting.

5. Discussion

Cun Y L et al. proposed a method to prune the network by using the information of the second derivative which is also known as Hessian [27]. While [28] proposed an approximate method to determinate which filter is unimportant by utilizing Tylor expansion, because the Hessian is usually unquantifiable. This method, though, does not match the accuracy of the greedy algorithm, save a lot of computing time. Using the first derivative to evaluate each filter's importance is far more efficient than the so-called oracle pruning. The method proposed in this paper is an oracle method indeed. Thus, our method is less efficient than the Tylor expansion approximating. Although the first thing to consider is inference time, the pruning cost still maters. In the future work, we will dedicate to a more economical method, including but not limited to Tylor expansion approximation.

Acknowledgments

This work was supported by the National Key Research and Development Program (NO.2016YFD0702100), the National Natural Science Foundation of China (Nos.61432008and 71732003), the Postgraduate Research and Practice Innovation Program of Jiangsu Province (No.SJCX17-0004).

Reference

[1] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database[C]//Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009: 248-255.
[2] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.
[3] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[4] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions[C]. Cvpr, 2015.
[5] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
[6] He K, Zhang X, Ren S, et al. Identity mappings in deep residual networks[C]//European Conference on Computer Vision. Springer, Cham, 2016: 630-645.

[7] Huang G, Liu Z, Weinberger K Q, et al. Densely connected convolutional networks[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, 1(2): 3.
[8] Chen L C, Papandreou G, Kokkinos I, et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs[J]. IEEE transactions on pattern analysis and machine intelligence, 2018, 40(4): 834-848.
[9] Girshick R. Fast r-cnn[J]. arXiv preprint arXiv:1504.08083, 2015.
[10] Liu W, Anguelov D, Erhan D, et al. Ssd: Single shot multibox detector[C]//European conference on computer vision. Springer, Cham, 2016: 21-37.
[11] Taigman Y, Yang M, Ranzato M A, et al. Deepface: Closing the gap to human-level performance in face verification[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2014: 1701-1708.
[12] Luo J H, Wu J, Lin W. Thinet: A filter level pruning method for deep neural network compression[J]. arXiv preprint arXiv:1707.06342, 2017.
[13] Zhou A, Yao A, Guo Y, et al. Incremental network quantization: Towards lossless cnns with low-precision weights[J]. arXiv preprint arXiv:1702.03044, 2017.
[14] Li H, Kadav A, Durdanovic I, et al. Pruning filters for efficient convnets[J]. arXiv preprint arXiv:1608.08710, 2016.
[15] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint arXiv:1503.02531, 2015.
[16] Romero A, Ballas N, Kahou S E, et al. Fitnets: Hints for thin deep nets[J]. arXiv preprint arXiv:1412.6550, 2014.
[17] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:1704.04861, 2017.
[18] Zhang X, Zhou X, Lin M, et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices[J]. arXiv preprint arXiv:1707.01083, 2017.
[19] Morcos A S, Barrett D G T, Rabinowitz N C, et al. On the importance of single directions for generalization[J]. arXiv preprint arXiv:1803.06959, 2018.
[20] Sandler M, Howard A, Zhu M, et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[C]//Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 4510-4520.
[21] Rastegari M, Ordonez V, Redmon J, et al. Xnor-net: Imagenet classification using binary convolutional neural networks[C]//European Conference on Computer Vision. Springer, Cham, 2016: 525-542.
[22] Courbariaux M, Hubara I, Soudry D, et al. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1[J]. arXiv preprint arXiv:1602.02830, 2016.
[23] LeCun Y, Cortes C, Burges C J. MNIST handwritten digit database[J]. AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist, 2010, 2.
[24] Goodfellow I, Pouget-Abadie J, Mirza M, et al. Generative adversarial nets[C]//Advances in neural information processing systems. 2014: 2672-2680.
[25] Kingma D P, Welling M. Auto-encoding variational bayes[J]. arXiv preprint arXiv:1312.6114, 2013.
[26] Zeiler M D, Fergus R. Visualizing and Understanding Convolutional Networks[C]// European Conference on Computer Vision. Springer, Cham, 2014:818-833.
[27] Cun Y L, Denker J S, Solla S A. Optimal brain damage[C]// International Conference on Neural Information Processing Systems. MIT Press, 1989:598-605.
[28] Molchanov P, Tyree S, Karras T, et al. Pruning Convolutional Neural Networks for Resource Efficient Inference[J]. 2016.