IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 22 August 2023, accepted 12 November 2023, date of publication 16 November 2023,
date of current version 22 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3333902

RESEARCH ARTICLE

Multi-Step Training Framework Using Sparsity Training for Efficient Utilization of Accumulated New Data in Convolutional Neural Networks

JEONG JUN LEE, (Graduate Student Member, IEEE), AND HYUN KIM, (Senior Member, IEEE)

Department of Electrical and Information Engineering, Research Center for Electrical and Information Technology, Seoul National University of Science and Technology, Seoul 01811, South Korea

Corresponding author: Hyun Kim (hyunkim@seoultech.ac.kr)

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Korea Government (MSIT) (Development of Self-Learnable Mobile Recursive Neural Network Processor Technology) under Grant 2020-0-01304, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A6A1A03032119.

ABSTRACT With the accumulation of large amounts of data that have the most significant impact on network performance and the development of hardware accelerators that can support such data processing, convolutional neural networks (CNNs) have achieved remarkable advances in various applications such as computer vision. In particular, in recent years, various new data suitable for user environments have been obtained from numerous mobile devices, and accordingly, methods have to be developed to train CNNs with only new data on the existing pre-trained network without access to the data used for the pre-training process on server platforms. Nevertheless, only a few studies have considered efficient training schemes for new data. In this study, we propose a multi-step training framework that efficiently utilizes accumulated new data for CNN training. In detail, to significantly improve the performance of CNNs, the proposed method creates unimportant filters while preserving knowledge of previous data through selective sparsity training and can effectively utilize these filters for CNN training of new data through diverse re-initialization and adaptive online distillation techniques. In addition, the proposed multi-step sparsity training with multi-step loss enables iterative network training of new data. The results of extensive experiments performed on various datasets show that the proposed method outperforms many existing methods, including fine-tuning.

INDEX TERMS Convolutional neural network (CNN), continual learning, sparsity training, knowledge distillation.

I. INTRODUCTION

With the development of GPUs and hardware accelerators [1], [2], convolutional neural networks (CNN) have shown remarkable success in computer vision tasks such as classification [3], [4], object detection [5], [6], [7], and segmentation [8], [9], [10]. However, for some applications that demand very few errors, such as autonomous driving and medical image analysis, further performance improvements are still required, and considerable research efforts are being invested [11], [12], [13]. Although various approaches have been examined to improve performance, increasing the amount of training data is the most fundamental approach. The amount of training data has the most direct influence on the performance of networks regardless of the field (e.g., image, sound, language, etc.) [14], [15].

Recently, as more people use mobile devices, various new data that are suitable for user environments are being acquired from these devices. If these data could be used for network training, it is possible to secure a network optimized for data distribution specific to each user's environment, away from the general data distribution. However, since the data acquired from these mobile devices is unlabeled, the task of labeling a large amount of unlabeled data involves

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

a high cost [16]. Therefore, pseudo-labeling methods [17], [18], [19] that improve network performance by generating labels from large, readily available unlabeled data are actively being researched in many tasks [16], [20]. However, even if pseudo-labels are correctly generated, the generated new data (i.e., pseudo-labeled data) must be efficiently trained on existing networks. Nevertheless, because current studies related to pseudo-labeling focus only on data generation, further research is needed on strategies to train new data efficiently and continuously. It is noteworthy that retraining all existing training data together to train new data is inefficient, and when this training is performed on mobile devices, access to the training set used for pre-training is impossible. That is, training only new data on the existing pre-trained network without access to previous data used for pre-training is important.

Until now, when networks need to be continuously trained on new data, fine-tuning is used, which is a very simple method of training new data by adjusting the learning rate in the network pre-trained with previous data [21]. However, as fine-tuning focuses only on optimizing networks for new data, forgetting the knowledge of previous data is relatively easy, and this approach has the disadvantage that it cannot have high performance without a pre-trained network using a huge dataset (e.g., ImageNet) [22]. Moreover, even if the network is pre-trained on a huge dataset, the case of performing multiple training procedures (*i.e.*, multi-step training) according to the continuous creation of new data has not been widely considered. Besides fine-tuning, approaches to improve the performance of CNNs or continual learning methods in which a new class is added can be used to train new data. The approaches adopted in [11], [12], and [13] enhance the learning ability of CNNs in various ways, and new data can be trained with higher performance. However, because it is difficult to prevent forgetting the knowledge of previous data during training new data, achieving high performance is relatively challenging. Continual learning [23], [24], [25] is an approach to solve the catastrophic forgetting phenomenon in which knowledge about a previously trained class is forgotten when a new class that has not been previously trained is added. It deals with the same situation in that new data are added, but because it focuses on solving catastrophic forgetting, it is difficult to train new data with high performance when applied to a situation where new data with the same class is continuously added.

In this study, we propose a multi-step training framework that efficiently trains accumulated new data without previous data by utilizing sparsity training. The proposed method proceeds in four steps. First, we prepare a pre-trained network that has been trained on previous data. Second, through proposed selective sparsity training (SST) that selects target parameters adaptively according to the training progress, pre-trained network compresses without degradation of performance for previous data, creating useless parameters with low importance. Third, to efficiently reuse useless parameters for the training of new data, we selectively re-initialize only some useless parameters in consideration of the parameter state of the current network. Finally, we train all parameters (*i.e.*, parameters that remain due to high importance and parameters that are re-usable through re-initialize) of the network with new data through the proposed adaptive online distillation (AOD). In the case of training sequentially (*i.e.*, multi-step), it proceeds through repetition of the four-step training process (*i.e.*, single-step) described above. Notably, multi-step training involves a problem in that it is difficult to preserve knowledge about previous data by performing sparsity training only with current data. Therefore, we overcome this problem by designing a multi-step loss (MSL) function using only knowledge distillation loss without cross-entropy loss using a pre-trained network as a teacher. The results of extensive experiments on the CIFAR-100 [26], Tiny-ImageNet [27], and ImageNet [28] datasets show that the proposed method significantly improves the performance of training new data.

The contributions of this work are summarized as follows:

- A selective sparsity training method is proposed to train new data.
- We develop an adaptive re-initialization method and an adaptive online distillation training method to achieve high performance by efficiently training useless parameters created by sparsity training with new data.
- A multi-step loss function is designed for multi-step training.

The remainder of this paper is organized as follows. Section II introduces existing studies related to approaches for training new data and sparsity training. Section III presents a detailed description of the proposed multi-step training framework, and Section IV provides the experimental results and an analysis of the proposed method. Finally, Section V concludes this paper.

II. RELATED WORKS

A. APPROACHES FOR TRAINING NEW DATA

Several approaches have been developed to train pre-trained CNNs with new data. Fine-tuning, a simple and common strategy of transfer learning, trains new data by adjusting the learning rate for a new purpose in the pre-trained network using a huge dataset such as ImageNet [28]. However, fine-tuning can be applied only in a limited environment because it is generally difficult to achieve high performance without the pre-trained network using a huge dataset [22] and even if such a pre-trained network were available, existing methods only considers single-step training; multi-step training is not considered. That is, when new data are continuously generated and accumulated, several training procedures are required, but there is a problem that fine-tuning several times decreases training efficiency and degrades performance.

When training new data, approaches to improving the learning ability of CNNs can also be used. In these approaches, various methods to improve performance such as training additional knowledge obtained through knowledge

distillation [29], adversarial training that controls the difference between the distributions of inputs [11], and reactivation of invalid parameters [30], [31] have been studied. However, training new data through these approaches does not accompany the solution to the problem of forgetting knowledge about previous data in the training process, so the performance for previous data degrades and it is difficult to achieve high performance in multi-step training that requires preserving knowledge obtained from previous data.

Continual learning [23], [24], [25], [32], [33], an approach to training new classes, can also be used to train new data. Continual learning was conducted with a focus on solving the catastrophic forgetting phenomenon for the previous class, which usually occurs while training new classes. However, because continual learning focuses on solving catastrophic forgetting, it is difficult to achieve high performance in training where new data with the same class are continuously added.

B. SPARSITY TRAINING

Sparsity training, combined with pruning [34], is a network compression method used to apply CNNs to real applications, which enables us to identify insignificant parameters in networks. A representative sparsity training method in [35] imposes L1 regularization on the scaling factors in batch normalization (BN) layers so that the BN scaling factor of some parameters converges to 0, which simplifies identifying insignificant filter without any changes to existing CNN architectures. The training objective function of the sparsity training in [35] is expressed as follows:

$$L = \sum_{(x,y)} l(f(x, W), y) + \lambda \sum_{s \in \tau} | s | \tag{1}$$

where (x,y) and W denote the (training input, target) and trainable weights, respectively. In (1), the first and second sum-terms correspond to the cross-entropy loss of a CNN and L1 norm sparsity-induced penalty on all the scaling factors, respectively, and λ balances the two terms. This means that the sparsity training in [35] is performed with all parameters of the network as targets, which makes all BN scaling factors of the network converge as a whole. This is not a big problem for pruning where sparsity training is only performed once, but if the sparsity training is used for multi-step training methods that require training to be performed several times, all BN scaling factors eventually converge to 0, which makes it impossible for the network to achieve high performance.

III. PROPOSED METHOD

A. APPROACH OVERVIEW

We propose a multi-step training framework using sparsity training for the efficient training of accumulated new data without access to previous data. The multi-step training framework is a framework that can continuously train new data to the network through repetition of the single-step training framework. Figure 1 illustrates the proposed single-step training procedure. We associate a scaling factor (reused from the BN layer) with each filter in all convolution layers so that the filter scaling factor converges to 0. By performing training with a filter unit, new data can be trained efficiently while preserving knowledge of previous data. As shown in Figure 1, after preparing a pre-trained network trained with previous data (*i.e.*, Figure 1 (a)), we create useless filters by converging the scaling factor of specific filters to 0 through sparsity training without performance degradation (*i.e.*, Figure 1 (b)). Subsequently, the filter scaling factor converged to 0 is re-initialized and reused for training again (*i.e.*, Figure 1 (c)), and we train the network with new data (*i.e.*, Figure 1 (d)). As a result, although the actual network size is the same, by creating useless filters through sparsity training and using them for new data training, the proposed method significantly improves network performance. Multi-step training is performed through repetition of single-step training by using the results of the current step (*i.e.*, the results of Figure 1 (d)) as a pre-trained network in the next step.

However, if the proposed training procedure is performed using existing methods, achieving high performance is challenging owing to various limitations. Existing sparsity training methods are not suitable for multi-step training of new data because of the limitations of forgetting knowledge of previous data and reducing the overall parameter values. In addition, when re-initializing the scaling factor of useless filters, the existing initialization method [36], [37] cannot be used. This is because during previous training, most parameters already have optimal values, and re-initialization should be performed only for the scaling factors of some useless filters while considering the state of other filter scaling factors in the network. Moreover, when training new data after re-initialization, because existing methods [21], [29], [38] do not have the ability to preserve the knowledge of previous data and efficiently train new data simultaneously, they are difficult to be used for new data training owing to their poor performance.

B. SELECTIVE SPARSITY TRAINING

The proposed SST is performed by targeting a few filter scaling factors selected adaptively, rather than all filter scaling factors. By preventing the reduction of factors other than the scaling factor of the useless filter, we overcome the problem of existing sparsity training [35] that the scaling factors of all filters in networks converge to 0 when performing training several times. We defined SP_r ($0 < SP_r < 1$) as the ratio of the target filter currently being sparsity trained and $Target_r$ ($0 < Target_r < 1$) as the final sparsity strength set by the user. Because a small L1 norm scaling factor has a small effect on training, the target filter scaling factor is selected as much as the SP_r in the order of the scaling factor with a small L1 norm, and the SP_r is determined adaptively to the training progress (epoch) as follows:

$$SP_r = \begin{cases} \dfrac{Target_r - 1}{E_{cut}} * E_{cur} + 1 & (E_{cur} < E_{cut}) \\ Target_r & (E_{cur} \geq E_{cut}) \end{cases} \tag{2}$$

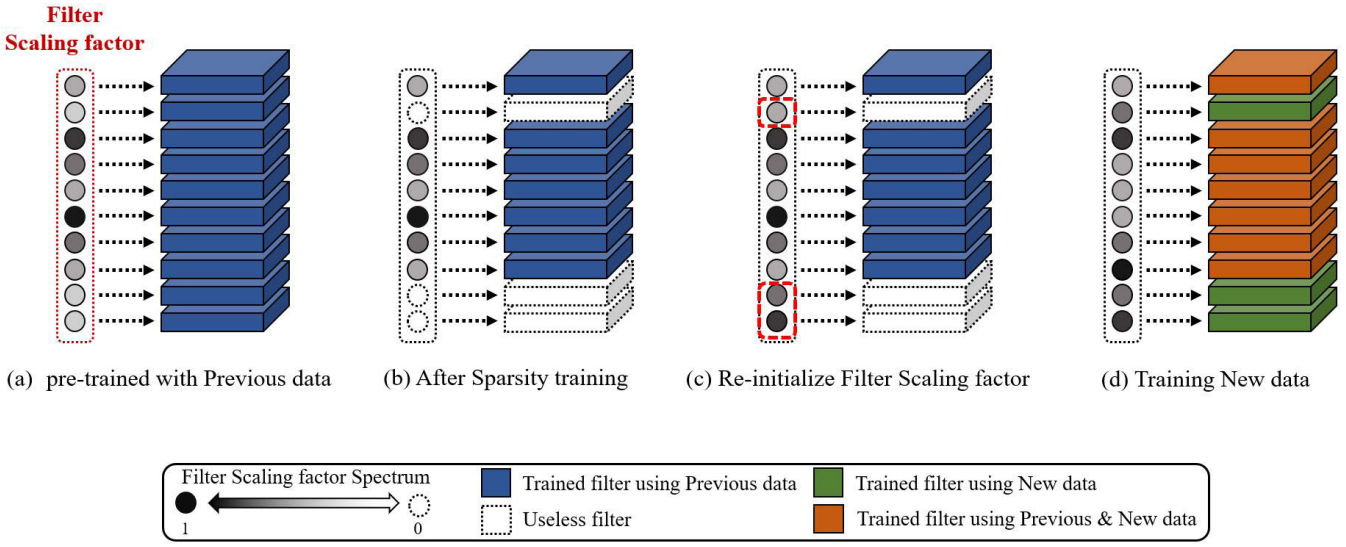


FIGURE 1. Procedure for the proposed single-step training on the pre-trained network.

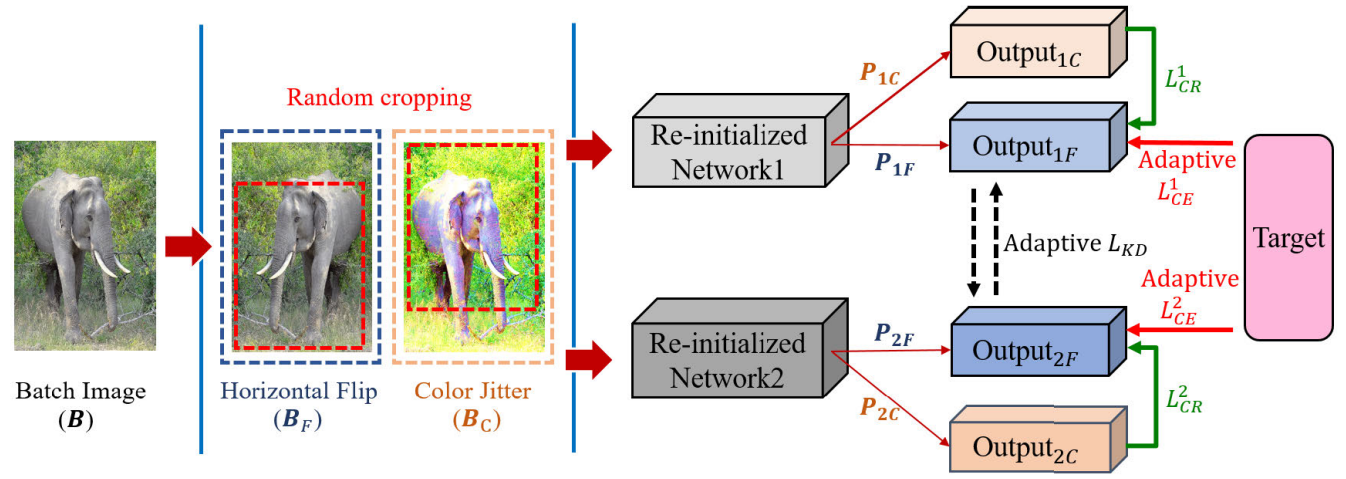


FIGURE 2. Overview of the adaptive online distillation for efficient training of new data.

where E_{cur} is the current epoch, and E_{cut} is the epoch when SP_r matches $Target_r$. When training begins, SP_r starts at 1, and sparsity training is performed targeting all filter scaling factors. However, as training progresses, SP_r decreases linearly and eventually matches the $Target_r$ at the E_{cut} epoch. To determine the optimal E_{cut}, several factors need to be considered. First, it is necessary to quickly match the SP_r and $Target_r$ to prevent unnecessary reduction of the filter-scaling factor. Second, if SP_r is decreased too rapidly, there is a higher chance of misjudging which filter's scaling factor is important. Third, if L1 regularization for sparsity training is applied too strongly, performance can degrade significantly. Therefore, we conduct an experiment to empirically determine the optimal E_{cut} considering these three points, and consequently define E_{cut} as an epoch in which the learning rate generally decreases as follows:

$$E_{cut} = \frac{1}{3}E_{max} \tag{3}$$

where E_{max} is the maximum number of training epochs. If filter scaling factors as much as the $Target_r$ converge to 0 before the E_{cut} epoch, SP_r immediately has the same value as the $Target_r$ to prevent unnecessary reduction of other filter scaling factors. To verify the $Target_r$ setting, comparative experiments are conducted based on the $Target_r$ in the SST section of the ablation study.

C. DIVERSE RE-INITIALIZATION TO REUSE FILTERS

After the proposed SST, the scaling factor of filters identified as useless is almost 0, and re-initialization is necessary for efficient training of new data. However, since all filters except those targeted for re-initialization have knowledge of previous data, re-initialization should be performed carefully. The proposed scaling factor re-initialization method re-initializes to a random value within an appropriate range, considering the size of the scaling factors of all filters in the network. When the filter scaling factors are sorted in order of size,

the scaling factors near the $Target_r$ have very small values, converging to almost 0, and the maximum scaling factor applied to only a few specific filters has a very large value. If these values are included in the re-initialization range, proper re-initialization may not be performed. Therefore, we determined the optimal re-initialization range of the filter scaling factor, R ($0 < R < 1$), as follows:

$$Target_r + I < R < 1 - I \tag{4}$$

where I is a term ranging from 0 to 1 that determines the appropriate R. A higher value of I results in a narrower range R with less diversity but a higher probability of including an appropriate value, while a lower value of I results in a wider range R with more diversity but a lower probability of including an appropriate value.

D. ADAPTIVE ONLINE DISTILLATION

As a result of the proposed SST and re-initialization, after securing an extra filter that enables the training of new data while preserving the knowledge of previous data, we improve the training efficiency of new data through the proposed AOD. The proposed AOD method adaptively combines knowledge distillation loss (L_{KD}) with cross-entropy loss (L_{CE}) to enable two student networks to effectively perform collaborative learning, and consistency regularization loss (L_{CR}) is added to improve network consistency. It should be noted that many existing online distillation approaches [29], [39], [40] focus only on the type of knowledge of student networks and the distillation strategy, and do not consider the performance of each student network. An overview of the proposed AOD is shown in Figure 2. As input, we use a pair of randomly cropped images of B_F, the horizontal flip version of the original batch image (B), and B_C, which is a color jitter version with corrected saturation, brightness, and illuminance. Because both input images represent the same class, the predicted class must be matched. Therefore, we improve the prediction consistency of the network by introducing the Kullback-Leibler divergence (KL), which is often used in CNNs, as a L_{CR}, as follows:

$$L_{CR} = KL(P_F \parallel P_C) \tag{5}$$

where P_F and P_C are the prediction logits for B_F and B_C, respectively. To achieve a higher performance than that obtained by the improved prediction consistency of the network, we use two student networks. To make collaborative learning more effective through knowledge distillation of two different student networks with different knowledge, before starting AOD, in the re-initialization stage, the value of I, which has a great influence on the range of re-initialization range R, is set differently to securing the diversity of filters reused. That is, two different student networks are created by performing diverse re-initialization using different values of I after SST. Training two student networks for knowledge distillation inevitably creates a relatively superior student network and an inferior student network. For the superior network, it is better to train with a high coefficient of L_{CE} rather than transfer the knowledge from the inferior network. However, for the inferior network, since it is difficult to outperform the superior network by training with the same dataset and loss function as the superior network, it is better to transfer the knowledge from the superior network with a high coefficient of L_{KD}. Shortly, it is more reasonable to train a superior network using a low coefficient of L_{KD} and an inferior network using a high coefficient of L_{KD} rather than two networks distilling each other's knowledge with the same coefficient. To simplify the notation, when k=1,2, we denote two student networks as N_k, the prediction logit of each network for a pair of inputs as P_{kF},P_{kC}, and the loss functions of each network as L^k_{CE}, L^k_{KD}, L^k_{CR}. The coefficient W^k_{KD} of L^k_{KD} to be applied to network N_k is determined by the ratio of cross-entropy loss (L^k_{CE}) as follows:

$$W^1_{KD} = \frac{L^1_{CE}}{L^1_{CE} + L^2_{CE}}, \quad W^2_{KD} = \frac{L^2_{CE}}{L^1_{CE} + L^2_{CE}}. \tag{6}$$

The knowledge distillation loss L^k_{KD} and the total loss L_k of each network are as follows:

$$L^1_{KD} = JS(P_{1F} \parallel P_{2F}), \quad L^2_{KD} = JS(P_{2F} \parallel P_{1F}), \tag{7}$$

$$L_1 = W^2_{KD} * L^1_{CE} + W^1_{KD} * L^1_{KD} + \alpha * L^1_{CR}, \quad L_2 = W^1_{KD} * L^2_{CE} + W^2_{KD} * L^2_{KD} + \alpha * L^2_{CR}, \tag{8}$$

where in (7), JS represents the Jensen-Shannon divergence, and in (8), α is a term that balances the total loss L_k.

E. MULTI-STEP LOSS FOR MULTI-STEP SPARSITY TRAINING

The multi-step training procedure is a repetition of the single-step training procedure, but one additional problem needs to be overcome to achieve high performance. In other words, unlike single-step training, which performs sparsity training using pre-training data, multi-step training (*i.e.*, second step, third step, etc.) performs sparsity training only with current data, so it has to overcome the problem of forgetting knowledge about previous data. Therefore, we design an MSL in addition to SST and then perform training by freezing the fully connected layer. To distill the knowledge of previous data, the network trained so far is used as a teacher network, frozen during training, and when the performance of the student network outperforms the teacher network's performance, the student network is updated to the new teacher network for distilling more powerful knowledge to the student network. The MSL designed without L_{CE} preserves the knowledge from previous data by distilling the attention map of each block of the teacher network as well as the prediction logit of the teacher network to figure out which part of the feature map is in focus. The designed MSL, L_{MS},

is formulated as follows:

$$L_{MS} = KL(P_T \parallel P_S) + (\sum_{i=1}^{n} L_{AT})/n, \tag{9}$$

where P_T and P_S are the prediction logits of the teacher and student networks for the previous data, respectively, n is the total number of blocks in the network, i is the block index, and L_{AT} is the attention loss in [41]. It should be noted that the first and second terms of (9) are the prediction logits distillation loss and the attention map distillation loss, respectively.

IV. EXPERIMENTS

A. EXPERIMENTAL ENVIRONMENTS

To evaluate the performance of the proposed multi-step training framework, we conducted extensive experiments related to each contribution, hyperparameter setting, and comparison with other approaches using commonly used architectures (*i.e.*, ResNet [3], Wide-ResNet [42], and MobileNetv2 [43]) and various datasets (*i.e.*, CIFAR-100 [26], Tiny-ImageNet [27], and ImageNet [28]) in image classification tasks. For use in the single-step and multi-step experiments, all datasets are split so that they do not overlap. In the single-step experiment, all datasets are randomly split by 1:1 and used as previous data and new data, respectively, without overlapping images; in the multi-step experiment, all datasets are randomly split in a 1:1:1:1 ratio without overlapping images and 4-step training is conducted using partitioned Data1, Data2, Data3, and Data4, respectively. It should be noted that Data1 is trained from scratch whereas the remaining data are trained on the pre-trained network. We use Top-1 classification accuracy (%) as an evaluation metric and report the average and standard deviation of three experiments to provide a more convincing analysis.

B. IMPLEMENTATION DETAILS

All experiments are conducted using NVIDIA GeForce RTX 3090 GPUs and NVIDIA GeForce RTX 2080 GPUs. We use stochastic gradient descent (SGD) as the optimizer and an initial learning rate of 0.1 for the first training and sparsity training, and 0.001 otherwise. The detailed environmental settings for CIFAR-100 include a batch size of 256, a momentum of 0.9, weight decay of 0.0005, overall training epochs of 300, and cosine annealing scheduler [44] is used. For Tiny-ImageNet and ImageNet training, a batch size of 64, momentum of 0.9, weight decay of 0.0001, and overall training epoch of 90 are used, and the learning rate is divided by 10 for every 30 epochs. For the diversity of two student networks used in AOD, each $I = \{0.1, 0.2\}$ is used, and $\alpha = 0.5$ is set to balance the loss.

C. ABLATION STUDY

1) SELECTIVE SPARSITY TRAINING

To verify the proposed SST, we compared it with the base sparsity training (BST) from [35] using Tiny-ImageNet in ResNet50. Figure 3 displays the distribution and average of the filter scaling factors of the three networks: the pre-trained network, the network trained with BST, and the network trained with SST. As shown in Figures 3 (a) and (b), both methods generated useless filters (*i.e.*, filters in which the scaling factor converges to 0) after sparsity training. In Figure 3(c), the proposed SST with pink color maintains the average scaling factor of the remaining filters, except for the useless filter, which is similar to that of the pre-trained network with blue color, but the BST with green color shows that the average scaling factor is significantly reduced. Thus, the scaling factor does not converge to 0 even if the proposed SST is performed several times.

TABLE 1. Top-1 Accuracy(%) according to the $Target_r$ and E_{cut}.

$Target_r$ / E_{cut}	$E_{max}/2$	$E_{max}/3$	$E_{max}/4$
0.1	$60.68_{\pm 0.15}$	$60.73_{\pm 0.11}$	$60.59_{\pm 0.12}$
0.3	$61.11_{\pm 0.12}$	$\mathbf{61.35}_{\pm 0.12}$	$61.03_{\pm 0.12}$
0.5	$61.02_{\pm 0.22}$	$61.21_{\pm 0.21}$	$60.99_{\pm 0.17}$

To determine the optimal values of E_{cut} and $Target_r$, a single-step experiment is conducted using ResNet50 on Tiny-ImageNet. Table 1 presents the experimental results under various conditions (i.e., a total of 9 cases obtained by combining three types of $Target_r$ and E_{cut}, respectively). Table 1 shows that the best performance was observed when $Target_r$ was set to 0.3 and E_{cut} to $E_{max}/3$. This means that the highest performance can be achieved when matching the value of sparsity strength with $Target_r$ at 1/3 of the entire training epoch, while 70% of the entire network stores information about previous data and the remaining 30% learns information about new data. However, since the performance varies significantly with changes in $Target_r$ compared to E_{cut}, we performed additional experiments on various networks to identify an appropriate $Target_r$ while keeping E_{cut} fixed at $E_{max}/3$.

Table 2 shows the results of single-step experiments using Tiny-ImageNet in ResNet34, ResNet50, and ResNet101 to determine the optimal value for $Target_r$. It should be noted that when the $Target_r$ is set too high, many useless filters are created, and valuable knowledge from previous data is lost; on the other hand, when it is set too low, the number of filters that can train new data completely decreases, making it difficult to efficiently train new data.The best performance was observed in all three networks when $Target_r$ was set to 0.3. This means that $Target_r$ of 0.3 allows the network to learn information about both previous data and new data in the most balanced way. Therefore, we empirically determined that $Target_r = 0.3$ is the optimal value and all subsequent experiments were conducted with this optimal $Target_r$ and $E_{cut} = E_{max}/3$.

2) DIVERSE RE-INITIALIZATION AND ADAPTIVE ONLINE DISTILLATION

In (7), AOD uses the adaptive L_{KD} and L_{CR}. To compare the effects of each loss when training new data, Table 3

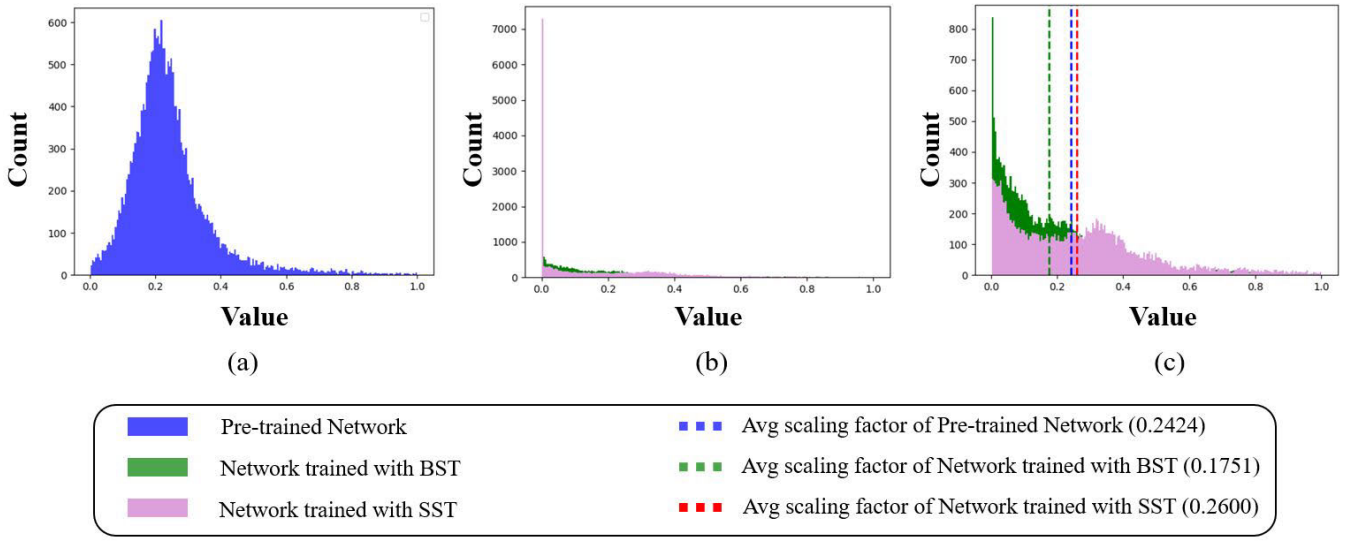


FIGURE 3. Distribution and average of the filter scaling factors of the three networks. (a) shows the distribution of the filter scaling factors of the pre-trained network, (b) shows the distribution of the filter scaling factors after BST and SST, and (c) shows the distribution and average of remaining scaling factors excluding scaling factors of useless filters.

TABLE 2. Top-1 Accuracy(%) according to the $Target_r$ on various networks.

Network	$Target_r$		
	0.1	0.3	0.5
ResNet34	60.14±0.15	**60.27**±0.12	58.98±0.13
ResNet50	60.73±0.11	**61.35**±0.12	61.21±0.21
ResNet101	60.82±0.13	**61.69**±0.14	61.16±0.15

TABLE 3. Top-1 Accuracy(%) according to the various loss functions in AOD.

Metric	General training	General L_{KD}	Adaptive L_{KD}	Adaptive L_{KD} + L_{CR}
Acc.(%)	66.08±0.15	66.89±0.15	68.21±0.25	**69.71**±0.19
Gain(%)	-	0.81	2.13	3.63

TABLE 4. Top-1 Accuracy(%) comparison with and without diverse re-initialization(Dri) and AOD after SST.

Metric	w/ FT	w/ AOD	w/ AOD&Dri
Acc.(%)	60.83±0.28	61.17±0.18	**61.35**±0.12
Gain(%)	-	0.34	0.52

shows the results of single-step experiments conducted using CIFAR-100 and ResNet110. Compared with general training, general L_{KD} and adaptive L_{KD} achieve a performance improvement of 0.81% and 2.13%, respectively. This shows that adding L_{KD} to general training improves the training performance of new data, and L_{KD} adaptive to L_{CE} achieves better performance than general L_{KD}. In particular, AOD with the adaptive L_{KD} and L_{CR} achieves the highest accuracy of 69.71% with a performance improvement of 3.63%.

Table 4 presents the results of verification experiments conducted on ResNet50 using Tiny-ImageNet to evaluate the performance of diverse re-initialization and AOD. When

TABLE 5. Top-1 Accuracy(%) comparison according to the teacher network update in MSL.

Step	Fixed teacher network	Teacher network update	Gain(%)
1	39.59±0.29	39.59±0.29	-
2	48.61±0.19	48.61±0.19	-
3	52.32±0.11	52.68±0.15	0.36
4	53.49±0.15	**54.90**±0.12	1.41

TABLE 6. Top-1 Accuracy(%) comparison w/ and w/o MSL in BST and SST.

Step	BST		SST	
	w/o MSL	w/ MSL	w/o MSL	w/ MSL
1	39.59±0.29	39.59±0.29	39.59±0.29	39.59±0.29
2	48.51±0.18	48.51±0.18	48.61±0.19	48.61±0.19
3	50.07±0.15	50.41±0.14	50.58±0.21	52.68±0.15
4	50.26±0.17	50.66±0.25	52.44±0.23	**54.90**±0.12

training new data after SST, comparing the single-step experiment results of fine-tuning with the proposed method shows that applying AOD and diverse re-initialization achieves performance improvements of 0.34% and 0.18%, respectively. These results show that the performance of the proposed AOD is better than fine-tuning when training new data after SST, and in particular, the performance of AOD can be further improved through diverse re-initialization.

3) MULTI-STEP LOSS FOR MULTI-STEP SPARSITY TRAINING

In MSL, if the student network outperforms the teacher network, it is updated as the new teacher network. Table 5 shows the results of multi-step training experiments conducted using Tiny-ImageNet on ResNet50 to verify the superiority of teacher network updates. In steps 1 and 2, there is no performance difference due to the teacher network update in MSL; however, a clear performance difference occurs from

TABLE 7. Top-1 Accuracy(%) comparison in single-step training and multi-step training.

Experiment	Dataset	Network	Training data	Baseline (fine-tuning)	HPO[22]	DML[30]	LwM[39]	Ours
Single-step	CIFAR-100	ResNet32	Previous	61.94±0.34	61.94±0.34	61.94±0.34	61.94±0.34	61.94±0.34
			New	64.92±0.03	65.10±0.04	65.13±0.06	65.78±0.14	**66.14**±0.07
		ResNet110	Previous	63.19±0.31	63.19±0.31	63.19±0.31	63.19±0.31	63.19±0.31
			New	65.90±0.27	66.02±0.29	66.11±0.15	66.48±0.21	**69.71**±0.19
		WRN28-10	Previous	69.69±0.21	69.69±0.21	69.69±0.21	69.69±0.21	69.69±0.21
			New	72.08±0.11	72.72±0.13	73.15±0.09	72.76±0.15	**76.53**±0.09
	Tiny-ImageNet	ResNet34	Previous	51.91±0.25	51.91±0.25	51.91±0.25	51.91±0.25	51.91±0.25
			New	55.18±0.21	56.68±0.24	56.83±0.15	57.54±0.17	**60.27**±0.12
		ResNet50	Previous	53.08±0.27	53.08±0.27	53.08±0.27	53.08±0.27	53.08±0.27
			New	57.86±0.18	58.25±0.21	58.12±0.11	58.10±0.19	**61.35**±0.12
		ResNet101	Previous	55.21±0.26	55.21±0.26	55.21±0.26	55.21±0.26	55.21±0.26
			New	60.62±0.23	60.82±0.13	60.68±0.17	60.78±0.20	**61.69**±0.14
		MobileNetv2	Previous	54.69±0.40	54.69±0.40	54.69±0.40	54.69±0.40	54.69±0.40
			New	58.15±0.12	57.78±0.08	58.21±0.06	58.96±0.03	**59.00**±0.08
	ImageNet	ResNet34	Previous	69.09±0.10	69.09±0.10	69.09±0.10	69.09±0.10	69.09±0.10
			New	71.48±0.14	71.47±0.19	71.52±0.10	71.54±0.10	**71.98**±0.07
Multi-step	CIFAR-100	ResNet32	Data1	51.00±0.15	51.00±0.15	51.00±0.15	51.00±0.15	51.00±0.15
			Data2	54.53±0.15	54.33±0.07	54.53±0.08	55.67±0.04	58.04±0.12
			Data3	57.06±0.11	56.76±0.09	57.08±0.08	58.08±0.02	60.54±0.14
			Data4	58.62±0.07	58.12±0.08	58.40±0.11	59.90±0.10	**60.90**±0.12
		ResNet110	Data1	53.45±0.51	53.45±0.51	53.45±0.51	53.45±0.51	53.45±0.51
			Data2	56.48±0.13	56.63±0.09	56.50±0.06	57.62±0.01	59.80±0.17
			Data3	59.19±0.02	59.14±0.05	59.23±0.12	60.11±0.15	63.18±0.06
			Data4	60.66±0.13	60.36±0.01	60.49±0.09	61.64±0.05	**65.27**±0.15
	Tiny-ImageNet	ResNet50	Data1	39.59±0.29	39.59±0.29	39.59±0.29	39.59±0.29	39.59±0.29
			Data2	44.63±0.41	44.66±0.32	44.54±0.29	45.11±0.23	48.61±0.19
			Data3	46.28±0.27	46.88±0.21	46.72±0.20	47.52±0.16	52.68±0.15
			Data4	48.48±0.23	47.98±0.11	48.02±0.13	48.91±0.12	**54.90**±0.12
		MobileNetv2	Data1	45.02±0.27	45.02±0.27	45.02±0.27	45.02±0.27	45.02±0.27
			Data2	49.28±0.16	48.86±0.12	48.71±0.02	49.39±0.03	50.40±0.01
			Data3	50.51±0.09	50.26±0.15	50.14±0.01	51.49±0.05	53.09±0.02
			Data4	51.96±0.03	51.32±0.05	51.22±0.01	52.58±0.01	**54.64**±0.01

step 3, and in step 4, a performance improvement of 1.41% is achieved compared to when the teacher network is fixed. This shows that continuously updating the network with a better teacher leads to better performance than fixing the teacher network in step 3 or higher where MSL is applied.

Next, to evaluate the superiority of the proposed method in multi-step training, we apply MSL to both BST and SST on ResNet50 with Tiny-ImageNet and compare the results of the multi-step experiment in Table 6. When training up to Step 2, the performance difference between BST and SST is not significant, but the performance gap widens as the step increases, confirming that the proposed SST is much better in multi-step training. Furthermore, since Step 2 training belongs to the single-step, performance does not change depending on whether MSL is applied or not. However, in the multi-step experiment where Data3 and Data4 are trained, the networks with the proposed MSL show better performance in both BST and SST. In particular, the proposed SST w/MSL shows a large performance difference of 4.64% compared to the BST w/o MSL, verifying that the proposed SST and MSL are excellent in multi-step training.

D. COMPARISON OF CLASSIFICATION ACCURACY

We compare the proposed multi-step training framework to standard fine-tuning and various recently proposed methods, including HPO [21], DML [29], and LwM [38]. HPO [21] is a recent study that has conducted many experiments and suggested new hyperparameter settings, such as learning rate, batch size, momentum, and weight decay, for fine-tuning based on the relationship between previous data and new data. HPO with improved fine-tuning was selected as a comparison target to show the performance improvement limit when training new data only by fine-tuning in the existing network. DML [29] demonstrated that collaborative learning of multiple student networks performed better than knowledge distillation from a powerful teacher network and proposed an online distillation method in which an ensemble of student networks collaboratively distill knowledge to each other.

In multi-step training, it is important to continuously train new data with high performance. Therefore, DML, which improves performance by supplementing the weaknesses of each network through an ensemble of various student networks, was selected as a comparison target for fair performance comparison with the proposed AOD. LwM [38], a recent study related to continual learning, achieved higher performance by distilling the prediction logit and final attention map of networks. LwM, which approached the catastrophic forgetting problem using distillation loss, was selected as a comparison target to verify the catastrophic forgetting prevention effect of the proposed MSL in multi-step training.

Table 7 presents the results of the single-step experiments with various network architectures. The proposed method outperforms the comparative approaches in all single-step experiments with very high performance. Specifically, on the CIFAR-100 dataset, the proposed method improves the performance by 1.22%, 3.81%, and 4.45% compared to the baseline in ResNet32, ResNet110, and WRN28-10, respectively. On the Tiny-ImageNet dataset, the proposed method shows performance improvements of 5.09%, 3.49%, 1.07%, and 0.85% compared to the baseline in ResNet34, ResNet50, ResNet101, and MobileNetv2, respectively. Additionally, the proposed method achieves 0.5% higher performance than the baseline in ResNet34 on the ImageNet dataset. The proposed method shows the best performance in all eight single-step experiments using various datasets and networks, and achieves a significant performance improvement of 5.09% when training Tiny-ImageNet in ResNet34. In order to achieve high performance in single-step experiments, it is most important that the network learns new data while preserving knowledge about previous data. Therefore, these results show that the proposed SST efficiently learned new data through Dri and AOD while preserving knowledge of previous data well in the network.

Table 7 also presents the multi-step experiment results conducted in various network architectures. In the proposed method, ResNet32 and ResNet110 trained on CIFAR-100 show performance improvements of 2.28% and 4.61%, respectively, compared to the baseline, and ResNet50 and MobileNetv2 trained on Tiny-ImageNet show performance improvements of 6.42% and 2.68%, respectively, compared to the baseline. As a result, compared to other approaches, the proposed method achieves excellent performance despite undergoing sparsity training several times with only current data, which verifies the effectiveness of the designed MSL for multi-step training. Moreover, the fact that the highest performance is achieved in all steps further confirms the superiority of the proposed multi-step training framework.

V. CONCLUSION

In this paper, we have proposed a novel multi-step training framework for efficiently training accumulated new data without access to previous data. Our approach achieves this by creating useless filters through SST, maintaining the performance of previous data, and then performing diverse re-initialization to reuse these useless filters for training new data, followed by efficient training with AOD. When new data are accumulated and training is needed again, multi-step training is performed by applying the MSL to SST. Extensive experiments have been conducted to demonstrate the excellent performance of the proposed approach on various networks and datasets, both in single-step and multi-step scenarios.

REFERENCES

[1] 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 Integr. (VLSI) Syst.*, vol. 27, no. 8, pp. 1861–1873, Aug. 2019.

[2] S. Ki, J. Park, and H. Kim, "Dedicated FPGA implementation of the Gaussian TinyYOLOv3 accelerator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 10, pp. 3882–3886, Oct. 2023.

[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] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," 2017, *arXiv:1712.04621*.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[6] 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. (ICCV)*, Oct. 2019, pp. 502–511.

[7] 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. (AICAS)*, Aug. 2020, pp. 16–20.

[8] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9156–9165.

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

[10] S. I. Lee and H. Kim, "GaussianMask: Uncertainty-aware instance segmentation based on Gaussian modeling," in *Proc. 26th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2022, pp. 3851–3857.

[11] C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, "Adversarial examples improve image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 816–825.

[12] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3712–3721.

[13] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 558–567.

[14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–10.

[15] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.

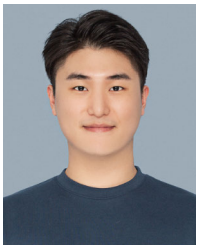
[16] J. Jeong, S. Lee, J. Kim, and N. Kwak, "Consistency-based semi-supervised learning for object detection," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–10.

[17] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Proc. Workshop Challenges Represent. Learn. (ICML)*, vol. 3, no. 2, 2013, p. 896.

[18] H. Pham, Z. Dai, Q. Xie, and Q. V. Le, "Meta pseudo labels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11557–11568.

[19] Z. Hu, Z. Yang, X. Hu, and R. Nevatia, "SimPLE: Similar pseudo label exploitation for semi-supervised classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 15099–15108.

[20] Z. Wang, Y. Li, Y. Guo, L. Fang, and S. Wang, "Data-uncertainty guided multi-phase learning for semi-supervised object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 4568–4577.
[21] H. Li, P. Chaudhari, H. Yang, M. Lam, A. Ravichandran, R. Bhotika, and S. Soatto, "Rethinking the hyperparameters for fine-tuning," 2020, *arXiv:2002.11770*.
[22] S. Kornblith, J. Shlens, and Q. V. Le, "Do better ImageNet models transfer better?" in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2661–2671.
[23] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2935–2947, Dec. 2018.
[24] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, and A. Grabska-Barwinska, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, Mar. 2017.
[25] R. Aljundi, F. Babiloni, M. Elhoseiny, M. Rohrbach, and T. Tuytelaars, "Memory aware synapses: Learning what (not) to forget," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 139–154.
[26] A. Krizhevsky, "Learning multiple layers of features from tiny images," Master's thesis, University of Toronto, Toronto, ON, Canada, 2009.
[27] Y. Le and X. Yang, "Tiny ImageNet visual recognition challenge," *CS 231N*, vol. 7, no. 7, p. 3, 2015.
[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, 2012, pp. 1–9.
[29] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4320–4328.
[30] F. Meng, H. Cheng, K. Li, Z. Xu, R. Ji, X. Sun, and G. Lu, "Filter grafting for deep neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6599–6607.
[31] J. J. Lee and H. Kim, "Versatile kernel reactivation for deep convolutional neural networks," *Electron. Lett.*, vol. 58, no. 19, pp. 723–725, Sep. 2022.
[32] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7765–7773.
[33] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," 2016, *arXiv:1606.04671*.
[34] N. J. Kim and H. Kim, "FP-AGL: Filter pruning with adaptive gradient learning for accelerating deep convolutional neural networks," *IEEE Trans. Multimedia*, vol. 25, pp. 5279–5290, 2022.
[35] 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. 2736–2744.
[36] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[38] P. Dhar, R. V. Singh, K.-C. Peng, Z. Wu, and R. Chellappa, "Learning without memorizing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5138–5146.
[39] G. Wu and S. Gong, "Peer collaborative learning for online knowledge distillation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 10302–10310.
[40] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 3430–3437.
[41] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," 2016, *arXiv:1612.03928*.
[42] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.
[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[44] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," 2016, *arXiv:1608.03983*.



JEONG JUN LEE (Graduate Student Member, IEEE) received the B.S. degree in electrical and information engineering from the Seoul National University of Science and Technology, Seoul, South Korea, in 2021, where he is currently pursuing the M.S. degree in electrical and information engineering. His research interests include continual learning, knowledge distillation, and various algorithms of computer vision and deep learning.

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