IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 24 July 2024, accepted 15 August 2024, date of publication 4 September 2024, date of current version 12 September 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3454329

RESEARCH ARTICLE

Adaptive Scaling Filter Pruning Method for Vision Networks With Embedded Devices

HYUNJUN KO[1], (Graduate Student Member, IEEE), JIN-KU KANG[1], (Senior Member, IEEE), AND YONGWOO KIM[2], (Member, IEEE)

[1]Department of Electrical and Computer Engineering, Inha University, Incheon 22212, Republic of Korea
[2]Department of Technology Education, Korea National University of Education, Cheongju 28173, Republic of Korea

Corresponding author: Yongwoo Kim (yongwoo.kim@knue.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by Korean Government through the Ministry of Science and ICT (MSIT), South Korea, under Grant 2022R1G1A1007415; and in part by the MSIT through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and Communications Technology Planning and Evaluation (IITP) under Grant IITP-2021-0-02052.

ABSTRACT Owing to improvements in computing power, deep learning technology using convolutional neural networks (CNNs) has recently been used in various fields. However, using CNNs on edge devices is challenging because of the large computation required to achieve high performance. To solve this problem, pruning, which reduces redundant parameters and computations, has been widely studied. However, a conventional pruning method requires two learning processes, which are time-consuming and resource-intensive, and it is difficult to reflect the redundancy in the pruned network because it only performs pruning once on the unpruned network. Therefore, in this paper, we utilize a single learning process and propose an adaptive scaling method that dynamically adjusts the size of the network to reflect the changing redundancy in the pruned network. To verify the performance of each method, we compare the performance of the proposed methods by conducting experiments on various datasets and networks. In our experiments using the ImageNet dataset on ResNet-50, pruning FLOPs by 50.1% and 74.0% resulted in a decrease in top-1 accuracy by 0.92% and 3.38%, respectively, and improved inference time by 26.4% and 58.9%, respectively. In addition, pruning FLOPs by 27.37%, 36.84% and 46.41% using the COCO dataset on YOLOv7, reduced mAP(0.5-0.95) by 1.2%, 2.2% and 2.9%, respectively, and improved inference time by 12.9%, 16.9% and19.3%.

INDEX TERMS Computer vision, convolutional neural network, inference time, network compression, pruning.

I. INTRODUCTION

Recently, deep learning technologies have been used actively in computer vision, speech recognition, natural language processing, etc. [1], [2], because of the increasing availability of training data and computing resources. In the field of computer vision, classification networks [3], [4], [5], [6], [7], detection networks [8], [9], [10] and segmentation networks [11], [12], [13] using convolutional neural networks (CNNs) are being actively researched. However, to achieve high performance, networks are becoming deeper and wider, and using many CNNs will require lots of computer resources. In addition, the increasing need to run computer vision networks on edge devices with limited computational resources is driving the need for model compression research. Therefore, lots of model compression research is being conducted for real-time operation.

Pruning and quantization are common examples in model compression research. Hybrid pruning [14] fuses two or more network compression methods such as low-rank decomposition, quantization, and knowledge distillation for developing the performance of compression models. Quantization converts parameters stored as 32-bit floating-point to 8 or less-bit integers of 16 or less floating-point. As a consequence, quantization reduces memory and computation by reducing data size and simplifies hardware using integer operations.

The associate editor coordinating the review of this manuscript and approving it for publication was Siddhartha Bhattacharyya.

Pruning removes redundant weights or filters from CNNs with minimal performance degradation. In other words, pruning reduces memory and computation by reducing the amount of data. According to the target, pruning is separated into weight and filter pruning. First, weight pruning methods determine the importance of each weight and prune them individually. This method could achieve a high compression rate but creates a sparse matrix. Consequently, we cannot expect the inference time acceleration in a typical GPU environment without special software or hardware. On the other hand, filter pruning methods prune each filter as a group by comparing their importance. This method has a relatively lower compression ratio than weight pruning methods. However, it only reduces the network width while maintaining the network structure so that it can achieve inference time acceleration in a typical GPU environment. This advantage is the reason why filter pruning [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34] is actively researched.

There are two research approaches in filter pruning methods: predefined and automatic. Predefined methods [25], [26], [27], [28] allow the user to set the sparsity of each layer to obtain a compressed network with the desired structure. However, it has the disadvantage that the sparsity ratio of the optimal network is highly dependent on the user's experience because it is a hyper-parameter. On the other hand, automatic methods [15], [16], [17], [18], [19] compare the importance of filters in the entire network using several criteria to obtain the target pruned network by setting the pruning ratio automatically. The typical filter pruning methods use sparsity learning to determine each filter's importance. The pruning stage removes unimportant filters. Finally, Execute the fine-tuning stage to recover performance degradation. However, this method requires two training processes, sparsity learning and fine-tuning, which require a long training time and lots of power. Therefore, several pieces of research [16], [17], [31] are being conducted to simplify these processes.

In this paper, we proposed an adaptive scaling method, which dynamically adjusts the size of the network by randomly regrowing some of the pruned filters to consider the redundancy change of each filter. Finally, we propose a single learning process to obtain a pruned network with maximally preserved performance in a single training. The contributions of this paper are as follows.

- To address the time-consuming disadvantage due to two learning processes in existing studies [15], [19], [20], [21], [22], [23], [24], [26], [29], [30], we propose a single learning process that can quickly obtain a lightweight model in one learning process.
- Inspired by the Channel exploration (CHEX) method [16], we randomly recover some filters during the pruning process to make the network have a flexible size and use an adaptive scaling method that can consider the changing dependency of the filter to obtain a target pruned network with maximum performance preservation with a short training time.
- Experiments using ResNet-56 [3] and VGG-16 [4] on CIFAR-10 [35] with the proposed method have better performance than other methods by up to 0.38% at similar pruning ratios. Experiments using ResNet-50 [3] on ImageNet [36] have better performance than others by up to 2.27%. In addition, we demonstrated high performance and fast inference time compared to other methods using YOLOv7 [8] on COCO [37] and VOC [38], proving the compatibility of our proposed method in various environments.

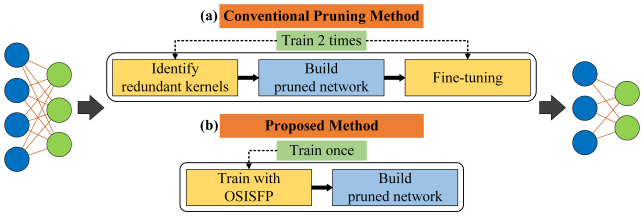

FIGURE 1. Compare the workflow between the conventional pruning method (a) and the proposed method (b).

The structure of the rest of the paper is organized as follows. We review existing filter pruning methods In Section II. Section III describes the details of our adaptive scaling filter pruning method. In Section IV and Section V, we present experimental results on different networks and datasets. In Section VI, we conclude the paper.

II. RELATED WORKS

Unlike weight pruning, which prunes by measuring the importance of individual weights, filter pruning removes groups of weights in a single filter to accelerate inference time. As mentioned in Section I, filter pruning has two research approaches: predefined and automatic. The predefined pruning method defines the sparsity of each layer individually by the user to prune. For example, L1-norm pruning [27] assumes that filters with higher L1-norms in each convolutional layer are more important. This method prunes filters in a small order to achieve the target pruning ratio and compares the importance only within each layer. Hrank [25] extracts feature maps from the network and determines that the larger the rank of each feature map, the more important it is. So Hrank prunes filters in a small order of rank to achieve a target pruning ratio. EagleEye [18] employs adaptive batch normalization to mitigate the impact of performance loss from pruning and finds the optimal pruned network by predicting the final accuracy across different randomly pruned networks. CHIP [28] uses channel independence to indicate the orthogonality of each channel; therefore, CHIP prunes channels in a small order of channel independence. These methods can obtain a network with the desired structure easily. Still, discovering the optimal pruned network is challenging because users must be involved in selecting the pruning ratios for individual layers.

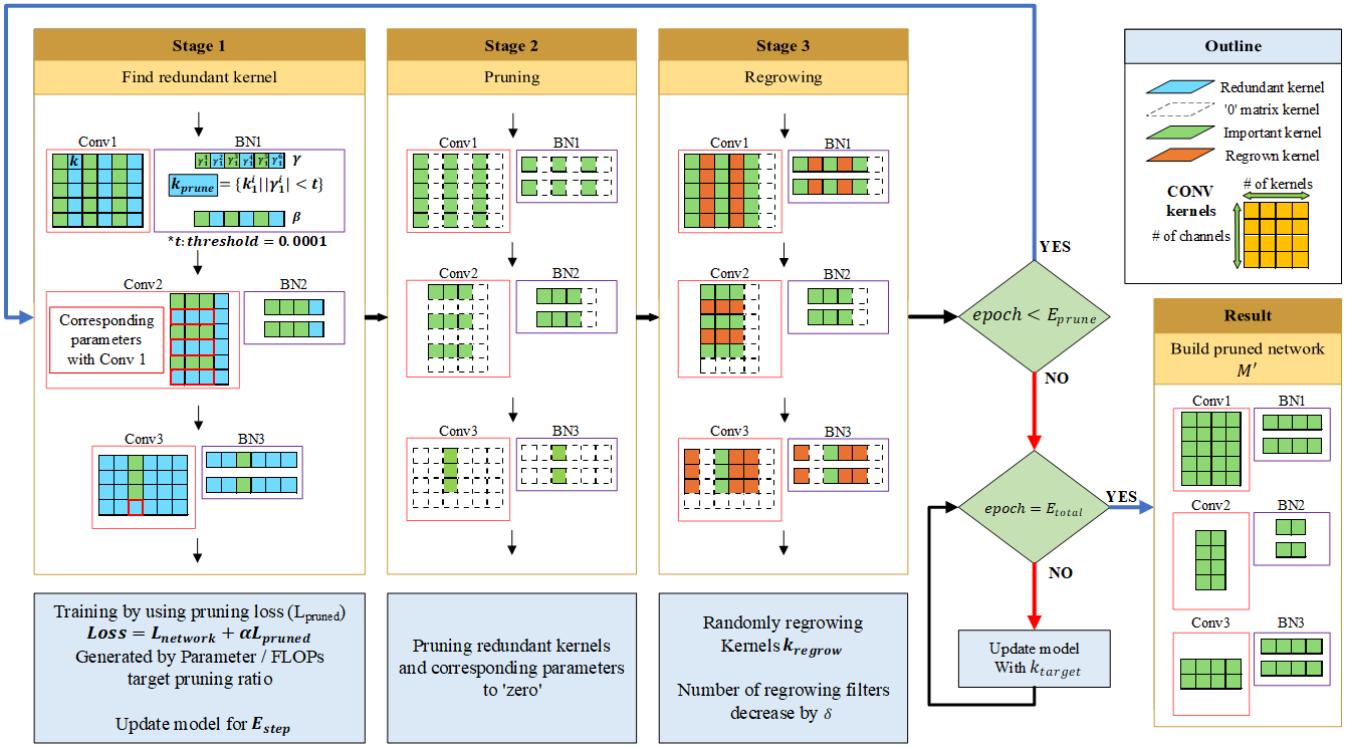

FIGURE 2. The overview of proposed adaptive scaling filter pruning (ASFP) method.

On the other hand, automatic methods use floating point operations (FLOPs) targets, parameters targets, or channel sparsity to specify the pruning ratio for each layer by comparing the importance of each filter. For example, network slimming [19] determines the importance of each feature map channel by the γ parameter of the batch-normalization [39] layer and prunes related filters in a smaller order. At this time, sparsity learning ensures that the γ associated with unimportant filters has a small value. CHEX [16] uses column subset search to determine the importance of each filter and set the unimportant parameters to zero. CHEX also uses regrowing, which recovers important filters by measuring the relationship between non-pruned and pruned filters as importance sampling to re-evaluate the importance of pruned filters. This method yields the same result as the last training result with the target pruned network, as the output feature map is zero, which means the result is unaffected. NISP [20] uses the final layer's ranking score to calculate the previous layers' neuron importance scores to remove unimportant filters. The disadvantages of these automatic methods are that they may choose unimportant filters without considering the behavior of the entire network, and the size of the pruned network cannot be predicted based on the criteria used as the target pruning ratio.

Pruning usually uses train-based pruning, which uses sparsity learning, pruning, and fine-tuning. The conventional processes of train-based pruning are described in Fig. 1 (a). First, sparsity learning uses various criteria to identify redundant structures. Pruning prunes redundant structures based on the results of sparsity learning. Finally, fine-tuning recovers the performance degradation caused by pruning. The conventional pruning method is time-consuming because it requires training twice.

Based on the works mentioned above, this paper proposes an adaptive scaling filter pruning (ASFP) that improves the time efficiency of pruning by eliminating the fine-tuning process and makes it operatable on edge devices by using both parameters and FLOPs target pruning ratio. Also, We conduct experiments on both classification and detection networks to verify our method on several computer vision networks.

III. PROPOSED METHOD

In this section, we describe ASFP in detail. First, we explain equations. Next, we describe the detailed process of our proposed method in Section III-B. An overall process of ASFP is described in Fig. 2.

A. NOTATIONS

Suppose a CNN has N number of convolutional layers and C_i means *i-th* convolutional layer. The filter, which is the parameter located in C_i, is $K_i \subseteq \mathbb{R}^{KC_i \times KC_{i-1} \times KH_i \times KW_i}$. KC_i means the number of filters and is equal to the channel of the output feature map. KC_{i-1} is the number of filters in the (i-1)-*th* convolutional layer and also means the number of channels in the input feature map. KH_i and KW_i means height and width of the filter C_i, respectively. The parameters

Algorithm 1 Overview of the ASFP Method

Input: Pre-trained network M
Total training epochs E_{total}
Training epochs between pruning steps E_{step}
Max training epochs using pruning step E_{prune}
Output: Pruned network M'

for each epoch E_{total} **do**
  if $e < E_{prune}$ **then**
    if $Mod(e, E_{step}) = 0$ **then**
      Find redundant filters k^i_{prune}=$\{k^i || \gamma^i| < t\}$
      Make 'Zero' k^i_{prune} and its corresponding parameters
      Compute regrowing ratio δ using (5)
      Compute the number of regrowing filters n^i_{regrow}=(# *of* k^i_{prune}) $\times$ δ
      Randomly find regrowing filters k^i_{regrow}=$\{k^i_{prune} \| choose\ n^i_{refrow}\ filters\}$
      Regrow k^i_{regrow} and its corresponding parameters by using recently used parameters
      Get set of pruning target filters k^i_{target}=k^i_{prune}-k^i_{regrow}
    end if
    Generate Loss using (4)
  end if
  Make 'Zero' k^i_{target} and its corresponding parameters
  Update network
end for
Build pruned network M'
Return M'

of the batch normalization layer are $\gamma_n \subseteq \mathbb{R}^{C_n}$ and $\beta_n \subseteq \mathbb{R}^{C_n}$. The convolutional and batch normalization layers are iterated over, and the output of the *i-th* layer is $FM_i \subseteq \mathbb{R}^{B\times C_i\times H_i\times W_i}$. B is the batch size and H_i and W_i is the height and width of FM_i, respectively. Fig. 3 shows the feature map of CNN and the parameters of the convolutional and batch normalization layers. In the convolutional layer parameters, each row represents a channel in the filter and each column represents a channel in the output feature map. In Algorithm 1, e denotes the current epoch, E_{step} denotes the epoch interval between the pruning stage. E_{prune} denotes the maximum epoch to use the pruning stage. E_{total} denotes the number of total epochs until a pruned network is obtained. These values are determined experimentally.

B. ADAPTIVE SCALING FILTER PRUNING

Fig. 2 describes the overall structure of the adaptive scaling filter pruning (ASFP) method. There are mainly two differences compared to the conventional method. First, as shown in (a) of Fig. 1, the conventional method requires re-training or fine-tuning to recover the lost performance after obtaining the pruned network. Otherwise, ASFP does not require additional training after obtaining the pruned network, as shown in (b) of Fig. 1. Second, since we iterate pruning and regrowing every E_{step} until E_{prune} in Fig. 2, we can consider the change of each filter redundancy due to pruning and use the regrowing ratio δ to achieve the target pruning ratio gradually. After E_{prune}, we train the network and fix pruning target filters k^i_{target} until E_{total}. The following shows the details of each step, and the complete method is described in Algorithm 1.

1) PARAMETERS AND FLOPS SPARSITY

We use both parameters sparsity and FLOPs sparsity to set the target pruned network. So, we generate pruning loss in (3) introduced in target capacity filter pruning [15]. We use γ, a parameter of the batch normalization layer [39], as a criterion for distinguishing important filters. The batch normalization layer is applied between the convolutional layer and the activation function for scaling through γ and shifting through β to prevent internal covariance shifts. The feature map is associated with the γ, which eventually leads to an unimportant feature map if γ is small. Also, since each γ corresponds to one channel in the feature map, and each channel is generated by the filter operation of the KC_n in the previous *n-th* convolutional layer, we can determine that the filter KC_n is unimportant.

$$\theta(\gamma, t) = \begin{cases} 0, & if\ |\gamma| \leqq t \\ 1, & if\ |\gamma| > t \end{cases} \tag{1}$$

$$\frac{\partial\theta(\gamma, t)}{\partial\gamma} = \begin{cases} -1, & if\ \gamma \leqq 0 \\ 1, & if\ \gamma > 0 \end{cases} \tag{2}$$

The L_{prune} is obtained by applying the computational amount and the number of parameters of the original network($F_{original}$, $P_{original}$), the computational amount and the number of parameters of the current network(F_{prune}, P_{prune}) being trained by applying loss, and the computational amount and the number of parameters of the target pruned network(F_{target}, P_{target}) as shown in (3). In particular, to obtain the computational amount and the number of parameters of the current network, γ, the parameter of the batch normalization layer, is used as an indicator function as shown in (1). Since the indicator function is discontinuous, backpropagation is impossible, so it is backpropagated using STE as shown in (2).

$$L_{prune} = \left(\frac{F_{prune} - F_{target}}{F_{original}}\right)^2 + \left(\frac{P_{prune} - P_{target}}{P_{original}}\right)^2 \tag{3}$$

$$L_{total} = L_{original} + \alpha L_{prune} \tag{4}$$

By using the pruning loss function in (3) to generate the loss function in (4) and proceed with training, we can induce γ to be less than threshold t through training to have the target pruning network computation and the number of parameters. At this time, if the pruning loss function is too large, it will cause the performance of the final network to be adversely affected, so use α to adjust the ratio of network loss($L_{original}$) and pruning loss(L_{prune}) to maintain the performance of the target network as much as possible.

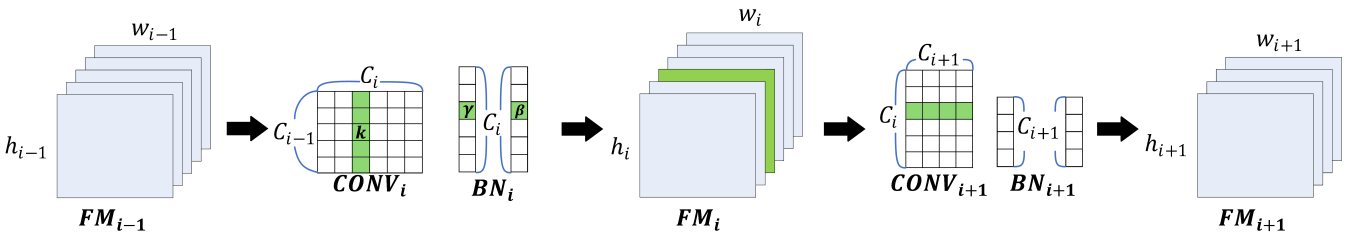

FIGURE 3. Structure of parameters in convolutional neuron network and indicate an example of corresponding parameters(green).

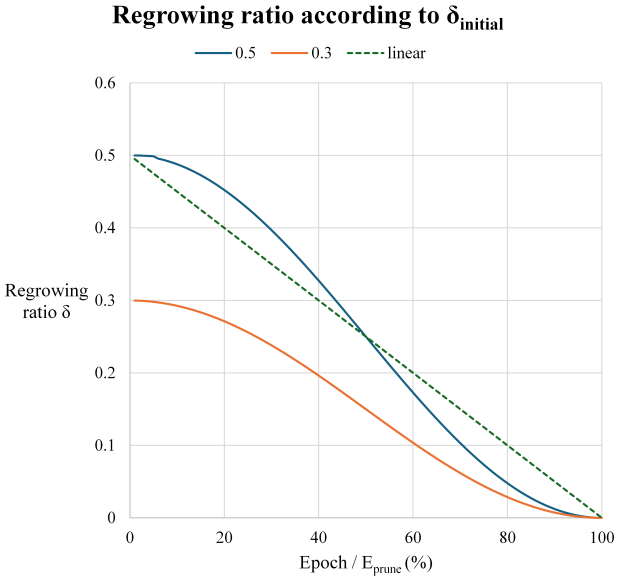

FIGURE 4. Regrowing ratio according to initial value and function.

2) PRUNING TO ZERO

In our proposed method, we set the redundant filters and their corresponding parameters to zero in the pruning stage for a single learning process so that they do not affect the feature maps and hence do not affect the final result. This ensures that there is no performance degradation when removing those parameters to get a pruned network. Looking on Line 4 in Algorithm 1, we first get the set of corresponding filters k^i_{prune} where $|\gamma|$ is less than a threshold t. Based on this, we remove those filters and their corresponding parameters on Line 5. The corresponding parameters can be determined as follows.

The associated parameters can be found in the convolutional and batch-normalization layers. If we determine that a particular filter in the convolutional layer, shown in Fig. 3 as green, is redundant, we set it to zero and also set the associated parameters in the subsequent batch normalization layer γ and β to zero, and one channel of output feature map generated by the zero parameters will also have a zero matrix. Also, setting the channel of one of the filters in the $(n+1)$-*th* convolutional layer, which operates on the channel of the corresponding feature map as input, to zero. By finding the redundant filters and setting the corresponding parameters to zero, ASFP can improve the existing two training sessions into one training session. So we could reduce the training time and power consumption.

3) REGROWING FOR ADAPTIVE SCALING

The adaptive scaling method considers the changing importance of filters as learning progresses, and utilizes regrowing to randomly select some of the pruned filters to recover to the value just before pruning, and reduces the number of recovered filters using cos decay to make the size of the model variable to obtain the target pruning network while maintaining maximum accuracy. The overall method is mentioned in Algorithm 1, Lines 6 to 9. In line 6, we compute the regrowing ratio δ, which controls the number of filters that are regrown to achieve the target pruning ratio. This is done in the form of a cos decay like (5), setting an initial regrowing ratio, $\delta_{initial}$, and configuring it to go to zero when the current epoch(e) reaches E_{prune}. Depending on the relationship between e and E_{prune}, the case of $\delta_{initial}$ for 0.3 and 0.5 with cos decay and the case of linear decay are shown in Fig. 4. In this paper, we use cos decay because it has been shown experimentally to perform better than linear.

$$\delta = \frac{1}{2}\left(1 + cos\left(\frac{e}{E_{prune}}\pi\right)\right) \times \delta_{initial} \tag{5}$$

The adaptive scaling method has two goals. The first is to regrow and re-evaluate the importance of each filter because the importance of each filter is not accurately determined at the beginning of training. Second, it allows the model size to be flexible, leading to better performance. To summarize, the conventional method using the one-shot pruning method uses sparsity learning to determine filter redundancy and fixes the size of the model through a single pruning to obtain a pruned network. However, redundancy in a situation where redundant filters affect the results and interconnections exist can be judged to be different from the filter redundancy for an optimal pruned network. Therefore, to account for the change in filter importance caused by pruning, regrowing for adaptive scaling can be used to gradually prune the model size to the target pruned network to account for the changing redundancy caused by pruning, resulting in higher performance.

In CHEX [16], the value of the filter that is recovered when regrowing is Most recently used(MRU). The reason for this

is that the performance of MRU is 0.5% to 1.9% higher than randomly recovering to 0. Therefore, we use MRU in this paper, and since the filters subject to regrowing include not only the filters pruned in the immediately preceding pruning stage but also the filters pruned several steps earlier, we save the last learned parameters in every iteration so that they can be used in regrowing.

As a way to select the regrowing filter, the CHEX method uses an importance sampling method based on channel orthogonality. The reason for this is to consider the inter-channel dependency of the pruned filter and the active filter. However, this method considers the relationship between all filters, contrary to one of the goals of regrowing, which is to consider the change in filter redundancy caused by gradually reducing the model size by pruning. In other words, the regrowing target obtained by the relationship of all filters does not represent the importance of the pruned network. Therefore, in this paper, the regrowing target is selected by random selection instead of importance sampling, and it is shown in Section V that higher performance can be obtained experimentally.

4) FIX REDUNDANT FILTERS TO ZERO

After pruning and regrowing are completed, the zero parameters will change to non-zero values as learning progresses. This means that the redundant parameters will affect the output feature map and a single learning process cannot be satisfied. Therefore, redundant filters are excluded from learning in the proposed method by fixing them to 0 at each iteration. The method is as follows.

Once the pruning and regrowing stages are completed, we can obtain k^i_{target}, the set of pruned filters as shown in Algorithm1 Line 10, by substrate k^i_{regrow} from k^i_{prune}. Therefore, we set k^i_{target} to zero at each iteration so that it does not affect the output feature map, as shown on Line 14.

IV. EXPERIMENTS

This section compares ASFP results of several classification and object detection networks with different datasets using the Pytorch framework.

A. IMAGE CLASSIFICATION

In Table. 1, we evaluate the results after pruning on ResNet-56 [3] and VGG-16 [4] using the CIFAR-10 [35] dataset. In Table. 2. we evaluate the accuracy after pruning ResNet-50 [3] using the ImageNet [36] dataset. We also compare the accuracy according to the FLOPs reduction ratio in Fig. 5.

1) CIFAR-10

The CIFAR-10 dataset has 60,000 images with ten classes divided into 50,000 training images and 10,000 validation images. For training, image size and batch size are 32 and 128, respectively. The learning rate is 0.01, and the momentum is 0.9. We train for E_{total} of 150 epochs with E_{prune} of 80 and E_{step} of 2.

TABLE 1. The comparison results of conventional and proposed ASFP methods on ResNet-56/VGG-16 networks using the CIFAR-10 dataset.

Method (Automatic[1])	Top-1 Accuracy (%) Baseline	Pruned	Δ	FLOPs (↓ %[2])	Parameters (↓ %[2])
ResNet-56					
L1-norm [27] (N)	93.04	93.01	-0.03	90.90M (27.6)	0.73M (13.7)
NISP [20] (Y)	93.04	93.01	-0.03	81.00M (35.5)	0.49M (42.4)
GAL [21] (Y)	93.26	93.38	+0.12	78.30M (37.6)	0.75M (11.8)
SCP [30] (N)	93.69	93.23	-0.46	N/A (N/A)	N/A (51.5)
HRank [25] (N)	93.26	93.17	-0.09	62.72M (50.8)	0.49M (42.4)
TCFP [15] (Y)	93.26	89.57	-3.69	64.59M (49.4)	0.42M (50.6)
CHIP [28] (N)	93.26	94.16	+0.90	65.94M (47.4)	0.48M (42.8)
Ours (Y)	93.26	**93.39**	**+0.13**	**63.47M (50.3)**	**0.42M (50.6)**
VGG-16					
SSS [22] (Y)	93.96	93.02	-0.94	183.13M (41.6)	3.93M (73.8)
GAL [21] (Y)	93.96	93.42	-0.54	171.89M (45.2)	2.67M (82.2)
CHIP [28] (N)	93.96	93.86	-0.10	131.17M (58.1)	2.76M (81.6)
Ours-1[3] (Y)	93.96	**93.44**	**-0.52**	**148.94M (52.7)**	**2.37M (84.2)**
HRank [25] (N)	93.96	92.34	-1.62	108.61M (65.7)	2.64M (82.4)
HRank [25] (N)	93.96	91.23	-2.73	73.70M (76.8)	1.78M (88.1)
SOSP [29] (Y)	94.18	92.71	-1.47	N/A (86.3)	N/A (97.8)
Ours-2[4] (Y)	93.96	**92.41**	**-1.55**	**46.00M (85.4)**	**1.92M (87.2)**

[1]Indicate whether using automatic method(Y) or predefined method(N)
[2]Pruned ratio: the higher value means more pruned
[3]Aiming to prune 55% of the FLOPs and 85% of the parameters
[4]Aiming to prune 85% of the FLOPs and 85% of the parameters

First, the experimental results on ResNet-56 show ASFP achieves 0.13% accuracy improvement even though 50.3% and 50.6% FLOPs and parameters are pruned compared to the baseline, respectively. It also improves the accuracy by 0.22% compared to HRank [25] with a similar pruning ratio. In particular, compared to GAL [21], ASFP helped to find the optimal pruned network with 12.7% fewer FLOPs and 38.8% fewer parameters but 0.01% higher accuracy. On the other hand, compared to ASFP, CHIP [28] outperforms by 0.77%. Still, it has the disadvantage that it uses a pre-defined method, which requires performing many combinations to get the optimal network. On the other hand, ASFP specifies the sparsity of each layer with an automatic method, so it is more simple to obtain the optimal network.

TABLE 2. The comparison results of conventional and proposed ASFP methods on the ResNet-50 network using the ImageNet dataset.

Method	Automatic[1]	Top-1 Accuracy (%)		Top-5 Accuracy (%)		FLOPs	Parameters	Inference time
		Accuracy (%)	Δ	Accuracy (%)	Δ	(↓ %[2])	(↓ %[2])	(ms)
Baseline	-	76.15	-	92.87	-	-	-	91.42
SFP [26]	N	74.61	-1.54	92.06	-0.81	41.80	N/A	N/A
Autopruner [23]	Y	74.76	-1.39	92.15	-0.72	48.70	N/A	N/A
Taylor [24]	Y	74.50	-1.68	N/A		44.90	44.50	N/A
SOSP [29]	Y	74.39	-1.76	N/A		51.00	49.00	N/A
CHEX [16]	Y	77.40	+ 1.25	N/A		51.5	N/A	91.34
Ours-1[3]	**Y**	**75.23**	**-0.92**	**92.40**	**-0.47**	**50.10**	**30.70**	**67.20**
GAL [21]	Y	71.95	-4.20	90.94	-1.93	43.00	16.90	N/A
HRank [25]	N	71.98	-4.17	91.01	-1.86	62.10	46.00	52.29
CHIP [28]	N	73.30	-2.85	91.48	-1.39	76.70	68.60	37.91
CHEX [16]	Y	76.00	+ 0.77	N/A		74.1	N/A	91.36
Ours-2[4]	**Y**	**72.32**	**-3.83**	**90.89**	**-1.98**	**74.00**	**64.50**	**37.50**

[1]Indicate whether using automatic method(Y) or predefined method(N)
[2]Pruned ratio: the higher value means more pruned
[3]During training, the pruning target for FLOPs is 50% and for parameters is 70%
[4]During training, the pruning target for FLOPs is 75% and for parameters is 65%

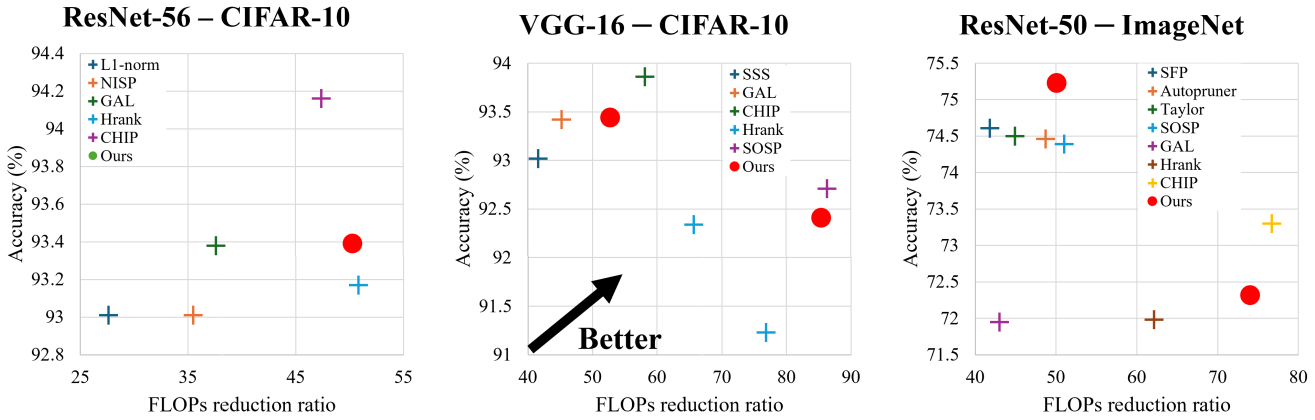

FIGURE 5. The comparison results of conventional and proposed ASFP methods using various classification networks and datasets.

The results for VGG-16 show that ASFP has a 1.55% decrease in accuracy compared to baseline with 85.4% and 87.2% pruning of FLOPs and parameters, respectively, which is a 1.18% increase in accuracy compared to HRank with a similar pruning ratio and a 0.07% increase in accuracy compared to network with a smaller pruning ratio. In the case of SOSP, its performance is 0.3% higher than Ours-2. But Ours-2 can obtain a pruned network faster than SOSP because Ours-2 uses one training process and SOSP uses two training processes. In Ours-1, The results with 52.7% and 84.2% pruning of FLOPs and parameters, respectively, showed a 0.52% accuracy decrease. This is 0.02% to 0.42% more accurate than other methods with similar pruning ratios. Thus, the experimental results of ASFP on ResNet-56 and VGG-16 using CIFAR-10 perform well on various networks.

2) ImageNet

The ImageNet dataset has 1,200,000 training images and 50,000 validation images with 1,000 classes. The training image size and batch size are 224 and 256, respectively. For inference time, we utilize a pruned network whose code is publicly available and executable. We train for E_{total} of 150 epochs, assigning 80 epochs to E_{prune} and 5 epochs to E_{step}. We set the image size and batch size to 224 and 1, respectively, and measured it on NVIDIA Jetson Nano.

In Table. 2, we evaluate the results of ASFP with previous studies using ResNet-50 and ImageNet datasets. We execute experiments with various FLOPs and parameters pruning ratios and find that Ours-1 achieves a 4.78% reduction from baseline when FLOPs and parameters are pruned by 77.72% and 69.09%, respectively. In Ours-2, parameters and FLOPs are pruned by 64.55% and 74.09%, respectively, which is a reduction of 3.83% from baseline, which is an improvement of up to 0.84% compared to other methods with higher pruning ratios. These results show that the proposed method works appropriately on various networks and various datasets.

TABLE 3. The comparison results of conventional and proposed ASFP methods on YOLOv7 network using PASCAL VOC and COCO datasets with different pruning ratios.

Dataset	Method	mAP(0.5) (%)	Δ	mAP(0.5-0.95) (%)	Δ	FLOPs (↓ %[1])	Parameters (↓ %[1])	Inference time (ms)			
								network	NMS	Total	Δ
VOC	Baseline	90.0	-	71.2	-	-	-	63.1	3.3	69.4	-
	TCFP [15]	87.0	-3.0	66.5	-4.7	34.59	29.46	54.9	3.2	58.1	-8.3
	Ours	86.9	-3.1	67.4	-3.8	36.14	32.95	51.2	3.2	54.4	-12.0
	TCFP [15]	86.0	-4.0	65.1	-6.1	44.54	40.00	51.0	3.2	54.2	-12.2
	Ours	86.5	-3.5	66.3	-4.9	45.41	42.81	49.3	3.2	52.5	-13.9
	TCFP [15]	85.7	-4.3	65.0	-6.2	49.95	49.50	49.7	3.1	52.8	-13.6
	Ours	86.1	-3.9	65.8	-5.3	50.82	50.74	48.6	3.2	51.8	-14.6
COCO	Baseline	69.7	-	51.2	-	-	-	64.1	3.7	67.8	-
	TCFP [15]	67.3	-2.4	49.1	-2.1	27.27	29.25	59.0	3.4	62.4	-5.4
	Ours	68.3	-1.4	50.0	-1.2	27.37	27.92	55.7	3.3	59.0	-8.8
	TCFP [15]	67.1	-2.6	48.8	-2.4	36.84	38.83	55.0	3.3	58.3	-9.5
	Ours	67.5	-2.2	49.0	-2.2	36.84	38.83	53.1	3.2	56.3	-11.5
	TCFP [15]	66.3	-3.4	48.0	-3.2	46.99	48.72	53.7	3.3	57.0	-10.8
	Ours	66.8	-2.9	48.3	-2.9	46.41	47.40	51.5	3.2	54.7	-13.1

[1]Pruned ratio: the higher value means more pruned

In the case of CHEX and CHIP, their performance is higher than Ours-2, but Ours-2 has a faster inference time than experiments of CHEX and CHIP with a similar pruning ratio to Ours-2. In particular, the comparison with Hrank shows an improvement of 1.39 times. In particular, when compared to CHEX, the accuracy is 3.68% lower, but the inference time is 2.43 times faster. The reason for this is that the code provided in CHEX [40] does not provide code to remove redundant filters from the pruned network, and in particular, it does not consider the skip connections present in ResNet-50, resulting in an unworkable network if all filters with zero are removed as suggested in the paper.

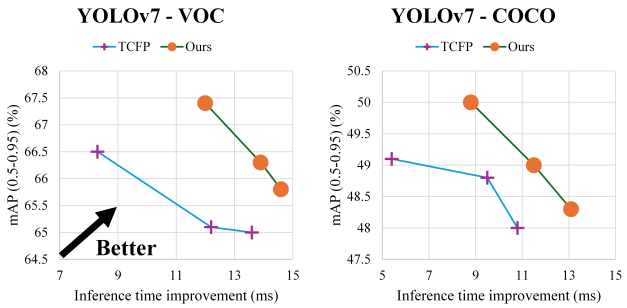

FIGURE 6. The experimental results comparing inference time of TCFP [15] and proposed ASFP methods on YOLOv7 networks using the PASCAL VOC and COCO datasets.

B. OBJECT DETECTION

In Table. 3, we experiment and compare the performance after pruning on YOLOv7 [8] using the PASCAL VOC [38] dataset and COCO [37] dataset. In the inference time, the network is the time it takes to get the output from the pruned network, and NMS is the time it takes to perform non-maximum suppression, and these two times are summed and expressed as the total and the difference between the total time and the inference time of the baseline is shown as Δ. We also compare the accuracy according to the inference time improvements in Fig. 6.

1) PASCAL VOC

The PASCAL VOC dataset has 20 classes and consists of about 16,000 training images and 5,000 validation images. The image size and batch size are set to 640 and 64, respectively, and 300 epochs of training were performed. The other settings are the same as the default settings of YOLOv7. Inference time is measured on NVIDIA Jetson Xavier NX with the same image size and batch size set to 1. The training process spans 300 epochs, including 200 epochs for E_{prune} and five epochs for E_{step}.

In Table. 3, we compare TCFP [15] and ASFP by obtaining pruned networks with three different pruning ratios. Compared to baseline, ASFP has 36.14%, 45.41%, and 50.82% fewer FLOPs, 32.95%, 42.81%, and 50.74% fewer parameters, and 3.8%, 4.9%, and 5.3% less accuracy on mAP(0.5-0.95). This is 0.9%, 1.2%, and 0.9% more accurate than TCFP, respectively. In addition, inference time shows that non maximum suppression(NMS) is similar between networks with similar pruning ratios, but significantly improves in network time, with total time improvements of 12.0ms, 13.9ms, and 14.6ms over baseline, respectively, which is up to 44.6% better than TCFP.

2) MS COCO

The MS COCO dataset has 80 classes and consists of about 11.5k training images and 5k validation images. The image size and batch size are set to 640 and 64, respectively, and 300 epochs of training are performed. The other settings are the same as the default settings of YOLOv7. Inference time is measured on NVIDIA Jetson Xavier NX with the same image size and batch size set to 1. Our training consists of

TABLE 4. The comparison results of regrowing ratio decision criterion with 50% target pruned ResNet-56 using CIFAR-10 dataset.

Method	$\delta_{initial}$	Top-1 Accuracy (%)
Baseline	-	93.26
Cosine	0.3	92.88
	0.4	92.77
	0.5	**93.39**
Linear	0.3	93.16
	0.4	92.77
	0.5	92.99

300 epochs in E_{total}, with 200 epochs allocated to E_{prune} and five epochs to E_{step}.

In Table. 3, we compare TCFP and ASFP by obtaining pruned networks with three different pruning ratios. Compared to the baseline, ASFP reduces FLOPs by 27.37%, 46.41% parameters by 27.92%, and 47.40% with only 1.2% and 2.9% mAP(0.5-0.95) reduction, while TCFP reduced FLOPs by 27.27%, 46.99% parameters by 29.25%, and 48.72% with 2.1% and 3.2% mAP (0.5-0.95) degradation. It shows that the proposed method has higher accuracy with a smaller model size than TCFP. In addition, the inference time shows a slight improvement in NMS and a significant improvement in network time, resulting in a total time improvement of 8.8ms - 13.1ms over baseline and up to 62.9% improvement over TCFP. These results show that the proposed ASFP obtains a higher-performance pruned network with one-step training without retraining, and the optimal pruned network can be obtained by pruning redundant filters through adaptive scaling.

V. ABLATION ANALYSIS

We demonstrate the effectiveness of ASFP on key features through ablation studies. All experiments are compared using the ResNet-56 and CIFAR-10 datasets with FLOPs and parameters pruned at a 50% pruning target ratio. Other settings are the same as in the previous experiments.

A. REGROWING RATIO DECISION CRITERION

Table. 4 compares the results according to the method of determining the regrowing ratio and the initial value. We compare the accuracy of the pruned network by setting the initial value as 0.3, 0.4, and 0.5 while comparing the cos and Linear methods shown in Fig. 4. As a result, the accuracy of the cos method is 93.39% when the initial value is 0.5, while the accuracy of the Linear method is 93.16% when the initial value is 0.3. Therefore, using (5) and using $\delta_{initial}$ as 0.5 can achieve at least 0.23% higher performance than using Linear.

B. TRAINING CRITERION

Table. 5 shows the difference in accuracy between the pruning method and the regrowing method. The conventional pruning method shown in (a) of Fig. 1 is used in the case of the retraining method, and the experiment is conducted according to whether the regrowing method is used or not

TABLE 5. The Comparison results of training criterion with 50% target pruned ResNet-56 using CIFAR-10 dataset.

Method	Regrowing[1]	Top-1 Accuracy (%)
Baseline	-	93.26
Retraining[2]	N	93.05
	Y	93.20
Ours	**Y**	**93.39**

[1]Indicate whether using regrowing method
[2]Using conventional pruning method(sparsity learning, pruning, re-training)

TABLE 6. The comparison results of regrowing criterion with 50% target pruned ResNet-56 using CIFAR-10 dataset.

Method	Top-1 Accuracy (%)
Baseline	93.26
CHEX [16] (Importance sampling)	93.04
Ours (Random selection)	**93.39**

in the sparsity learning stage. If using the re-training method, the regrowing method improved by 0.15% than without the regrowing method. The performance of ASFP with the regrowing method was improved by 0.19% compared to the conventional method with regrowing. Therefore, we can analyze that the regrowing method contributes to finding the optimal pruned network and that the method of fixing the redundant parameters to zero when pruning in the proposed method helps preserve the pruned network's performance as much as possible.

C. TARGET REGROWING FILTER DECISION CRITERION

Table. 6 shows the difference in accuracy based on how the target regrowing filter is defined. We trained using the same target pruning ratio and the same hyperparameters. First, using the importance sampling method as CHEX, the accuracy is 93.04%, but using the random selection method used in our proposed method, the performance is 93.39%, which is an improvement of 0.35%. Therefore, we can conclude that the proposed random selection is more helpful in obtaining a pruned network with high performance.

D. LIMITATIONS

ASFP has been experimented only in classification and object detection tasks among vision networks using CNN, so it is difficult to guarantee its performance in tracking and segmentation. Also, since it analyzes the gamma of each batch normalization layer, it has the disadvantage that it must be modified according to the structure of each network such as residual connections. This is a problem for quickly applying to various networks, but it has the advantage that it is easy to adjust the pruning range to suit the characteristics of each network. In particular, in the case of YOLOv7, the last layer for detection was not pruned to maintain high performance in order to maximize accuracy.

VI. CONCLUSION

In this paper, we proposed adaptive scaling filter pruning (ASFP) as a network compression method for running various computer vision networks on edge devices. This method utilizes a single learning process that reduces two learning processes to a single learning process and proposes adaptive scaling to consider the redundancy difference due to pruning. By using this method, we could obtain pruned networks with less accuracy degradation than other methods on ResNet-56, ResNet-50, and VGG-16 classification networks using CIFAR-10 and ImageNet datasets. In particular, we obtained pruned networks with up to 0.84% higher accuracy in experiments using ImageNet and ResNet-50. In the YOLOv7 object detection network, we also achieved a high pruning ratio on PASCAL VOC and MS COCO datasets with less accuracy loss than other methods and achieved up to 21.0% improvement in inference time on embedded systems using NVIDIA Jetson Xavier NX. These experiments demonstrate that the proposed ASFP can be used in classification networks and object detection networks. To overcome the limitations mentioned in Section V-D, we will apply ASFP on object tracking networks and segmentation networks.

ACKNOWLEDGMENT

The EDA tool was supported by the IC Design Education Center (IDEC), Korea.

REFERENCES

[1] S. Naoumi, A. Bazzi, R. Bomfin, and M. Chafii, "Complex neural network based joint AoA and AoD estimation for bistatic ISAC," *IEEE J. Sel. Topics Signal Process.*, early access, Apr. 10, 2024, doi: 10.1109/JSTSP.2024.3387299.

[2] M. Delamou, A. Bazzi, M. Chafii, and E. M. Amhoud, "Deep learning-based estimation for multitarget radar detection," in *Proc. IEEE 97th Veh. Technol. Conf. (VTC-Spring)*, vol. 32, Jun. 2023, pp. 1–5.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[5] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Jan. 1998.

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y.-M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 7464–7475.

[9] W. Liu, "SSD: Single shot multibox detector," in *Proc. 14th Eur. Conf. Comput. Vis.* Amsterdam, The Netherlands: Cham, Switzerland: Springer, Oct. 2016, pp. 21–37.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. 18th Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, vol. 9351, Munich, Germany. Cham, Switzerland: Springer, 2015, pp. 234–241.

[12] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder–decoder with atrous separable convolution for semantic image segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 801–818.

[13] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[14] X. Xu, M. S. Park, and C. Brick, "Hybrid pruning: Thinner sparse networks for fast inference on edge devices," 2018, *arXiv:1811.00482*.

[15] J. Jeon, J. Kim, J.-K. Kang, S. Moon, and Y. Kim, "Target capacity filter pruning method for optimized inference time based on YOLOv5 in embedded systems," *IEEE Access*, vol. 10, pp. 70840–70849, 2022.

[16] Z. Hou, M. Qin, F. Sun, X. Ma, K. Yuan, Y. Xu, Y.-K. Chen, R. Jin, Y. Xie, and S.-Y. Kung, "CHEX: CHannel EXploration for CNN model compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 12277–12288.

[17] T. Chen, B. Ji, T. Ding, B. Fang, G. Wang, Z. Zhu, L. Liang, Y. Shi, S. Yi, and X. Tu, "Only train once: A one-shot neural network training and pruning framework," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 19637–19651.

[18] B. Li, B. Wu, J. Su, and G. Wang, "EagleEye: Fast sub-net evaluation for efficient neural network pruning," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, 2020, pp. 639–654.

[19] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.

[20] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[21] 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. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2785–2794.

[22] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 304–320.

[23] J.-H. Luo and J. Wu, "AutoPruner: An end-to-end trainable filter pruning method for efficient deep model inference," *Pattern Recognit.*, vol. 107, Nov. 2020, Art. no. 107461.

[24] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11256–11264.

[25] 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. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1526–1535.

[26] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," 2018, *arXiv:1808.06866*.

[27] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," 2016, *arXiv:1608.08710*.

[28] Y. Sui, M. Yin, Y. Xie, H. Phan, S. A. Zonouz, and B. Yuan, "Chip: Channel independence-based pruning for compact neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 34, 2021, pp. 24604–24616.

[29] M. Nonnenmacher, T. Pfeil, I. Steinwart, and D. Reeb, "SOSP: Efficiently capturing global correlations by second-order structured pruning," 2021, *arXiv:2110.11395*.

[30] M. Kang and B. Han, "Operation-aware soft channel pruning using differentiable masks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5122–5131.

[31] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, "DepGraph: Towards any structural pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2023, pp. 16091–16101.

[32] Y. Li, K. Adamczewski, W. Li, S. Gu, R. Timofte, and L. Van Gool, "Revisiting random channel pruning for neural network compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 191–201.

[33] X. Ding, T. Hao, J. Tan, J. Liu, J. Han, Y. Guo, and G. Ding, "ResRep: Lossless CNN pruning via decoupling remembering and forgetting," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4490–4500.
[34] H. Wang and Y. Fu, "Trainability preserving neural pruning," 2022, *arXiv:2207.12534*.
[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Comput. Sci. Univ. Toronto, Toronto, ON, Canada, 2009.
[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[37] T. Lin, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, Zurich, Switzerland. Cham, Switzerland: Springer, 2014, pp. 740–755.
[38] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge: A retrospective," *Int. J. Comput. Vis.*, vol. 111, no. 1, pp. 98–136, Jan. 2015.
[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[40] zejiangh. (2022). *Filter-Gap*. [Online]. Available: https://github.com/zejiangh/Filter-GaP

HYUNJUN KO (Graduate Student Member, IEEE) received the B.S. degree in electronics engineering from Inha University, Incheon, South Korea, in 2023, where he is currently pursuing the M.S. degree in electrical and computer engineering. His current research interests include deep learning and computer vision.

JIN-KU KANG (Senior Member, IEEE) received the B.S. degree from Seoul National University, Seoul, South Korea, in 1983, the M.S. degree in electrical engineering from New Jersey Institute of Technology, Newark, NJ, USA, in 1990, and the Ph.D. degree in electrical and computer engineering from North Carolina State University, Raleigh, NC, USA, in 1996. From 1983 to 1988, he was with Samsung Electronics, Inc., South Korea, where he was involved in memory design. In 1988, he was with Texas Instruments, South Korea. From 1996 to 1997, he was a Senior Design Engineer with Intel Corporation, Portland, OR, USA, where he was involved in high-speed I/O and timing circuits for processors. Since 1997, he has been with Inha University, Incheon, South Korea, where he is currently a Professor and leads the System IC Design Laboratory, Department of Electronics Engineering. His research interest includes high-speed/low-power mixed-mode circuit design for high-speed serial interfaces.

YONGWOO KIM (Member, IEEE) received the B.S. and M.S. degrees from Inha University, Incheon, South Korea, in 2007 and 2009, respectively and the Ph.D. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2019. From 2009 to 2017, he was a Senior Engineer with Silicon Works Company Ltd., Daejeon. From 2019 to 2020, he was a Senior Researcher with the Artificial Intelligence Research Division, Korea Aerospace Research Institute, Daejeon. From 2020 to 2024, he was an Assistant Professor with the Department of System Semiconductor Engineering, Sangmyung University, Cheonan, South Korea. Since 2024, he has been with Korea National University of Education, Cheongju, South Korea, where he is currently an Assistant Professor with the Department of Technology Education. His current research interests include image/video processing algorithm, super-resolution, and deep learning hardware architecture for vision processing.