Adaptive FSP: Adaptive Architecture Search with Filter Shape Pruning

Aeri Kim, Seungju Lee, Eunji Kwon, and Seokhyeong Kang

Department of Electrical Engineering
Pohang University of Science and Technology, Pohang, South Korea
{arkim17, seungju825, eunjikwon, shkang}@postech.ac.kr

Abstract. Deep Convolutional Neural Networks (CNNs) have high memory footprint and computing power requirements, making their deployment in embedded devices difficult. Network pruning has received attention in reducing those requirements of CNNs. Among the pruning methods, Stripe-Wise Pruning (SWP) achieved a further network compression than conventional filter pruning methods and can obtain optimal kernel shapes of filters. However, the model pruned by SWP has filter redundancy because some filters have the same kernel shape. In this paper, we propose the Filter Shape Pruning (FSP) method, which prunes the networks using the kernel shape while maintaining the receptive fields. To obtain an architecture that satisfies the target FLOPs with the FSP method, we propose the Adaptive Architecture Search (AAS) framework. The AAS framework adaptively searches for the architecture that satisfies the target FLOPs with the layer-wise threshold. The layer-wise threshold is calculated at each iteration using the metric that reflects the filter's influence on accuracy and FLOPs together. Comprehensive experimental results demonstrate that the FSP can achieve a higher compression ratio with an acceptable reduction in accuracy.

Keywords: Deep Learning Optimization · Structured Pruning · Convolution Neural Networks.

1 Introduction

Deep Convolutional Neural Networks (CNNs) have been widely used in computer vision applications, such as image classification [6, 31], object detection [30, 29, 21], and segmentation [1, 24]. These successes rely on the tremendous number of parameters and computations of the networks. However, their high requirements in storage and computing resource make the networks hard to deploy in edge devices. To address this problem, numerous studies have proposed different approaches to compress the networks. Popular approaches include quantization [2], compact network design [10, 39], and network pruning [5, 23].

Network pruning methods can be classified into several categories: element-wise pruning, filter pruning, and stripe-wise pruning. Element-wise pruning [5] is the most fine-grained non-structured pruning method that removes the individual weights to obtain a high compression rate without sacrificing accuracy. However, because the positions of non-zero weights are irregular and random,

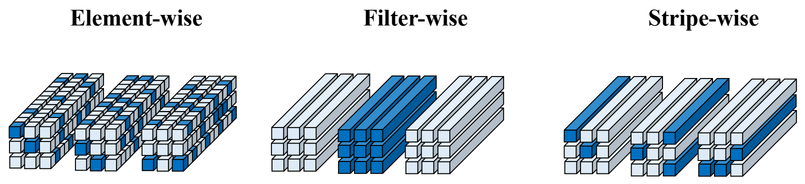


Fig. 1: Different pruning types of convolution layers.

element-wise pruning has irregular memory access and thus cannot achieve practical performance improvement without dedicated hardware. In contrast, filter pruning [23, 15, 9] is a structured method that removes the unimportant filters, which can directly achieve real performance improvement in general processors. However, filter pruning is less fine-grained than element-wise pruning; therefore, the further compression is limited. To overcome this weakness of filter pruning, Stripe-Wise Pruning (SWP) [27] has been introduced. SWP combines the strengths of element-wise pruning and filter pruning methods. Thus, SWP achieves finer granularity than traditional filter pruning and can still be accelerated in general processors. Fig. 1 shows the different pruning types of convolution layers.

SWP can obtain the optimal kernel shapes of filters by pruning unimportant stripes after training the importance of stripes in the filter using a learnable matrix called a FilterSkeleton (FS). The follow-up studies of SWP [12, 20, 22] focused on obtaining further optimized kernel shapes; therefore, the improvement of compression rate is insignificant or rather diminished. Moreover, when we visualize the filters of each layer in the model after SWP, several filters are found to have the same kernel shape as shown in Fig. 2. The kernel shapes significantly influence receptive fields, which are crucial for feature extraction. This indicates that the features extracted by filters with the same kernel shape have higher similarity than the others. Furthermore, this property causes filter redundancy, a phenomenon in which filters extract similar features that exist in duplicate. Thus, by pruning these filters, filter redundancy can be effectively reduced with little impact on accuracy. Motivated by these observations, we propose the Filter Shape Pruning (FSP) method, which prunes the model after SWP by kernel shape while preserving the receptive fields of kernel shapes. Moreover, we use "the FSP rule", a crucial concept in FSP, that maintains receptive fields. The FSP method can highly compress networks with a slight loss in accuracy.

In this study, we propose the Adaptive Architecture Search (AAS) framework that efficiently applies FSP to the networks. The AAS framework adaptively searches for an architecture that meets the target FLOPs and fine-tunes the pruned model from scratch. Previous adaptive pruning studies [33, 40] proposed a metric that only considers the accuracy or computational intensity of the pruned model. In contrast, we propose a metric that considers the effect of the filters on accuracy and FLOPs. This metric can find filters that generate many FLOPs while less sensitive in accuracy. Furthermore, using the proposed metric during the AAS framework, we use the adaptive layer-wise threshold that reflects

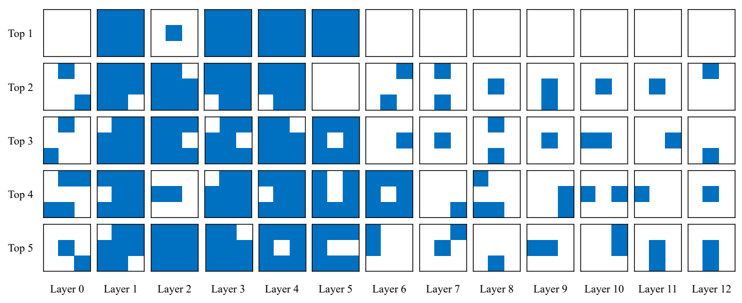


Fig. 2: Visualization of the VGG-16 filters pruned by SWP [27]. We illustrate the Top-5 filters according to their frequency in each layer, and the blue color indicates the remaining stripe after SWP.

the architecture, which varies at each iteration. Using our proposed framework, we can obtain an architecture consisting of filters that have good scores in both accuracy and FLOPs.

The main contributions of this study can be summarized as follows:

– We propose the FSP method that efficiently reduces filter redundancy while maintaining receptive fields of kernel shapes. Furthermore, we prune the network using "the FSP rule" to preserve the receptive fields. Using FSP, we can achieve a higher compression ratio than other pruning methods.
– We propose the AAS framework using a layer-wise threshold that adaptively changes. The layer-wise threshold is calculated by a metric reflecting the influence of the filter on accuracy and FLOPs together. The framework obtains the architecture that satisfies the target FLOPs using the FSP method.

The rest of the paper is organized as follows. Section 2 covers related works. Section 3 presents preliminaries. Section 4 describes the FSP method and the AAS framework. Section 5 provides the experimental results and analysis. Section 6 covers additional experiments. Finally, we summarize and conclude this study in Section 7.

2 Related Work

Filter Pruning Filter pruning is a structured pruning method that prunes unimportant filters. Therefore, it does not require dedicated hardware/libraries. Li et al. [15] pruned unimportant filters based on the l_1 norm of the filter weights. Liu et al. [23] used the scaling factor γ, a trainable variable of batch normalization, to remove unimportant channels from the output channels of each layer. He et al. [17] discovered that the average rank of several feature maps generated by a single filter is always the same, regardless of batch size. The authors formulated a process to prune filters with low-rank feature maps based on the principle that they contain less information. Lin et al. [35] proposed a metric that measures the correlations among different feature maps to perform efficient filter pruning using channel independence. The authors considered the less independent feature map as containing less useful information and pruned its corresponding filter. The networks pruned by filter pruning methods can be accelerated without dedicated hardware. However, because the filter pruning method uses a bigger unit than element-wise pruning, it has the disadvantage of a low compression rate. To solve this problem, SWP has been introduced.

Stripe-Wise Pruning Meng et al. [27] proposed Stripe-Wise Pruning (SWP) to prune more fine-grained networks than conventional filter pruning methods and can still be accelerated without dedicated hardware. SWP obtains the optimal kernel shape of the filter by learning the importance of the filter stripes. Liu et al. [22] developed the pruning framework called Squeezing More Out of Filters (SMOF) that reduces the kernel size with the "peeling" method and the number of filters of CNNs with Filter Mask, which learns the importance of filters. Huo et al. [12] proposed Balanced-Stripe-Wise Pruning (BSWP). BSWP balances stripes by adding a regulation term to the loss, reflecting the frequency of kernel stripes and the number of filter stripes within each layer. Liu et al. [20] proposed a two-stage approach that automatically finds the optimal kernel shape. The authors add three regulation terms to the loss: sparse regulation, direction-wise regulation, and group-wise regulation. Furthermore, they proposed a binary search algorithm to find the pruning threshold to meet the constraint. Because these studies focused on obtaining further optimized kernel shapes, the improvement in compression ratio is small or rather decreased. Therefore, the benefits of the fine granularity of SWP are underused. To address this problem, we propose the Filter Shape Pruning (FSP) method, which prunes the networks using the kernel shape while fully using the fine granularity of SWP.

Adaptive Pruning Method The adaptive pruning method determines the pruning rate by considering the trade-off between accuracy and reduction in computations at each iteration. Therefore, this method has a smaller accuracy drop than the non-adaptive pruning strategy, which prunes a fixed percentage of filters. Singh et al. [33] proposed a framework called Play and Prune (PP), which jointly prunes and fine-tunes the networks with an adaptive weight threshold. The initial weight threshold is obtained using an optimization formula, and the weight threshold is adaptively determined by considering the accuracy of the remaining filters. Zhao et al. [40] presented an adaptive and activation-based pruning approach to generate models that automatically satisfy each of the three target tasks: accuracy-critical, memory-constructed, and latency-sensitive. This work uses the average of activation-based attention maps to determine the importance of filters and prunes the networks by adaptively adjusting the global threshold. However, these studies used a metric that reflects either the accuracy or FLOPs when adaptively obtaining pruning rates. To solve this problem, we propose a metric that considers both accuracy and FLOPs, which allows us to find unimportant filters in both aspects.

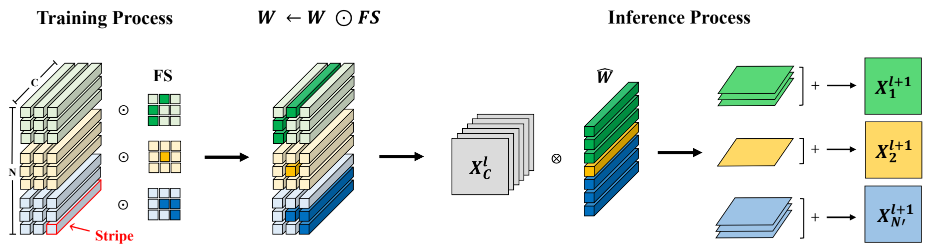


Fig. 3: Overall process of SWP. During the training process, FilterSkeleton (FS) learns the importance of stripes. After the training process, valid stripes are multiplied by filters and we obtain the optimal kernel shapes. Only valid stripes create an output feature map in the inference process.

3 Preliminaries

In this section, we review the key concepts of Stripe-Wise Pruning (SWP) [27] as shown in Fig. 3. SWP has been introduced to learn the optimal kernel shape with pruning and achieves $K^2\times$ finer granularity than filter pruning (assuming the kernel size is $K \times K$). Meng et al. [27] introduced a learnable matrix I, namely FilterSkeleton (FS), to learn the optimal kernel shape. In the convolution process, I is multiplied by filter W, which can be expressed as

$$X^{l+1}_{n,h,w} = \sum_{c}^{C}\sum_{i}^{K}\sum_{j}^{K} I^l_{n,i,j} \times W^l_{n,c,i,j} \times X^l_{c,h+i-1,w+j-1}, \tag{1}$$

where $X^l_{c,h,w}$ denotes a l-th convolutional layer feature map. N and C are the number of filters and channels of the input feature map in layer l, respectively; and n and c are the n-th filter and c-th input channel, respectively. K denotes kernel size, and i and j are indexes of the width and height of the kernel. I is first initialized with an all-one matrix, and the l_1 norm penalty of I is added to the original loss term.

For efficient pruning, stripes with $I^l_{n,i,j}$ less than threshold T are no longer updated in the training process, and these stripes are pruned after the training process. We define the pruned FS as $\hat{I}^l_{n',i,j}$, where n' denotes the n'-th filter in N', and N' is the number of filters with at least one stripe remaining in the kernel after pruning. Finally, $\hat{I}^l_{n',i,j}$ is merged to W (*i.e.*, $\hat{W} \leftarrow W \odot \hat{I}$), and only $\hat{W}$ is used during inference. In the inference process shown in Fig. 3, $\hat{W}$ has only valid stripes; therefore, the calculation order of Equ. (1) can be modified so that the channel-direction is calculated first. In other words, Equ. (1) can be reformulated as follows:

$$X^{l+1}_{n,h,w} = \sum_{i}^{K}\sum_{j}^{K}\Big(\sum_{c}^{C} \hat{W}^l_{n,c,i,j} \times X^l_{c,h+i-1,w+j-1}\Big) . \tag{2}$$

The channel-direction calculation of Equ. (2) can be implemented by 'im2col'; therefore, the model after SWP is available in the general processor.

4 Proposed Method

4.1 Filter Shape Pruning (FSP)

Filter Shape Pruning (FSP) is a method that prunes filters by each kernel shape obtained by Stripe-Wise Pruning (SWP). SWP is a pruning method to obtain the optimal kernel shape by learning the importance of stripes in the filter via a learnable matrix called the FilterSkeleton (FS) in the training process. After training, stripes with a FS value less than the global threshold are pruned, and then we can gain the optimal kernel shapes consisting of valid stripes in each filter. As shown in Fig. 2, filters with the same kernel shape exist at each layer. As the kernel shape of the filter is a significant factor affecting receptive fields, filters with the same kernel shape have similar receptive fields, which causes filter redundancy. FSP can efficiently reduce this filter redundancy to obtain a highly compressed model.

The pruning targets of FSP are filters with overlapping kernel shapes in each layer, and filters with the same kernel shape can be grouped together. Thus, N' filters in layer l can be expressed as follows:

$$W^l_{N'} = \left\{S^l_0,\ S^l_1,\ \cdots,\ S^l_a,\ \cdots,\ S^l_A\right\}, \tag{3}$$

where A is the number of types of kernel shape in layer l, and a is the a-th kernel shape type. S^l_a is a set of filters with the same kernel shape type. In other words, N' filters in layer l can be classified by kernel shape; therefore, there can be multiple filters with kernel shape of a-th type, and we define a set of those filters as S^l_a. If more than two filters exist within S^l_a, the corresponding S^l_a is the pruning target set. Here, the most important point is to retain at least one filter within each S^l_a to preserve the receptive field of the kernel shape. In other words, the maximum compression rate of FSP is defined when all S^l_a have only one filter. Therefore, it is not necessary to search for an architecture adaptively to gain the maximum compression rate. We observed that small N networks such as ResNet56/110 are less likely to overlap kernel shapes than others; therefore, we propose the weight inheritance technique which uses the weights after SWP. The weight inheritance technique can reduce the overhead of searching and fine-tuning for small N networks while obtaining the maximum compression rate; it describes how to combine all filters in S^l_a as one new filter, and only the new filter is used for fine-tuning. To create the new filter, the weights $W^l_{n'}$ of all filters in S^l_a are multiplied with their FilterSkeleton $I^l_{n'}$, and the multiplication values are added together. Also, a new FilterSkeleton is initialized with the matrix in which only the kernel shape position (i, j) is filled with '1'; otherwise filled with '0'. This process can be formulated as

$$W^l_m = \sum^{num\_f} W^l_{n'} \odot I^l_{n'}\ ,\quad I^l_{m,i,j} = \begin{cases} 1, & \text{if (i, j)} \subset \text{validstripes} \\ 0, & \text{otherwise} \end{cases}, \tag{4}$$

where num_f denotes the number of filters in S^l_a. We applied FSP to ResNet56/110 using the weight inheritance technique in our experiments.

The purpose of FSP is to remove redundant filters with the same kernel shape until reaching the target FLOPs while preserving the minimum diversity of kernel shapes. Therefore, FSP can be expressed as a problem of determining the number of filters to remove in each pruning target set S_a^l. To solve this problem, we propose the Adaptive Architecture Search (AAS) framework. In FSP, removing redundant filters with the same kernel shape is more important if there are multiple filters in S_a^l. However, if there is only one filter in S_a^l, FSP stops the removal of filters in S_a^l to preserve the receptive field of the kernel shape, and we define this "the FSP rule". We describe the AAS framework and "the FSP rule" in Section 4.2.

4.2 Adaptive Architecture Search (AAS)

We propose the AAS framework that determines the number of filters to be removed from each S_a^l to obtain the architecture that meets the target FLOPs. For selecting filters to remove, we introduce a metric that considers the influence of the filter on accuracy and FLOPs and use the layer-wise threshold T^l obtained by this metric. In other words, the framework removes filters that have smaller importance than T^l. Then, T^l adaptively changes according to the proposed metric updated at each iteration. In the framework, if only one filter remains in an S_a^l, the S_a^l is excluded from the pruning target sets to preserve the receptive field of the kernel shape.

Layer-wise Threshold We introduce a metric that determines how much to increase the layer-wise threshold, and this metric simultaneously considers the influence of the filter on accuracy and FLOPs. To calculate the metric, we define the filter importance as $F_{n'}^l$ and FLOPs importance as $M_{n'}^l$. $F_{n'}^l$ represents the filter's influence on the accuracy, and the larger the $F_{n'}^l$, the greater the impact of the filter on accuracy. $F_{n'}^l$ is defined follows:

$$F_{n'}^l = \frac{\sum_i^K \sum_j^K \hat{I}_{n',i,j}^l}{n_s}, \tag{5}$$

where n_s is the number of valid stripes in the filter, which is the number of nonzero elements in $\hat{I}_{n'}^l$. $M_{n'}^l$ represents the FLOPs generated by the filter and is defined as follows:

$$M_{n'}^l = H' \times W' \times C \times n_s, \tag{6}$$

where H' and W' denote the output feature map height and width, respectively.

To determine how much to increase the layer-wise threshold, we first need to determine the effect of each layer on accuracy and FLOPs. Therefore, we calculate the layer score L_s^l and layer FLOPs L_m^l using Equ. (5) and Equ. (6), and they are expressed as

$$L_s^l = \frac{\sum_{n'}^{N'} F_{n'}^l}{N'} \ , \quad L_m^l = \frac{\sum_{n'}^{N'} M_{n'}^l}{N'} \ . \tag{7}$$

Using Equ. (7), the relative layer's accuracy importance AL^l and FLOPs importance FL^l compared to all convolution layers are as follows:

$$AL^l = \frac{L_s^l}{\sum_l^L L_s^l} \ , \quad FL^l = \frac{L_m^l}{\sum_l^L L_m^l} \ . \tag{8}$$

Finally, the formulation of the metric with AL^l and FL^l is as follows:

$$\Delta T^l = \frac{\alpha \cdot (1 - AL^l) + \beta \cdot FL^l}{metric\_norm}, \tag{9}$$

where α and β are the weight parameters of the metric. We will discuss the variation of parameters and FLOPs according to the values of α and β in Section 6.2. $metric\_norm$ regulates the variation of the threshold. Since the threshold is too large if the model's FLOPs are smaller than the target FLOPs, $metric\_norm$ is multiplied by ϵ ($\epsilon > 1$) to make ΔT^l small. We used $\epsilon = 2$ in our experiments. The larger the AL^l, the greater the effect of the filter on accuracy; thus, we have to prune the smaller AL^l filters. Therefore, we use $(1 - AL^l)$ for the metric.

Using the metric (Equ. (9)), the layer-wise threshold T^l is updated as follows:

$$T^{l'} = T^l \cdot (1 + \Delta T^l) \ . \tag{10}$$

The FSP Rule The key concept of FSP is to retain at least one filter in S_a^l to preserve the receptive fields of kernel shapes. To maintain the key concept of FSP, the AAS framework follows the FSP rule, which deals with maintaining receptive fields throughout the framework. After pruning, if there is only one filter in S_a^l, S_a^l is eliminated from the pruning target sets in the next iteration. In the case that all filters in S_a^l are less than T^l, only the filter with the most significant $F_{n'}^l$ remains, and S_a^l is eliminated from the pruning target in the next iteration.

The Procedure of Adaptive Architecture Search (AAS) The AAS procedure is illustrated in Fig. 4. First, the framework classifies the filters of the model after SWP (Equ. (3)), and sets S_a^l, which has more than two filters, as the pruning target set. Second, our framework saves or rewinds the state. The framework evaluates the FLOPs of the pruned model in the end, and if the FLOPs of the pruned model are smaller than the target FLOPs in iteration i, $metric\_norm$ increases to make smaller ΔT^L of iteration $i+1$. To obtain a model close to the target FLOPs by applying the adjusted ΔT^L, the model before pruning at iteration i is required, so the framework has to save the pruned model of iteration $i-1$. Therefore, the framework checks whether $metric\_norm$ is updated and if not updated, the framework saves the latest model. If $metric\_norm$ is updated, the framework rewinds the current state to the saved state. Next, the framework calculates the metric (Equ. (9)) and update the layer-wise threshold T^L (Equ. (10)). Then, the framework compares the filter importance $F_{n'}^l$ with T^l for each layer and prunes the model according to "the FSP rule". Finally, the framework evaluates the FLOPs of the pruned model, which are described

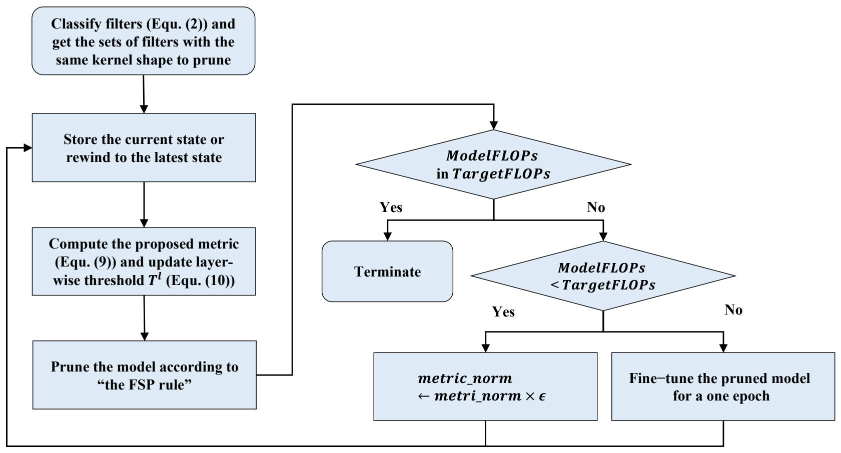

Classify filters (Equ. (2)) and get the sets of filters with the same kernel shape to prune
Store the current state or rewind to the latest state
Compute the proposed metric (Equ. (9)) and update layer-wise threshold T^l (Equ. (10))
Prune the model according to "the FSP rule"
ModelFLOPs in TargetFLOPs
Yes
No
Terminate
ModelFLOPs < TargetFLOPs
Yes
No
metric_norm ← metri_norm × ϵ
Fine−tune the pruned model for a one epoch

Fig. 4: AAS framework procedure. $ModelFLOPs$ represents the FLOPs of the pruned model and $TargetFLOPs$ represents the target FLOPs boundary. When the pruned model meets $TargetFLOPs$, it is fine-tuned from scratch.

as $ModelFLOPs$. If $ModelFLOPs$ is within the target FLOPs, the framework is terminated. If it is smaller than the target FLOPs, ΔT^L is adjusted by increasing $metric\_norm$. The adjusted ΔT^L makes $ModelFLOPs$ at the next iteration close to the target FLOPs. If $ModelFLOPs$ does not reach the target FLOPs, the framework fine-tunes the pruned model for one epoch. This process adjusts the FS values of the pruned model in iteration i so that the FS values suitable for the current architecture can be reflected in the next iteration. The proposed metric reflects the changed architecture for each iteration; therefore, the framework can find the optimal architecture that satisfies the target FLOPs. After obtaining the pruned model that satisfies the target FLOPs, the pruned model is fine-tuned from scratch.

5 Experiments

5.1 Experimental Settings

We conducted experiments on two popular datasets (Cifar-10 [13] and ImageNet [3]) and two different architectures (VGG-16 [32] and ResNet[7]). We applied VGG-16 and ResNet-56/110 to Cifar-10. Our method was based on a model after Stripe-Wwase Pruning (SWP); thus, we obtain the baseline of SWP [1] mainly using hyper-parameters similar to those used in [27]. The SWP baseline of VGG-16 was trained using the same hyper-parameter as [27], except the training epoch. We used 115 epochs for training, and the initial learning rate was 0.1 and was

[1] The baseline obtained by SWP influences the performance of our method. The better the SWP baseline, the better the performance compared with our results. We report the SWP re-implementation results used as the baseline.

divided by 10 at epochs 45 and 75. We used $\alpha = 0.5$ and $\beta = 0.5$. The SWP baseline of ResNet-56 was trained for 160 epochs using the same hyper-parameter as [27]. However, there no results were obtained for ResNet-110 in [27]; therefore, we searched hyper-parameters for the networks. We used a batch size of 128, and the other parameters were the same as the baseline of ResNet56. We used the AAS framework to the SWP baseline of VGG-16 and fine-tuned from scratch for 115 epochs with the cosine annealing [25] scheduler. For ResNets, we applied the weight inheritance technique and fine-tuned for 50 epochs with the AdamW [26] optimizer and cosine annealing scheduler. The learning rate was 0.001, and the weight decay was 0.05. The other VGG-16 and ResNets hyper-parameters were the same as those used in the SWP baseline training.

We applied ResNet-18 to ImageNet, and the SWP baseline of ResNet-18 was trained using the same hyper-parameters as [27]. We use the AAS framework to the SWP baseline of ResNet18 and fine-tune from scratch for 90 epochs using the AdamW optimizer and cosine annealing scheduler. The learning rate was 0.001, and the weight decay was 0.01. We used $\alpha = 1$ and $\beta = 2$. The other ResNet-18 parameters were the same as those used for the SWP baseline training.

5.2 Results on Cifar-10

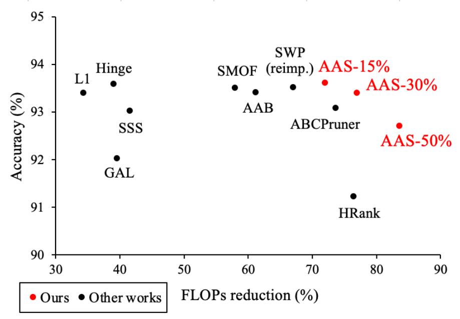
VGG-16 Table 1 shows the performance of our method and other pruning methods. AAS-15% indicates that the target FLOPs were 15% less than that of the SWP baseline. Furthermore, our SWP baseline is the re-implementation result of SWP in Table 1. Compared to L1, Hinge, GAL, SSS, SMOF, and AAB, AAS-15% achieved a significantly large reduction in both FLOPs and parameters with higher accuracy. AAS-30% was advantageous in all aspects compared to ABCPruner and HRank. AAS-50% significantly reduced the FLOPs (83.6%) and parameters (96.73%) and was the only one that reduced the FLOPs by more than 80%. Fig. 5 shows the comparison of our method with other pruning methods, and the higher the dot in the upper right, the better the performance of the method. In Fig. 5a, the advantages of the FSP were clearer. Our results showed better performance considering both the FLOPs reduction and accuracy. These results indicate that FSP reduces filter redundancy while maintaining the receptive field of the kernel so that it can further compress the networks with a small accuracy loss.

ResNet-56/110 Table 1 shows the experimental results of ResNet-56/110 on Cifar-10. Max indicates that we apply the maximum compression rate of FSP to the networks using the weight inheritance technique. Our SWP baselines are re-implementation results of SWP in Table 1. The Max of ResNet-56 reduced FLOPs by 80.51% and deleted parameters by 78.22%. In addition, our results had the best performance in all aspects among the studies in Table 1. Compared to our SWP baseline, our results achieved higher accuracy and improved reduction of parameters and FLOPs. Fig. 5b shows a graph comparing Max to other studies in terms of FLOPs reduction and accuracy. Max is at the furthest upper right than the other works, which confirms that our method performed better in terms of accuracy and FLOPs.

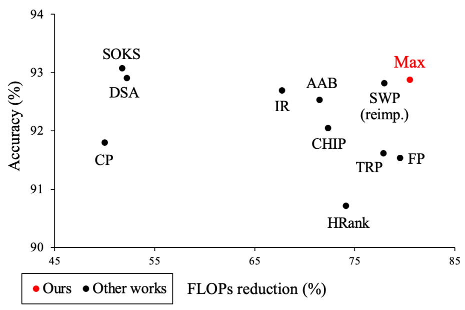
Table 1: Comparing the FSP method with state-of-the-art pruning methods for VGG-16 and ResNet-56/110 on Cifar-10. We have sorted in the order of small reduction in FLOPs, and our methods show the most significant reduction in FLOPs and parameters.

Model	Method	Accuracy(%)	FLOPs ↓ (%)	Param ↓ (%)
VGG-16	Baseline	93.25	0	0
	L1 [15]	93.40	34.2	64
	Hinge [16]	93.59	39.07	80.05
	GAL [19]	92.03	39.6	77.6
	SSS [11]	93.02	41.6	73.8
	SMOF [22]	93.50	58	80.9
	AAB [40]	93.41	61.17	72.85
	SWP (reimp.)	93.52	67.02	93.1
	SWP [27]	93.65	71.16	92.66
	Ours (AAS-15%)	**93.61**	**72.01**	**94.43**
	ABCPruner [18]	93.08	73.68	88.68
	HRank [17]	91.23	76.5	92
	Ours (AAS-30%)	**93.40**	**76.99**	**95.47**
	Ours (AAS-50%)	**92.71**	**83.6**	**96.73**
ResNet-56	Baseline	93.10	0	0
	CP [9]	91.8	50	-
	DSA [28]	92.91	52.2	-
	SOKS [20]	93.08	51.73	54.12
	IR [36]	92.70	67.7	-
	AAB [40]	92.54	71.44	-
	CHIP [34]	92.05	72.3	71.8
	HRank [17]	90.72	74.1	68.1
	SWP [27]	92.98	77.7	75.6
	SWP (reimp.)	92.82	77.95	73.82
	TRP [38]	91.62	77.83	-
	FP [14]	91.54	79.50	70.59
	Ours (Max)	**92.88**	**80.51**	**78.22**
ResNet-110	Baseline	93.50	0	0
	L1 [15]	93.30	38.6	32.4
	GAL [19]	92.55	48.5	44.8
	FP [14]	93.73	48.52	44.77
	ABCPruner [18]	93.58	65.04	67.41
	HRank [17]	92.65	68.6	68.7
	SWP(reimp.)	93.63	70	71.43
	CHIP [34]	93.63	71.6	68.3
	Ours (Max)	**93.42**	**74.8**	**77.11**

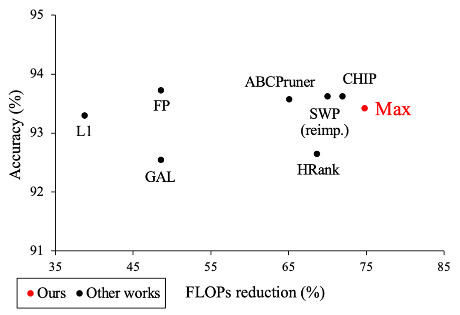
The results of ResNet-110 is also reported in Table 1. Compared with L1, GAL, and HRank, our results had better performance on all fronts. Particularly, our method removed 2.38× parameters and 1.94× FLOPs than L1. In compari-



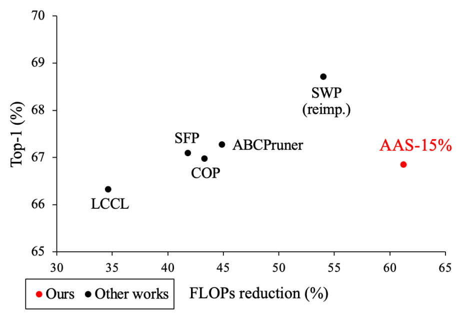
(a) VGG-16



(b) ResNet-56



(c) ResNet-110



(d) ResNet-18

Fig. 5: FLOPs reduction and accuracy comparison with other pruning methods on Cifar-10 and ImageNet. (a)-(c) were the results of Cifar-10, and (d) was the result of ImageNet.

son with FP and CHIP, Max reduced FLOPs and parameters with little drop in accuracy. Fig. 5c demonstrates that our method achieved the highest reduction in FLOPs with only a slight drop in accuracy. These results demonstrate that FSP can obtain a high compression rate with little accuracy loss by maintaining the receptive fields of kernel shapes.

5.3 Results on ImageNet

The results for ResNet-18 on ImageNet are shown in Table 2. Our SWP baseline is the re-implementation result of SWP in Table 2. AAS-15% reduced the FLOPs by 15% and deleted parameters by 36.74% based on the SWP baseline. Compared with LCCL, our results reduced more FLOPs with higher accuracy. Furthermore, our results achieved a 61.2% FLOPs and 57.74% parameters reduction with small accuracy loss compared to SFP, COP, and ABCPruner. Fig. 5d shows that AAS-15% had the highest compression rate among the other pruning methods with similar accuracy.

Table 2: Comparing the AAS framework with state-of-the-art pruning methods on ImageNet. We applied our framework to ResNet-18. We sorted in the order of smallest reduction in FLOPs, and our method obtained the highest compression ratio in FLOPs and parameters.

Model	Method	Top-1(%)	Top-5(%)	FLOPs ↓ (%)	Param ↓ (%)
ResNet-18	Baseline	69.76	89.08	0	0
	LCCL [4]	66.33	86.94	34.6	-
	SFP [8]	67.1	87.78	41.8	-
	COP [37]	66.98	-	43.3	45.1
	ABCPruner [18]	67.28	87.67	44.88	43.55
	SWP (reimp.)	68.72	88.63	53.98	41.83
	SWP [27]	69.59	89.04	54.58	-
	Ours (AAS-15%)	**66.86**	**87.05**	**61.2**	**57.74**

Table 3: Effectiveness of pruning filters while preserving the receptive field of the kernel shape. We applied VGG-16 on Cifar-10 and ResNet-18 on ImageNet.

Model	Method	Param (M)	FLOPs (M)	Accuracy (%)
VGG-16	**FSP**	**0.68**	**144.33**	**93.40**
	Simple baseline	0.78	156.14	92.79
ResNet-18	**FSP**	**4.94**	**1415.95**	**66.86**
	Simple baseline	6.01	1424.57	65.64

6 Ablation Study

6.1 Effect of Preserving the Receptive Field

We experimentally proved the effect of preserving the receptive field of the kernel shape, which is the core concept of FSP. We compared FSP with a 'Simple baseline,' which removes filters with the same kernel shape after SWP in the order of the smallest filter importance (Equ. (5)). We experimented VGG-16 on Cifar-10 and ResNet-18 on ImageNet, and used the same SWP baseline for both FSP and the simple baseline. We fine-tuned both FSP and the simple baseline to the same conditions as Section 5.1. The result is shown in Table 3.

In Table 3, although FSP has fewer parameters and FLOPs than the simple baseline, it showed higher accuracy in both VGG-16 and ResNet-18. In the simple baseline case, we observed that some kernel shape sets had no filters. That is, the receptive fields corresponding to those sets were erased, which resulted in an additional accuracy drop. The results demonstrate that the core concept of FSP, which is to preserve the receptive field of the kernel shape, plays a key role in reducing the loss in accuracy.

Table 4: Variance of parameters, FLOPs, and accuracy according to α and β. We set $\alpha = 1 - \beta$.

α	β	Param $\downarrow$ (%)	FLOPs $\downarrow$ (%)	FLOPs/Param	Accuracy (%)
0.1	0.9	20.19(0.83M)	30.18(144.46M)	1.49	93.07
0.3	0.7	27.88(0.75M)	30.35(144.10M)	1.09	93.21
0.5	0.5	34.62(0.68M)	30.24(144.33M)	0.87	93.40
0.7	0.3	38.46(0.64M)	30.22(144.37M)	0.79	93.18
0.9	0.1	40.38(0.62M)	30.03(144.76M)	0.74	92.96

6.2 Weight Parameters of the Metric : α and β

α and β are weight parameters used for the proposed metric and determine whether to assign weight to accuracy importance or FLOPs importance. We investigated the difference between the parameters and FLOPs according to α and β for the general case. We experimented with VGG-16 on Cifar-10 under the same conditions except for α and β and used AAS-30% as a pruning method. We reported parameters and FLOPs reduction compared to the SWP baseline. Table 4 shows the results. The larger the β compared to the α, the larger the FLOPs reduction compared to the parameters reduction. In other words, the larger the β, the more the framework preferentially prunes filters that generate many FLOPs. Considering the accuracy, the overall performance was good when $\alpha = 0.5$ and β=0.5.

7 Conclusion

In this study, we propose the Filter Shape Pruning (FSP) method, which prunes networks using the kernel shape of the filter while preserving the receptive field of the kernel shape. In addition, we proposed the Adaptive Architecture Search (AAS) framework to search for the architecture that satisfies the target FLOPs with the FSP method. The AAS framework adaptively searches the architecture that meets the target FLOPs with the layer-wise threshold. The layer-wise threshold is updated at each iteration by the proposed metric that considers the effect of the filter on both accuracy and FLOPs. The experimental results demonstrated that the FSP method could obtain a higher compression rate than other pruning methods with an acceptable accuracy loss by preserving the receptive fields of kernel shapes.

Acknowledgements This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-01172, DRAM PIM Design Base Technology Development) and by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No.2019R1A5A1027055).

References

1. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
2. Chen, W., Wilson, J., Tyree, S., Weinberger, K., Chen, Y.: Compressing neural networks with the hashing trick. In: International conference on machine learning. pp. 2285–2294. PMLR (2015)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
4. Dong, X., Huang, J., Yang, Y., Yan, S.: More is less: A more complicated network with less inference complexity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5840–5848 (2017)
5. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. In: IJCAI International Joint Conference on Artificial Intelligence (2018)
9. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision. pp. 1389–1397 (2017)
10. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
11. Huang, Z., Wang, N.: Data-driven sparse structure selection for deep neural networks. In: Proceedings of the European conference on computer vision (ECCV). pp. 304–320 (2018)
12. Huo, Z., Wang, C., Chen, W., Li, Y., Wang, J., Wu, J.: Balanced stripe-wise pruning in the filter. In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4408–4412. IEEE (2022)
13. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
14. Li, D., Chen, S., Liu, X., Sun, Y., Zhang, L.: Towards optimal filter pruning with balanced performance and pruning speed. In: Proceedings of the Asian Conference on Computer Vision (2020)
15. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)
16. Li, Y., Gu, S., Mayer, C., Gool, L.V., Timofte, R.: Group sparsity: The hinge between filter pruning and decomposition for network compression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 8018–8027 (2020)

17. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 1529–1538 (2020)
18. Lin, M., Ji, R., Zhang, Y., Zhang, B., Wu, Y., Tian, Y.: Channel pruning via automatic structure search. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. pp. 673–679 (2021)
19. Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., Doermann, D.: Towards optimal structured cnn pruning via generative adversarial learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2790–2799 (2019)
20. Liu, G., Zhang, K., Lv, M.: Soks: Automatic searching of the optimal kernel shapes for stripe-wise network pruning. IEEE Transactions on Neural Networks and Learning Systems (2022)
21. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
22. Liu, Y., Guan, B., Xu, Q., Li, W., Quan, S.: Smof: Squeezing more out of filters yields hardware-friendly cnn pruning. arXiv preprint arXiv:2110.10842 (2021)
23. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision. pp. 2736–2744 (2017)
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)
25. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
26. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
27. Meng, F., Cheng, H., Li, K., Luo, H., Guo, X., Lu, G., Sun, X.: Pruning filter in filter. Advances in Neural Information Processing Systems **33**, 17629–17640 (2020)
28. Ning, X., Zhao, T., Li, W., Lei, P., Wang, Y., Yang, H.: Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In: European Conference on Computer Vision. pp. 592–607. Springer (2020)
29. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
30. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
32. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
33. Singh, P., Verma, V.K., Rai, P., Namboodiri, V.P.: Play and prune: Adaptive filter pruning for deep model compression. arXiv preprint arXiv:1905.04446 (2019)
34. Sui, Y., Yin, M., Xie, Y., Phan, H., Aliari Zonouz, S., Yuan, B.: Advances in Neural Information Processing Systems **34**, 24604–24616 (2021)
35. Sui, Y., Yin, M., Xie, Y., Phan, H., Aliari Zonouz, S., Yuan, B.: Chip: Channel independence-based pruning for compact neural networks. Advances in Neural Information Processing Systems **34** (2021)

36. Wang, H., Zhang, Q., Wang, Y., Yu, L., Hu, H.: Structured pruning for efficient convnets via incremental regularization. In: 2019 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2019)
37. Wang, W., Fu, C., Guo, J., Cai, D., He, X.: Cop: customized deep model compression via regularized correlation-based filter-level pruning. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. pp. 3785–3791 (2019)
38. Xu, Y., Li, Y., Zhang, S., Wen, W., Wang, B., Qi, Y., Chen, Y., Lin, W., Xiong, H.: Trp: trained rank pruning for efficient deep neural networks. In: Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. pp. 977–983 (2021)
39. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6848–6856 (2018)
40. Zhao, K., Jain, A., Zhao, M.: Adaptive activation-based structured pruning. arXiv preprint arXiv:2201.10520 (2022)