Functionality-Oriented Convolutional Filter Pruning

BMVC 2019 Submission # Anonymous for Blind Review

Abstract

The sophisticated structure of Convolutional Neural Network (CNN) models allows for outstanding performance, but at the cost of intensive computation load. To reduce this cost, many model compression works have been proposed to eliminate insignificant model structures, such as pruning the convolutional filters which have smaller absolute weights. However, most of these works merely depend on quantitative significance ranking without qualitative filter functionality interpretation or thorough model structure analysis, resulting in considerable model retraining cost. Different from previous works, we interpret the functionalities of convolutional filters and identify the model structural redundancy as repetitive filters with similar feature preferences. In this paper, we proposed a functionality-oriented filter pruning method, which can precisely remove the redundant filters without compromising the model functionality integrity and accuracy performance. Experiments with multiple CNN models and databases testified the unreliability of conventional weight-ranking based filter pruning methods, and demonstrate our method's advantages in terms of computation load reduction (at most 68.88% FLOPS), accuracy retaining (<0.34% accuracy drop), and expected retraining independence.

1 Introduction

Nowadays, Convolutional Neural Networks (CNNs) have been widely applied to various cognitive tasks (*esp.* image recognition) [11, 15, 28]. This great success is benefited from CNNs' sophisticated model structures, which utilize multiple inter-connected layers of convolutional filters to hierarchically abstract input data features and assemble accurate prediction results [2, 4]. However, the increasing layer width and depth boost not only accuracy but also computation load. For example, 2.3× parameter size increment from AlexNet to VGG-16 will introduce 21.4× more computation load in terms of FLOPs [11, 19].

Many optimization works have been proposed to relieve the CNN computation cost [3, 8, 12], and the filter pruning is considered as one of the most efficient approaches. Filter pruning methods identify the convolutional filters with the smallest significance based on different metrics. By removing these filters and repetitive retraining the model, the computation load can be significantly reduced while the prediction accuracy is well retained [6, 23, 29]. However, such significance-ranking based filter pruning methods have been increasingly questioned: On the one hand, the correlation between the filter weight and functionality significance (*e.g.*, in [12]) is not theoretically proven. Some works have found that pruning certain small filters may cause severe accuracy drop than pruning large ones [21]. On the other hand, the excessive dependence on retraining process seriously questions the rationality of the conventional filter significance identification. Some works show that the

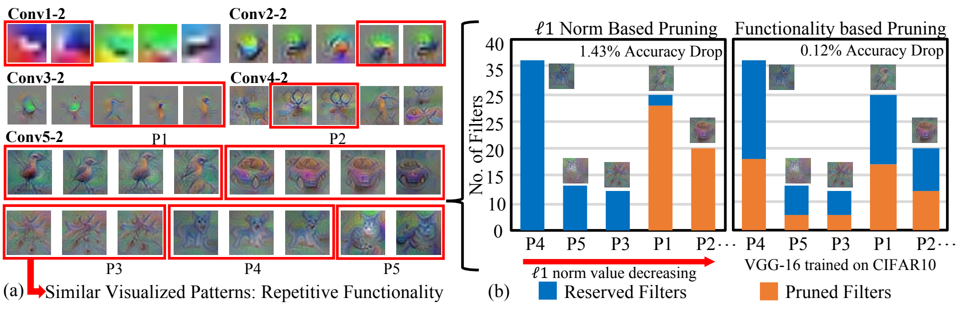


Figure 1: (a) An illustration of the visualized patterns of convolutional filters. (b) ℓ_1 norm based filter pruning and functionality-oriented pruning.

retraining process actually reconstructs the CNN models due to massive filters are inappropriately pruned [13]. Therefore, to re-examine the convolutional filter pruning, rather than merely quantitatively ranking the filter significance, it is urgent to qualitatively interpret filter functionalities and identify actual model redundancies.

In this work, targeting image-cognitive CNNs, we utilized CNN visualization techniques to interpret the convolutional filter functionality [22, 25, 26]. Specifically, we adopted the Activation Maximization method to synthesize a specific input image for each filter [22]. These images can trigger the maximum activation of corresponding filters, thus representing each filter's preferred input feature pattern as its particularly functionality. As shown in Fig. 1 (a), with the layer depth increasing, the visualized filter functionality patterns evolve from fundamental colors and shapes into recognizable objects.

Based on the visualized interpretation, we discover that a certain number of repetitive filters with similar functionality patterns exist in each layer, which can be grouped into multiple functionality clusters (denoted by red boxes). The functionality repetition in each cluster can be considered as a model structural redundancy in the filter level.

Motivated by this discovery, we propose a functionality-oriented filter pruning method. In our methods, regardless of the weight values, we find the best pruning ratio of every cluster in every layer to reduce the model structural redundancy. After the pruning, only a small amount of retraining effort is required to fine tune the accuracy performance. Fig. 1 (b) presents a set of filter pruning examples to demonstrates the difference between our proposed method and the conventional ℓ_1-norm based filter pruning method. With the same amount of filters pruned, we can see that, our method precisely address the redundant filters in every clusters and layers, resulting ignorable accuracy drop.

Experiment results show that, on CIFAR-10, our method reduces more than 43% and 44.1% FLOPs on ResNet-56 and VGG-16 respectively, achieving 0.05% and 0.72% relative accuracy improvement. On CIFAR100, our method reduces more than 37.2% FLOPs on VGG-16 without losing accuracy. On ImageNet, with 50.64% and 23.8% FOLPs reduction, our method can accelerate VGG-16 and ResNet32 without accuracy drop.

We also observe the filter functionality transition in the retraining process. Experiments reveal that, the conventional filter pruning methods may significantly compromise filters' functionality integrity, resulting in considerable retraining cost. While, our method demonstrates expected accuracy retaining capability and retraining independence.

In the following sections, we will proceed into specific design and evaluation details.

2 Related work

Convolutional Filter Pruning Previous filter pruning works can be roughly divided into two categories: (1) *Post-Training Filter Pruning* methods are applied to pre-trained CNN models by identifying and pruning the insignificant filters based on particular weights ranking schemes. For example, [12], [5], and [17] utilized ℓ_1-norm, ℓ_2-norm, and Taylor expansion respectively to rank the filter weights for pruning in each layer. (2) *Training Phase Filter Pruning* methods apply particular regulation constraints to the CNN model training phase, and enforce a certain amount of filters to become sufficiently small to be safely pruned (*e.g.*, structured sparsity learning [20], structured Bayesian pruning [18], and ℓ_0 regularization [14]). Usually, the second category methods can achieve better optimization performance due to their profound effects in the early training stage. However, the first category methods have better practicability with wider application scenarios. In this work, we will focus on renovating the first category methods and compare our methods with corresponding state-of-the-art.

Convolutional Filter Visualization As the convolutional filters are designed to capture certain input features, the semantics of the captured feature can conclusively indicate the functionality of each filter [16, 24, 27]. However, the functionality is hard to be directly interpreted, which significantly hinders qualitative CNN model analysis and optimization development [2]. Recently, many CNN visualization works have been proposed to analyze CNN models in a functionality perspective: [26] established the correlation between each convolutional filter and a specific semantic concept; [22] designed a novel visualization technique – Activation Maximization – to illustrated a filter's maximum activation pattern, which represents the filter's exclusive feature preference as its functionality. In this work, we will utilize the Activation Maximization (AM) as our major visualization tool for filter analysis.

3 Functionality-Oriented Filter Pruning Methods

Proposed Method Overview Based on the convolutional filter functionality and structural redundancy analysis, we proposed a functionality-oriented filter pruning method. The proposed method consists of the following major steps:

(1) *Filter Functionality Interpretation*: Given a pre-trained model, each filter's functionality is firstly interpreted by AM visualization;

(2) *Functionality Redundancy Identification*: Based on proper similarity analysis on the visualized functionality patterns, the filters with repetitive functionalities are clustered together. These repetitive filters can be considered as redundant filters to be pruned;

(3) *Filter Significance Identification*: Inside each cluster, based on the gradients analysis, each filter's relative accuracy contribution is further evaluated. Such a filter significance identification will be applied to determine the pruning priority in cluster level pruning;

(4) *Model-wise Filter Pruning*: Given a global pruning ratio R, we multiply it with the layer-wise coefficients to get each layer's actual pruning ratio r_l. The layer-wise coefficients are determined by each layer's pruning accuracy impact as we will show later. For each layer, this layer-wise pruning ratio r_l will be applied to every filter cluster.

(5) *Model Fine-tuning*: After model pruning, a small number of retraining iterations might be applied to recover potential accuracy drop.

The algorithm details are presented as follows:

Layer	K	Filters	Ratio
Conv1_1	15	62	96.8%
Conv1_2	17	64	100%
Conv2_1	20	128	100%
Conv2_2	31	121	94.5%
Conv3_1	26	251	98.0%
Conv3_2	24	251	98.0%
Conv3_3	14	239	93.4%
Conv4_1	27	447	87.3%
Conv4_2	28	437	85.4%
Conv4_3	38	439	85.7%
Conv5_1	46	500	97.7%
Conv5_2	61	503	98.2%
Conv5_3	40	506	98.8%
Sum	387	3942	93.3%

Figure 2: Filter Cluster Summary

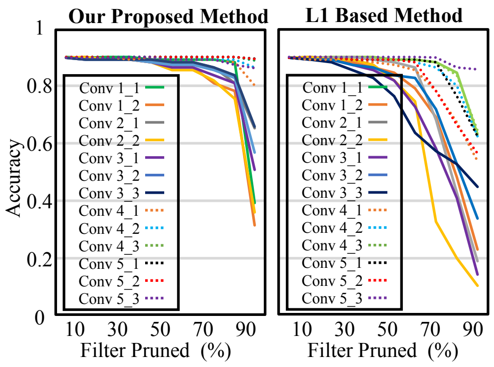


Figure 3: Individual Layer's Accuracy Impact

Filter Functionality Interpretation In AM visualization, each filter's functionality is defined as its feature extraction preference from the CNN inputs. The feature extraction preference of the i_{th} filter $\mathcal{F}_i^l$ in the l_{th} layer is represented by a synthesized input image X that can cause the maximum activation of $\mathcal{F}_i^l$ (*i.e.* the convolutional feature map value). The synthesis process of such an input image can be formulated as:

$$V(\mathcal{F}_i^l) = \underset{X}{\operatorname{argmax}} A_i^l(X), \qquad X \leftarrow X + \eta \cdot \frac{\partial A_i^l(X)}{\partial X}, \tag{1}$$

where $A_i^l(X)$ is the activation of filter $\mathcal{F}_i^l$ 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 is iteratively changed along the $\partial A_i^l(X)/\partial X$ increment direction to achieve the maximum activation. Eventually, X demonstrates a specific visualized pattern $V(\mathcal{F}_i^l)$, which contains the filter's most sensitive input features with certain semantics, and represents the filter's functional preference for feature extraction.

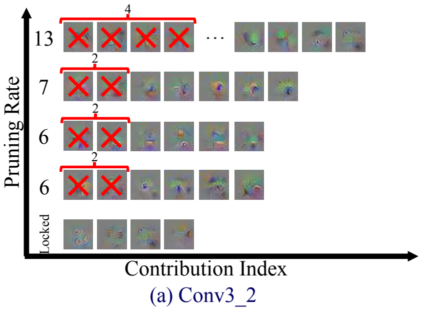
Functionality Redundancy Identification To identify the functionality redundancy, we apply k-means algorithm with pixel-level Euclidean-distance to cluster the filters with similar visualized patterns in each layer. The pixel-level Euclidean-distance between the AM visualized patterns of filter $\mathcal{F}_k^l$ and $\mathcal{F}_i^l$ is formulated as:

$$S_E[V(\mathcal{F}_i^l), V(\mathcal{F}_k^l)] = \|V(\mathcal{F}_i^l) - V(\mathcal{F}_k^l)\|^2. \tag{2}$$

which indicates the functionality similarity of any two convolutional filters.

To determine the proper number of clusters (*i.e.* K), we perform a grid search from one to half of the total filter number (*i.e.* $I_l/2$) in each layer. With a larger cluster number, smaller pattern differences are taken into consideration, causing many clusters may only contain one filter with extremely minimal similarity with others. These filters are considered as non-clustered filters and merged in a locked cluster, which are considered to have unique features and will not be pruned. The maximal K during the grid search is selected as final parameter.

In Fig. 2, we show the filter cluster distribution based on k-means analysis. For a VGG-16 model with 13 convolutional layers, the filters in each layer are grouped into 14 to 61 clusters. With the layer depth increment, the cluster number also becomes larger: in layer Conv5_2, the cluster number is as large as 61. This is because of the feature complexity increment with more divergent visualized graphic patterns. Meanwhile, our proposed method can effectively cluster most filters. The minimum cluster ratio across all convolutional layers remains above



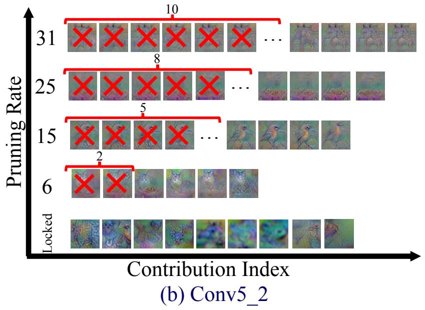


Figure 4: Case study of the filter functionality-oriented filter pruning on the Conv3_2 and Conv5_2 of VGG-16. The convolutional filters are shown by their visualized patterns, and aligned according to the contribution index increment.

85%. In average, about 93% filters are well clustered through the whole model, indicating our method's sufficient filter redundancy analysis capability.

Filter Significance Identification Ideally, the filters with the same functionality can substitute each other. However, they may still have slight contribution difference to the prediction accuracy. To identify each filter's contribution to the output, we use the first-order Taylor expansion to approximate the CNN output variation under the filter's impact:

$$Z(A_i^l + \Delta) = Z(A_i^l) + \frac{\partial Z(A_i^l)}{\partial A_i^l} \cdot \Delta, \ where \ \Delta = A_i^l \to 0, \tag{3}$$

where $Z(A_i^l + \Delta)$ is the CNN output loss and the A_i^l is i_{th} filter's output feature map in layer l. When filter $\mathcal{F}_i^{l-1}$ in the $l-1$ layer is pruned, the filter's output feature map is corresponding set to zero, i.e. changing the i_{th} dimension of A^l to zero. Therefore, the influence on Z can be qualitatively evaluated by its coefficient $\frac{\partial Z(A_i^l)}{\partial A_i^l}$.

Before pruning, each filter $\mathcal{F}_i^{(c,l)}$ in the cluster C_l^k of layer l is firstly ranked by the contribution index, which is calculated by examining the average gradients:

$$I(\mathcal{F}_i^{(c,l)}) = \frac{1}{N} \sum_{n=1}^{N} \left\| \frac{\partial Z(F, A^l)}{\partial A_i^l(x_n)} \right\|, \tag{4}$$

where $Z(F, A^l)$ is the CNN output loss of a test image x_n, and $A_i^l(x_n)$ is the feature map of filter $\mathcal{F}_i^{(c,l)}$ for each test image x_n.

Filter Pruning and Fine-Tuning Based on the functionality redundancy and filter significance identification, we can proceed to the stage of model-wise convolutional filter pruning. Given a global model pruning ratio R, the layer-wise pruning ratio of each layer r_l is determined by multiplying R and the layer-wise coefficients, which are determined according to each layer's pruning accuracy impact. For example, Fig.3 shows the layer-wise network accuracy impact according to our method and ℓ_1-norm based method. Clearly, the shallow layers demonstrate higher accuracy impact, while the deeper layers have lower impact. Therefore, we prune more gently for shallower layers but more aggressive for deeper layers. The layer-wise coefficients of the pruning ratios r_l from the shallow layers to deep layers (conv1_x to conv5_x) is thus set to 0.25:0.125:0.125:0.375:0.375. For other models, dedicated pruning ratios are discussed in later experiments.

Meanwhile, we can see that our proposed pruning has a slower accuracy degradation rate compared with the ℓ_1-norm based pruning method in majority of layers. This means the redundant filters can be more accurately identified and pruned through the proposed interpretive functionality-oriented filter pruning approach.

Fig. 4 shows our functionality-oriented filter pruning examples with the convolutional layers of Conv3_2 (a) and Conv5_2 (b) with $r_l = 30\%$. The convolutional filters are shown by their visualized functionality patterns, and the filters with similar patterns are grouped into multiple clusters. When the layer-wise pruning ratio is assigned, the filters with small contribution index values will be firstly pruned. According to our proposed method, larger clusters are pruned with more filters. This make senses since such balanced functionality reduction will lead to non-biased functionality composition as the original model's. Therefore, we can see different amounts of filters are pruned between different clusters but all the pruning ratios are $\sim$30% as shown in the Fig. 4.

Leveraging the convolutional filters' qualitative functionality analysis, our proposed method is expected to leverage the neural network interpretability for more accurate redundant filter allocation, faster pruning speed, and optimal computation efficiency.

After the model-wise pruning, a small amount of model fine-tuning will be applied to recover the potential accuracy drop, as we will show in later experiments.

4 Experiments

We evaluate our proposed method with both single-branch CNN models (*i.e.*, ConvNet, VGG [19]) and a multiple-branch CNN model (*i.e.*, ResNet [4]) on CIFAR10/100 [10] and a subset of ImageNet [1]. A data argumentation procedure is applied to CIFAR10/100 dataset through the horizontal flip and random crop, generating a 4-pixel padded training dataset. The visualization analysis and filter pruning are implemented in Caffe environment [9]. The contribution index is calculated during the training phase with the whole training dataset for more accurate CNN model analysis. The retraining process for our method is executed only by 40 epochs (*i.e.*, 1/4 of the original training epochs) with a constant learning rate of 0.001. While the retraining process for the compared state-of-the-art will be continuously applied until model convergence. Specifically, the compared works include ℓ_1-norm Pruning [12], Taylor Pruning [17], Geometric Median Pruning (GM) [6], and Channel Pruning [7].

4.1 CIFAR Experiment

In this section, we first evaluate our method on two CIFAR image datasets, namely CIFAR-10 and CIFAR-100. The experimental results are shown in Table 1.

ConvNet on CIFAR-10 Our ConvNet is designed based on the AlexNet model with 5 convolutional layers, which contain 64-128-256-512-512 filters respectively. All the convolutional filters are constructed with 3x3 matirces. The baseline test accuracy of our ConvNet is 90.05%. As shown in the Table 1, ours method pruned 40% convolutional filters without accuracy drop and achieved 37.4% computation load reduction We also implemented the ℓ_1-norm and Taylor pruning for comparison. These methods pruned the same amount of filters as our method to achieve the same amount of FLOPs reduction. As shown in the Table 1, our method achieves better performance in terms of both pruned accuracy and retrained accuracy.

VGG-16 on CIFAR-10 We implemented a VGG-16 model following the same pre-processing and hyper-parameters configurations as [12], which consists of 13 convolutional layers and 2 fully-connected layers. Each convolutional layer is followed by a batch normalization

Table 1: CIFAR Pruning Comparison

CNN Models	Pruning Method	Baseline acc. (%)	FLOPs (x10^8)	FLOPs $\downarrow$ (%)	Prune acc. (%)	Retrain acc. (%)
ConvNet (CIFAR10)	ℓ_1-norm*	90.05	5.34	37.4	83.29	88.53
	Taylor*	90.05	5.34	37.4	85.29	89.42
	Ours (40%)	90.05	5.34	37.4	**87.88**	**90.04**
VGG-16 (CIFAR10)	ℓ_1-norm [12]	93.25	2.06	34.2	-	93.40
	Taylor*	93.25	1.85	44.10	73.24	92.31
	GM [6]	93.58	5.34	35.9	80.38	**94.00**
	Ours (45%)	93.25	1.85	**44.10**	**91.13**	93.30
ResNet-56 (CIFAR10)	ℓ_1-norm [12]	93.04	0.91	27.60	-	93.06
	Taylor*	92.85	0.71	43.00	76.32	92.01
	Channel [7]	92.80	0.56	**50.00**	-	93.23
	Ours (40%)	92.85	0.71	43.00	81.13	**93.30**
VGG-16 (CIFAR100)	ℓ_1-norm*	73.14	1.96	37.32	63.21	72.31
	Taylor*	73.14	1.96	37.32	65.19	72.52
	Ours (45%)	73.14	1.96	37.32	**68.21**	**73.21**

"*" indicates our implementation. Ours(40%) means 40% filters are pruned by our method.

layer. As shown in Table 1, our method has better performance than the ℓ_1-norm and Taylor pruning methods. Under a pruning ratio of 45%, our proposed method can achieve 44.1% FLOPs reduction with the accuracy well-retained. Although the GM method achieves an even improved accuracy performance after the retraining process, it only reduced 35.9% computation load, about 18.6% less than our proposed method.

ResNet-56 on CIFAR-10 ResNet-56 is a multiple-branch CNN model, which contains three convolutional layers connected by projection mapping channels, one global average pooling layer, and one fully-connected layer. After trained on CIFAR-10 from scratch using the same training parameters as [4], the model can achieve a baseline accuracy of 92.85%. To avoid changing the input and output feature maps of each residual block, same as ℓ_1, we only prune filters from the first layers of each block. Considering ResNet56 has many layers, we set the pruning ratio of all convolutional layers to the same value in this experiment. Our method demonstrate better accuracy retaining capability compared to the ℓ_1-norm and Taylor pruning. Comparing to the Channel Pruning, although our computation load reduction is slightly smaller, our implementation is much straightforward, considering the Channel Pruning requires additional multiple-branch enhancement to generalize itself to ResNet.

VGG-16 on CIFAR-100 We further evaluated our pruning method on CIFAR100 dataset. Using the same VGG-16 model, our method introduces less accuracy drop compared to state-of-the-art with the same computation load reduction. Meanwhile, the accuracy drop caused by our pruning method can be fully recovered with 0.07% accuracy improvement.

4.2 ImageNet Experiment

In evaluations on ImageNet, 10 and 100 classes images out of 1000 classes, namely ImageNet-10 and ImageNet-100 are utilized in this work. Each class contains 1300 training images and 50 validation images. We evaluate our proposed method for the VGG-16, and ResNet32 on the ImageNet-10 and ImageNet-100 respectively.

The VGG-16 is implemented as the same architecture as the VGG-16 on the CIFAR. The proportion of the pruning ratios from shallow layers to deep layers (conv1_x to conv5_x) is

Table 2: ImageNet Pruning Comparison

Network	Method	Baseline acc. (%)	FLOPs (x10^{10})	FLOPs $\downarrow$ (%)	Prune acc. (%)	Retrain acc. (%)
VGG-16	ℓ_1-norm*	93.76	0.57	62.99	21.22	92.26
(ImageNet-10)	Taylor*	93.76	0.57	62.99	24.37	93.12
	Ours(40%)	93.76	0.57	62.99	**25.13**	**94.34**
ResNet-32	ℓ_1-norm*	88.31	1.55	32.90	80.37	84.75
(ImageNet-10)	Taylor*	88.31	1.55	32.90	81.03	85.12
	Ours(30%)	88.31	1.55	32.90	**81.25**	**88.25**
VGG-16	ℓ_1-norm*	78.51	0.76	50.64	36.35	76.56
(ImageNet-100)	Taylor*	78.51	0.76	50.64	40.28	77.32
	Ours(30%)	78.51	0.76	50.64	**43.25**	**79.51**
ResNet-32	ℓ_1-norm*	75.34	1.76	23.81	63.25	72.52
(ImageNet-100)	Taylor*	75.34	1.76	23.81	68.14	73.43
	Ours(20%)	75.34	1.76	23.81	**70.21**	**75.40**

"*" indicates our implementation. Ours(40%) means 40% filters are pruned by our method.

set to 2:1:1:3:3. The ResNets-32 for ImageNet have three stages of residual blocks, which contain 3, 4, and 2 residual blocks respectively, and each residual block has three convolutional layers. Same as ResNet-56, in each residual block, only the first two convolutional layers are pruned to keep the input and output feature maps to be identical. The pruning ratio of all convolutional layers is set to the same value for simplicity. The result of ImageNet pruning comparison is shown in the Table 2.

ImageNet-10 Table 2 shows that our method outperforms previous methods on ImageNet10 dataset. For the VGG-16, large accuracy loss occurs in the three pruning method. However, our proposed method induces less accuracy degradation compared with the ℓ_1 and Taylor. With re-training, our method achieves higher accuracy than the baseline accuracy. For the ResNet-32, it is hard to recover the accuracy drop through retraining on the ImageNet-10 dataset, and hence acceptable pruning rate in this scenario is relatively small. As shown in the Table 2, we can achieve 32.90% FLOPs reduction on ResNet-32 with only 0.06% accuracy drop.

ImageNet-100 With larger image data complexity and more class composition complexity, the feature extraction becomes more complex. It's become more challenge for the filter functionality identification and clustering. Our method can still outperform previous methods on ImageNet100 dataset. As shown in the Table 2, we can achieve 50.64% FLOPs reduction on VGG-16 with 1% accuracy improvement. For the ResNet-32, our method can still achieve less accuracy and better retraining accuracy improvement than the previous method.

4.3 CNN Model Retraining Analysis

In most filter pruning works, the retraining process is essential to compensate the accuracy drop. However, as aforementioned, its role still lacks certain research. In this work, we also analysis the retraining process quantitatively and qualitatively in filter pruning.

Fig. 5 compares the retraining processes of our method and the ℓ_1-norm based pruning method with ConvNet and VGG-16 on CIFAR-10. Comparing our method (solid line) with the ℓ_1-norm based method (dashed line), we can observe that: The models pruned by our method always demonstrate quicker accuracy recovery. Take VGG-16 as the example, our method is close to the convergence accuracy after only after 100 iterations with less retraining

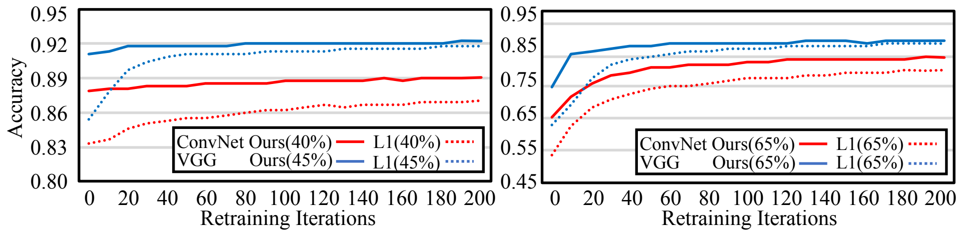


Figure 5: Pruned model accuracy recovery by retraining.

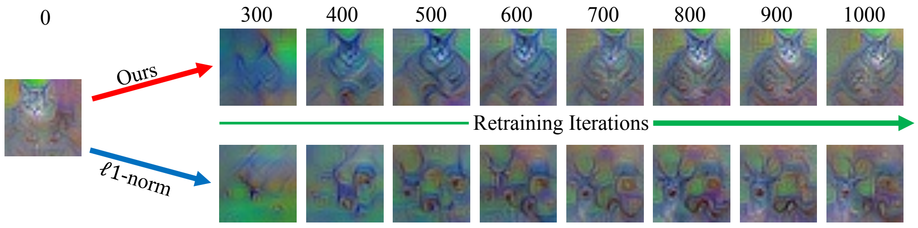


Figure 6: Filter functionality transformation during retraining.

dependency. While the accuracy of the ℓ_1-norm based method is still $\sim$3% lower than ours, resulting in significantly more retraining effort.

We also use AM visualization to analysis the filter functionality transition during the retraining process as shown in Fig. 6. We randomly choose and visualize one preserved filter after pruning with our method and the ℓ_1-norm based method. During the retraining process, we visualize its functionality pattern every 100 iterations. From Fig. 6, we can see that, the visualized pattern after the ℓ_1-norm pruning demonstrates dramatic change. This indicates that the ℓ_1-norm based method significantly defects the original CNN model's functionality integrity, and the retraining process has to reconfigure the filter's functionality for compensation. Meanwhile, the filter functionality pattern remains unchanged in our method, which indicates our method's precise redundant filter identification.

5 Conclusion

In this work, through Activation Maximization based visualization and functionality analysis, we firstly demonstrate that convolutional neural network filter redundancy exists in the form of functionality repetition. Therefore, we propose that such functional repetitive filters could be effectively pruned from neural network to provide computation redundancy reduction. Based on such motivation, we propose an interpretable functionality-oriented filter pruning method: By first interpreting and clustering filters with same functions together, we remove the repetitive filters with smallest significance and contribution in a balanced manner inside each cluster. Therefore, this implicitly helps maintain the similar functionality composition as the original model, and thus brings less damage to the model accuracy. Extensive experiments on CIFAR and ImageNet demonstrate the superior performance of our pruning method over state-of-the-art methods. By analyzing the functionality changing of remaining filters in the retraining process, we further proves our assumption that ℓ_1-norm based pruning partially destructs original neural network's functionality composition. By contrast, our method shows consistent neuron functionality during retraining process, demonstrating less harm to original model functionality.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.

[2] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, 126(5): 476–494, 2018.

[3] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 770–778, 2015.

[5] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[6] Yang He, Ping Liu, Ziwei Wang, and Yi Yang. Pruning filter via geometric median for deep convolutional neural networks acceleration. *arXiv preprint arXiv:1811.00250*, 2018.

[7] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the International Conference on Computer Vision*, 2017.

[8] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[10] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1098–1105, 2012.

[12] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[13] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[14] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

[15] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. *arXiv preprint arXiv:1809.09478*, 2018.

[16] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[17] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *arXiv preprint arXiv:1611.06440*, 3, 2016.

[18] Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems*, pages 6775–6784, 2017.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[20] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016.

[21] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018.

[22] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[23] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[24] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[25] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[26] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

[27] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the importance of individual units in cnns via ablation. *arXiv preprint arXiv:1806.02891*, 2018.

[28] Fengda Zhu, Linchao Zhu, and Yi Yang. Sim-real joint reinforcement transfer for 3d indoor navigation. *arXiv preprint arXiv:1904.03895*, 2019.

[29] Huiyuan Zhuo, Xuelin Qian, Yanwei Fu, Heng Yang, and Xiangyang Xue. Scsp: Spectral clustering filter pruning with soft self-adaption manners. *arXiv preprint arXiv:1806.05320*, 2018.