Demystifying Neural Network Filter Pruning

Zhuwei Qin[1], **Fuxun Yu**[2], **Chenchen Liu**[3] **Xiang Chen**[4]
[1,2,4]Department of Electrical Computer Engineering, George Mason University, Fairfax, VA 22030
[3]Department of Electrical Computer Engineering, Clarkson University, Potsdam, NY 13699
`zqin@gmu.edu`[1], `fyu2@gmu.edu`[2], `chliu@clarkson.edu`[3], `xchen26@gmu.edu`[4]

Abstract

Based on filter magnitude ranking (e.g. ℓ_1 norm), conventional filter pruning methods for Convolutional Neural Networks (CNNs) have been proved with great effectiveness in computation load reduction. Although effective, these methods are rarely analyzed in a perspective of filter functionality. In this work, we explore the filter pruning and the retraining through qualitative filter functionality interpretation. We find that the filter magnitude based method fails to eliminate the filters with repetitive functionality. And the retraining phase is actually used to reconstruct the remained filters for functionality compensation for the wrongly-pruned critical filters. With a proposed functionality-oriented pruning method, we further testify that, by precisely addressing the filter functionality redundancy, a CNN can be pruned without considerable accuracy drop, and the retraining phase is unnecessary.

1 Introduction

The great success of CNN is benefited from its complex algorithm and architecture at a cost of intensive computation load. Therefore, many CNN filter pruning works have been proposed to alleviate this issue [1-7]. While these works demonstrated expected performance, most of them are merely based on quantitative ranking of the filters' magnitude. However, how the magnitude ranking really reflect the filters' functionality and contribution to the classification still remains a lack of research. Therefore, in this work, we utilized the filter visualization technique to interpret one of the most representative magnitude ranking based filter pruning method (i.e. ℓ_1 norm [1]), as well as the retraining process in a perspective of filter functionality. From the analysis, we find that:

- The magnitude ranking based filter pruning method fails to select filters with functionality redundancy, resulting in inevitable accuracy drop;
- Filters suffer from significant functionality changes during retraining phase, which indicates that the magnitude ranking based filter pruning method may defect certain critical filter functionality and requires filter reconstruction for compensation;
- With functionality-oriented pruning method, a CNN can actually be pruned without considerable accuracy drop and the retraining phase is relatively unnecessary.

2 Analyzing Magnitude Ranking based Filter Pruning through Filter Functionality Interpretation

In this section, we utilize a well-established CNN visualization technique [8] – Activation Maximization (AM) to interpret the filter functionality. In the CNN visualization analysis, the filter functionality is usually defined as the feature extraction preference, which can be represented by a synthesized input image that causes the highest feature map activation for a filter during the convolution process. Mathematically, the visualization process can be formulated as:

$$V(F_i^l) = \arg\max_X A_i^l(X), \qquad X \leftarrow X + \eta \cdot \frac{\partial A_i^l(X)}{\partial X}, \tag{1}$$

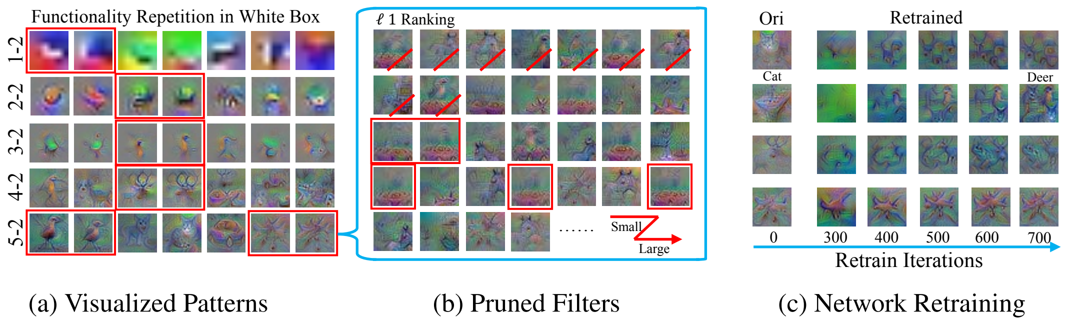

Figure 1: Case study of the ℓ_1 based filter pruning on the Conv5_2 of VGG-16 and network retraining interpretation based on filter functionality.

where $A_i^l(X)$ is the activation of filter F_i^l from an input image X, η is the gradient ascent step size. With X initialized as an input image of random noises, each pixel of this input image is iteratively changed along the $\partial A_i^l(X)/\partial X$ increment direction to achieve the highest activation. Eventually, X demonstrates a specific visualized pattern $V(F_i^l)$, which contains the filter's most sensitive input features with certain semantics, and represents the filter's functional preference for feature extraction.

For preliminary demonstration, an VGG-16 model trained on the CIFAR-10 dataset is adopted [1]: (1) Fig. **??** (a) demonstrates filters' visualized patterns from different layers for functionality interpretation analysis. From Fig. **??** (a), we find many similar patterns inside each layer (denoted in red blocks). The filters with similar functionality may repetitively extract the same feature and introduce significant network redundancy. (2) Fig. **??** (b) demonstrates a case study of a pruned layer (Conv5_2) based on the magnitude ranking of ℓ_1 norm. As shown in the figure, the filters' visualized patterns are ranked by the ℓ_1 norm in an ascending order, where the pruned filters are marked by red slashes. We can observe that the ℓ_1 norm based pruning preserves all the filters with high magnitudes, even significant functionality repetition exists among those filters. Therefore, the ℓ_1 norm based pruning fails to address the functionality redundancy in the model.

According to the functionality interpretation, the filters with small magnitudes could also demonstrate distinct feature extraction preferences, and contribute to the feature extraction integrity and diversity. The magnitude ranking based pruning method may overlook the significance of small filters and defect the information retrieval for critical features, resulting in inevitable accuracy drop.

3 Filter Functionality Transition During Retraining Phase

In most neural network pruning works, the retraining phase is a mandatory operation for maintaining the accuracy performance. However, the iteratively pruning and retraining operation also introduce computation cost. In this section, we analyze the mechanism and necessity of the retraining phase.

As shown in Fig. **??** (c), we randomly select four filters that have not been pruned by the ℓ_1 norm method. Then we use the visualization to revel the filters' original functionality and the functionality transition during different retraining iterations, e.g. every 100 iterations. From Fig. **??** (c), we can observe that: (1) For most filters, the visualized patterns are dramatically changed by the ℓ_1 morn

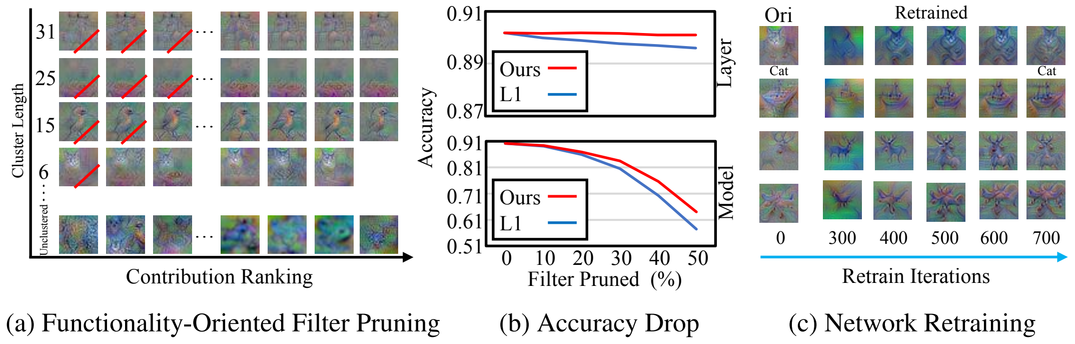

Figure 2: Case study of the filter functionality-oriented filter pruning on the Conv5_1 of VGG-16.

based pruning method. For example, the content of the visualized patterns in the first row changes from a cat to a deer. Such changes indicate that the retraining phase eventually reconstruct the remaining filters' functionality to compensate the accuracy drop. (2) During the retraining iterations, the filters' functionality construction is gradually implemented, which indicates that a certain amount of retraining iterations and the corresponding computation cost are inevitable.

Therefore, the nature of retraining process is to dramatically reconstruct the network rather than filter functionality fine-tuning. Based on the previous analysis, such reconstruction might be introduced by missing filters with significant functionality but small magnitude, and the remaining filters are reconstructed for functionality compensation.

4 Functionality-Oriented Filter Pruning

Different from the ℓ_1 norm based filter pruning method, we propose a functionality-oriented filter pruning method, which are expected to precisely reduce the filter functionality redundancy.

Fig. **??** (a) illustrates an intuitive example of our method in the layer Conv5_1: Each row represents one filter cluster, where the filters with similar functionalities are grouped by applying K-means analysis to the Euclidean distance of the visualized patterns. The last row shows the filters with extremely minimal similarity to each other, which are not considered for pruning due to their possible instinct functionality. As each cluster contain different filter numbers, the filters are sorted by their contribution index γ_i, which is evaluated by the back-propagation gradients analysis:

$$\gamma_i = \frac{1}{N}\sum_{n=1}^{N}\left\|\frac{\partial Z}{\partial A_i(x_n)}\right\|, \tag{2}$$

where the Z and $A_i(x_n)$ is the CNN output and filter i's activation for each test image n respectively. Given certain filter pruning amount, the relative pruning rate for each cluster can be determined by cluster's volume size: the cluster with more repetitive filters will be pruned more aggressively. In each cluster, the filters with least contribution will be pruned first.

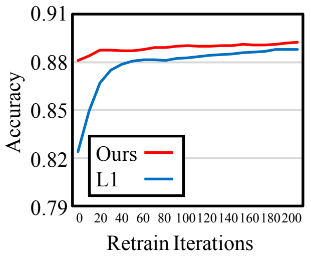

Figure 3: Pruned Model Accuracy Recovery

The aforementioned method can be applied to each convolutional layers concurrently to reduce the whole CNN functionality redundancy. Fig. **??** (b) shows the layer-wise (Conv5_1) and model-wise CNN accuracy drop under different pruning ratio respectively. We can see that our method has a slower accuracy drop compared with the ℓ_1 norm based pruning method in both layer-wise and model-wise pruning. In the Fig. **??** (c), the retraining process of our method is also qualitatively evaluated. We can observed that, regardless the retraining iterations, the remained filters' functionality remain unchanged. For example, the content of visualized pattern in the first row is still a cat at iteration 700, while the filter functionality is even clearer after retraining. That explains why our pruning method has less accuracy drop.

Meanwhile, we also quantitatively evaluate the retraining phase's effectiveness with the proposed method. As shown in Fig. **??**, the red line represents the pruned model accuracy recovery based on our proposed method whereas the blue line represents the ℓ_1 norm based pruning method. We can see that the model pruned by our method demonstrates much less impact on the accuracy. And the accuracy change is mainly introduced by filter fine-tuning, which also takes much less iteration numbers. Therefore, with more interpretable and accurate repetitive filter identification and functionality-oriented pruning, the costly retraining phase becomes less necessary.

5 Conclusion

In this work, we interpret the magnitude filter pruning including retraining phase in a perspective of filter functionality. We show that the filter magnitude pruning method fails to choose filters with functional redundancy. By further analyzing the functionality transition of remaining filters in the retraining phase, we revealed that the magnitude based pruning actually partially destructs original neural network's functionality composition. The nature of retraining phase is dramatically network reconstruction rather than recover the filter functionality. By contrast, our proposed functionality-oriented method demonstrated consistent filter functionality during retraining phase, indicating less harm to original network functional composition.

References

[1] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. (2017) Pruning Filters for Efficient ConvNets. in Proceedings of *the International Conference on Learning Representations* (ICLR), pp. 1-13.

[2] J.-H. Luo, J. Wu, and W. Lin. (2017) ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. in Proceedings of *the International Conference on Computer Vision* (ICCV), pp. 5058-5066.

[3] Y. He, X. Zhang, and J. Sun. (2017) Channel Pruning for Accelerating Very Deep Neural Networks. in Proceedings of *the International Conference on Computer Vision* (ICCV), pp. 1389-1397.

[4] H. Hu, R. Peng, YW. Tai, and CK. Tang. (2016) Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv:1607.03250*.

[5] S. Han, H. Mao, and W. J. Dally. (2014) Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv:1510.00149*.

[6] M. Jaderberg, A. Vedaldi, and A. Zisserman. (2014) Speeding up convolutional neural networks with low rank expansions. *arXiv:1405.3866*.

[7] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. (2016) Learning structured sparsity in deep neural networks. in Proceedings of *the Advances in Neural Information Processing Systems* (NIPS), pp. 2074-2082.

[8] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. (2015) Understanding neural networks through deep visualization. *arXiv:1506.06579*.