Fine-Grained Channel Pruning for Deep Residual Neural Networks

DOI: 10.1007/978-3-030-61616-8_1
Handle: http://hdl.handle.net/1942/33444

Fine-grained Channel Pruning for Deep Residual Neural Networks*

Siang Chen[1], Kai Huang[1](✉), Dongliang Xiong[1], Bowen Li[1], and Luc Claesen[2]

[1] Institute of VLSI Design, Zhejiang University, Hangzhou, China
{huangk,11631032,xiongdl,11631033}@zju.edu.cn
[2] Engineering Technology - Electronics-ICT Dept, Hasselt University, 3590 Diepenbeek, Belgium
luc.claesen@uhasselt.be

Abstract. Pruning residual neural networks is a challenging task due to the constraints induced by cross layer connections. Many existing approaches assign channels connected by skip-connections to the same group and prune them simultaneously, limiting the pruning ratio on those troublesome filters. Instead, we propose a Fine-grained Channel Pruning (FCP) method that allows any channels to be pruned independently. To avoid the misalignment problem between convolution and skip connection, we always keep the residual addition operations alive. Thus we can obtain a novel efficient residual architecture by removing any unimportant channels without the alignment constraint. Besides classification, We further apply FCP on residual models for image super-resolution, which is a low-level vision task. Extensive experimental results show that FCP can achieve better performance than other state-of-the-art methods in terms of parameter and computation cost. Notably, on CIFAR-10, FCP reduces more than 78% FLOPs on ResNet-56 with no accuracy drop. Moreover, it achieves more than 48% FLOPs reduction on MSR-ResNet with negligible performance degradation.

Keywords: Channel pruning · Residual neural network · Efficient network structure.

1 Introduction

Despite the superior performance of deep convolutional neural networks in machine learning, the massive computation and storage consumption prevents its deployment in resource constraint devices. Pruning is a promising way for convolutional neural network (CNN) model size compression by identifying and removing unnecessary neurons without significant performance degradation. Recent studies on neural network pruning can be divided into either non-structured [5] or structured pruning [21], the former prunes weight independently, thus always results in structures that are unfriendly for hardware acceleration, while

* Supported by the National Key R&D Program of China (2018YFB0904900, 2018YFB0904902).

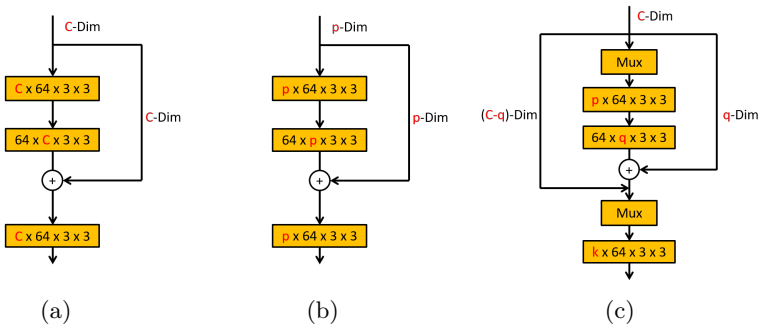

Fig. 1. An illustration of **(a)**: baseline structure. **(b)**: structure pruned by group strategy. **(c)**: structure pruned by our fine-grained strategy. The red letter denotes the channel number.

the latter aims at removing parameters in units of filters which can take advantage of fast dense matrix multiplication [4]. Among the structured pruning methods, channel pruning (a.k.a filter pruning) [9] directly eliminates entire channels in each layer with no special hardware design required.

As the neural network becomes deeper and wider, it is a challenge to effectively train a very deep model. One good solution is residual learning [6], which leverages the shortcut connection between layers to reformulate the layers as learning residual information. The conception of residual learning has been widely used to design efficient neural network architectures. ResNet is one of the most popular residual architectures, which performs residual mapping by a shortcut and element-wise addition. Directly applying pruning methods to residual neural networks, however, brings some problems. Specifically, pruning filters of the last convolution in each basic block independently will lead to the misalignment between the skip connection and the corresponding output feature maps. Therefore, various works have been made to tackle these problems. [17] avoid pruning these troublesome layers. [16] prune pre-activation models by inserting an additional channel selection layer before the first convolution in each residual block. [13] apply the mixed block connection to avoid such problem. Recently, [3, 4, 22] all propose the Group Pruning for those layers connected by skip connections. Unfortunately, pruning in a group technique will lead to models shown in Figure. 1(b) such that all corresponding connections of one eliminated filter should be removed simultaneously, limiting the performance at especially high pruning ratios.

In this paper, we propose a novel Fine-grained Channel Pruning (FCP) approach as shown in Figure. 1(c), which solves the constraint that the pruning problem encounters when pruning residual neural networks. Instead of focusing on measuring the importance of filters, we insert gate function into all channels between layers, and transfer the problem of optimizing filter numbers into minimizing data transmissions. By paying attention on estimating the importance of each channel independently, we allow both input and output

channels of convolutions to be pruned. The FCP method provides a larger search space for unimportant filter selections, thus can achieve a more fine-grained channel allocation between layers. Our contributions are summarized as follows:

(1) We analyze the pruning of residual neural network in detail and observe that the state-of-the-art group strategy is a coarse pruning that is still limited by the alinement constraint.

(2) We propose FCP to allow any channels between residual blocks to be pruned independently while keeping constant numbers of skip connections. By performing such pruning strategy, we can obtain a novel efficient residual network structure, of which connections can fully skip the residual building block.

(3) We demonstrate the effectiveness of FCP on both classification and image representation (super-resolution) tasks. The extensive experiments show that FCP can prune more parameters and FLOPs with less performance drop than state-of-the-art methods.

2 Related Work

Model pruning has shown great success in neural network compression by removing unimportant neurons with negligible performance degradation. Despite the deep compression of parameters, pruning individual weights [5] always leads to unstructured models, which makes it difficult to implement realistic speedup unless special software and hardware are designed. Therefore, many researches focus on filter pruning. [14] prune filters in each layer with small *l1*-norm magnitude. [16] impose sparsity-induced regularization on the scaling factor in batch normalization layers, and identify insignificant channels with small scaling factors. [8] prune the most replaceable filters containing redundant information by analyzing geometric median distance. Our work also falls into the category of channel pruning.

The problem of vanishing gradient prevents neural networks from becoming deeper to demonstrate higher performance. To address this problem, [6] apply element-wise addition on the feature maps between two residual blocks, which is known as ResNet. [10] connect each layer to every other layer in a feed-forward fashion, which fully exploits the advantages of skip connections and reduces the number of parameters as well. While the existence of skip connections makes it effective to train a very deep network, methods for pruning plain networks like VGG [19] and AlexNet [12] can not be applied to residual models directly: pruning the filters of each layer independently will result in misalignment of feature maps between residual blocks. [17] avoid this problem by only pruning the internal layers in residual blocks. [16] place a channel selection layer before the first convolution in each residual block to mask out insignificant channels, and leave the last convolution layer unpruned, which only works for pre-activation networks. [13] use a mixed block connectivity to avoid redundant computation. Recently, [3, 4, 22] propose to assign the layers connected by pure skip connections into the same group, thus the filters in the same group can be

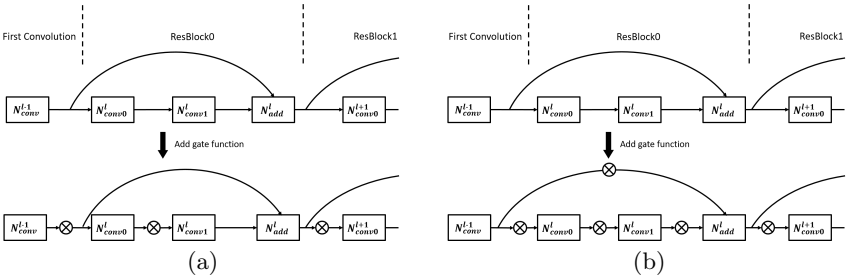

Fig. 2. Structures of ResBlock with node transformation before and after inserting the gate function (symbol $\bigotimes$). **(a)**: group pruning. **(b)**: our fine-grained pruning

pruned simultaneously. However, the above methods still can not remove each channel between residual blocks independently.

3 Approach

3.1 Rethinking pruning residual neural networks

To understand the relationship of channels between residual blocks more clearly in following sections, we first transform the operation on feature maps in the network into **Node**. Consider pruning two consecutive blocks in ResNet, as shown in Figure. 2(a) and 2(b). The structure consists of two basic Nodes:

1)**Node**$_{add}$: indicates the element-wise addition for channels of the same index from the output of the previous convolution and the corresponding skip connection, the operation can be defined as:

$$O_{add}^{l,c} = O_{conv1}^{l,c} + O_{conv}^{l-1,c} \tag{1}$$

where $O_{add}^{l,c}$ denotes the c-th output channel of Node$_{add}$ in l-th layer, $O_{conv1}^{l,c}$ denotes the c-th channel from the previous convolution.

2)**Node**$_{conv}$: indicates the regular convolution operation. Take the first convolution in ResBlock as an example:

$$O_{conv0}^{l+1,k} = \sum_{i=1}^{C} O_{add}^{l,c} * W^{l+1,c,k} \tag{2}$$

C is the total number of input channels, $O_{conv0}^{l+1,k}$ denotes the k-th output channel of convolution, $W_{l+1,c,k}$ denotes c-th input channel and k-th output channel weight. For representational simplicity, the kernel operation, bias term, BN and activation layers are not included in our formulation. Existing methods only estimate the importance of the output channel of convolutions as shown in Figure. 2(a), in order to allow all filters to be pruned, they regard the channel relationship as:

$$C(O_{conv}^{l-1}) = C(O_{add}^{l}) = C(O_{conv1}^{l}) = C(O_{conv1}^{l+1}) \tag{3}$$

C(x) denotes the set of input channels in x. Based on Eq. (1) and Eq. (2), pruning these channels is under the constraint that the output channel number of the Node^{l}_{conv1}, channel number of skip connection and input channel of $\text{Node}^{l+1}_{conv0}$ should be maintained the same. Therefore, importance score for these filters in the group are accumulated together, which makes them harder to be pruned, and always results in dense connections between residual blocks and very few connections inner residual blocks especially for high pruning ratios. Instead, we consider the problem of pruning from the perspective of gating feature maps. Different from [22] that only add gates after convolutions, we try to insert gate function on each channel except the output channel of Node^{l-1}_{conv} as shown in Figure. 2(b), thus the operations for Node^{l}_{add} become:

$$\begin{aligned} O^{l,c}_{add} &= I^{l,c}_{add} + R^{l,c}_{add}, \\ I^{l,c}_{add} &= g^{l,c}_{add1} \times O^{l,c}_{conv1}, \\ R^{l,c}_{add} &= g^{l,c}_{add2} \times O^{l-1,c}_{conv} \end{aligned} \tag{4}$$

while for $\text{Node}^{l+1}_{conv0}$:

$$\begin{aligned} O^{l+1,k}_{conv0} &= \sum_{c=1}^{C} I^{l+1,c}_{conv0} * W^{l+1,c,k}, \\ I^{l+1,c}_{conv0} &= g^{l+1,c}_{conv0} \times O^{l,c}_{add} \end{aligned} \tag{5}$$

Here $g^{l,c}_{add}$ is the gate function for the corresponding channel, of which the value is 0 for masking. $I^{l,c}_{add}$ and $R^{l,c}_{add}$ is the c-th output channel of Node^{l}_{conv1} and the skip connection after the gate function respectively. According to Eq. (1) and Eq. (2), the channel relationship is actually:

$$C(O^{l}_{add}) = C(O^{l-1}_{conv}) \cup C(O^{l}_{conv1}) \tag{6}$$

$$C(I^{l+1}_{conv0}) = C(O^{l}_{add}) \times g^{l+1}_{conv0} \tag{7}$$

Combined with Eq. (5), the channel set of weights in each convolution only depends on input feature map for each Node, which means pruning the input of each Node equals to pruning filters. Note that here we only analyze the channels between residual block. For channels inner the residual block or in the plain neural network, we treat them as a special case without skip connections. Therefore the problem of pruning filters in residual neural network can be transformed into the optimization of feature map channels.

3.2 Channel Importance

The biggest difference between our approach and group pruning is that we prune each channel independently, therefore the problem arises on how to measure the importance of both input and output channel in normalization. Magnitude-based [14, 8] or utilizing BN scaling factor [16] is not applicable for such pruning

strategy, we leverage Taylor expansion [18] in this work, and extend this method to a more general one. Instead of only considering the output of each convolution, we multiply each channel between convolutions by a trainable scaling factor α, then we estimate the change in loss function caused by setting α to zero, thus we get the importance score of the corresponding channel:

$$IS(\alpha) = |\frac{\partial L}{\partial \alpha}\alpha| \tag{8}$$

which can be easily computed during back-propagation. Therefore, the gate function can be defined as:

$$g(\alpha) = \begin{cases} 0, & IS(\alpha) < T \\ \alpha, & otherwise \end{cases} \tag{9}$$

where T is a global threshold that depends on pruning ratio and is computed by sorting the importance score.

Note that we can also prune the R_{add} in the same way as in the other channels. We however do not prune R_{add}, there are three reasons: First and most important, keep constant numbers of residual connections can avoid the problem of misalignment for output channels of last convolution in each block. Second, in the case that I_{add} is pruned, R_{add} can still provide information for following layers, which retains the full capacity of network to some extent. Third, pruning R_{add} can not eliminate filters directly.

According to Eq. (8), the importance score is zero when gradient or α is zero, while the gradient depends on training process, we can induce more sparsity by adding a sparse constraint on α.

$$L_{FT} = L_D + \lambda \sum_{\alpha \in \Phi} |\alpha| \tag{10}$$

where L_D is the loss function on data, λ is the penalty.

To guarantee the importance score is accurate enough, we compute Eq. (8) of each channel by accumulating individual contribution in one epoch, and prune p percentage of the total channels each time besides zero scaling factors, then we fine-tune the network for T epochs based on the loss function Eq. (10). We iteratively conduct the prune and fine-tune step until meeting the compression requirement.

3.3 Analysis of Residual Neural Network

In this section, we analyze the possible efficient structures for ResNet pruned and reconstructed by our approach. For simplicity, we take two consecutive basic blocks with at most two channels for each convolution as examples in Figure. 3(a), a regular ResNet can be treated as a combination of replicated structures. Possible pruning for channels inter ResBlock can be summarized as follows:

1) Pruning I_{conv}. As shown in Figure. 3(b), we only remove the input channel 0 of Node^{l}_{conv0} while keeping the corresponding output channel of $\text{Node}^{l-1}_{conv1}$,

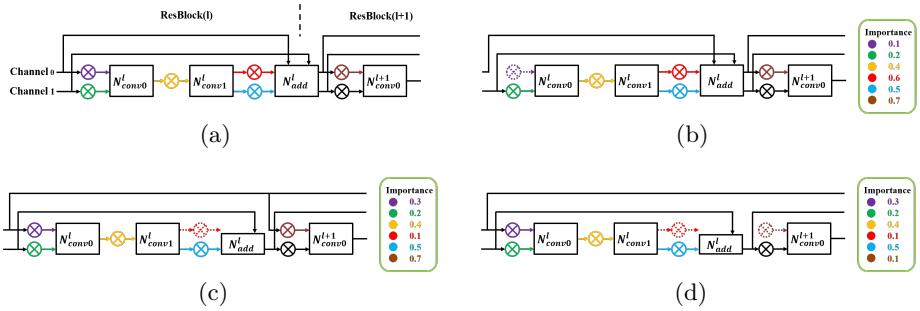

Fig. 3. Illustration of possible architectures pruned and reconstructed by FCP in channel-wise view, the dotted lines denote pruned channels. **(a)**: structure before pruning. **(b)**: only prune input channel. **(c)**: only prune output channel. **(d)**: prune both input and output channels

$O_{add}^{l-1,c}$ will bypass through Node_{conv0}^{l} and connect to Node_{conv1}^{l} as a skip connection. We only need to store the pruned or the remaining channel indices to assure correct input channel feature maps flow into $\text{Node}_{conv0}^{l+1}$, which is negligible to the total amount of parameters.

2) Pruning I_{add}. we eliminate the output channels of weight in Node_{conv1}^{l}, but we do not prune I_{conv0}^{l+1} simultaneously so that the skip connection of channel 0 becomes a pure input for $\text{Node}_{conv0}^{l+1}$ in Figure. 3(c). Similarly, We should store the pruned or remaining channel index to exclude the pruned channel from addition operation.

3) Pruning both I_{conv} and I_{add}. we can remove the output channel of Node_{conv1}^{l} and the input channel of $\text{Node}_{conv0}^{l+1}$ as shown in Figure. 3(d). However, different from group pruning or pruning in plain neural networks, we allow the skip connection to bypass through $\text{Node}_{conv0}^{l+1}$ and connect to $\text{Node}_{conv1}^{l+1}$ directly.

4 Experiments

In this section, We demonstrate the benefits of FCP for ResNet on both classification and super-resolution tasks. For classification, we use two standard benchmarks: CIFAR-10 and CIFAR-100 [11]. CIFAR-10 contains 50000 training images and 10000 testing images of size 32×32, which are categorized into 10 different classes. CIFAR-100 is similar to CIFAR-10 but has 100 classes. For super-resolution, we conduct MSRResNet [20] on the DIV2K dataset [2], which contains 800 high-resolution images.

To compare with other state-of-the-art pruning methods for residual neural networks, we define different pruning levels as follows:

Skip. Only prunes the channels inner ResBlock.

In-only. Only prunes the input channels for the first convolution in each ResBlock.

Out-only. Only prunes the output channels for the last convolution in each ResBlock.

Group. This strategy prunes the channels connected by pure shortcut connections together.

4.1 Experimental Settings

Training setting. For CIFAR-10 and CIFAR-100 datasets, we use the default parameter settings as [7]. For super-resolution on DIV2K dataset, we refer to the open-source platform BasicSR. The HR images are cropped into small images with size 480×480 by step 240, and the number of small images is 32208. The LR images with size 32×32 are randomly cropped from small images, then ratated by 90°, 180°, 270° and flip them horizontally. We optimize the weights via ADAM with batchsize=16, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The initial learning rate is set to $2\times\ 10^{-4}$ and reduced to half every 500 epochs for 1500 epochs totally.
Pruning setting. We prune models by following the Tick-Tock setup in [22]. All the networks are pruned 0.2% filters in each Tick stage for 10 epochs, followed by one Tock phase of 10 epochs for classification and 20 epochs for super-resolution, respectively. In the case of CIFAR-10 and CIFAR-100, the learning rate used in the Tick stage is set to 10^{-3}, we use the 1-cycle strategy to linearly increase the learning rate from 10^{-3} to 10^{-2} in the first half of the iteration, then linearly decrease it from 10^{-2} to 10^{-3}, sparse constraint λ is set to 10^{-3}. For DIV2K, the learning rate is set to $2\times\ 10^{-7}$ in the Tick stage, and increases from $2\times\ 10^{-7}$ to $2\times\ 10^{-5}$ in the first half of the iteration, and then linearly decreases from $2\times\ 10^{-7}$ to $2\times\ 10^{-5}$. For fine-tuning stage, we use the same learning rate strategy as the Tock phase, the difference is that fine-tune epochs are 40 for classification and 125 for super resolution.

4.2 Results on Classification

CIFAR-10. Table. 1 shows the results. Our FCP achieves a better performance than other state-of-the-art pruning methods for ResNet. For example, GBN [22] use the group strategy to prune ResNet-56 by 70.3% FLOPs with only 0.03% accuracy drop, we can however achieve no accuracy drop while pruning 7.75% more FLOPs and 5.44% more parameters. FPGM [8] prune ResNet-20 by 42.2% FLOPs with 1.11% accuracy drop, we can achieve even 0.09% better accuracy than baseline with more FLOPs pruned. Figure. 4 shows the pruning result of ResNet-56 on CIFAR-10, active channel numbers between ResBlocks are not limited to be the same, and some layers are even totally pruned to allow feature maps of previous layer directly flow into next layer.
CIFAR-100. As shown in Table. 1, results on ResNet-32 and ResNet-164 demonstrate that FCP outperforms previous methods on CIFAR-100 again. For ResNet-164, there are few works on this pre-activation model which adds more constraints on pruning, but FCP can still reduce more than 60% FLOPs with a even 0.19% accuracy increase.

Table 1. Comparison of pruning ResNet on CIFAR-10 and CIFAR-100

Dataset	Depth	Method	Baseline acc.(%)	Acc. ↓(%)	Params ↓(%)	Flops ↓(%)
CIFAR-10	20	FPGM[8]	92.20	1.11	-	42.20
		GBN[22]	92.07	0.68	36.08	44.53
		Ours	92.07	**-0.09**	**36.62**	**48.46**
	32	FPGM[8]	92.63	0.32	-	41.50
		GBN[22]	93.22	0.46	35.93	44.59
		Ours	93.22	**0.10**	**38.70**	**52.10**
	56	He et al.[9]	92.80	1.00	-	50.00
		FPGM[8]	93.59	0.10	-	52.60
		GBN[22]	93.10	0.03	66.70	70.30
		Ours(60%)	93.10	**-0.37**	61.00	70.08
		Ours(78%)	93.10	0.00	**72.14**	**78.05**
	110	FPGM[8]	93.68	-0.16	-	52.30
		GBN[22]	94.02	-0.05	58.04	54.17
		Ours	94.02	**-0.24**	**58.89**	**68.01**
CIFAR-100	32	FPGM[8]	69.77	1.25	-	41.50
		GBN[22]	70.27	1.38	20.82	44.36
		Ours	70.27	**0.32**	**20.96**	**50.02**
	164	Li et al.[16]	76.63	0.54	-	50.60
		Ours	76.25	**-0.19**	**34.04**	**60.07**

These results validate the effectiveness of FCP, which can produce a more compressed ResNet model with nearly the same or better performance compared to the original model.

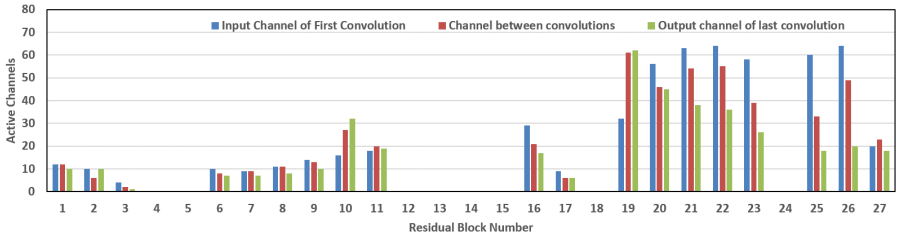

Fig. 4. Result of 70% FLOPs pruned by our method on ResNet-56-CIFAR-10

4.3 Comparison of different pruning levels

To fairly validate the effectiveness of FCP at different pruning ratios, we compare different pruning levels using the same pruning criterion and settings in this paper.

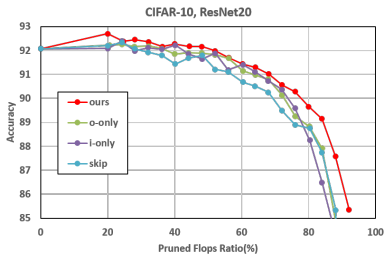

Fig. 5. The effect of varying percentages of FLOPs by different pruning levels

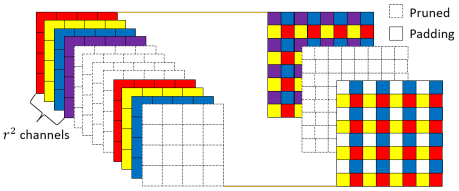

Fig. 6. Padding-and-Pruning strategy for the pixelshuffle layer

Figure. 5 shows the results. The skip strategy produces the smallest search space of channels and is more sensitive to pruning in most cases. Since in-only and out-only both can be treated as subsets of FCP, their results are similar. FCP is more fine-grained than the other three pruning levels, thus is more robust against pruning ratio and performs better especially at deeper pruning ratios.

4.4 Results on Super Resolution

Table 2. Quantitative results of evaluated methods for x4 SR

Method	Params	FLOPs	Set5 PSNR/SSIM	Set14 PSNR/SSIM	B100 PSNR/SSIM	Manga109 PSNR/SSIM
Bicubic	-	-	28.63/0.8138	26.21/0.7087	26.04/0.6719	25.07/0.7904
EDSR	43090K	2894.5G	32.46/0.8968	28.80/0.7876	27.71/0.7420	31.02/0.9148
MSRResNet	1517K	146.0G	32.22/0.8952	28.63/0.7826	27.59/0.7357	30.48/0.9089
CARN	1592K	90.8G	32.13/0.8937	28.60/0.7806	27.58/0.7349	30.45/0.9073
Li et al.[14]	861K	78.69G	32.03/0.8931	28.54/0.7803	27.53/0.7346	30.23/0.9056
FPGM[8]	859K	83.94G	31.95/0.8917	28.48/0.7790	27.48/0.7332	30.03/0.9033
GBN[22]	863K	75.76G	32.09/0.8944	28.58/0.7815	27.56/0.7356	30.36/0.9075
Ours (60%)	973K	90.29G	32.18/0.8947	28.61/0.7823	27.58/0.7362	30.44/0.9084
Ours (50%)	799K	75.73G	32.15/0.8946	28.58/0.7816	27.57/0.7358	30.40/0.9080

In MSRResNet, upscaled features are generated by the pixelshuffle layer, which reshapes feature maps from $H \times W \times r^2 C$ to $rH \times rW \times C$ in a periodic way. Here the input image is assumed to be of size $H \times W \times C$, r is the scale factor. Thus there is a constraint on pruned channels of convolution before the pixelshuffle layer, we apply a novel padding-and-pruning approach to address this problem. As shown in Figure. 6, for each periodic r^2 channels with more than one channels remaining after pruning, we extend them to the original r^2 channel size

by padding, and others are pruned away. Those padding channels will become blank pixels after the pixelshuffle layer, which means the final HR images consist of pixels directly from bilinear interpolation. This also makes sense that some pixels by bilinear interpolation may be good enough and convolutions for those pixels can be skipped.

We evaluate our pruned model on four standard benchmark datasets, including Set5, Set14, B100 and Manga109. Results are evaluated with PSNR and SSIM on Y channels of transformed YCbCr space. To show the effectiveness of FCP, we implement the state-of-the-art pruning methods: L1-norm based [14], FPGM [8], GBN [22].

Table. 2 shows the parameters, FLOPs and performance for $\times 4$ SR. FCP can reduce more parameters and FLOPs while maintaining higher PSNR and SSIM on all datasets than other approaches. When compared to state-of-the-art models, we can achieve nearly 64% parameters and 62% computation cost of the baseline model with negligible performance drop, and SSIM on dataset B100 can be even better than original model. Our 60% pruned model performs better than CARN [1] on most of the datasets, but the parameters of our pruned model is 619K less. EDSR [15] optimizes performance by increasing network depth and width, but too much parameters and FLOPs limit the application on resource-constrained devices, while our pruned model has almost $54\times$ reduction in model size and $38\times$ reduction in computation cost. These results demonstrate that FCP can achieve better performance with a comparable compression ratio on pixel-level tasks.

5 Conclusion

In this paper, we propose a fine-grained channel pruning (FCP) approach for deep residual networks. Unlike previous works that prune in a group technique, we allow any channels between convolution layers to be pruned, our approach multiplies a scaling factor on each channel, then we compute the importance score based on Taylor expansion, finally we obtain the compact model by removing unimportant channels independently, which reveals a novel residual structure for efficient model design. Extensive experiments show that FCP outperforms other state-of-the-art filter pruning approaches on both classification and super-resolution tasks.

References

1. Ahn, N., Kang, B., Sohn, K.: Fast, accurate, and lightweight super-resolution with cascading residual network. In: ECCV 2018. pp. 256–272 (2018). https://doi.org/10.1007/978-3-030-01249-6_16
2. Bevilacqua, M., Roumy, A., Guillemot, C., Alberi-Morel, M.: Low-complexity single-image super-resolution based on nonnegative neighbor embedding. In: BMVC. pp. 1–10 (2012). https://doi.org/10.5244/C.26.135

3. Ding, X., Ding, G., Guo, Y., Han, J.: Centripetal SGD for pruning very deep convolutional networks with complicated structure. In: CVPR. pp. 4943–4953 (2019)
4. Gao, S., Liu, X., Chien, L., Zhang, W., Alvarez, J.M.: VACL: variance-aware cross-layer regularization for pruning deep residual networks. CoRR **abs/1909.04485** (2019), http://arxiv.org/abs/1909.04485
5. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural networks pp. 1135–1143 (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
7. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. In: IJCAI. pp. 2234–2240 (2018). https://doi.org/10.24963/ijcai.2018/309
8. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: CVPR. pp. 4340–4349 (2019)
9. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: ICCV. pp. 1398–1406 (2017). https://doi.org/10.1109/ICCV.2017.155
10. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: CVPR. pp. 2261–2269 (2017). https://doi.org/10.1109/CVPR.2017.243
11. Krizhevsky, A.: Learning multiple layers of features from tiny images. In: Technical report (2009)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NeurIPS. pp. 1106–1114 (2012)
13. Lemaire, C., Achkar, A., Jodoin, P.: Structured pruning of neural networks with budget-aware regularization. In: CVPR. pp. 9108–9116 (2019)
14. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: ICLR (2017)
15. Lim, B., Son, S., Kim, H., Nah, S., Lee, K.M.: Enhanced deep residual networks for single image super-resolution. In: CVPR Workshops. pp. 1132–1140 (2017). https://doi.org/10.1109/CVPRW.2017.151
16. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: ICCV. pp. 2755–2763 (2017). https://doi.org/10.1109/ICCV.2017.298
17. Luo, J., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: ICCV. pp. 5068–5076 (2017). https://doi.org/10.1109/ICCV.2017.541
18. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: CVPR. pp. 11264–11272 (2019)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015), http://arxiv.org/abs/1409.1556
20. Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., Loy, C.C.: ESRGAN: enhanced super-resolution generative adversarial networks. In: ECCV Workshops. pp. 63–79 (2018). https://doi.org/10.1007/978-3-030-11021-5_5
21. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: NeurIPS. pp. 2074–2082 (2016)
22. You, Z., Yan, K., Ye, J., Ma, M., Wang, P.: Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In: NeurIPS. pp. 2130–2141 (2019)