LETTER

Filter Level Pruning Based on Similar Feature Extraction for Convolutional Neural Networks

Lianqiang LI[†], *Student Member*, **Yuhui XU**[†], *and* **Jie ZHU**[†a)], *Nonmembers*

SUMMARY This paper introduces a filter level pruning method based on similar feature extraction for compressing and accelerating the convolutional neural networks by k-means++ algorithm. In contrast to other pruning methods, the proposed method would analyze the similarities in recognizing features among filters rather than evaluate the importance of filters to prune the redundant ones. This strategy would be more reasonable and effective. Furthermore, our method does not result in unstructured network. As a result, it needs not extra sparse representation and could be efficiently supported by any off-the-shelf deep learning libraries. Experimental results show that our filter pruning method could reduce the number of parameters and the amount of computational costs in Lenet-5 by a factor of 17.9× with only 0.3% accuracy loss.

key words: *CNNs, filter, pruning, feature extraction, k-means++, structured*

1. Introduction

Convolutional Neural Networks (CNN) are famous for their success in various of artificial intelligence tasks [1]–[3]. However, most of parameters in CNN models are usually redundant. This redundancy makes it hard for CNN models to be deployed in both computation limited and memory limited devices.

To address this issue, there are a number of works have been proposed [4]–[7]. Among them, pruning is one of the most outstanding methods to alleviate the complexity of CNNs. Han et al. [7] introduced a simple yet effective pruning strategy. It was reported that Lenet-5 could be compressed by a factor of 12×. But such improvements cannot transfer directly to faster inference by modern deep learning libraries due to its unstructured pattern. To avoid the shortcoming, [8], [9] have studied structured pruning methods based on removing filters according to their importance indices. However, these importance criteria do not yield satisfactory results because of their unreasonable tactics.

In this paper, we present a novel filter level pruning method for convolutional neural networks by k-means++ [10] algorithm. First, we employ the k-means++ algorithm to enforce the filters to enter specific clusters. Second, we will retain the filter which is the closest to the cluster center and prune the others in every cluster. Then the pruned model will be fine-tuned to recover accuracy. Our approach aims to find the correlation among filters in feature extraction and prune the analogous ones. Naturally, we could obtain more lightweight CNN models. We evaluate the proposed pruning strategy by the classical CNN model, Lenet-5 [1], on MNIST. Results show that our method has the ability to reduce the number of parameters and the amount of inference costs in Lenet-5 by a factor of 17.9× with only 0.3% accuracy loss, which has significant improvements than state-of-the-arts.

The rest of this paper is organized as follows. We describe the related works briefly in Sect. 2. Section 3 introduces the proposed filter level pruning in detail. Section 4 presents the experimental results. Finally, we summarize the conclusions and present our future research direction in Sect. 5.

2. Related Works

In this section, we will give a brief introduction to some of pruning methods.

Srinivas et al. [6] proposed a data-free approach to carry out CNN models compression. They managed to avoid employing any training data by minimizing the expected squared difference of logits. Compared to the former works, they removed a neuron at a time instead of removing some weights at a time. All of the reduction is implemented on fully connected layers. There is no pruning operation on convolutional layers. In practice, convolutional layers are the computation intensive layers. Thus, this method has great limitations.

Han et al. [7] introduced a pruning approach by applying L1/L2-norm regularizations to remove the small weights. The basic idea of their work is that a weight connectivity should be pruned if it is less than a predefined threshold. Both convolutional layers and fully connected layers could be pruned by this strategy. Their method achieved the compression ratio on Lenet-5 by a factor of 12×. Although the performance is inspiring, the pruning would result in unstructured pattern in weights. This shortcoming requires long fine tuning time which may exceed the original network training by a factor of 3×.

In line with our work, several works [8], [9] have explored structured pruning by removing filters. The main challenge is to evaluate the importance of filters. Li et al. [8] judged the importance of each filter by calculating absolute weight sum. This criterion is simple and naive. Small weights also have large effects on the final accuracy. Hu et al. [9] introduced Average Percentage of Zeros to assess the

---

Manuscript received November 9, 2017.
Manuscript revised December 19, 2017.
Manuscript publicized January 18, 2018.

[†] The authors are with the Dept. of Electronic Engineering, Shanghai Jiao Tong University (SJTU), Shanghai, 200240, China.
a) E-mail: zhujie@sjtu.edu.cn (Corresponding author)

DOI: 10.1587/transinf.2017EDL8248

importance of each filter. A filter may be selected as unimportant one if most outputs of the filter are zero. Their tactic seems more reasonable. Nevertheless, this work needs lots of extra calculations and the compression ratio is not satisfying.

3. The Proposed Filter Level Pruning for CNNs

The input feature map in convolutional layer i could be viewed as a 3-D tensor $\boldsymbol{I_i} \in R^{N_i \times H_i \times W_i}$, where N_i, H_i and W_i are the number of channels, height and width of input feture map, respectively. The weights in this layer could be expressed as a 4-D tensor $\boldsymbol{W_i} \in R^{N_{i+1} \times N_i \times h_i \times w_i}$, where N_{i+1} is the number of filters, h_i and w_i are the spatial height and width of filters. $\boldsymbol{W_i}$ consists of a bunch of 3-D filters $\boldsymbol{W_i^n}$ $(1 \le n \le N_i)$. They perform convolutional operation with the input could produce another 3-D tensor, output feature map, $\boldsymbol{O_i} \in R^{N_{i+1} \times H_o \times W_o}$, where H_o and W_o are the height and width of the output feature map. $\boldsymbol{O_i}$ is also the input for the $(i + 1)$th convolutional layer.

As we were studying the prior structured pruning methods, we found two main deficiencies. On the one hand, their important indices are little unconvincing. It may be logical that we could prune some parameters if their weights are smaller than a specific threshold in fully connected layers. But filters with smaller sum of weights do not mean less important than larger ones in convolutional layers. Different filters could capture different features. On the other hand, their optimization procedures are too difficult to converge in a short time because there are several constraints on weights at one time. To solve these drawback, we improve structured pruning method by more reasonable and effective strategy. The core of our filter level pruning method is shown in Algorithm 1.

Algorithm 1 Filter level pruning based on feature selection by k-means++

```
Input: A trained CNN model, target layer $i$ and tolerance threshold $\alpha$ for
    accuracy loss
Output: Pruned CNN model
Initialize k=$N_i$-1
repeat
    Use k-means++ to force the $\boldsymbol{W_i^n}$ $(1 \le n \le N_i)$ into k clusters
    Keep the filter which is the most closet to centroid for each cluster,
    prune the others and their output feature maps
    Fine tuning the pruned model as training process
    k=k-1
until Accuracy loss is larger than $\alpha$
```

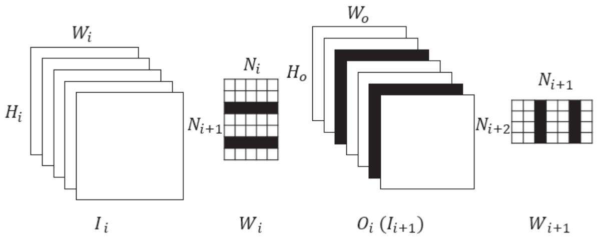


Fig. 1 The illustration of the proposed pruning method

We merge filters with highly correlated relationships into one instead of removing filters by additional regularizations. Specifically, we use k-means++ algorithm to identify the similar filters for each layer of trained CNN models, and we only keep the filter which is the closest to the centroid for every cluster. The motivation of this idea is straightforward. If some filters belong to one cluster, they are similar to each other in capturing features. The proposed filter level pruning can reduce feature map directly, which condenses CNN models into thinner ones. The thinner models are able to gain acceleration via modern deep learning libraries because no additional sparse representation is required. Figure 1 shows the illustration of our method briefly. If we prune $(N_{i+1} - k)$ filters for layer i, we could reduce $(N_{i+1} - k)N_i h_i w_i H_o W_o$ operations as the pruned filers' corresponding feature maps would be removed. The channels which are related to those removed feature maps in the next convolutional layer would also be removed. Actually, keeping k filters of layer i will reduce $(N_{i+1} - k)/N_{i+1}$ of the computation costs for both layers i and $i + 1$.

Our method could also be used in fully connected layer. However, we choose to follow the approach in [7]. There are two reasons. First, we can get higher compression ratio. Furthermore, the function of fully connected layer is to deal with different combinations of features learned from convolutional layer. The method in [7] would be more sensible here. Despite it results in unstructured weight matrices, computational costs in fully connected layer are so small that we can ignore the unstructured influence.

4. Experiments

In this section, we evaluate the effectiveness of our method in the classical CNN model, LeNet-5, on MNIST by Caffe [11]. Lenet-5 has two convolutional layers and two fully connected layers, which achieves 0.87% error rate on MNIST. We employ the pipeline that pruning a trained model layer by layer and then fine tuning iteratively. Our fine tuning is same to the training process, without introducing any other tricks except the learning rate is set to be 0.1 times as large as the original one. The tolerance threshold for accuracy loss in Algorithm 1 is 0.3%.

Table 1 gives the performance of our method[†] and the reported performance in [6], [7] and [9] in terms of the percentage of retained parameters (Paras), the percentage of retained floating point operations (FLOPs) and the final accuracy. Please note that the approach in [8] is similar to ours, and its authors did not present results on Lenet-5. We reproduce their algorithm and apply the same pruning ratios of ours into [8] to get a more intuitive contrast.

The results in Table 1 show that the proposed filter level pruning method by k-means++ has advantages over the state-of-the-arts with respect to the number of param-

---

[†] More details are in https://github.com/shiyuetianqiang

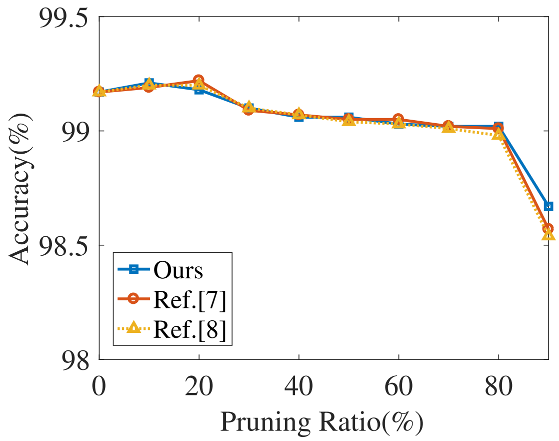
Table 1 The performance of different pruning methods on LeNet-5

Method Layer		Ours	Ref. [6]	Ref. [7]	Ref. [8]	Ref. [9]
Conv1	Paras (%)	**20**	100	66	**20**	100
	FLOPs (%)	**20**	100	66	**20**	100
Conv2	Paras (%)	16	100	**12**	16	48
	FLOPs (%)	**3.2**	100	10	**3.2**	48
Fc1	Paras (%)	**5**	6	8	**5**	50.4
	FLOPs (%)	**5**	6	6	**5**	50.4
Fc2	Paras (%)	**5**	100	19	**5**	100
	FLOPs (%)	**5**	100	10	**5**	100
Total	Paras (%)	**5.66**	12.66	8	**5.66**	50.89
	FLOPs (%)	**5.63**	83.6	16	**5.63**	55.06
Accuracy (%)		98.83	94.18	99.23	98.17	**99.26**

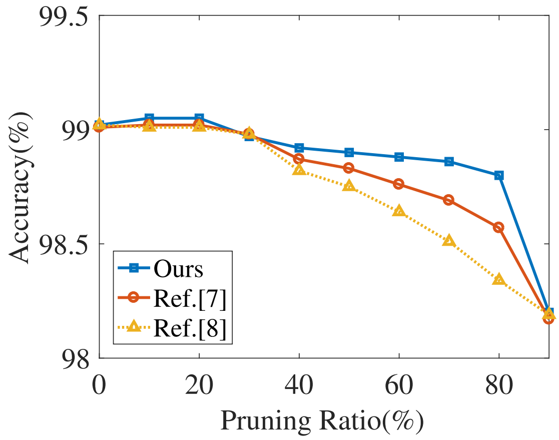
eters and the amount of computational costs. More concretely, our method has the ability to reduce the number of parameters and FLOPs in Lenet-5 on MNIST by a factor of 17.9× with only 0.3% accuracy loss due to its cross-layer linkage of structured pruning. As for [6], it only concentrates on pruning the first fully connected layer, which helps it to retain 12.66% parameters compared to original Lenet-5. But the computational complexity does not decrease too much and the loss in accuracy is unacceptable due to its reckless pruning strategy. As far as [7] is concerned, although its accuracy is 0.07% higher than the baseline, the number of retained parameters and FLOPs is still more than ours. More crucial is that this approach needs extra sparse representation and special hardware for acceleration. The work in [8] obtains the same compression ratios as ours in parameters and computational complexity because it also employs filter level pruning method. However, the pruning strategy in [8] is too native to retain the accuracy. The accuracy in [9] is the highest one, which is 0.13% higher than the baseline. In fact, it only deals with the second convolutional layer and the first fully connected layer. The final pruning ratio is relatively low. Our method, [7] and [8] could achieve similar or even better accuracy performance in such a low pruning ratio.

To ensure fair comparison and validate the superiority of the proposed method thoroughly, we explore the accuracy of our method, [7] and [8] against the pruning ratio on the convolutional layers of LeNet-5. The three algorithms carry out pruning on all of the convolutional layers. Please note that the performance of the second convolutional layer is obtained on the basis of 80% pruning ratio for the first convolutional layer. Results are shown in Fig. 2.

We could see from Fig. 2 (a) that the accuracy of the three algorithms would drop with the increase of pruning ratio in the first convolutional layer. In addition, there is no apparent difference among them. We could also see that the accuracy of the proposed algorithm has slight improvements when pruning ratio is low according to Fig. 2 (b). This may be because that our approach avoids overfitting to some extent under such circumstances. It is clear that the accuracy performance of our method is superior to the others when pruning ratio is from 40% to 80% while the accuracy performance of the three algorithms is similar to each other when pruning ratio is 90%. The explanation is that enforcing highly correlated filters to enter one cluster would not hurt the accuracy heavily as the fine tuning process could make up for it easily. However, with the number of clustering centroids becomes less, the fine tuning procedure has more difficulties in compensating for the accuracy loss. As a result, the performance of all of the algorithms including ours would drop heavily at last.



(a) Comparison of accuracy loss with the same sparsity in the first convolutional layer



(b) Comparison of accuracy loss with the same sparsity in the second convolutional layer

Fig. 2 Comparison of accuracy loss with the same sparsity for LeNet-5

5. Conclusion

In this work, we put forward to a filter level pruning method based on similar feature extraction for convolutional neural networks by k-means++ algorithm. The proposed method belongs to structured pruning. It leverages similarities among the filters in a layer to detect redundant ones and merge them. Compared with the exiting methods, our method shows significant improvements both in the number of parameters and the amount of the FLOPs with negligible loss in accuracy.

In the future, we would like to carry out our method on large convolutional neural networks. Moreover, devising another strategy for filter selection is also worthy to be studied. In a word, filter level pruning is a promising approach for compression and acceleration, but how to optimize its strategy still needs a lot of work.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (grant Nos.61371147, and 11433002).

References

[1] 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.2278–2324, 1998.

[2] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in Neural Information Processing Systems, pp.1097–1105, 2012.

[3] T. Matsuo and N. Shimada, "Construction of latent descriptor space and inference model of hand-object interactions," IEICE Transactions on Information and Systems, vol.E100-D, no.6, pp.1350–1359, 2017.

[4] J.H. Ko, B. Mudassar, T. Na, and S. Mukhopadhyay, "Design of an energy-efficient accelerator for training of convolutional neural networks using frequency-domain computation," Proceedings of the 54th Annual Design Automation Conference 2017, DAC '17, New York, NY, USA, pp.59:1–59:6, ACM, 2017.

[5] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," arXiv preprint arXiv:1702.03044, 2017.

[6] S. Srinivas and R.V. Babu, "Data-free parameter pruning for deep neural networks," Procedings of the British Machine Vision Conference 2015, pp.31.1–31.12, 2015.

[7] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," Advances in Neural Information Processing Systems, pp.1135–1143, 2015.

[8] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H.P. Graf, "Pruning filters for efficient convnets," arXiv preprint arXiv:1608.08710, 2016.

[9] H. Hu, R. Peng, Y.W. Tai, and C.K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," arXiv preprint arXiv:1607.03250, 2016.

[10] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, pp.1027–1035, 2007.

[11] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," arXiv preprint arXiv:1408.5093, 2014.