FP-AGL: Filter Pruning With Adaptive Gradient Learning for Accelerating Deep Convolutional Neural Networks

Nam Joon Kim, *Student Member, IEEE*, and Hyun Kim, *Member, IEEE*

***Abstract*—Filter pruning is a technique that reduces computational complexity, inference time, and memory footprint by removing unnecessary filters in convolutional neural networks (CNNs) with an acceptable drop in accuracy, consequently accelerating the network. Unlike traditional filter pruning methods utilizing zeroing-out filters, we propose two techniques to achieve the effect of pruning more filters with less performance degradation, inspired by the existing research on centripetal stochastic gradient descent (C-SGD), wherein the filters are removed only when the ones that need to be pruned have the same value. First, to minimize the negative effect of centripetal vectors that gradually make filters come closer to each other, we redesign the vectors by considering the effect of each vector on the loss-function using the Taylor-based method. Second, we propose an adaptive gradient learning (AGL) technique that updates weights while adaptively changing the gradients. Through AGL, performance degradation can be mitigated because some gradients maintain their original direction, and AGL also minimizes the accuracy loss by perfectly converging the filters, which require pruning, to a single point. Finally, we demonstrate the superiority of the proposed method on various datasets and networks. In particular, on the ILSVRC-2012 dataset, our method removed 52.09% FLOPs with a negligible 0.15% top-1 accuracy drop on ResNet-50. As a result, we achieve the most outstanding performance compared to those reported in previous studies in terms of the trade-off between accuracy and computational complexity.**

***Index Terms*—Adaptive gradient learning, convolutional neural networks, filter pruning, light-weight technique, taylor expansion.**

I. Introduction

In RECENT years, convolutional neural networks (CNNs) have been used in various computer vision tasks, including image classification [1]–[4], object detection [5]–[8], and segmentation [9], [10], [55] owing to their remarkable development through the efforts of several researchers. However, it is extremely challenging to utilize deep CNNs on resource-limited mobile devices or autonomous vehicles since the performance improvement of these CNNs is generally achieved through the use of deeper network structures with more hidden layers, resulting in large model sizes, high computational costs, and a heavy memory footprint [6], [11], [12]. In detail, in order to utilize CNN-based applications in mobile devices, a model of less than 600M Floating point operations per second (FLOPs) and less than 5W power consumption is required [54]. However, it is reported that running a CNN with 1 billion connections (i.e., CNN that are not lightweight) at 20 frames per second (FPS) requires 12.8W ($= 20\times1\text{G}\times640\text{pJ}$) only for DRAM access [18]. Moreover, CNN-based object detection/segmentation models for autonomous driving applications must guarantee real-time operation of at least 30 FPS with high-resolution images in the embedded platform for smooth and safe driving [5], and consequently, there is a constant demand for reducing the network size.

To enable the utilization of CNNs in real-world applications in response to these demands, numerous studies related to network compression and acceleration have been conducted [13]–[39], [56]–[64]. Among them, pruning is a commonly used and popular approach to reduce the model size of CNNs; in this method, redundant weights or filters are removed while ensuring an acceptable level of degradation in accuracy. Weight pruning [16]–[21], [45], [56], in particular, can be used to achieve a very high compression efficiency by removing redundant weights in a filter. However, as it leads to unstructured sparsity in CNNs, accelerating the inference phase of CNNs in a real GPU environment without the use of special software/hardware support is impossible. On the contrary, filter pruning [22]–[39], [57]–[60] not only reduces computational costs significantly by removing the filters of the convolution layers, but also leads to an actual acceleration of the inference phase in the GPU environment without special software/hardware support. In addition, it has high scalability and compatibility that enables its easy application to various CNN models, making it widely usable in a variety of domains.

Due to the aforementioned advantages of filter pruning, many filter pruning methods have been actively investigated, and state-of-the-art (SOTA) methods have succeeded in significantly reducing the model size of CNNs while maintaining a high accuracy. There are various approaches to perform filter pruning,

Manuscript received 21 January 2022; revised 12 May 2022 and 27 June 2022; accepted 4 July 2022. Date of publication 11 July 2022; date of current version 30 October 2023. This work was supported in part by the Industrial Fundamental Technology Development Program under Grant 20019367, in part by the Development of Low Power AI Architecture for AIoT, in part by the Ministry of Trade, Industry & Energy of Korea, in part by the Basic Science Research Program through the National Research Foundation of Korea, and in part by the Ministry of Education under Grant NRF-2019R1A6A1A03032119. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Mai Xu. (*Corresponding author: Hyun Kim.*)

The authors are with the Department of Electrical and Information Engineering and the Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, Korea (e-mail: rlarla2626@seoultech.ac.kr; hyunkim@seoultech.ac.kr).

Digital Object Identifier 10.1109/TMM.2022.3189496

which include the use of pre-defined architectures [22], [24], [25], [28], [36] and automatically determined architectures [23], [27], [35], [37], [40]–[43]. Among them, the automatically determined architectures have the advantage that filters can be removed globally so that the optimal structure of the network can be found and the FLOPs can be significantly reduced. However, it evades some problems that occur when pruning the networks with complicated structures such as ResNet [1], which are often used as the backbone in object detection or segmentation [9]. For example, Li *et al.* [22] and He *et al.* [25] pruned only the first layer of each residual block. Ding *et al.* [28] proposed a new optimization method called centripetal stochastic gradient descent (C-SGD) for network slimming. Because C-SGD can produce redundancy patterns without any heuristic knowledge, it can solve the problem of constrained filter pruning of networks with complicated structures such as identity mapping [1] or dense connection [3]. However, because the centripetal constraints of C-SGD [28], referred to as *centripetal vectors* in this paper, are added not only to the convolution layer weights but also to the parameters of batch normalization (BN) [44]. This eventually acts as a penalty vector, and performance degradation cannot be avoided. In addition, gradients modified to maintain the distance between filters, referred to as *averaged gradients* in this study, do not follow a normal gradient descent, thereby making it difficult to reach the optimal minimum.

Inspired by C-SGD [28], we redesign the update rule of C-SGD to achieve better performance of pruning in CNN-based computer vision tasks. First, we consider the change in loss induced by the converging of filters in the clusters to a single point (*i.e.*, the midpoint of filters) using Taylor expansion, and then redesign the centripetal vector by reflecting the change in loss. In addition, we propose an adaptive gradient learning (AGL) technique to adaptively update the weights using the original gradient and the average gradient to prevent the deformation of the gradients obtained through back-propagation as much as possible. Therefore, we optimize the loss-function with a normal gradient descent by partly utilizing the original gradient. It should be noted that the proposed AGL not only mitigates the performance degradation, but also accelerates the process of converging the filters to a single point. Experiments on various benchmark datasets (*i.e.*, CIFAR-10, ILSVRC-2012, PASCAL VOC, and SBD) and networks (*i.e.*, image classification, object detection, and segmentation) show that the proposed filter pruning method with AGL (FP-AGL) is superior to other SOTA pruning methods and has high compatibility. In particular, FP-AGL can reduce the FLOPs of ResNet-56 and ResNet-50 by approximately 60% on the CIFAR-10 and the ILSVRC-2012, respectively, with a negligible accuracy drop compared to the baseline, thereby achieving a SOTA performance. The major contributions of this paper are summarized as follows:

- In order to compensate for the problem that the existing method [28] is not loss-aware by imposing centripetal vectors on all convolution weights and BN parameters, we propose the redesigned centripetal vector (RCV) that redesigns the existing centripetal vector using Taylor expansion, which can reflect the change in loss, and the RCV can alleviate the performance degradation caused by pruning.
- In order to compensate for the problem that the existing method [28] has the weakness of deviating from the direction of normal gradient descent due to the exclusion of the original gradient, we propose the AGL that maintains the direction of original gradient descent as much as possible, and the AGL can mitigate the performance degradation and accelerate the process of filter convergence.
- Through experiments on various tasks and datasets, we prove that the proposed FP-AGL, a combination of RCV and AGL, has high scalability and compatibility for a variety of applications.

The rest of the paper is organized as follows. Section II describes studies related to weighted pruning and filter pruning. Section III provides a detailed description of the proposed FP-AGL. The experimental results are presented in Section VI, and finally, Section VII concludes the paper.

II. Related Works

A. Weight Pruning

Weight pruning can achieve a high compression ratio by removing the weights of filters through heuristics or optimization processes. Cun *et al.* [45] remove redundant weights after determining the importance of weights based on the Hessian matrix of the loss-function, and Han *et al.* [20] propose a dense-sparse-dense training framework that restores connections after pruning for the regularization of deep neural networks. The lottery ticket hypothesis [17] finds the winning tickets with faster convergence speed and higher accuracy than those of the baseline network. Morcos *et al.* [21] conduct experiments to verify that winning tickets are able to generally work well in various architectures, optimizers, and datasets, and showed through numerous experiments that winning tickets can achieve good performance even in large datasets such as ImageNet [46]. Ding *et al.* [56] propose an asymmetric convolution block using 1-D asymmetric convolution to enhance the representational power of standard square convolution. Ding et al [19] also propose momentum-SGD that changes the gradient flow for lossless pruning and end-to-end training, and it is possible to find better winning tickets than the lottery ticket hypothesis [17] through momentum-SGD. However, these weight pruning methods have a disadvantage in that, it is impossible to accelerate the inference phase without dedicated hardware or software owing to irregular sparsity.

B. Filter Pruning

Studies on filter pruning have been being actively conducted in recent years. This is because this technique can compensate for the disadvantages of weight pruning through the acceleration of the actual inference phase without any dedicated hardware or software support. Li *et al.* [22] remove filters based on the l_1-norm of the filter after analyzing the sensitivity of each layer from the pre-trained network. Ding *et al.* [57] propose an auto-balanced regularization method that uses l-2 regularization to penalize insignificant filters and stimulate important filters to become more and more important. After that, they repeat the pruning-retrain process until the desired compression ratio

is satisfied. Li *et al.* [59] propose the hinge filter pruning and decomposition through group sparsity. Aflalo *et al.* [60] propose a novel pruning method that simultaneously optimizes the accuracy and FLOPs and distills knowledge from the inner layers of the unpruned network. Liu *et al.* [23] and Ye *et al.* [26] obtain structured sparsity by imposing sparsity-induced regularization on the scaling factor in the batch normalization layers [44] during the training phase and then remove filters below a predefined threshold during the pruning phase. He *et al.* [24] select filters for every epoch based on the l_2-norm, and the selected filters are then removed in a soft manner. He *et al.* [25] also softly prune the filters using the geometric median rather than the norm-based criterion for filter importance. Ding *et al.* [58] propose a data-dependent soft pruning method based on long short-term memory (LSTM) for learning hierarchical characteristics to select an appropriate layer for pruning. In contrast to the existing deterministic methods, Zhao *et al.* [30] prune redundant channels after evaluating the distribution of channel saliency using the Bayes' rule. Gao *et al.* [27] add tiny auxiliary connections to the convolution layer to skip insignificant convolution operations during the inference phase, and these connections reduce computations while maintaining the network capacity. Molchanov *et al.* [29] use the Taylor expansion to evaluate the importance of filters, remove the least important filters, and fine-tune the network to achieve a comparable accuracy. However, most of the abovementioned studies have not completely addressed the problem of constrained filter pruning in networks with complicated structures such as those with residual connections [1]. C-SGD [28] uses a new optimization method to perfectly prune a network with a complicated structure. During training, C-SGD makes some filters identical to one another and then prunes the redundant ones. This method easily resolves the constrained filter pruning problem because filters that need to be pruned are determined without any heuristic knowledge.

C. Centripetal Stochastic Gradient Descent

In this section, we review the pruning technique of C-SGD [28] in detail; the FP-AGL proposed in this paper is inspired from this technique. First, the associated symbols and notations are described. We first assume a loss function, *L(D,W)*, and consider a classification task involving cross-entropy. In *L(D,W)*, *D* denotes the training set and *W* denotes the trainable parameters of the CNNs. The trainable parameters include the weights of the convolutional layer as well as the scaling factors and bias of the batch normalization [44].

C-SGD [28] aims to aggregate the filters belonging to each cluster into a single point and eventually make them identical. After the training is complete, only one filter per cluster is left in the pruning phase and the remaining ones are removed. Through this process, the problem of constrained filter pruning in networks with complicated structures, such as identity mapping [1] and dense connection [3], can be effectively solved. Specifically, as described in the following equation, the goal of C-SGD is to make the values of the two filters F_0 and F_1 identical, where i is the current iteration.

$$\lim_{i\to\infty} \|F_0^{(i)} - F_1^{(i)}\| = 0 \tag{1}$$

To satisfy (1), the following equations make the two filters F_0 and F_1 gradually closer to one another.

$$F_0^{(i+1)} \leftarrow F_0^{(i)} - \eta\frac{1}{2}\left(\frac{\partial L}{\partial F_0^{(i)}} + \frac{\partial L}{\partial F_1^{(i)}}\right) + \varepsilon\frac{1}{2}\left(F_1^{(i)} - F_0^{(i)}\right) \tag{2}$$

$$F_1^{(i+1)} \leftarrow F_1^{(i)} - \eta\frac{1}{2}\left(\frac{\partial L}{\partial F_0^{(i)}} + \frac{\partial L}{\partial F_1^{(i)}}\right) + \varepsilon\frac{1}{2}\left(F_0^{(i)} - F_1^{(i)}\right) \tag{3}$$

The second terms in (2) and (3) are used to maintain the difference between the two filters, and the third terms, including ε, gradually narrow the distance between these filters. We define these two terms again as the *averaged gradient* and the *centripetal vector*, respectively. Parameter ε is the centripetal strength [28], and parameter η is the learning rate. $\frac{1}{2}(\frac{\partial L}{\partial F_0^{(i)}} + \frac{\partial L}{\partial F_1^{(i)}})$ is the averaged gradient of F_0 and F_1 and constrains the distance between the two filters so that the distance between them does not increase. However, because these gradients deviate from the direction of the normal gradient descent, a wider space must be searched through additional iterations to reach the optimal local minima. In this manner, the centripetal vector gradually narrows the distance between the two filters and eventually converges to a single point. However, because the movement of these filters follows the approach of L_2-regularization, when the difference between the two filters is large, the filters come close to each other quickly, but when the difference between the two values is small, the filters come close to each other slowly. Therefore, the distance between the filters can never be zero, which renders perfect pruning impossible [41]. In addition, because many redundant-induced penalties (*i.e.*, centripetal vectors) are added to the scaling factors and biases of the batch normalization layers [44] as well as the weights of the convolution layers, performance degradation is inevitable. In Section III, we present the FP-AGL technique that effectively addresses the aforementioned problems of C-SGD [28].

III. Proposed Method

A. Overview of the Proposed FP-AGL

The overall operational flow of the proposed FP-AGL is presented in Fig. 1. We utilize the RCV in Fig. 1(a) and AGL in Fig. 1(b) to collect the two filters, Filter_0 (*i.e.*, green boxes in Fig. 1(c)) and Filter_1(*i.e.*, orange boxes in Fig. 1(c)), into a single point. Detailed descriptions of the proposed RCV and AGL techniques are presented in Sections III–B and III-C, respectively. As shown in Fig. 1(c), the difference between Filter_0 and Filter_1 is gradually converged to 0 through RCV-based FP-AGL training to create redundancy in Filter_0 to minimize the loss due to pruning, and the process of convergence to a single point can be accelerated by performing normal training for the remaining filters through AGL. Fig. 1(d) shows that after FP-AGL training is completed, there is almost no difference between Filter_0 and

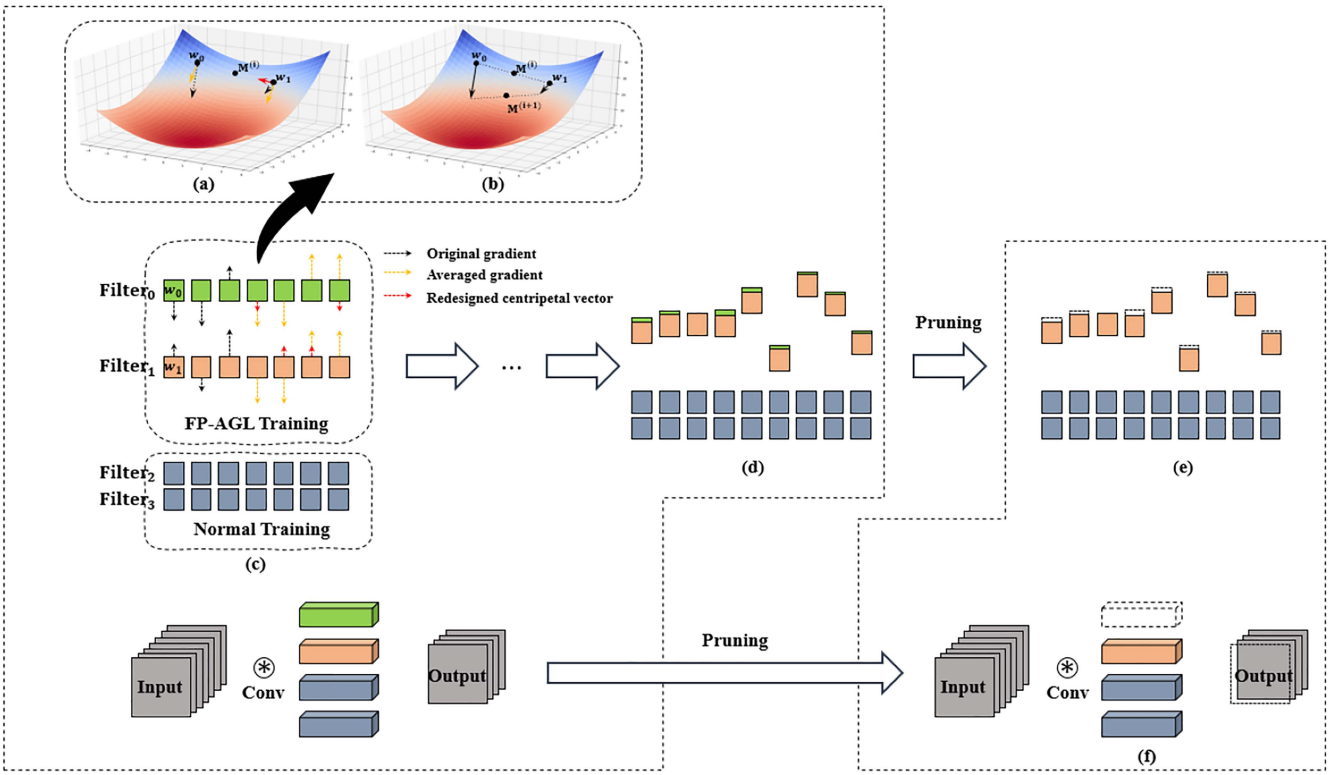


Fig. 1. Operation flow of the proposed FP-AGL.

Filter_1 (*i.e.*, green boxes and orange boxes are well overlapped). Because these two filters depicted in Fig. 1(d) have almost the same values, a compact network in Fig. 1(f) can be created without any loss of accuracy by pruning the redundant filter, Filter_0 with green boxes, as shown in Fig. 1(e). It should be noted that although the proposed method is described assuming two filters for each cluster to help the reader understand, the proposed method can be applied by grouping all filters by two filters (i.e., in pairs for all filters) even if there are more filters in the cluster.

B. Redesigning Centripetal Vector

As described in Section II-C, the centripetal vectors that cause the convolutional layer filters to come close together in the cluster are added based on the number of weights of the filters to be pruned [28]. Because this becomes a redundant-induced penalty that distorts the original gradients to optimize the loss function, performance degradation is inevitable. To address this problem, similar to the methods described in [19], [29], [42], and [47], we use the first-order Taylor expansion, which can approximate the change in loss when specific filters are removed from the network (*e.g.*, pruning filters). Although the second-order Taylor expansion can approximate the loss more accurately than the first-order, it has been reported that the performance of first-order and second-order Taylor expansion is comparable in measuring loss change [29]. Therefore, the proposed FP-AGL performs model pruning using first-order Taylor expansion considering the trade-off between accuracy and computational amount. It should be noted that it is very difficult to use the second-order in the training process of a large model due to the huge amount of computation of the Hessian matrix and memory constraints. The pruning method using this first-order Taylor expansion evaluates the importance I_m of weights as follows:

$$I_m = \left| \frac{\partial L(D, W)}{\partial w_m} \cdot w_m \right| \tag{4}$$

where $\frac{\partial L(D,W)}{\partial w_m}$ and w_m are the gradient and weight, respectively. The advantage of using this Taylor-based method is that it is possible to easily evaluate the importance of weights with a simple equation using only the weight and gradient.

The process of redesigning the centripetal vector using this first-order Taylor expansion is as follows: first, if there are two weights, w_0 and w_1, in each filter, a centripetal vector must be determined, where the weights converge to a single point without any performance loss. The network loss when w_0 and w_1 converge to w^* using the Taylor expansion can be expressed as follows:

$$L(D, W_{w_0 \to w^*}) = L(D, W) - \frac{\partial L(D, W)}{\partial w_0} \cdot (w_0 - w^*) \tag{5}$$

$$L(D, W_{w_1 \to w^*}) = L(D, W) - \frac{\partial L(D, W)}{\partial w_1} \cdot (w_1 - w^*) \tag{6}$$

where we ignore the higher-order term because it is computationally expensive. Notably, C-SGD [28], which imposes the centripetal vectors for all the weights without considering the change in loss, is not loss-aware, whereas our method defines the change in loss when w_0 and w_1 converge to w^* as I_0 and

I_1, which denote the importance of w_0 and w_1, respectively, and this importance becomes a criterion for redesigning the weight updates for optimizing the centripetal vectors. I_0 and I_1 can be obtained as follows:

$$\begin{aligned} I_0 &= |L(D,W) - L(D, W_{w_0 \to w^*})| \\ &= \left| \frac{\partial L(D,W)}{\partial w_0} \cdot (w_0 - w^*) \right| \end{aligned} \tag{7}$$

$$\begin{aligned} I_1 &= |L(D,W) - L(D, W_{w_1 \to w^*})| \\ &= \left| \frac{\partial L(D,W)}{\partial w_1} \cdot (w_1 - w^*) \right| \end{aligned} \tag{8}$$

In (7) and (8), the gradients related to the loss function (i.e., $\frac{\partial L(D,W)}{\partial w_0}$ and $\frac{\partial L(D,W)}{\partial w_1}$) can be easily computed during the process of back-propagation. In addition, because w^* is the midpoint between w_0 and w_1 (i.e., $w^* = \frac{w_0+w_1}{2}$), I_0 and I_1 can be simplified as follows:

$$I_0 = \left| \frac{\partial L(D,W)}{\partial w_0} \cdot \left(\frac{w_0 - w_1}{2} \right) \right| \tag{9}$$

$$I_1 = \left| \frac{\partial L(D,W)}{\partial w_1} \cdot \left(\frac{w_1 - w_0}{2} \right) \right| \tag{10}$$

In (9) and (10), since $|\frac{w_0-w_1}{2}|$ and $|\frac{w_1-w_0}{2}|$ are equal, only need to compare the absolute values of the gradients (i.e., $|\frac{\partial L(D,W)}{\partial w_0}|$ and $|\frac{\partial L(D,W)}{\partial w_1}|$) to evaluate the importance of the two weights. Existing studies in [42] and [47] estimated the importance as in (4) with high complexity due to the product of weights and gradients, but our method efficiently eliminates this computation overhead because we only compare the absolute values of gradients for evaluating the importance.

For convenience of expression, we use "mask", which has a binary value $\in \{01\}$ that determines whether or not to remove the centripetal vector. Masks, m_0 and m_1 , are determined by comparing the absolute values of the two gradients and can be expressed as follows:

$$m_0 = \begin{cases} 1 & if \left| \frac{\partial L(D,W)}{\partial w_0} \right| < \left| \frac{\partial L(D,W)}{\partial w_1} \right| \\ 0 & otherwise \end{cases} \tag{11}$$

$$m_1 = \begin{cases} 1 & if \left| \frac{\partial L(D,W)}{\partial w_0} \right| > \left| \frac{\partial L(D,W)}{\partial w_1} \right| \\ 0 & otherwise \end{cases} \tag{12}$$

A mask value of 1 means that the corresponding centripetal vector is maintained. It should be noted that m_0 and m_1 always have different values.

Finally, we redesign the update rule of the centripetal vectors as follows:

$$w_0^{(i+1)} \leftarrow w_0^{(i)} - \eta \frac{1}{2} \left(\frac{\partial L}{\partial w_0} + \frac{\partial L}{\partial w_1} \right) + m_0 \varepsilon \left(\frac{w_1 - w_0}{2} \right) \tag{13}$$

$$w_1^{(i+1)} \leftarrow w_1^{(i)} - \eta \frac{1}{2} \left(\frac{\partial L}{\partial w_0} + \frac{\partial L}{\partial w_1} \right) + m_1 \varepsilon \left(\frac{w_0 - w_1}{2} \right) \tag{14}$$

Algorithm 1: FP-AGL Algorithm

Input: Training dataset $\mathcal{D}$, pretrained model $\mathcal{M}$, training iteration $\mathcal{T}$**,** averaged gradient $\boldsymbol{g_{avg}}$

Output: The compact model $\mathcal{M}'$

for $i \leftarrow 1$ to $\mathcal{T}$ **do**
  if $|w_{0est}^{(i+1)} - w_{1est}^{(i+1)}| < |w_0^{(i)} - w_1^{(i)}|$ **then**
    $w_0^{(i+1)} \leftarrow w_0^{(i)} - \eta \frac{\partial L}{\partial w_0^{(i)}}$
    $w_1^{(i+1)} \leftarrow w_1^{(i)} - \eta \frac{\partial L}{\partial w_1^{(i)}}$
  else
    $g_{avg} = \frac{1}{2}(\frac{\partial L}{\partial w_0^{(i)}} + \frac{\partial L}{\partial w_1^{(i)}})$
    if $|\frac{\partial L}{\partial w_0^{(i)}} < \frac{\partial L}{\partial w_1^{(i)}}|$ **then**
      $w_0^{(i+1)} \leftarrow w_0^{(i)} - \eta g_{avg} + m_0 \varepsilon (\frac{w_1 - w_0}{2})$
      $w_1^{(i+1)} \leftarrow w_1^{(i)} - \eta g_{avg}$
    else
      $w_0^{(i+1)} \leftarrow w_0^{(i)} - \eta g_{avg}$
      $w_1^{(i+1)} \leftarrow w_1^{(i)} - \eta g_{avg} + m_1 \varepsilon (\frac{w_0 - w_1}{2})$
    end if
  end if
end for
Prune model $\mathcal{M}$ to $\mathcal{M}'$

In (13) and (14), the second terms on the right side are the averaged gradients that constrain the distance between weights to be constant. The third terms on the right side are the RCVs, which contain the mask values defined in (11) and (12). By using these binary masks with different values at all times, the number of redundant-induced penalties (*i.e.*, centripetal vectors) for pruning is reduced to half compared to C-SGD [28], thereby minimizing the performance loss. The application process of this RCV is well shown in the yellow arrows (*i.e.*, averaged gradient) and red arrows (*i.e.*, RCV) in Fig. 1(c).

C. Adaptive Gradient Learning

The existing C-SGD [28] updates the weight using the sum of the averaged gradient (to maintain the distance between the weights) and the centripetal vector (to make the weights come closer to one another). However, this weight update leads to performance degradation as the normal gradient descent gets deviated. In addition, as mentioned in Section II-C, because centripetal vectors behave similar to L_2-regularization, the difference in the weight values cannot completely converge to zero (*i.e.*, can be close to some extent) due to the nature of L_2-regularization. Therefore, several training iterations are required, thereby increasing the computational time ($= 1^{\text{st}}$ problem). In addition, if the convergence is not complete to a single point, even if the model is fine-tuned after pruning, it may be trapped into bad local minima ($= 2^{\text{nd}}$ problem).

We therefore present the AGL method that can address the two aforementioned problems simultaneously by updating the weights in an adaptive manner, and the application process of FP-AGL is summarized in Algorithm1. The proposed adaptive

update rule is formulated as follows:

$$w_0^{(i+1)} = \begin{cases} w_0^{(i)} - \eta \frac{\partial L}{\partial w_0^{(i)}} \; if \; \left| w_{0est}^{(i+1)} - w_{1est}^{(i+1)} \right| < \left| w_0^{(i)} - w_1^{(i)} \right| \\ w_0^{(i)} - \eta \frac{1}{2} \left(\frac{\partial L}{\partial w_0^{(i)}} + \frac{\partial L}{\partial w_1^{(i)}} \right) + m_0 \varepsilon \left(\frac{w_1 - w_0}{2} \right) \; else \end{cases} \tag{15}$$

$$w_1^{(i+1)} = \begin{cases} w_1^{(i)} - \eta \frac{\partial L}{\partial w_1^{(i)}} \; if \; \left| w_{0est}^{(i+1)} - w_{1est}^{(i+1)} \right| < \left| w_0^{(i)} - w_1^{(i)} \right| \\ w_1^{(i)} - \eta \frac{1}{2} \left(\frac{\partial L}{\partial w_0^{(i)}} + \frac{\partial L}{\partial w_1^{(i)}} \right) + m_0 \varepsilon \left(\frac{w_0 - w_1}{2} \right) \; else \end{cases} \tag{16}$$

where $w_{0est}^{(i+1)}$ and $w_{1est}^{(i+1)}$ are w_0 and w_1 of the *(i+1)-th* iteration, respectively, predicted from the current (i.e., *i-th*) iteration. It should be noted that we can predict the difference between the weight values of the *(i+1)-th* iteration if we know the magnitude and direction of *i-th* gradients obtained through backward propagation and the magnitude of *i-th* weights (see Appendix A for details). In case of $|w_{0est}^{(i+1)} - w_{1est}^{(i+1)}| < |w_0^{(i)} - w_1^{(i)}|$ in (15) and (16), the weights are updated with the original gradient (*i.e.*, lines 2-3 in Algorithm 1). In other words, when the estimated distance between w_0 and w_1 in the *(i+1)-th* training iteration is less than that in the *i-th* training iteration, the distance between the two weights can be reduced; this coincides with our objective of bringing the two weights closer to one another. Because the weight update under these conditions uses the original gradients, the loss-function can be optimized in the normal gradient descent direction, thereby maintaining the performance of the original model. However, if the distance between w_0 and w_1 in the *(i+1)-th* training iteration is larger than that in the *i-th* training iteration (*i.e.*, line 4 in Algorithm 1), the distance between the two weights increases. In this case, the weights are updated by using the RCVs proposed in Section III-B. As the proposed AGL method maintains the original gradients by updating the weights, it can reach the optimal local-minimum faster than the case where the weights by are updated only using the averaged gradient in C-SGD [28], thereby enabling faster and more accurate filter pruning.

IV. Experimental Results

A. Implementation Details

CIFAR-10: The CIF AR-10 [48] dataset consists of 32 × 32 50000 training images and 10000 test images categorized into 10 classes. The baseline model is trained with a batch size of 64 for 300 epochs (≈240000 training steps) using SGD from scratch. The initial learning rate is set to 0.1, and is divided by 10 at 50% and 75% of the total training steps. To improve the convergence speed and training performance, a Nesterov momentum of 0.9, and a weight decay of 10^{-4} are used. Training using the proposed FP-AGL is set with an initial learning rate of 5×10^{-2} and centripetal strength [28] of 5×10^{-4}. The rest of the hyper-parameter settings are same as the baseline.

ILSVRC-2012: The ImageNet ILSVRC-2012 [46] dataset consists of 1.28 million training images and 50000 test images categorized into 1000 classes. The baseline model is trained with a batch size of 128 for 80 epochs (≈800000 training steps) using SGD from scratch. The initial learning rate is set to 0.1, and this rate is divided by 10 at 50% and 75% of the total training steps. To improve the convergence speed and training performance, a Nesterov momentum of 0.9, and a weight decay of 10^{-4} are used. The hyper-parameter settings during training using the proposed FP-AGL is same as that for CIFAR-10 [48].

B. Experimental Results on CIFAR-10

In Table I, we compare the performance of the proposed method with those of other methods using ResNet-56 [1], a representative network used for image classification tasks, on the CIFAR-10 dataset [48]. In all the cases for which the results are presented, "Ours" is the result obtained by performing one-shot pruning without fine-tuning after FP-AGL training with sufficient iteration. "Acc. (%)" and "Pruned Acc. (%)" refer to top-1 Accuracy of the baseline model and pruned model, respectively. "Drop (%)" and "Pruned FLOPs (%)" denote the accuracy degradation and FLOPs reduction of the pruned model as compared to the baseline, respectively. It should be noted that we present the baseline accuracy results of the comparative studies as they are because the ResNet baseline accuracy presented in each study varies due to different environments of each study (e.g., experimental setting, implementation language, etc.). Accordingly, in order to perform a fair comparison, we conduct the performance comparison in terms of trade-off between FLOPs reduction (Pruned FLOPs (%)) and relative accuracy drop (Drop (%)) rather than absolute accuracy (i.e., Pruned Acc. (%)). ResNet-56 consists of three stages of residual blocks (16-32-64), and each stage has 16, 32, and 64 filters. As presented in Table I, Ours(11-22-44) means that the number of filters remaining in each stage of the pruned ResNet-56 using FP-AGL is 11, 22, and 44 after the pruning. This notation applies equally to Ours(13-26-52) and Ours(10-20-40).

Experimental results show that our proposed ResNet-56 achieves better performance than that of the other pruning methods except for C-SGD [28]. In detail, Ours(13-26-52) is superior to Variational [30], PF [22], and LEGR [34], which achieved 20%∼30% of FLOPs reduction, in terms of the trade-off between FLOPs and accuracy. LEGR [34] achieved the SOTA performance by enhancing the accuracy by 0.2% with 30% FLOPs reduction, whereas Ours(13-26-52) enhanced the top-1 accuracy by 0.4% even over the baseline when pruning 33.72% of the FLOPs. In addition, Ours(11-22-44), proposed to achieve a 50% level of FLOPs reduction, enhances the top-1 accuracy by 0.1% over the baseline even when pruning 52.49% FLOPs. Compared to the performance of AMC [37], this can be considered to be an outstanding result showing a large margin of 1% level (*i.e.*, +0.1% vs. -0.9%) in an accuracy drop even though Ours(11-22-44) further reduces FLOPs by 2.49% compared to AMC [37]. When comparing the proposed FP-AGL with the existing SOTA studies with a pre-defined pruning ratio, SFP [24], FPGM [25], and LFPC [36], Ours(11-22-44) has a lower accuracy drop at similar levels of pruned FLOPs. These results

TABLE I
PERFORMANCE COMPARISON OF THE PRUNED RESNET-56 ON THE CIFAR-10 DATASET

Network	Method	Acc. (%)	Pruned Acc. (%)	Drop (%)	Pruned FLOPs↓ (%)
ResNet-56	Variational [30]	93.04	92.26	-0.78	20.49
	PFEC [22]	93.04	93.06	+0.02	27.6
	LEGR [34]	93.9	94.1	+0.2	30
	Ours(13-26-52)	**93.59**	**93.99**	**+0.4**	**33.72**
	GhostNet [64]	93.0	92.7	-0.3	49.6
	AMC [37]	92.8	91.9	-0.9	50
	Hrank [33]	93.26	93.17	-0.09	50
	DMC [32]	93.62	93.69	+0.07	50
	SCP [38]	93.69	93.23	-0.46	51.5
	Ours(11-22-44)	**93.59**	**93.69**	**+0.1**	**52.49**
	FPGM [25]	93.59	93.49	-0.1	52.6
	SFP [24]	93.59	93.35	-0.24	52.6
	LFPC [36]	93.59	93.24	-0.35	52.9
	Knapsack [60]	94.5	93.83	-0.69	53.8
	ABCPruner [35]	93.26	93.23	-0.03	54.13
	KSE [63]	93.03	92.88	-0.15	60
	C-SGD [28]	93.39	93.44	+0.05	60.85
	Ours(10-20-40)	**93.59**	**93.49**	**-0.1**	**60.92**

TABLE II
PERFORMANCE COMPARISON OF THE PRUNED RESNET ON THE IMAGENET ILSVRC-12 DATASET

Network	Method	Acc. (%)	Pruned Acc. (%)	Drop (%)	Pruned FLOPs↓ (%)
ResNet-34	Taylor-FO [29]	73.31	72.83	-0.48	22.25
	FPGM [25]	73.92	72.63	-1.29	41.1
	Ours	**73.02**	**72.72**	**-0.3**	**43.14**
	DMC [32]	73.30	72.57	-0.73	43.4
ResNet-50	FPGM [25]	76.15	75.59	-0.56	42.2
	Ours	**75.92**	**75.8**	**-0.12**	**43.18**
	Hrank [33]	76.15	74.98	-1.17	43.77
	C-SGD [28]	75.33	74.93	-0.4	46.24
	GhostNet [64]	75.3	75.0	-0.3	46.34
	LEGR [34]	76.1	75.3	-0.8	47
	Hinge [59]	76.15	75.70	-0.45	50
	Knapsack [60]	78.47	77.8	-0.67	50.21
	Ours*	**75.92**	**75.77**	**-0.15**	**52.09**
	FPGM [25]	76.15	74.83	-1.32	53.5
	Ours	**75.92**	**75.28**	**-0.64**	**54.91**
	C-SGD [28]	75.33	74.54	-0.79	55.76
	ABCPruner [35]	76.01	73.52	-2.49	56.01
	Ours	**75.92**	**74.82**	**-1.1**	**60.23**
	LFPC [36]	76.15	74.46	-1.69	60.8
	Hrank [33]	76.15	71.98	-4.17	62.1
	Ours*	**75.92**	**74.94**	**-0.98**	**62.15**
ResNet-101	Taylor-FO [29]	77.37	77.35	-0.02	39.74
	FPGM [25]	77.37	77.32	-0.05	42.2
	Ours	**77.16**	**77.40**	**+0.24**	**43.45**

show that our method is the most efficient under similar network sizes. Although Ours(10-20-40) causes an accuracy drop of 0.15% more than C-SGD [28], we show the superior performance of FP-AGL compared to C-SGD [28] in the various experiments that follow (i.e., Tables II, IV, V, VI, VII, and VII). In addition, it should be noted that the results of 'Ours' presented in Table I are obtained by the one-shot pruning method, whereas the result of C-SGD presented in [28] was obtained through the iterative pruning method.

C. Experimental Results on ILSVRC-2012

We compared the proposed FP-AGL and previous studies on the ILSVRC-2012 dataset [46] using ResNet-34, 50, and 101 [1], which are frequently used as backbones in object detection and segmentation. As presented in Table II, our method shows SOTA performance under various pruned FLOPs. Because the existing studies on filter pruning can be divided into two categories: ones that adopt the iterative method and others that

adopt the one-shot method, we evaluated our method using both these schemes and have presented our results accordingly. To be more specific, we used "Ours∗" and "Ours" as the iterative and one-shot pruning schemes, respectively. As in Table I, "Acc. (%)" and "Pruned Acc. (%)" refer to the top-1 Accuracy of the baseline model and pruned model, respectively, and "Drop (%)" and "Pruned FLOPs (%)" denote the accuracy degradation and FLOPs reduction of the pruned model as compared to the baseline, respectively. We compare the performance of our proposed method (*i.e.*, "Ours") with that of Taylor-FO [29], FPGM [25], and DMC [32] in ResNet-34. Among them, in comparison with DMC [32], which has the most similar FLOPs reduction, the proposed method mitigated the accuracy drop by 0.43%. Compared to Taylor-FO [29], even though much more FLOPs of 20.89% are reduced (*i.e.*, 43.14% *vs.* 22.25%), the accuracy drop is rather 0.18% lower (*i.e.*, -0.3% *vs.* -0.48%).

For ResNet-50, both of our FP-AGL schemes (*i.e.*, "Ours" and "Ours∗") show better performance in all the cases when compared to the methods that pruned 40%–50% of the FLOPs. In particular, compared to C-SGD [28], which inspires our method and showed the best performance among the existing studies, the proposed method with the iterative scheme (*i.e.*, "Ours∗") shows a higher FLOP reduction (*i.e.*, 52.09% *vs.* 43.18%) as well as a lower accuracy drop (*i.e.*, -0.15% *vs.* -0.4%). This shows that the proposed method well compensates for the disadvantages of C-SGD. C-SGD also reduced FLOPS by 55.76% with an accuracy drop of 0.79%, but the proposed method with one-shot pruning (*i.e.*, "Ours") can achieve a FLOPs reduction of 54.91% with an accuracy drop of 0.64%, which in turn indicates that the proposed FP-AGL can mitigate the accuracy drop better than C-SGD in an almost similar FLOPs reduction. It should be noted that C-SGD performed iteratively pruning through more epochs to obtain 55.76% of FLOPs reduction, whereas the newly added experimental result of FP-AGL (i.e., 54.91% of FLOPs reduction) is achieved with one-shot pruning. "Ours" with 60.23% of FLOPs reduction also shows a 0.59% lower accuracy drop than the SOTA one-shot pruning method, LFPC [36], in achieving a similar level of FLOPs reduction (*i.e.*, approximately 60%).

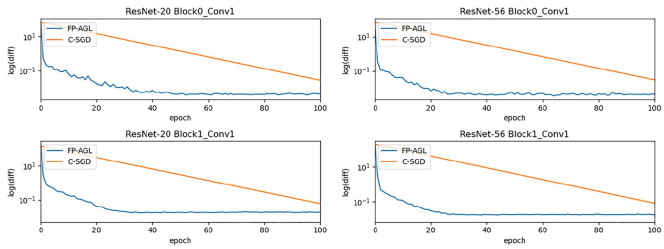
Even for ResNet-101, "Ours" improved the accuracy by 0.24% over the baseline, although the FLOPs reduction was 3.71% and 1.25% higher than those of Taylor-FO [29] and FPGM [25], respectively. These results show that the proposed FP-AGL is effective even when an aggressive one-shot pruning scheme is being used. In addition, we confirm that the FP-AGL has high scalability and compatibility, achieving generally good performance on complicated networks such as ResNet-34, 50, and 101.

D. Performance Analysis By Each Scheme (Ablation Study)

In order to analyze the impact of each algorithm of FP-AGL (*i.e.*, RCV and AGL), we verify the step-by-step performance on the CIFAR-10 dataset [48] using ResNet-20, 32, 56, and 164 [1]. Table III shows the accuracy (*i.e.*, Acc. (%)) for each algorithm at the FLOPs reduction of approximately 60%. All results are obtained by performing one-shot pruning without fine-tuning. Experimental results show that each of RCV and AGL achieves

TABLE III
PERFORMANCE EVALUATION ACCORDING TO EACH PROPOSED SCHEME

Network	Baseline (%)	RCV	AGL	Acc.(%)	Drop (%)	Diff. (%)	Pruned FLOPs↓ (%)
ResNet-20	91.98			89.14 (Scratch)	2.84	-	60.66
		✓		89.86	2.12	0.72	
			✓	89.70	2.28	0.56	
		✓	✓	90.11	1.87	0.97	
ResNet-32	93.12			90.75 (Scratch)	2.37	-	60.76
		✓		91.41	1.71	0.66	
			✓	91.61	1.51	0.86	
		✓	✓	91.86	1.26	1.11	
ResNet-56	93.59			92.85 (Scratch)	0.74	-	60.92
		✓		93.32	0.27	0.47	
			✓	93.45	0.14	0.6	
		✓	✓	93.49	0.1	0.64	
ResNet-164	94.28			93.51 (Scratch)	0.77	-	60.89
		✓		93.72	0.56	0.21	
			✓	93.93	0.35	0.42	
		✓	✓	94.11	0.17	0.6	



(a) (b)

Fig. 2. Comparison of the process of convergence to a single point in FP-AGL and C-SGD on (a) ResNet-20 and (b) ResNet-56. The "diff" is the summation of the differences between convolution layer filters in clusters.

the effect of mitigating a significant level of accuracy drop, and the smallest accuracy drop can be achieved in the same FLOPs reduction when using both AGL and RCV. Although the difference in performance when AGL and RCV+AGL is applied in ResNet-56 is relatively small, AGL and RCV individually have a significant effect on performance improvement in the remaining networks except for ResNet-56 (i.e., ResNet-20, 32, and 164). It can be seen that the proposed methods are dependent on the network structure. These results show that even in the case of one-shot pruning, each of the two proposed techniques helps to reach the optimal local optima, and these two techniques are well compatible to achieve optimal pruning performance.

In order to prove that the proposed AGL enables fast convergence of filters to a single point, we compare the process in which the filters converge to a single point in both the FP-AGL and C-SGD methods. As can be seen in Fig. 2, which shows the convergence process of filters in the block0_conv1 and blcok1_conv1 layers of ResNet-20 and ResNet-56 on Cifar-10 dataset, the use of the proposed AGL makes the convergence

TABLE IV
PERFORMANCE COMPARISON IN OTHER ACTIVATION FUNCTIONS EXCEPT RELU

Network / Dataset	Method	Activation	FT?	Acc. (%)	Drop (%)	Pruned FLOPs↓(%)
ResNet20 / CIFAR-10	Baseline	Mish	-	92.42	-	-
	Slimming [23]	Mish	✓	90.76	-1.66	32.93
	C-SGD [28]	Mish	×	91.26	-1.16	34.15
	FP-AGL	Mish	×	**91.69**	**-0.73**	**34.15**
DarkNet53 / IMAGENET	Baseline	Leaky	-	77.17	-	-
	Slimming [23]	Leaky	✓	70.63	-6.54	32.04
	C-SGD [28]	Leaky	×	76.54	-0.63	33.89
	FP-AGL	**Leaky**	×	**76.87**	**-0.3**	**33.89**

speed of filters approximately 5.3 times and 8.3 times faster than C-SGD, respectively.

E. Compatibility With Other Activation Functions

Since ReLU makes negative values zero, it automatically induces sparsity in activations. For this reason, most of the filter pruning methods were targeted only for networks using ReLU (*e.g.*, VGG, ResNet, and DenseNet) [38], [49]–[51]. However, recently, networks that use activation functions other than ReLU (*i.e.*, Leaky ReLU or Mish [52]) are increasing, and these activation functions cannot maintain the advantage of sparsity. Therefore, it is very important to verify the performance of pruning in various activation functions other than ReLU. In Table IV, we compare and verify the performance of the proposed technique and previous studies, Slimming [23] and C-SGD [28], in ResNet-20 [1] and DarkNet-53 [7], which use Mish and Leaky ReLU as activation functions, respectively. The value of the penalty factor of previous studies is set to 5×10^{-2}. It should be noted that all C-SGD [28] results presented hereafter are the results of one-shot pruning after training C-SGD using evenly clustering in the same training setting (i.e., training epochs, optimizer, learning rate, etc.) as the proposed FP-AGL. In Table IV, "FT" denotes whether the pruned network is fine-tuned or not. FP-AGL shows higher accuracy than Slimming [23] and C-SGD [28] in similar FLOPs reduction. The large accuracy drops of Slimming [23] in various activation functions occur because L1-regularization is applied only to the scaling factor of batch normalization, and pruning is not complete due to the remaining bias ($\neq 0$). On the other hand, C-SGD [28] and FP-AGL, which make the filters equal by considering the bias, have better performance than Slimming [23] and do not require fine-tuning. In addition, as described in Section III, FP-AGL can reach better local minima by using both of RCV that reduces the number of penalty vectors by half and AGL that updates weights while maintaining the original gradient as much as possible. As a result, FP-AGL can achieve better performance than C-SGD [28] for various activation functions.

F. Performance Evaluation on Lightweight Networks

In Table V, we conduct additional experiments on MobileNet-V1 [2] and MobileNet-V2 [66], which are difficult to prune because they are already sufficiently lightweight. We compare the top-1 accuracy drop under the same FLOPs constraint as the previous experiments. Experimental results in Table V show that the proposed FP-AGL achieves significantly less accuracy drop than C-SGD [28] in various FLOPs reduction ratios. It can also be observed that the difference in accuracy drop between the two methods (i.e., FP-AGL and C-SGD) increases as the FLOPs reduction ratio increases. As a result, these experimental results show that the proposed method is a more effective pruning method even at the extreme FLOPs reduction ratio.

TABLE V
PERFORMANCE COMPARISON WITH C-SGD [28] USING MOBILENET-V1 AND MOBILENET-V2 ON THE CIFAR-10 DATASET

Network	Method	Base Acc. (%)	Pruned Acc. (%)	Acc. Drop (%)	Pruned FLOPs↓(%)
MobileNet -V1	C-SGD [28]	90.44	88.55	-1.89	60.2
	FP-AGL		**99.12**	**-1.32**	**60.2**
	C-SGD [28]		89.49	-0.95	43.0
	FP-AGL		**89.88**	**-0.56**	**43.0**
MobileNet -V2	C-SGD [28]	93.13	91.68	-1.45	59.3
	FP-AGL		**92.16**	**-0.97**	**59.3**
	C-SGD [28]		92.55	-0.58	42.3
	FP-AGL		**92.94**	**-0.19**	**42.3**

In addition, to measure the accuracy of C-SGD and FP-AGL at various centripetal strengths (i.e., 0.1, 0.15, 0.2, 0.25, and 0.3) for a more even comparison, we conduct additional experiments using MobileNet-V1 with a FLOPs reduction of 60.2% on the Cifar-10 dataset. As can be seen in Fig. 3, the proposed method achieves better accuracy than C-SGD on all centripetal strengths of C-SGD. These experimental results show that the proposed method has the additional advantage of reducing significant training cost because it does not require any hyper-parameter setting, unlike C-SGD, which requires multiple training processes to find the optimal hyper-parameter (i.e., centripetal strength).

G. Scalability With Object Detection and Semantic Segmentation

We also tested the scalability of the proposed FP-AGL with object detection and semantic segmentation tasks by using transfer learning. The experiment for object detection is performed on the PASCAL VOC dataset [53] with 20 classes, and YOLOv3 [7], a representative one-stage object detection model. Concretely speaking, as YOLOv3 uses Darknet-53 as the backbone,

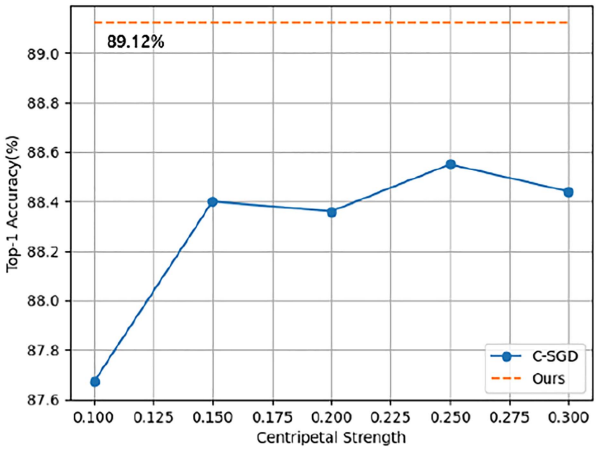


Fig. 3. Top-1 accuracy (%) of MobileNet-V1 on the CIFAR-10 dataset according to various centripetal strengths.

TABLE VI
TRANSFER LEARNING RESULTS WITH YOLOV3 ON PASCAL VOC 2007

Network	Size	Backbone FLOPs↓(%)	mAP@0.5 (%)	FPS
Unpruned (Backbone: Original DarkNet-53)	416	-	79.68	75
Pruned FP-AGL (Backbone: Pruned DarkNet-53)		**42.87**	**79.49**	**112**
Pruned C-SGD [28] (Backbone: Pruned DarkNet-53)		42.87	78.9	112

we prune all layers except the first layer of this Darknet-53 on ImageNet [46] using the proposed FP-AGL and C-SGD [28]. The hyper parameter settings of the training are the same as those described in Section IV-A. The pruned DarkNet-53 with the proposed FP-AGL and C-SGD shows 0.3% and 0.75% lower accuracy, respectively, as compared to the baseline, on the 42.87% of FLOPs reduction. Next, we train YOLOv3 with these pruned DarkNet-53 networks on the PASCAL VOC 2007+2012 training set. As presented in Table VI, the proposed method achieves a 24% of FPS improvement with only a 0.19% drop in mean average precision (mAP) on the VOC metric (IoU = .5) compared to the baseline (i.e., 'Unpruned'), and consequently achieves better performance than C-SGD by 0.59% of mAP.

In a similar way to the object detection model, semantic segmentation models are constructed by combining the FCN module [55], which consists only of a convolution layer without a fully connected layer, with Darknet-53 as a backbone, and we perform fine-tuning on all combined models (i.e., 'Unpruned', 'Pruned FP-AGL', and 'Pruned C-SGD' models). As the training setting, all of the original model and the pruned models on the SBD dataset [65] use a fixed learning rate of $1.0e^{-10}$ for 30 epochs. As shown in Table VII, the pruned segmentation model with the proposed FP-AGL achieves an accuracy improvement of 0.18% compared to the baseline in mean intersection over union (mIOU), which represents the accuracy of segmentation, and outperforms the pruned model with C-SGD by 0.4% of mIOU. In addition, the pruned segmentation model can improve FPS by 33.3% through pruning.

TABLE VII
TRANSFER LEARNING RESULTS WITH THE SEMANTIC SEGMENTATION TASK ON THE SBD DATASET

Network	Backbone FLOPs↓(%)	mIoU (%)	FPS
Unpruned (Backbone: Original DarkNet-53)	-	54.12	27
Pruned FP-AGL (Backbone: Pruned DarkNet-53)	**42.87**	**54.3**	**36**
Pruned C-SGD [28] (Backbone: Pruned DarkNet-53)	42.87	53.9	36

TABLE VIII
COMPARISON OF SCRATCH TRAINING PERFORMANCE OF THREE METHODS (NORMAL SGD, C-SGD, AND FP-AGL) ON THE CIFAR-10 DATASET

Model	Normal SGD (%)	C-SGD (%)	FP-AGL (%)
Res-20 (10-20-40)	89.14	89.48	89.75
Res-32 (10-20-40)	90.75	91.19	91.34
VGG * 1/8	81.85	82.51	82.75

Considering that the real-time object detection model YOLOv3 and semantic segmentation model are quite vulnerable to light-weighting techniques such as quantization and pruning, these results demonstrate the superiority of the proposed FP-AGL.

H. Scratch Training Performance

We conduct additional experiments to verify the scratch training performance of FP-AGL on the ResNet and VGG networks, and the results are presented in Table VIII. In Table VIII, Res (10-20-40) denotes the remaining structure after pruning and scratch training of the original ResNet structure (16-32-64) (i.e., each stage has 16, 32, and 64 filters) by each method. VGG $*$ 1/8 indicates that the original width of VGG is slimmed by 1/8. Experimental settings are the same as those presented in Section IV-A, and one-shot pruning is used for clear performance comparison. Experimental results show that FP-AGL achieves the highest accuracy in all networks, and consequently, it can be seen that the proposed FP-AGL is a more efficient training method than normal SGD and C-SGD [28] even in scratch training.

V. CONCLUSION

Although various studies on filter pruning are being conducted, the development of superior filter pruning methods in terms of the trade-off between performance and network complexity remains a key issue because deeper CNNs are continuously being proposed in accordance with the trend of constantly

pursuing better performance. In this paper, based on the analysis of the problems of the existing C-SGD, we proposed the FP-AGL to effectively reduce the model size without performance degradation of CNNs. The proposed method redesigns the centripetal vector in a loss-aware manner using Taylor expansion, and adaptively applies the centripetal vectors to minimize the distortion of the original gradients and to enable fast convergence of weights to a single point. Experiments using various benchmark datasets and networks demonstrated that the proposed FP-AGL is the most efficient pruning method achieving a performance better than those of the SOTA methods. In conclusion, the proposed FP-AGL is expected to provide great support when utilizing CNNs in mobile platforms, thereby accelerating the commercialization of deep learning algorithms.

APPENDIX A

This appendix presents the method to predict the difference between the weight values of the *(i+1)-th* iteration by using the magnitude and direction of *i-th* gradients obtained through backward propagation and the magnitude of *i-th* weights.

$|w_{0est}^{(i+1)} - w_{1est}^{(i+1)}|$ is always smaller than $|w_0^{(i)} - w_1^{(i)}|$ when

Case ①: $(w_0^{(i)} < w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} < 0$ and $\frac{\partial L}{\partial w_1^{(i)}} > 0$

Case ②: $(w_0^{(i)} < w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} < 0$ and $\frac{\partial L}{\partial w_1^{(i)}} < 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| > |\frac{\partial L}{\partial w_1^{(i)}}|$

Case ③: $(w_0^{(i)} < w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} > 0$ and $\frac{\partial L}{\partial w_1^{(i)}} > 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| < |\frac{\partial L}{\partial w_1^{(i)}}|$

Case ④: $(w_0^{(i)} \geq w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} > 0$ and $\frac{\partial L}{\partial w_1^{(i)}} < 0$

Case ⑤: $(w_0^{(i)} \geq w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} < 0$ and $\frac{\partial L}{\partial w_1^{(i)}} < 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| < |\frac{\partial L}{\partial w_1^{(i)}}|$

Case ⑥: $(w_0^{(i)} \geq w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} > 0$ and $\frac{\partial L}{\partial w_1^{(i)}} > 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| > |\frac{\partial L}{\partial w_1^{(i)}}|$

Otherwise, $|w_{0est}^{(i+1)} - w_{1est}^{(i+1)}|$ is always greater than of equal to $|w_0^{(i)} - w_1^{(i)}|$ when

Case ⑦: $(w_0^{(i)} < w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} > 0$ and $\frac{\partial L}{\partial w_1^{(i)}} < 0$

Case ⑧: $(w_0^{(i)} < w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} < 0$ and $\frac{\partial L}{\partial w_1^{(i)}} < 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| < |\frac{\partial L}{\partial w_1^{(i)}}|$

Case ⑨: $(w_0^{(i)} < w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} > 0$ and $\frac{\partial L}{\partial w_1^{(i)}} > 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| > |\frac{\partial L}{\partial w_1^{(i)}}|$

Case ⑩: $(w_0^{(i)} \geq w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} < 0$ and $\frac{\partial L}{\partial w_1^{(i)}} > 0$

Case ⑪: $(w_0^{(i)} \geq w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} < 0$ and $\frac{\partial L}{\partial w_1^{(i)}} < 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| > |\frac{\partial L}{\partial w_1^{(i)}}|$

Case ⑫: $(w_0^{(i)} \geq w_1^{(i)})$ and $\frac{\partial L}{\partial w_0^{(i)}} > 0$ and $\frac{\partial L}{\partial w_1^{(i)}} > 0$ and $|\frac{\partial L}{\partial w_0^{(i)}}| < |\frac{\partial L}{\partial w_1^{(i)}}|$

Please note that all notations in the Appendix are the same as those presented in the manuscript.

REFERENCES

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

[2] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[3] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[4] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 6848–6856.

[5] J. Choi, D. Chun, H. Kim, and H.-J. Lee, "Gaussian YOLOv3: An accurate and fast object detector using localization uncertainty for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 502–511.

[6] J. Choi, D. Chun, H.-J. Lee, and H. Kim, "Uncertainty-Based object detector for autonomous driving embedded platforms," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst.*, 2020, pp. 16–20.

[7] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*.

[8] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10781–10790.

[9] D. Bolya, C. Zhou, F. Xiao, and Y. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9157–9166.

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[11] D. T. Nguyen, H. Kim, H.-J. Lee, and I.-J. Chang, "An approximate memory architecture for a reduction of refresh power consumption in deep learning applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2018, pp. 1–5.

[12] D. T. Nguyen, T. N. Nguyen, H. Kim, and H.-J. Lee, "A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 27, no. 8, pp. 1861–1873, Aug. 2019.

[13] Y. Xu, W. Dai, Y. Qi, J. Zou, and H. Xiong, "Iterative deep neural network quantization with Lipschitz constraint," *IEEE Trans. Multimedia*, vol. 22, no. 7, pp. 1874–1888, Jul. 2020.

[14] Z. Wang, W. Hong, Y. Tan, and J. Yuan, "Pruning 3D filters for accelerating 3D ConvNets," *IEEE Trans. Multimedia*, vol. 22, no. 8, pp. 2126–2137, Aug. 2020.

[15] S. Kim and H. Kim, "Zero-Centered fixed-point quantization with iterative retraining for deep convolutional neural network-based object detectors," *IEEE Access*, vol. 9, pp. 20828–20839, 2021.

[16] B. Hassibi, D. G. Stork, and G. Wolf, "Optimal brain surgeon: Extensions and performance comparisons," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 263–270.

[17] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[19] X. Ding et al., "Global sparse momentum SGD for pruning very deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6379–6391.

[20] S. Han et al., "DSD: Dense-sparse-dense training for deep neural networks," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[21] A. Morcos, H. Yu, M. Paganini, and Y. Tian, "One ticket to win them all: Generalizing lottery ticket initializations across datasets and optimizers," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 4933–4943.

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[23] Z. Liu et al., "Learning efficient convolution networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2736–2744.

[24] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2234–2240.

[25] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. comput. Vis. Pattern Recognit.*, 2019, pp. 4340–4349.
[26] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *Proc. 6th Int. Conf. Learn. Representations*, 2018.
[27] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C. Xu, "Dynamic channel pruning: Feature boosting and suppression," in *Proc. 7th Int. Conf. Learn. Representations*, 2019.
[28] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for pruning very deep convolutional networks with complicated structure," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4943–4953.
[29] 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.*, 2019, pp. 11264–11272.
[30] C. Zhao et al., "Variational convolutional neural network pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2780–2789.
[31] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *J. Mach. Learn. Res.*, vol. 23, no. 3, pp. 1139–1147, 2013.
[32] S. Gao, F. Huang, J. Pei, and H. Huang, "Discrete model compression with resource constraint for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1899–1908.
[33] M. Lin et al., "HRank: Filter pruning using high-rank feature map," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1529–1538.
[34] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1518–1528.
[35] M. Lin et al., "Channel pruning via automatic structure search," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, 2020, pp. 673–679.
[36] Y. He et al., "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2009–2018.
[37] Y. He et al., "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 784–800.
[38] M. Kang and B. Han, "Operation-Aware soft channel pruning using differentiable masks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5122–5131.
[39] N. Kim and H. Kim, "Mask-Soft filter pruning for lightweight CNN inference," in *Proc. IEEE 17th Int. SoC Des. Conf.*, 2020, pp. 316–317.
[40] Z. Liu et al., "MetaPruning: Meta learning for automatic neural network channel pruning," in *Proc. IEEE/CVF Conf. comput. Vis. Pattern Recognit.*, 2019, pp. 3296–3305.
[41] X. Ding et al., "Lossless CNN channel pruning via gradient resetting and convolutional Re-parameterization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 4510–4520.
[42] J. Shi, J. Xu, K. Tasaka, and Z. Chen, "SASL: Saliency-adaptive sparsity learning for neural network acceleration," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 5, pp. 2008–2019, May 2021.
[43] X. Ding, G. Ding, Y. Guo, J. Han, and C. Yan, "Approximated oracle filter pruning for destructive CNN width optimization," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1607–1616.
[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, pp. 448–456, 2015.
[45] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 598–605.
[46] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[47] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 2130–2141.
[48] A. Krizhevsky, and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Univ. Toronto, Toronto, ON, Canada, 2009.
[49] C.-Y. Wang et al., "CSPNet: A new backbone that can enhance learning capability of CNN," in *Proc. IEEE Conf. comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 1571–1580.
[50] H. Hu, R. Peng, Y. W. Tai, and C. K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*.
[51] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. 5th Int. Conf. Learn. Representations*, 2017.
[52] D. Misra, "Mish: A self regularized non-monotonic activation function," in *Proc. 31st British Mach. Vis. Conf.*, 2020.
[53] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge results," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2007.
[54] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proc. 8th Int. Conf. Learn. Representations*, 2020.
[55] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[56] X. Ding, Y. Guo, G. Ding, and J. Han, "ACNet: Strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1911–1920.
[57] X. Ding, G. Ding, J. Han, and S. Tang, "Auto-balanced filter pruning for efficient convolutional neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 32, pp. 6797–6804.
[58] G. Ding, S. Zhang, Z. Jia, J. Zhong, and J. Han, "Where to prune: Using LSTM to guide data-dependent soft pruning," *IEEE Trans. Image Process.*, vol. 30, pp. 293–304, 2021.
[59] Y. Li, S. Gu, C. Mayer, L. V. Gool, and R. Timofte, "Group sparsity: The hinge between filter pruning and decomposition for network compression," in *Proc. IEEE Conf. comput. Vis. Pattern Recognit.*, 2020, pp. 8018–8027.
[60] Y. Aflalo, A. Noy, M. Lin, I. Friedman, and L. Zelnik, "Knapsack pruning with inner distillation," 2020, *arXiv:2002.08258*.
[61] W. Ahmed, A. Zunino, P. Morerio, and V. Murino, "Compact cnn structure learning by knowledge distillation," in *Proc. IEEE Int. Conf. Pattern Recognit.*, 2021, pp. 6554–6561.
[62] X. Chen, J. Zhu, J. Jiang, and C.-Y. Tsui, "Tight compression: Compressing CNN through fine-grained pruning and weight permutation for efficient implementation," *IEEE Trans. Comput.-Aided Design Integrated Circuits Syst.*, to be published, doi: 10.1109/TCAD.2022.3178047.
[63] Y. Li et al., "Exploiting kernel sparsity and entropy for interpretable CNN compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2800–2809.
[64] K. Han et al., "GhostNet: More features from cheap operations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1580–1589.
[65] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 991–998.
[66] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. -C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

Nam Joon Kim (Student Member, IEEE) received the B.S. and M.S. degrees in electrical and information engineering from the Seoul National University of Science and Technology, Seoul, Korea, in 2019 and 2021, respectively, where he is currently working toward the Ph.D. degree in electrical and information engineering. His research interests include network pruning, quantization, and efficient network design for deep neural networks.

Hyun Kim (Member, IEEE) received the B.S., M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, Korea, in 2009, 2011, and 2015, respectively. From 2015 to 2018, he was with the BK21 Creative Research Engineer Development for IT, Seoul National University, as a BK Assistant Professor. In 2018, he joined the Department of Electrical and Information Engineering, Seoul National University of Science and Technology, where he is currently an Assistant Professor. His research interests include algorithm, computer architecture, memory system design, and digital system (SoC) design for low-complexity multimedia applications and deep neural networks.