Filter Contribution Recycle: Boosting Model Pruning with Small Norm Filters

Zehong Chen[1,2], Zhonghua Xie[1], Zhen Wang[1], Tao Xu[1] and Zhengrui Zhang[1*]
[1] School of Computer Science and Engineering, Huizhou University, Huizhou 516007, China
[e-mail: chenzh@hzu.edu.cn, eezhxie@gmail.com, wangzhen@hzu.edu.cn, xutao911@163.com, 437028792@qq.com]
[2] Guangdong Key Laboratory of Intelligent Information Processing and Shenzhen Key Laboratory of Media Security, Shenzhen 518060, China
*Corresponding author: Zhengrui Zhang

Received June 13, 2022; revised October 7, 2022; accepted November 6, 2022; published November 30, 2022

Abstract

Model pruning methods have attracted huge attention owing to the increasing demand of deploying models on low-resource devices recently. Most existing methods use the weight norm of filters to represent their importance, and discard the ones with small value directly to achieve the pruning target, which ignores the contribution of the small norm filters. This is not only results in filter contribution waste, but also gives comparable performance to training with the random initialized weights [1]. In this paper, we point out that the small norm filters can harm the performance of the pruned model greatly, if they are discarded directly. Therefore, we propose a novel filter contribution recycle (FCR) method for structured model pruning to resolve the fore-mentioned problem. FCR collects and reassembles contribution from the small norm filters to obtain a mixed contribution collector, and then assigns the reassembled contribution to other filters with higher probability to be preserved. To achieve the target FLOPs, FCR also adopts a weight decay strategy for the small norm filters. To explore the effectiveness of our approach, extensive experiments are conducted on ImageNet2012 and CIFAR-10 datasets, and superior results are reported when comparing with other methods under the same or even more FLOPs reduction. In addition, our method is flexible to be combined with other different pruning criterions.

Keywords: Model pruning, Structured pruning, Filter contribution recycle, Filter weight reutilization

The work of this paper is supported by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515111087, and the Guangdong Province ordinary universities characteristic innovation project under Grant 2021KTSCX127, and Guangdong Overseas Famous Teacher Project: Research on artificial intelligence and new generation intelligent manufacturing technology, and 2021 practice teaching base project of integration of science, industry and education in Guangdong Undergraduate Universities, and the 2020 School-level characteristic specialty project of Huizhou University under Grant TSZY2020001, and the Professorial and Doctoral Scientific Research Foundation of Huizhou University under Grant 2020JB035.

1. Introduction

During the past years, convolutional neural networks (CNNs) have achieved state-of-the-art performance in computer vision tasks such as image classification [2,3], object detection [4,5] and so on. To achieve excellent performance, the architecture of CNNs has become more and more complicated, and the number of parameters increases quickly. Deploying these huge models require expensive computation and memory costs, which prevent their usage on resource-limited devices, such as mobile phones [6,7]. Especially, a large number of sensors in the field of Industrial Internet of Things (IIoT) [8], produces gigantic massive data for facilitating a wide range application of IIOT. Due to the limitation of network transformation [9] and the requirement of real-time feedback, those edge devices can only be performed lightweight CNN models. Thus it is very necessary to obtain a lightweight CNN model, which not only has the advantage of low computation and memory costs, but also has the competitive performance comparing to the complicated one.

To address this issue, many model compression methods have been proposed, such as model pruning [10,11], knowledge distillation [12,13] and parameter quantization [14,15]. Among them, model pruning has the longest history, which can be dated back to 1980s [16]. Based on the intuition that a filter with unimportant knowledge [12] has few contribution to the model performance, model pruning methods directly discard unimportant filters from the original over-parameterized model, meanwhile, they maintain the performance almost unchanged or slightly decreased. The knowledge or contribution of a filter is usually measured by its norm, *e.g*. LASSO [17]. In short, the purpose of model pruning is to find and discard filters with relatively small contribution.

Generally, model pruning methods can be categorized into two different classes, non-structured pruning and structured pruning. Non-structured pruning [10,18] can achieve model sparsity, but it may be less efficient for saving the computation and memory costs. On the contrary, structured pruning [11,19], including filter, channel and block pruning, is more friendly to resource-limited devices. Therefore, structured pruning is more suitable to obtain the lightweight model and to accelerate model inference time in a variety of vision tasks.

However, most existing structured pruning methods suffer from one problem: ***The contribution waste of discarded filters***. In order to reach the target FLOPs, filters with small contribution are directly discarded in these methods [19,20]. We argued that these filters carry necessary information of the learned model. As can be seen in **Fig. 1**, when discarding those small filters with different pruning ratios, there is a distinct accuracy drop between those pruned models and the original one. The experiment intuitively indicates that a filter, even with small contribution, still has a non-negligible effect on the model accuracy. Therefore, discarding the filters with small contribution directly may not be the optimal way, it leads to a lot of information waste.

In addition, for structured pruning, a pruned model was fine-tuned by Li *et al.* [1] only obtained comparable or worse performance comparing with training the model with random initialized weights. The parameters of the pruned filter have few contribution for the final performance comparing to the random initialized ones. In the procedure of Li *et al*. [1], instead of exploring which contribution of small filters has, these filters are discarded directly. This is a general way in model pruning.

Is there an efficient way to maximally utilize those small norm filters in the model pruning algorithm? The answer is yes. In this paper, we propose Filter Contribution Recycle

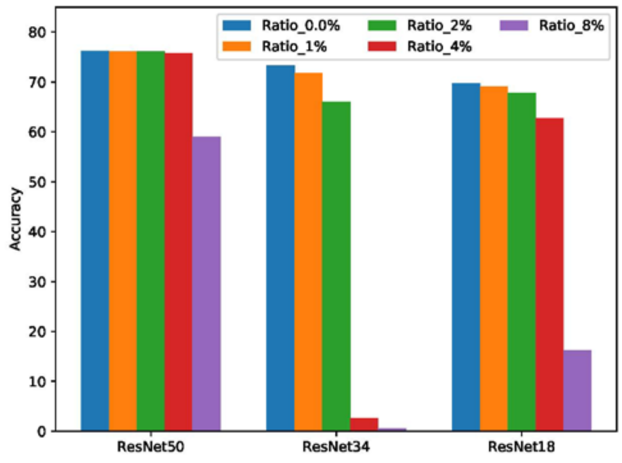

Fig. 1. Accuracies on ImageNet2012 dataset of ResNet models with three kinds of depth under different filter discarding ratios. All the filters are ranked in descending order according to their contribution, which is calculated with LASSO. We then directly discard those filters which ranked last with different ratios, for example: 0.0%/ 1%/ 2%/ 4%/ 8%, and evaluate the model without fine-tuning. It can be observed that the performance drops obviously for all the models. Especially the accuracy of ResNet34 decreases drastically to almost zero with ratio larger than 4%. The reason is that all the filters in the 10th convolutional layer[1] of the ResNet34 pretrained model are smaller than other convolutional layers. Those small filters have the highest priority to be discarded. When setting a high discarding ratio, the 10th convolutional layer could not maintain enough capacity to transfer information. All the pretrained models are downloaded from Pytorch website[2].

(FCR), a novel effective structured pruning approach, to address the fore-mentioned problem. FCR first collects contribution from all the droppable filters, and then reassembles those contribution to obtain a mixed contribution collector. Contribution contained by the collector is then assigned to part of the preserved filters. Through collecting and assigning, FCR establishes a contribution recycle from the droppable filters to the preserved ones. To achieve the target FLOPs, FCR also adopts a weight decay strategy for all the droppable filters. When the contribution of a droppable filter is less than a specified threshold, it will be discarded.

Our main contributions are summarized as follows:

1) We propose Filter Contribution Recycle (FCR), a novel structured model pruning method, with the motivation of reusing small norm filters to avoid filter contribution waste. FCR achieves a high-efficiency utilization of the droppable filters, which will not be discarded until not valuable at all. FCR is also flexible to be combined with other different pruning criterions.

2) FCR establishes an implicit connection between all the droppable filters and each top-ranking preserved filter. The connection makes the top-ranking preserved filters to inherit more contribution to become richer than before. Meanwhile, we validate that the parameters of the discarded filters are important for structured pruning. To get a better effectiveness, a pruning method could adopt an extra method like FCR to maximally utilize parameters against random initialized parameters for the pruned model.

3) Extensive experiments show that our approach achieves a superior performance comparing with other methods under the same or even more FLOPs reduction.

[1] The layer name layer2.0.downsample.0.weight in ResNet34
[2] https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py

2. Related Works

In this section, we discuss the related model pruning algorithms. According to different pruning strategies, model pruning approaches could be split into two categories: unstructured pruning and structured pruning.

Unstructured Pruning. The unstructured pruning algorithm pushes weights in the feature map to zero. The architecture of the sparse model actually is not modified. Those zeroed weights are dispersed in different feature maps, and the number of the zeroed weights in each feature map are generally not the same. [10] specified a pruning threshold to filter the small weight which is less than the threshold. [18] proposed a sparse momentum algorithm to train the CNN model efficiently. The algorithm relies on identifying weights in a layer, and redistributing and growing them across layers. Based on connection sensitivity, [21] introduced a saliency criterion to identify important connections in the network. [22] presented a systematic weight pruning approach which formulates the weight pruning problem as a non-convex optimization problem. In [22], more weights are set to zero to obtain a high sparsity ratio, but the inference time of the pruned model is generally approximate to that of the original one. The proposed approach in [22] cannot reach the theoretical inference time. Therefore, the memory cost and the inference time are not effective decrease.

Structured Pruning. To effectively reduce the inference time and the memory cost, the structured pruning algorithm zeros out part of filters in a convolutional layer. Those zeroed out filters generally has few contribution for the original model. To distinguish less contribution filters, different pruning criterions are used to estimate the importance of the filters. [11] dynamically pruned the filters with small L2-norm, and enabled the pruned filters to be updated when training the model after pruning. [19] proposed an iterative two-step algorithm to prune efficiently in each layer by a LASSO regression based on channel selection and least square reconstruction. [23] used a meta learning approach to prune channels automatically for very deep neural networks. [24] combined the filter learning with filter selection for model compression. [25] introduced a greedy algorithm to conduct channel selection and parameter optimization in an iterative way. [26] pruned unimportant filters to simultaneously accelerate and compress CNN models, in which, whether a filter needs to prune depends on the outputs of its next layer not its own layer. Li *et al.* [1] observed that the parameters of the pruned filter have few contribution for the final performance, and that a pruned model was fine-tuned only obtained comparable or worse performance comparing with training the model with random initialized weights.

3. Methodology

3.1 Preliminaries

For simplicity, we assume that a neural network has L layers, and the parameter of the lth convolutional layer C_l can be represented as $W^l \in \mathbb{R}^{n_l \times n_{l-1} \times k_l \times k_l}$, where k_l is the kernel size, and n_l is the number of filters in C_l. In this way, we can use $W_i^l \in \mathbb{R}^{n_{l-1} \times k_l \times k_l}$ as the parameter of the i th filter in C_l. In the filter pruning stage, we can divide all the convolutional filters W in the network into two subsets, *i.e.*, the discarded filter subset and the preserved filter subset, denoted as $\mathcal{D}$ and $\mathcal{P}$, respectively.

Given a set of training examples $\mathcal{X} = \{(x_m, y_m)\}_{m=1}^{M}$, where x_m is the m th input and y_m is the corresponding label, the objective of pruning is to maximally inherit the performance (*e.g.* accuracy) of the original neural network with minimum filters P, which can be formulated as:

$$\min_{\mathsf{P}} \left| \mathsf{L}(W;\mathsf{X}) - \mathsf{L}(\mathsf{P};\mathsf{X}) \right|, \mathsf{P} << W \tag{1}$$

where L is the loss function (*e.g.* cross-entropy loss). P starts with the whole set of parameters of W, and we iteratively identify and remove the small contribution filters until reaching the best trade-off between computation and accuracy.

3.2 Filter Divide Criterion

To achieve the target FLOPs with structured pruning, we should discard part of filters (convolutional kernels) in each convolutional layer. In our experiments, a filter could be discarded without waste its contribution if its LASSO value is less than a specific threshold T (*e.g.* $1 \times e^{-4}$ or lower), and the accuracy only slightly decrease. The filter discard criterion can be formulated as:

$$W = \begin{cases} \left\{ \left| w_j^l \right|_{Lasso} < \mathsf{T} \right\} \in \mathsf{D} \\ \left\{ \left| w_j^l \right|_{Lasso} \geq \mathsf{T} \right\} \in \mathsf{P} \end{cases} \tag{2}$$

where w_j^l denotes the j th filter in the l th convolutional layer in a neural network. The operation $\left| w_j^l \right|_{Lasso}$ is used to calculate the contribution of the filter w_j^l.

Generally, there are some innately dead filters in a well trained neural network. Even if discarding all the dead filters, there is not enough to reach the target FLOPs. We should discard more filters in the preserved filter subset. The LASSO values of all the filters in the preserved subset are greater than the threshold T, in which, there are a part of LASSO values are close to T, and the corresponding filters are more likely to be discarded. These discarded filters can be called droppable filters. In our method, we define the last 10% filters in preserved subset belong to the droppable filter subset U by the contribution rank. Then, a new subset named droppable filter set U is split from P. Thus, W could be denoted as: { D, U, P }.

3.3 Motivation

According to the filter divide criterion, the filters in a neural network can be divided into three groups. Though the filters in the droppable subset have few contribution, their effect cannot be ignored. The reason is analyzed in **Fig. 1**. Discarding the small contribution filter would degrade the effectiveness of the neural network quickly. We want to find a way to recycle the contribution of the droppable filter, and to enhance the performance of the pruned model or inherit more information from the original model. This is the motivation of this paper (**Fig. 2**), and the overall architecture of FCR method is described in **Algorithm 1**.

3.4 Filter Contribution Recycle

For simplicity, we take the l th convolutional layer without bias as our example layer, which has C_n^l filters. The parameter of this layer can be denoted as: $W^l = \{w_k, k \in [1, C_n^l]\}$, where

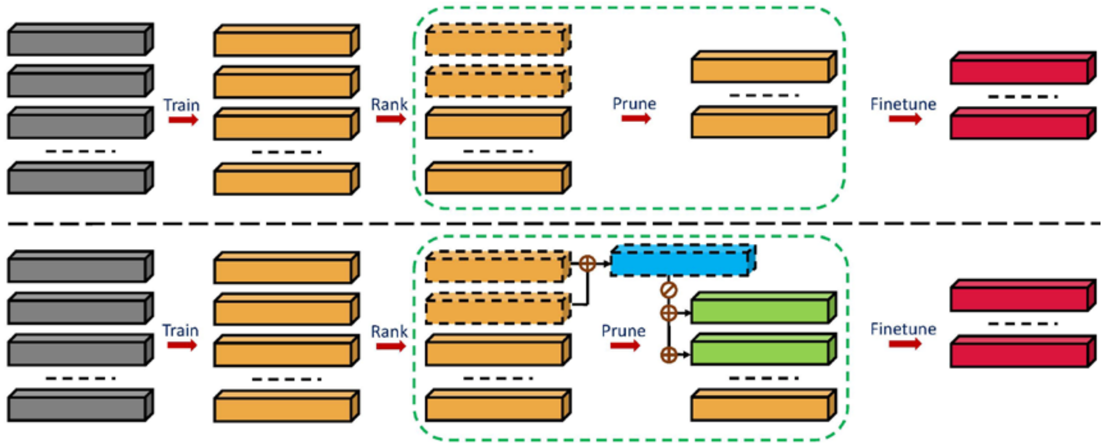

Fig. 2. Comparison of different pruning algorithms. (**Top**) The traditional model pruning. (**Bottom**) Model pruning with filter contribution recycle. The gray, yellow and red box donates the initial, optimized and fine-tuned convolutional filters, respectively. The blue box represents contribution collector. The green box represents optimized filter which has received contribution from the contribution collector (This figure is best viewed in color).

Algorithm 1: Algorithm Description of FCR

```
Input: Train data $X$, pretrained parameter: $W = \{w_i, 0 \le i \le n\}$, target FLOPs: $T_{flops}$,
       discard threshold: $\mathrm{T}$, calculate FLOPS function: $F_{flops}$
Output: The best pruned model and its parameters $W^*_{best}$
1  for epoch=1; epoch<epoch_max; epoch=epoch+1 do
2     Update the parameter $W$ based on $X$;
3     Calculate the LASSO value for each filter ($|w_i|_{Lasso}, 0 \le i \le n$) and rank total filters;
4     Divide out droppable filters and collect contribution collector $\mathrm{C}$ with (4);
5     Assign the contribution collector $\mathrm{C}$ to each top preserved filter by contribution rank with (5);
6     Decay total droppable filters with (6);
7     for i=1; i<n; i++ do
8        if $|w_i|_{Lasso} < \mathrm{T}$ then
9           Zero the filter $w_i$;
10       end
11    end
12    if $F_{flops}(W) < T_{flops}$ then
13       Break;
14    end
15 end
16 Obtain the pruned model with parameters $W^*$ from $W$;
17 Fine-tuning pruned model with $X$ to get the best pruned model $W^*_{best}$.
```

w_k represents the k th filter in W^l. Because the discarded filter subset D has no contribution to the neural network, we focus on other two subsets. The parameter W^l can be simplified to the formulation:

$$W^l = \left\{ \mathsf{U}\{w_{k_d}^u\},\ \mathsf{P}\{w_{k_p}^p\},\ k_u + k_p \le C_n^l \right\} \tag{3}$$

where k_u and k_p are the corresponding number of filters in the droppable filter subset U and the preserved filter subset P , respectively.

Filter contribution collector. The general pruning method discards the filters with small contribution directly. This way wastes the information of the droppable filters. In order to avoid this, we define a filter contribution collector C to collect the filter contribution for the droppable subset in each convolutional layer. Then C can be formulated as:

$$\mathsf{C} = \sum_{i=1}^{i \le k_u} \alpha\ w_i^u, \tag{4}$$
$$s.t.\ \ \alpha \in [0,1),\ w_k^u \in \mathsf{U}$$

where α is the filter contribution collecting rate, which controls the contribution shared volume for each droppable filter. We will discuss the impact of α to the model pruning in Subsection 4.4. The collector is a contribution inheritor, and it also is a contribution carrier of the droppable filters.

Filter contribution assignment. Training a neural network is a dynamic procedure, and the filter parameters are updated by the optimizer. According to the filter divide criterion, the status of a filter is not constant, and it has a probability to switch its group between the droppable subset and the preserved one. The droppable filters are probably updated near the boundary of the group divide, they are almost impossible to reach the top of the filter contribution rank. The top-ranking preserved filters have certain magnitude advantages against the droppable ones. Therefore, the top-ranking preserved filters are the best receivers to inherit the contribution from the collector. The procedure of the filter contribution assignment can be formulated as:

$$\begin{aligned} w_{iter+1}^p &= w_{iter}^p + \frac{1}{top_n}\mathsf{C} \\ &= w_{iter}^p + \frac{1}{top_n}\sum_{i=1}^{i \le k_u} \alpha\ w_i^u, \end{aligned} \tag{5}$$
$$s.t.\ \ w_{iter}^p \in \mathsf{P},\ w_{iter}^u \in \mathsf{U},\ top_n \in [1, k_p]$$

where w_{iter}^p represents a top-ranking preserved filter, *iter* denotes the training iteration, top_n is the number of the top-ranking preserved filters. To keep the model training stable, we divide C into top_n parts averagely, and then assign them to each top-ranking preserved filter.

Droppable filter decay strategy. To achieve the target of FLOPs reduction quickly, we define a filter parameter decay strategy for the droppable filters. The proposed strategy can be formulated as:

$$w_{iter+1}^u = (1-\alpha)\ w_{iter}^u,\ \ s.t.\ w_{iter}^u \in \mathsf{U} \tag{6}$$

where w_{iter}^u represents a droppable filter. While the droppable filters share α percent contribution, it decays the corresponding contribution to accelerate the speed of model pruning. The proposed strategy decays the magnitudes of the droppable filters iteratively to reach the target FLOPs reduction.

3.5 Discussion

The filter contribution recycle (FCR) not only achieves the contribution reutilization for the droppable filters, but also builds an implicit connection between all the droppable filters and each top-ranking preserved filter. When the model FLOPs is decreasing, the top-ranking preserved filters inherit more contribution to become richer than before, and the droppable filters continuously sharing contribution until they are useless. It is a mutually beneficial approach for all filters.

4. Experiments

To explore the performance of the proposed FCR, we evaluate it on two datasets including CIFAR-10 and ImageNet2012 with a variety of network architectures. Both the two datasets are widely used for classification tasks. FCR is an independent module, which could be combined with different filter contribution criterions (*e.g.* LASSO, L2-norm and Taylor-expansion). So we validate the effectiveness of FCR module combined with different criterions. In addition, we analyze the effectiveness of the filter contribution collecting rate α .

4.1 Implementation details

Pruning and Fine-tuning Stage. Our method starts with a high computation cost and well accuracy pretrained model. The overall training process is divided into two stages: the pruning stage and the fine-tuning one. In the pruning stage, according to the filter contribution criterion, the pretrained model discards the small norm filters iteratively in each convolutional layer. When the model has discarded enough filters to achieve the target FLOPs, we fine-tune the pruned model until the best accuracy is obtained.

Minimum Preserved Filter Rate Strategy. In general, there are different distributions of values among convolutional layers. For example, in the official pretrained model of ResNet34, the values of the 10th convolutional layer are obviously less than other layers. When we specify a large pruning rate, the 10th convolutional layer has the highest priority to be discarded. If the filters in the layer are discarded excessively, the remaining ones may not have enough capacity to transfer the data information to result in a poor effectiveness. Therefore, we set a minimum preserved filter rate for each layer to avoid excessive pruning. The preserved filter rate is an empirical value. In our experiments, if the preserved filter rate is set too small (less than 10%), the number of the preserved filters could not maintain the accuracy of the network in the pruning phase, which damaged the expression ability of the network. On the other hand, if the preserved filter rate with a larger value (more than 25%), there are some redundant filters to be preserved. To find the minimum number of filters is the purpose of model pruning. Therefore, the preserved filter rate is set to 15% in our approach.

FLOPs Computation for Shortcut. In the ResNet architecture, there are lots of shortcut connections. We follow the equation (8) in [11] to confirm the output shape to calculate FLOPs.

4.2 Results on CIFAR-10

CIFAR-10 is a small-scale dataset which has 50,000 training images and 10,000 testing images in 10 classes. We evaluate our method on CIFAR-10 with ResNet of two different depths (56/110). We set the parameter α in equation (4) to 0.1. In the pruning stage, the learning rate is 0.01. SGD optimizer is used in our experiments. When the pruned model reaches the target FLOPs, we fine-tune the model with the learning rate 0.01 and divide it by 10 at the fixed epoch (*e.g.*, 60, 90, 120, 150). All the training experiments are conducted on one Tesla V100 GPU with a batch size of 512. We pad the input data boundaries with 4 zero pixels as data augmentation.

To evaluate the effectiveness of the proposed method, we reduce 55.0% and 60.0% FLOPs on ResNet56 and ResNet110. The results are shown in **Table 1**. From **Table 1**, we can see that for ResNet56, FCR achieves more FLOPs reduction than FPGM [20], and the Pruned Top1 accuracy of our pruned model exceeds the model in [20] by 0.15%. For almost 60.0% FLOPs reduction, our method achieves much smaller accuracy decrease comparing with [9]. For ResNet110, to achieve the same accuracy on the pruned model, SASL [29] just reduces 51.7% FLOPs, but our method can reduce 54.9% FLOPs. These results on CIFAR-10 show that our method can achieve higher FLOPs reduction and maintain competitive performance against other methods.

Table 1. Pruning results of ResNet-56/110 on CIFAR-10

Depth	**Method**	**Base Top1(%)**	**Pruned Top1(%)**	**Top1↓(%)**	**FLOPs(1*e*8)**	**FLOPs↓(%)**
56	PFEC [27]	93.04	93.06	-0.02	0.91	27.6
	GAL [9]	93.26	93.38	-0.12	0.78	37.6
	CP [19]	92.80	91.80	1.0	-	50.0
	HRank [8]	93.26	93.17	0.09	0.63	50.0
	FPGM [20]	93.59	93.49	-0.1	0.59	52.6
	Ours	93.56	93.64	-0.08	0.57	55.1
	GAL [9]	93.26	91.58	1.98	0.50	60.2
	Ours	93.56	93.42	0.14	0.50	60.4
110	PFEC [27]	93.53	93.30	0.23	1.55	38.6
	GAL [9]	93.50	92.74	0.76	1.30	48.5
	SASL [29]	93.83	93.99	-0.16	-	51.7
	FPGM [20]	93.68	93.74	-0.16	1.21	52.3
	Ours	93.75	93.99	-0.24	1.14	54.9
	HRank [8]	93.50	93.36	0.14	1.06	58.2
	Ours	93.75	93.91	-0.16	1.01	60.0

4.3 Results on ImageNet

ImageNet2012 dataset is a large-scale dataset which contains 1.28 million training images and 50,000 validation images in 1,000 classes. We evaluate our method on ImageNet2012 with ResNet of three different depths (18/34/50). We set the parameter α in equation (4) to 0.1. In the pruning stage, the learning rate is 0.01. SGD optimizer is used in our experiments. When the pruned model reaches the target FLOPs, the learning rate starts from 0.01 and is divided by 10 at the fixed epoch (*e.g.* 45, 75, 105). All the training experiments are conducted on four Tesla V100 GPUs with a batch size of 512.

The results are shown in **Table 2**. In Table 2, we can observe that for ResNet18, our method obtains 68.72% top1 accuracy with 42.0% FLOPs reduction. Compared to FPGM [20], the accuracy of our method exceeds 0.38% in the similar FLOPs reduction. Our method gets 68.28% top1 accuracy with 51.0% FLOPs reduction. The top1 accuracy of FBS [30] decreases 2.44% around 50.0% FLOPs reduction, while that of our method only decreases 1.48%. For ResNet32, comparing with SFP [11], our method obtains a better result around 41.0% FLOPs. Meanwhile, our method reduces FLOPs 45.0% to obtain much smaller top1 accuracy decrease against the pretrained model. For ResNet50, under the similar 56.0% FLOPs reduction, the top1 accuracy of our method is 74.88%, which exceeds 2.19% compare with that of TRP [34]. From **Table 2**, we can see that the results of our method can obtain a better performance against other methods.

Table 2. Pruning results of ResNet18/34/50 on ImageNet2012 dataset. SFP-w/o-FT[11] indicates the SFP prune method without pretrained model in [11]

Depth	Method	Base Top1(%)	Pruned Top1(%)	Base Top5(%)	Pruned Top5(%)	Top1↓ (%)	Top5↓ (%)	FLOPs↓ (%)
18	FPGM [20]	70.28	68.34	89.63	88.53	1.94	1.10	41.8
	Ours	69.76	68.72	89.08	88.44	1.04	0.42	42.0
	DCP [25]	69.21	67.25	88.86	87.60	1.96	1.26	46.0
	FBS [30]	70.71	68.27	89.68	88.22	2.44	1.46	49.5
	Ours	69.76	68.28	89.08	88.05	1.48	1.03	51.0
34	PFEC [27]	73.23	72.17	-	-	1.06	-	24.2
	SFP-w/o-FT[11]	73.92	71.83	91.62	90.33	2.09	1.29	41.1
	Ours	73.31	72.32	91.42	90.69	0.99	0.73	41.7
	Ours	73.31	72.27	91.42	90.72	1.04	0.70	45.0
50	SFP [8]	76.15	62.14	92.87	84.06	14.01	8.27	41.8
	HRank [31]	76.15	74.98	92.87	92.33	1.17	0.53	43.8
	Taylor-FO [32]	76.18	74.50	-	-	1.68	-	45.0
	Ours	76.13	75.37	92.86	92.51	0.76	0.35	45.0
	LFC [20]	75.30	73.40	92.20	91.40	1.90	0.8	50.0
	FPGM [13]	76.15	74.83	92.87	92.32	1.32	0.55	53.5
	GAL [9]	76.15	71.80	92.87	90.82	4.35	2.05	55.0
	C-SGD [33]	75.33	74.54	92.56	92.09	0.79	0.47	55.8
	ThiNet [26]	75.30	72.03	92.20	90.99	3.27	1.21	55.8
	TRP [34]	75.90	72.69	92.70	91.41	3.21	1.49	56.5
	Ours	76.13	74.88	92.86	92.26	1.25	0.6	56.7
	GDP [35]	76.15	70.93	92.30	90.14	5.22	2.16	61.6
	HRank [8]	76.15	71.98	92.87	91.01	4.17	1.86	62.1
	Ours	76.13	73.42	92.86	91.49	2.71	1.37	72.6

4.4 Impact of filter contribution collecting Rate

The filter contribution collecting rate α in equation (4) is a critical parameter for our method, it controls the contribution shared volume. To explore the impact of α, we set different values (*e.g.*, 0.01/0.05/0.1/0.5) for it on ImageNet2012 with ResNet18. And the experiments are conducted with 60.0% FLOPs reduction. The experimental results of the pruning stage and the fine-tuning one are shown in **Fig. 3** and **Fig. 4**, respectively.

In **Fig. 3**, it is obvious that the pruning speed and accuracy are largely different among different collecting rates. In our method, the parameters of a network are updated not only by the SGD optimizer, but also by the FCR module. For a larger collecting rate (*e.g.* 0.5), the filter parameters could obtain greater update volume than other smaller collecting rates, but the accuracy is worse than others. This is because a larger collecting rate disturbs the balance among different filter groups and makes the training process unstable to result in poor accuracy in the pruning stage. To achieve the target FLOPs reduction, the time consumed by the smaller collecting rate (*e.g.* 0.01) is several times compared with those used by other collecting rates. The FCR method with a smaller collecting rate weakens the contribution

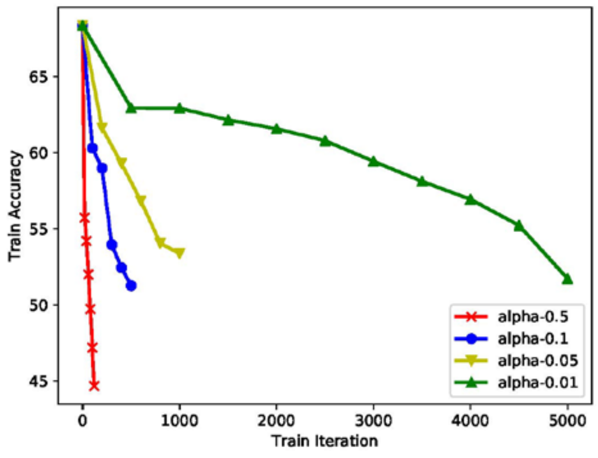

Fig. 3. The curve for model pruning stage with different filter collecting rates α in equation (4). The lowest point for each curve is the ending to achieve the target FLOPs reduction.

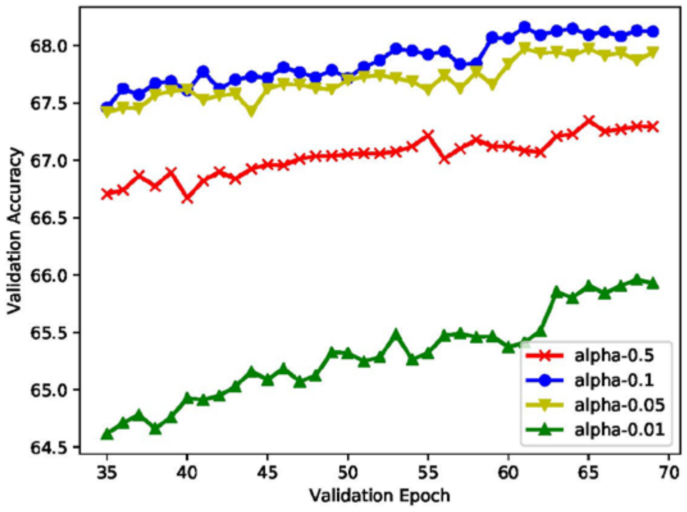

Fig. 4. The last 35 epochs validation accuracy in fine-tuning stage for different filter collecting rates.

transformation among the filters, and it actually becomes an enhanced weight decay module. In the fine-tuning stage, the intermediate rates 0.1 and 0.05 both take an advantage in accuracy. The accuracy advantage benefits from the proper collecting rate in the pruning stage, which purpose is to find the most valuable parameters and combination to inherit the contribution from the original model.

In addition, in the pruning stage, the rate of 0.01 achieves almost accuracy comparing with 0.1 and 0.05, but it is the lowest one in the fine-tune stage. The obvious gap results from the insufficient transformation of the filter contribution. Meanwhile, the gap shows that although the pruned model can maintain a competitive performance in the pruning stage, the remaining parameters and combinations are not the best ones and cannot achieve a best accuracy. On the other hand, the FCR module not only inherits massive effective filter contribution from the pretrained model, but also finds the optimal architecture among variety filter connections. To sum up, the experimental results show that the FCR module is useful for model pruning.

4.5 Combine with different contribution criterions

The proposed FCR is a flexible module, which can combine with different pruning criterions to build different pruning methods. To explore the composability of FCR, we combine FCR with LASSO, L2-norm, Taylor-expansion respectively to analyze their accuracy on ImageNet2012 using ResNet34. All the training parameters and training strategies are the same. The target FLOPs reduction is 50.1%. The experimental result is shown in **Fig. 5**, where we use lasso, l2 and taylor_fo to denote LASSO, L2-norm, and Taylor-expansion, respectively. The final accuracy is shown in **Table 3**. From **Table 3**, we can see that FCR combines with the LASSO criterion has a slight advantage to obtain the best accuracy. From **Fig. 5**, it is clear that FCR can achieve a competitive performance combines with different criterions, which is a superior auxiliary module in the pruning domain.

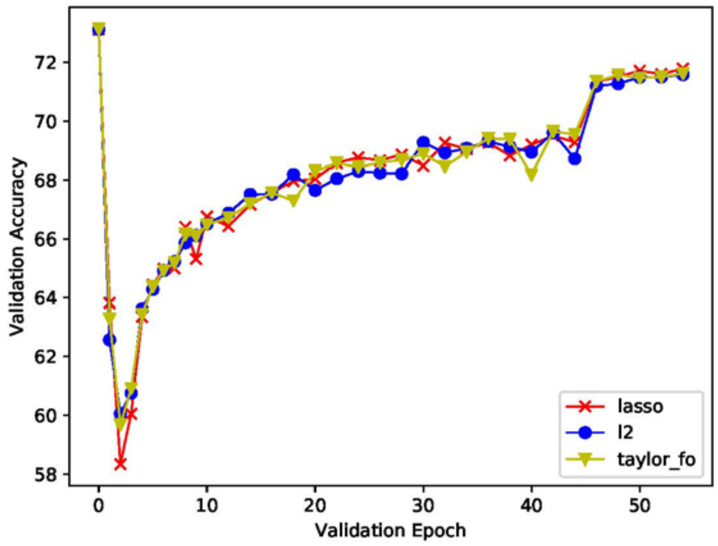

Fig. 5. Comparison of different filter contribution criterions with the FCR method.

Table 3. The pruning results of the FCR module combines with different criterions

Criterion	Pruned Top1(%)	Pruned Top5(%)	Top1↓(%)	Top5↓(%)
l2	71.71	90.38	1.60	1.04
taylor-fo	71.74	90.57	1.57	0.85
lasso	71.84	90.50	1.47	0.92

4.6 Is weight useful for model pruning?

Comparing with training a pruned model with random initialized weights, fine-tuning a pruned model only obtains comparable or worse accuracy [1]. Li *et al*. [1] argued that the weights of the pruned model could not improve the accuracy of the model, and that the architecture of the pruned model is more important than the pruned weights. In the pruning phase of the model in [1], the small norm filters are discarded directly, which results in a waste of filter contribution. This is a general method in the field of model pruning. We point out that the model architecture and the parameters are both useful for the pruned model. The goal of the model pruning is to find an optimal architecture and a minimal filter set to inherit filter contribution from the original model. The proposed FCR method is effective to improve the accuracy of the pruned model by recycling the contribution of the discarded filters. We compare the pruning results of Li *et al*. [1], Huang *et al*. [36] and the proposed FCR on ImageNet2012 with ResNet50 in **Table 4**, where we use Rethink and SSS to denote the methods used in [1] and [36], respectively. From **Table 4**, the accuracy of the proposed FCR is higher than those of Rethink and SSS in the similar FLOPs reduction. We concluded that, a method with the parameter reutilization, like the proposed FCR, can achieve better accuracy than those without the recycle strategy. And the comparison results implicitly demonstrate that the contribution of the parameters in the pruning stage cannot be ignored for the final effectiveness of the model.

Table 4. The pruning results of SSS [36], Rethink [1] and the proposed FCR on Imagenet with ResNet50

Depth	Method	Base Top1(%)	Pruned Top1(%)	Top1↓(%)	FLOPs (1*e*9)
50	SSS [36]	76.12	75.44	0.68	3.473
	Rethink [1]	76.12	76.17	-0.05	3.473
	Proposed FCR	76.13	75.97	0.15	3.403
	SSS [36]	76.12	74.18	1.94	2.818
	Rethink [1]	76.12	74.67	1.45	2.818
	Proposed FCR	76.13	75.37	0.76	2.250
	SSS [36]	76.12	71.82	4.30	2.329
	Rethink [1]	76.12	73.41	2.71	2.329
	Proposed FCR	76.13	74.88	1.25	1.74

5. Conclusion and Future Work

In this paper, we propose a novel filter contribution recycle (FCR) for filter pruning. Different from the existing methods, FCR designs a contribution collector to inherit contribution from the droppable filters, and assigns the collected contribution to the preserved filters which have higher probability to survive. FCR can effectively avoid the contribution waste of the small norm filters. In addition, FCR constructs a filter decay strategy for the droppable filters to discard it. FCR achieves a superior performance compared with the state-of-the-art pruning methods. In the future, we plan to work on how to design an auxiliary post-processing method after the pruning stage, which can enhance the performance of the pruned model further with the discarded parameters.

References

[1] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, “Rethinking the value of network pruning,” in *Proc. of ICLR 2019*, pp. 1-21, 2019. Article (CrossRef Link)

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. Article (CrossRef Link)

[3] A. Krizhevsky, I. Sutskever, and G. E Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no, 6, pp. 84-90, 2017. Article (CrossRef Link)

[4] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: keypoint triplets for object detection," in *Proc. of the IEEE/CVF International Conference on Computer Vision*, pp. 6568-6577, 2019. Article (CrossRef Link)

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, pp.1904–1916, 2015. Article (CrossRef Link)

[6] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "HRank: filter pruning using high-rank feature map," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1526-1535, 2020. Article (CrossRef Link)

[7] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. of the IEEE/CVF 963 Conference on Computer Vision and Pattern Recognition*, pp. 2785-2794, 2019. Article (CrossRef Link)

[8] H. Yang, Q. Yao, B. Bao, A. Yu, J. Zhang, A. V. Vasilakos, "Multi-associated parameters aggregation-based routing and resources allocation in multi-core elastic optical networks," *IEEE/ACM Transactions on Networking*, vol. 30, no. 5, pp. 2145-2157, 2022. Article (CrossRef Link)

[9] C. Li, H. Yang, Z. Sun, Q. Yao, B. Bao, J. Zhang, A. V. Vasilakos, "Federated hierarchical trust-based interaction scheme for cross-domain industrial IoT," *IEEE Internet of Things Journal*, pp. 1–1, 2022. Article (CrossRef Link)

[10] S. Han, H. Mao, and W. J Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2016. Article (CrossRef Link)

[11] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. of the 27th International Joint Conference on Artificial Intelligence*, pp. 2234–2240, 2018. Article (CrossRef Link)

[12] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *Proc. of Twenty-eighth Conference on Neural Information Processing Systems*, 2014.

[13] Z. Huang, and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," *arXiv preprint arXiv:1707.01219*, 2017. Article (CrossRef Link)

[14] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," in *Proc. of the 33rd International Conference on Neural Information Processing*, pp. 7950–7958, 2019. Article (CrossRef Link)

[15] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018. Article (CrossRef Link)

[16] Y. Cun, J. Denker, and S. Solla, "Optimal brain damage," in *Proc. of the 2nd International Conference on Neural Information Processing Systems*, pp. 598–605, 1989. Article (CrossRef Link)

[17] S. Kim and E. Xing, "Tree-guided group lasso for multi-task regression with structured sparsity," in *Proc. of the 27th International Conference on International Conference on Machine Learning*, pp. 543–550, 2010.

[18] T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: faster training without losing performance," *arXiv preprint arXiv:1907.04840*, 2019. Article (CrossRef Link)

[19] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. of the IEEE International Conference on Computer Vision*, pp. 1398-1406, 2017. Article (CrossRef Link)

[20] Y. He, P. Liu, Z. Wang, Z. Hu, and Y Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4335-4344, 2019. Article (CrossRef Link)

[21] N. Lee, T. Ajanthan, and P. Torr, "Snip: single-shot network pruning based on connection sensitivity," *arXiv preprint arXiv:1810.02340*, 2018. Article (CrossRef Link)

[22] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, “A systematic dnn weight pruning framework using alternating direction method of multipliers,” in *Proc. of the European Conference on Computer Vision (ECCV)*, pp. 191-207, 2018. Article (CrossRef Link)
[23] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K. Cheng, and J. Sun, “Meta-pruning: meta learning for automatic neural network channel pruning,” in *Proc. of the IEEE International Conference on Computer Vision*, pp. 3295-3304, 2019. Article (CrossRef Link)
[24] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang, and W. Liu, “Compressing convolutional neural networks via factorized convolutional filters,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3972-3981, 2019. Article (CrossRef Link)
[25] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, “Discrimination-aware channel pruning for deep neural networks,” in *Proc. of the 32nd International Conference on Neural Information Processing Systems*, pp. 883–894, 2018. Article (CrossRef Link)
[26] J. Luo, H. Zhang, H. Zhou, C. Xie, J. Wu, and W. Lin, “Thinet: pruning CNN filters for a thinner net,” *IEEE transactions on pattern analysis and machine intelligence*, vol.41, no.10, pp.2525–2538, 2019. Article (CrossRef Link)
[27] H. Li, A. Kadav, I. Durdanovic, H. Samet, and Hans Peter Graf, “Pruning filters for efficient convnets,” in *Proc. of 5th International Conference on Learning Representations*, 2017.
[28] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter pruning via geometric median for deep convolutional neural networks acceleration,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4335-4344, 2019. Article (CrossRef Link)
[29] J. Shi, J. Xu, K. Tasaka, and Z. Chen, “Sasl: saliency-adaptive sparsity learning for neural network acceleration,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol.31, no.5, pp. 2008–2019, 2021. Article (CrossRef Link)
[30] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C. Xu, “Dynamic channel pruning: feature boosting and suppression,” in *Proc. of 7th International Conference on Learning Representations*, 2019. Article (CrossRef Link)
[31] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, “Importance estimation for neural network pruning,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11264–11272, 2019. Article (CrossRef Link)
[32] P. Singh, V. Kumar Verma, P. Rai, and V. Namboodiri, “Leveraging filter correlations for deep model compression,” in *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 824-833, 2020. Article (CrossRef Link)
[33] X. Ding, G. Ding, Y. Guo, and J. Han, “Centripetal sgd for pruning very deep convolutional networks with complicated structure,” in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4938-4948, 2019. Article (CrossRef Link)
[34] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong, “Trp: trained rank pruning for efficient deep neural networks,” in *Proc. of the Twenty-Ninth International Joint Conference on Artificial Intelligence Main track*, pp.977–983, 2020. Article (CrossRef Link)
[35] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, “Accelerating convolutional networks via global & dynamic filter pruning,” in *Proc. of the 27th International Joint Conference on Artificial Intelligence*, pp. 2425–2432, 2018. Article (CrossRef Link)
[36] Z. Huang and N. Wang, “Data-driven sparse structure selection for deep neural networks,” in *Proc. of the European Conference on Computer Vision*, pp. 317–334, 2018. Article (CrossRef Link)

Zehong Chen received the Ph.D. degree in information and communication engineering from Shenzhen University, Shenzhen, China, in 2019. She joined the Huizhou University, Huizhou, China in August 2019. Now she is a teacher of School of Computer Science and Engineering of Huizhou University. Her current research interests include information security, privacy preserving and deep learning.

Zhonghua Xie is currently an associate professor with the School of Computer Science and Engineering of Huizhou University, China. His research interests include compressed sensing, deep learning, and image reconstruction. He received the Ph.D. degree in information and communication engineering from the South China University of Technology, China.

Zhen Wang received the Ph.D. degree in computer technology and application from the Faculty of Information Technology, Macau University of Science and Technology (MUST), China, in June 2017. From 2006 to 2017, he was a teacher of Beijing Institute of Technology, Zhuhai. He joined the Huizhou University, Huizhou, China in October 2017. Currently, he is a teacher of School of Computer Science and Engineering of Huizhou University. His research interests focus on vehicular networks and intelligent transportation system.

Tao Xu received the Ph.D. degree in computer science and technology from Northwestern Polytechnical University, Xi’an, China, in 2009. He has worked in teaching and scientific research with Huizhou University since 2009. His research interests include multimedia information processing and digital watermark.

Zhengrui Zhang received the Ph.D. degree in information and communication engineering from Shenzhen University, Shenzhen, China, in 2019. He has worked in teaching and scientific research with Huizhou University since 2020. His current research interests include medical image processing, deep learning and cloud computing.