IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 6 December 2022, accepted 17 December 2022, date of publication 27 December 2022,
date of current version 24 January 2023.

Digital Object Identifier 10.1109/ACCESS.2022.3232566

RESEARCH ARTICLE

A Channel Pruning Optimization With Layer-Wise Sensitivity in a Single-Shot Manner Under Computational Constraints

MINSU JEON, TAEWOO KIM, CHANGHA LEE, (Member, IEEE), AND CHAN-HYUN YOUN, (Senior Member, IEEE)
School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Chan-Hyun Youn (chyoun@kaist.ac.kr)

This work was supported in part by the Challengeable Future Defense Technology Research and Development Program of Agency for Defense Development, in 2022, under Grant 915027201; and in part by Samsung Electronics Company Ltd., under Grant IO201210-07976-01.

ABSTRACT In the constrained computing environments such as mobile device or satellite on-board system, various computational factors of hardware resource can restrict the processing of deep learning (DL) services. Recent DL models such as satellite image analysis mainly require larger resource memory occupation for intermediate feature map footprint than the given memory specification of hardware resource and larger computational overhead (in FLOP) to meet service-level objective in the sense of hardware accelerator. As one of the solutions, we propose a new method of controlling the layer-wise channel pruning in a single-shot manner that can decide how much channels to prune in each layer by observing dataset once without full pretraining. To improve the robustness of the performance degradation, we also propose a layer-wise sensitivity and formulate the optimization problems for deciding layer-wise pruning ratio under target computational constraints. In the paper, the optimal conditions are theoretically derived, and the practical optimum searching schemes are proposed using the optimal conditions. On the empirical evaluation, the proposed methods show robustness on performance degradation, and present feasibility on DL serving under constrained computing environments by reducing memory occupation, providing acceleration effect and throughput improvement while keeping the accuracy performance.

INDEX TERMS Single-shot pruning, channel pruning, lottery ticket hypothesis, DL model compression.

I. INTRODUCTION

Recent advances in deep learning (DL) models have achieved remarkable analysis performance in many computer vision tasks [1], [2]. Since the recent convolutional neural networks (CNNs) have grown in depth and complexity in pursue of high analysis performance, it becomes a challenge to deploy DL models on constrained computing environments such as mobile device or satellite on-board system.

As one of the solutions, pruning can reduce computational overhead and resource occupation of DL models, which enables deploying in constrained computing resources. The state-of-the-art pruning schemes [3], [4] attempt to decide weight parameters to prune in single-shot manner without pretraining that requires additional huge computational overhead. However, these single-shot based pruning schemes are mainly targeted on weight pruning, that can not directly achieve reduction in computational resource occupation of DL models without sparsity-aware designed hardware or software; even such sparsity-aware hardware or software are provided, they yield only a small effect on practical DL acceleration [5].

Therefore, it can be considered to apply the single-shot based channel pruning that can achieve reduction on computational resource occupation and acceleration on processing DL model directly by removing output channels of each layer for constrained computing environments. There are three main issues on channel pruning the DL model in a single-shot

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

manner for constrained computing environments: 1) how to adapt single-shot weight pruning to a channel pruning scheme, 2) risk of pruning a whole layer in channel pruning with global searching scheme, and 3) varied aspects on layer-wise pruning sensitivity and layer-wise computational properties.

To solve the problems, we propose a new scheme of layer-wise channel pruning in a single-shot manner that is robust to the performance (accuracy) degradation while meeting the computational constraints of target hardware resource. Our main contributions can be summarized as follows.

- Firstly, theoretical analyses on validity of two possible single-shot channel pruning criteria adapted from single-shot weight pruning scheme are conducted, and the valid one is defined as channel sensitivity.
- We also observe that pruning channels with globally searching scheme under lottery ticket hypothesis [6] is prone to remove a whole layer by difference on layer-wise channel sensitivity scale, and propose a layer-wise sensitivity to regulate critical performance degradation caused by excessive pruning on a certain layer.
- Finally, we formulate the optimization problem that decides layer-wise pruning ratios to minimize sensitivity score of target model while meeting the computational constraints of target hardware resource. We derive the optimal conditions, and propose the practical methods to search the optimal solutions that enable to consider diverse layer-wise aspects of pruning sensitivity and computational characteristics.

In the paper, empirical evaluation of the proposed methods is conducted on various datasets and network models. The proposed methods show improved robustness on performance degradation and feasibility on DL serving with its accelerating effect.

II. RELATED WORK

A. DL PROCESSING ON CONSTRAINED COMPUTING ENVIRONMENTS

Applications like personalized service on mobile device [7], [8], anomaly detection from IoT device [9], and satellite imagery analysis [10] on on-board processing system [11] require deploying DL models on constrained computing environment. As one of use cases, Cloudscout [11] deploys the custom designed CNN for nanosatellite to select eligible data by detecting cloud as binary masking form. As the available hardware resources and power budget are limited in such environment, light-weight DL model is designed by constructing short and small CNN, but the light-weight DL model inevitably shows lower performance than the deeper and wide CNNs in general [12].

The main computational constraints of deploying such deep CNNs under limited hardware resources are resource memory occupation and computation overhead (which is generally quantified in the number of floating point operations (FLOP)). Since the size of the target input image grows up on the recent practical applications, requirement of the resource memory occupation size for intermediate feature maps in CNN mainly exceeds the given hardware memory size [13], [14]. For example, as shown in Fig. 1, deploying Faster-RCNN [2] model with 1k x 1k input requires about 9GB resource memory occupation for memory footprint of intermediate feature maps and about 3TFLOP of computational overhead, however, the hardware accelerators for constrained computing environments like NVIDIA TX-1 [15] or Xilinx VC707 [16] can only accommodate 4GB memory at maximum and can achieve about 60∼500GFLOPS for DL inference processing which is far insufficient to compute within seconds level.

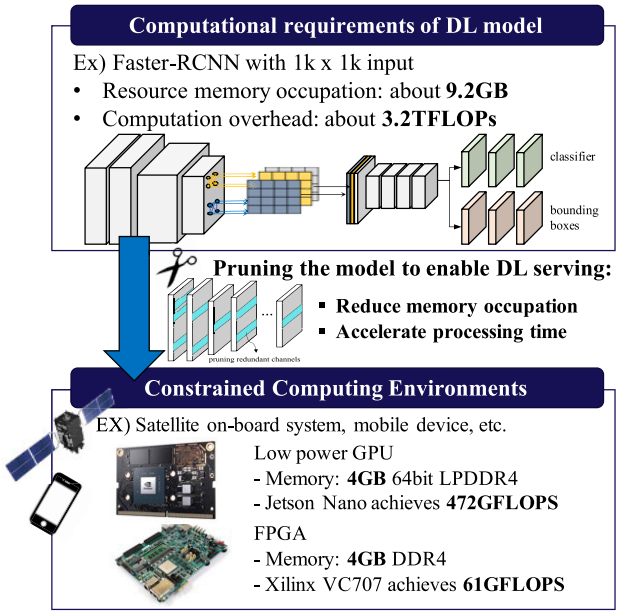

FIGURE 1. Necessity of pruning for serving DL model on constrained computing environments.

B. SINGLE-SHOT WEIGHT PRUNING

According to the time point when to prune the model, research on pruning can be divided into two main branches: (1) a form of pruning from pretrained state, and (2) a form of pruning from initial state by observing dataset only once. The schemes of pruning from pretrained state [17], [18] inevitably suffer performance degradation from the original pretrained model even though the further fine-tuning is conducted. A recent study observes the lottery ticket hypothesis [6] that there exists the sparse trainable subnetworks at initialization (called as winning ticket). However, finding this winning ticket from pretrained state requires additional computing overhead of pretraining the original model before retraining the pruned model. Therefore, recent works [3], [4] propose the single-shot based weight pruning criterion that can search weight parameters to prune in initial state without pretraining full iterations.

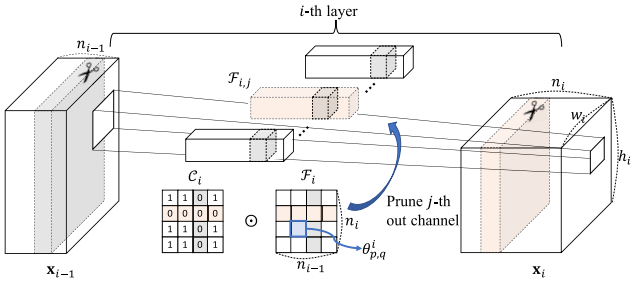

FIGURE 2. Illustration of pruning a output channel in a convolutional layer.

C. CHANNEL PRUNING

However, the weight pruning itself can not directly achieve reduction on resource memory occupation and computation overhead without sparsitiy-aware designed hardware and software [5], and its effect is relatively small on general GPU resources [19] comparing to the degree of pruning. Accordingly, channel pruning [20], [21] can be applied to directly reduce the resource memory occupation and computation overhead of DL model by removing output channels in each layer. As the channel pruning is more risky to the performance degradation by removing a bunch of weights linked to output channel at once, most studies [17], [21] mainly targeted on how efficiently recover the performance degradation for pruning from pretrained state than how to select efficient channels to prune.

On the other side, some studies [22], [23], [24] attempted to prune the channels as a form of neural architecture search (NAS) by constructing the loss function that considers the both task performance and model cost (*e.g.,* memory or FLOP) together. However, such NAS-based methods inevitably requires the additional heavy overhead of training the NAS model itself before training the pruned model.

To overcome these limitations, in this paper, we aim to propose an efficient channel pruning scheme for constrained computing environments in a single-shot manner.

III. SINGLE-SHOT BASED LAYER-WISE CHANNEL PRUNING WITH COMPUTATIONAL CONSTRAINTS

In this section, we firstly introduce how to adapt single-shot based weight pruning criterion to channel pruning scheme, and check its validity theoretically. Then, the layer-wise sensitivity is introduced to regulate excessive pruning on a certain layer, and we formulate the optimization problem for minimizing the whole pruning sensitivity score of the model. We then derive the optimal condition and propose a practical optimum searching method for each constraint respectively in the following subsections.

A. SINGLE-SHOT BASED CHANNEL PRUNING

Let n_i, h_i, and w_i denote the number of output channels, height and width of the output feature map in ith layer, respectively. As shown in Fig. 2, from the input x_{i-1}, a convolution layer conducts $n_i \cdot n_{i-1}$ convolution operations, and pruning can be represented as masking the filters denoted by $\mathcal{C}_i \odot \mathcal{F}_i$, where $\odot$ denotes element-wise product, and $\theta^i_{p,q}$ denotes each kernel parameter for linking pth output channel and qth input channel in ith layer. In the following, let denote $\mathcal{C}$ as a set of whole masking matrices in the network, and denote $\mathcal{C} \odot \mathcal{F}$ as masking in each layer by $\mathcal{C}_i \odot \mathcal{F}_i$.

TABLE 1. Computation time required for pretraining and retraining over channel pruning methods.

	Proposed	Lottery	Pt + Ft	Ori.
Accuracy (%)	81.90	84.29	79.05	80.00
Pretrain time (s)	15.6	1736.8	1736.8	1736.8
Retrain time (s)	850.5	992.5	176.9	-
Total time (s)	866.1	2729.3	1913.7	1736.8

Table 1 shows the motivation of our work for single-shot based channel pruning. In the table, the training elapsed time to achieve the best test accuracy among 160 epochs on each pruning methods is measured. The test environment is same with that of Table 2, 3 described in Section IV-A. The target model is ResNet-101 for satellite imagery dataset which is the most practical application.

As shown in the result, the conventional methods that prune from the pretrained state then conduct fine-tuning (denoted as Pt + Ft, and the fine-tuning time is presented as retraining time) [21] or then conduct retraining with re-initialization (denoted as Lottery) [6] show much more computational burden on total of pretraining (single-shot observing stage for the proposed method) and retraining (or fine-tuning) stage. Such overheads even exceed the overhead of training the original full model. Accordingly, we target to address the single-shot based channel pruning that can achieve acceleration on both pretraining stage and retraining (or fine-tuning) stage not only on the inference phase.

1) CHANNEL SENSITIVITY

As lack of study on single-shot based channel pruning criterion, we analyze two possible adapting ways for channel pruning from single-shot based weight pruning criterion. First possible criterion is summing up the single-shot based weight masking effect along the weights linked to the target output channel. Let L denotes a loss function (*e.g.,* cross-entropy loss), $\mathcal{D}$ denotes given dataset. When observing this criterion theoretically, from one of the single-shot based weight pruning criterion [3], weight-wise sensitivity is given as $s^i_{p,q} = \frac{|g^i_{p,q}(\mathcal{F};\mathcal{D})|}{\sum_{i'}\sum_{p'}\sum_{q'}|g^{i'}_{p',q'}(\mathcal{F};\mathcal{D})|}$ where $g^i_{p,q} = \frac{\partial L(\mathcal{C}\odot\mathcal{F};\mathcal{D})}{\partial c^i_{p,q}}|_{\mathcal{C}=1} \approx L(\mathcal{F};\mathcal{D}) - L(\mathcal{F};\theta^i_{p,q} = 0,\mathcal{D})$, and i, p, q denote index of layer, output channel, input channel respectively. However, pruning with this criterion can not guarantee being equal to solving the empirical risk minimization problem of network in the finite hypothesis space of pruning, which is stated as following property.

Property 1: Pruning a channel by $\sum_q s^i_{p,q}$ (where $s^i_{p,q}$ is single-shot based weight sensitivity in [3]) can not guarantee being equal to solving empirical risk minimization (ERM)

problem of the neural network in the finite hypothesis space of pruning.

Proof: Consider the weight-wise pruning criterion on SNIP [3],

$$s^i_{p,q} = \frac{|g^i_{p,q}(\mathcal{F}; \mathcal{D})|}{\sum_{i'} \sum_{p'} \sum_{q'} |g^{i'}_{p',q'}(\mathcal{F}; \mathcal{D})|}. \tag{1}$$

As mentioned in main content, i is index of layer, p or j denotes index of output channel, and q denotes index of input channel. Choosing a channel to prune by $\sum_q s^i_{p,q}$ as a form of summing up weight-wise scores linked to the target channel [21] is written as:

$$\arg\min_{i,j} \sum_q s^i_{p,q}. \tag{2}$$

Consider the pruning a channel by conducting element-wise product on filters with masking matrix $(\mathcal{C}_i \odot \mathcal{F}_i)$ is equal to finding $\mathcal{F}_i$ with which channel to be pruned $(\theta^i_{p,q} = 0, \forall q)$ in the finite hypothesis space H where the space consists of all possible channel pruning cases. Therefore, the problem of (2) can be written as:

$$= \arg\min_{\mathcal{F}_i|_{\theta^i_{j,q}=0,\forall q} \in H} \sum_q s^i_{p,q}. \tag{3}$$

Applying $\sum_q s^i_{p,q} = \frac{\sum_q |L(\mathcal{F};\mathcal{D}) - L(\mathcal{F};\theta^i_{p,q}=0,\mathcal{D})|}{\sum_{i'} \sum_{p'} \sum_{q'} |g^{i'}_{p',q'}(\mathcal{F};\mathcal{D})|}$ to (3), denominator term and $L(\mathcal{F}; \mathcal{D})$ term in numerator are constant with regard to j (wrapped as $\mathcal{F}_i|_{\theta^i_{j,q}=0,\forall q}$). As the cross entropy is usually used as loss function $L()$, assume $L > 0$. Then, the problem in (3) can be written as:

$$= \arg\min_{\mathcal{F}_i|_{\theta^i_{j,q}=0,\forall q} \in H} \frac{\sum_q |\alpha - L(\mathcal{F}; \theta^i_{p,q} = 0, \mathcal{D})|}{\beta} \tag{4}$$

$$\neq \arg\min_{\mathcal{F}_i|_{\theta^i_{j,q}=0,\forall q} \in H} L(\mathcal{F}; \theta^i_{p,q} = 0, \mathcal{D}), \tag{5}$$

where $\alpha, \beta \in \mathbb{R}$, and $\alpha > 0$, $\beta \geq 0$ are constants. Therefore, this problem equation does not guarantee equal to problem of empirical risk minimization (5). □

Alternatively, we define channel sensitivity by transforming single-shot based weight pruning criterion [3] to the form of direct output channel masking effect as follows.

$$s^i_j = \frac{|g^i_j(\mathcal{C} \odot \mathcal{F}; \mathcal{D})|}{\sum_{i'} \sum_{j'} |g^{i'}_{j'}(\mathcal{C} \odot \mathcal{F}; \mathcal{D})|}, \tag{6}$$

where $g^i_j(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) = \frac{\partial L(\mathcal{C} \odot \mathcal{F}; \mathcal{D})}{\partial c^i_j}|_{\mathcal{C}=1} \approx L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) - L(\mathcal{C}|_{c^i_j=0} \odot \mathcal{F}; \mathcal{D})$.

Pruning a channel with the defined criterion (s^i_j) can be stated as an equivalent problem of minimizing the empirical loss on the network, as shown in the following Lemma 1.

Lemma 1: If $L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D}) \geq L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) \geq 0, \forall (i,j) \in \{(i,j) | c^i_j = 1, \forall c^i_j \in \mathcal{C}\}$, *pruning a channel by* s^i_j *is equal to solving empirical risk minimization (ERM) problem of the neural network in the finite hypothesis space.*

Proof: In the score formulation transformed for channel pruning from weight-wise score of [3], impact of pruning j-th output channel in i-th layer (ΔL^i_j) is approximated to $g^i_j(\mathcal{C} \odot \mathcal{F}; \mathcal{D})$ for calculation efficiency in implementation as follows.

$$\begin{aligned} \Delta L^i_j(\mathcal{F}; \mathcal{D}) &= L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) - L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D}) \\ &\approx g^i_j(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) = \frac{\partial L(\mathcal{C} \odot \mathcal{F}; \mathcal{D})}{\partial c^i_j}|_{\mathcal{C}=1} \end{aligned} \tag{7}$$

Applying this to s^i_j of (6) obtains

$$s^i_j = \frac{|g^i_j(\mathcal{C} \odot \mathcal{F}; \mathcal{D})|}{\sum_{i'} \sum_{j'} |g^{i'}_{j'}(\mathcal{C} \odot \mathcal{F}; \mathcal{D})|} \tag{8}$$

$$\approx \frac{|L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) - L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D})|}{\sum_{i'} \sum_{j'} |L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) - L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D})|}. \tag{9}$$

Therefore, choosing a channel to prune by s^i_j is written as:

$$\arg\min_{i,j} s^i_j \tag{10}$$

$$= \arg\min_{i,j} \frac{|L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) - L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D})|}{\sum_{i'} \sum_{j'} |L(\mathcal{C} \odot \mathcal{F}; \mathcal{D}) - L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D})|}, \tag{11}$$

where denominator term and $L(\mathcal{C} \odot \mathcal{F}; \mathcal{D})$ term in numerator are constant with regard to i, j.

As the cross entropy is usually used as a loss function $L(\cdot)$, assume $L > 0$. Then, when denoting $L(\mathcal{C} \odot \mathcal{F}; \mathcal{D})$ in numerator and denominator term as constant $\alpha, \beta \in \mathbb{R}$, $\alpha, \beta \geq 0$ respectively, the equation of (11) becomes:

$$= \arg\min_{i,j} \frac{|\alpha - L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D})|}{\beta}. \tag{12}$$

This implies that, when $L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D}) \geq \alpha \geq 0$, the problem becomes equal to empirical risk minimization problem of the neural network in the finite hypothesis space H where the space consists of all possible channel pruning cases. Therefore, in such condition, (12) becomes:

$$= \arg\min_{i,j} L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D}) \tag{13}$$

$$= \arg\min_{\mathcal{F} \in H} L(\mathcal{F}; \mathcal{D}). \tag{14}$$

□

The exception condition $L(\mathcal{C} \odot \mathcal{F}; c^i_j = 0, \mathcal{D}) < L(\mathcal{C} \odot \mathcal{F}; \mathcal{D})$ also means the case where pruning reduces the loss from non-pruned state. Moreover, as solving ERM guarantees probably approximately correct (PAC) bound [25], under the same condition in Lemma 1, pruning a channel by s^i_j also guarantees PAC bound and its estimation error is upper bounded. As minimizing empirical loss tends to result in minimizing test error except overfitting, it can be expected

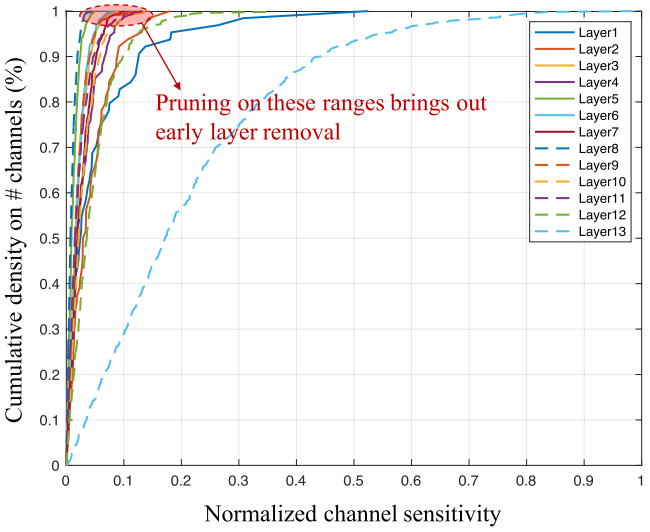

FIGURE 3. Cumulative density on the number of channels with regard to the channel sensitivity value s_j^i in each layer of VGG-16 with CIFAR-10, which shows the risk on early layer removal.

that pruning with the criterion from (6) can contribute to minimize test error.

2) LAYER-WISE SENSITIVITY

However, the defined single-shot based channel sensitivity reveals distributional difference over layers as shown in Fig. 3. Since the channel sensitivity scores are distributed on different bound for each layer and the bounds are different over layers, it is prone to prune a certain whole layer as the degree of pruning increases on global searching scheme. For example, as shown in Fig. 3, trying to prune the model with the normalized channel sensitivity criteria value of 0.2 just invokes removing whole channels of layers (Layer 8, Layer5, etc.). Such risk of pruning a whole layer results in critical performance degradation, and the early layer removal worsens the robustness of performance degradation with regard to the compression rate. The detailed evaluation results of such risk are also presented in Section IV.

To overcome such risk, in the methods of pruning from pretrained state, layer-wise pruning sensitivity curve that reveals layer-wise performance degradation with respect to the degree of pruning can be easily obtained from profiling process, and this layer-wise pruning sensitivity curve can be used to regulate pruning a certain layer excessively [21]. However, in the single-shot based pruning scheme, it is practically impossible to obtain the layer-wise pruning sensitivity curves as it can not directly obtain performance information unless training on each profiling points with full iterations.

Instead, in order to replace the task performance degradation profiled curve that cannot be acquired in the single-shot pruning scheme, we define a layer-wise sensitivity LS_i for single-shot pruning scheme in the inverse form of sum on total channel sensitivity scores on remaining channels after pruning (with $\mathcal{C}$) at i-th layer as follows:

$$LS_i(\mathcal{C}) = \frac{1}{\sum_{j'=0}^{n_i-1} s_{j'}^i - \sum_{j' \in \{j' | c_{j'}^i = 0, \forall c_{j'}^i \in \mathcal{C}_i\}} s_{j'}^i}. \tag{15}$$

The proposed layer-wise sensitivity (LS_i) basically follows the theoretical foundation (Lemma 1) of the global channel sensitivity but additionally regulates the excessive pruning on a certain layer, by constructing LS_i as a multiplicative inverse of the sum on the remaining global channel sensitivity in each layer. In other words, in terms of each layer, minimizing LS_i makes pruning the channels with small global channel sensitivity first in each layer with its theoretical foundation. In terms of inter-layer, excessive pruning on a certain layer invokes excessive increase on LS_i, and therefore the proposed layer-wise sensitivity regulates risk of early layer removal by trying to minimize LS_i over the layers.

Accordingly, in order to apply such mechanism to all the layers of target network, minimization of the product over all LS_i is set as the optimization objective (minimize $\prod_i LS_i$). Consequently, the channels with small global channel sensitivity s_j^i are pruned in each layer basically under its theoretical foundation (Lemma 1), while such objective (minimize $\prod_i LS_i$) also prevents $LS_i = \infty$ (*i.e.,* pruning a whole layer) for any layer i, which results in being robust on the task performance degradation.

Therefore, based on this property, we propose a global pruning scheme (`s-ls-global`) with layer-wise sensitivity as described in Algorithm 1. The proposed scheme searches channels to prune in whole network globally by selecting a channel that shows minimum $s_j^i \cdot LS_i(\mathcal{C}; c_j^i = 0)$ score channel-by-channel, iteratively updating layer-wise sensitivity, where layer-wise sensitivity term on score regulates to select a channel in excessively pruned layer on current searching space.

Algorithm 1 Global Channel Pruning Scheme With Layer-Wise Sensitivity: `s-Ls-global()`

Input: Target pruning ratio pr

$c_j^i \leftarrow 1$ for $\forall i, j$
calculate s_j^i for $\forall i, j$
update $LS_i(\mathcal{C})$ for $\forall i$
while $\sum_i \sum_j c_j^i > \sum_i n_i \cdot (1 - pr)$ **do**
  $(i^*, j^*) \leftarrow \arg\min_{(i,j) \in \{(i,j) | c_j^i = 1\}} s_j^i \cdot LS_i(\mathcal{C}; c_j^i = 0)$
  $\mathcal{C} \leftarrow \mathcal{C}|_{c_{j^*}^{i^*} = 0}$
  update $LS_{i^*}(\mathcal{C})$
end while
return $\mathcal{C}$

B. LAYER-WISE ADAPTIVE PRUNING UNDER COMPUTATIONAL CONSTRAINTS

As mentioned above, resource memory occupation size and computation overhead (FLOP) of the DL model are mainly

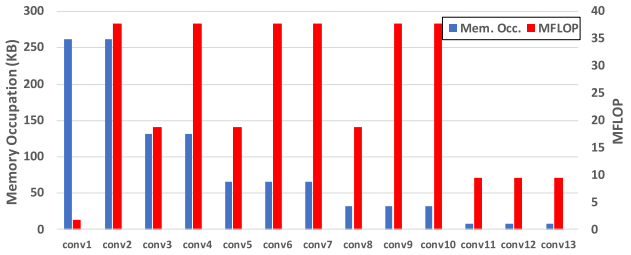

FIGURE 4. Layer-wise aspects of resource memory occupation size and computation overhead on VGG-16 with CIFAR-10.

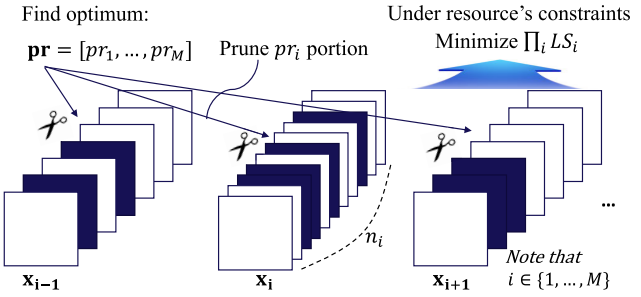

FIGURE 5. Illustration of layer-wise adaptive pruning, the different pruning ratio is allocated to each layer considering its layer-wise sensitivity and memory/FLOP characteristics.

limited by hardware resources. To meet such computational constraints by the proposed `s-ls-global` of Algorithm 1, the appropriate pruned model can be searched by incrementally controlling the target pruning ratio up to meet the computational constraints. However, this scheme does not consider the layer-wise aspects of computational characteristics, where both the resource memory occupation and computation overhead (FLOP) have different aspect each other and also over layers as shown in Fig. 4. Accordingly, inefficient pruning case that just selecting links of low impact on loss with low reduction effect of computational constraints can occur, and it can disrupt the robustness of performance degradation with regards to each computational constraint.

Therefore, from the idea of aforementioned layer-wise sensitivity, we additionally propose a layer-wise adaptive pruning scheme of determining the pruning ratio of each layer that minimize network sensitivity $\prod_i LS_i(pr_i)$ while meeting the computational constraints as shown in Fig. 5. Let pr_i denotes pruning ratio on i-th layer that is equivalent to $pr_i = 1 - \sum_j c_j^i / n_i$, $LS_i(pr_i)$ denotes layer-wise sensitivity score of i-th layer pruned with degree of pr_i by channel sensitivity s_j^i, and $\mathbf{pr} = [pr_1, \ldots, pr_M]$ denotes pruning policy for a whole network with M layers. In turn, the resource memory occupation constraint (mainly occupied by memory footprint of intermediate feature maps on large input image) and the computation overhead constraint can be written as following equations respectively.

$$\sum_i |\mathbf{x}_i|_0 n_i (1 - pr_i) \le r_{mem} \sum_i |\mathbf{x}_i|_0 n_i \quad (16)$$

$$\sum_i (1 - pr_{i-1})(1 - pr_i) CO_i \le r_F \sum_i CO_i \quad (17)$$

where r_{mem}, r_F denotes the target constraint level to meet hardware resource requirements (*i.e.,* remaining ratio comparing to requiring quantity of original full model), and CO_i denotes quantity of computation overhead in FLOP at i-th layer. In the following, as $LS_i(pr_i)$ is convex, we can further construct the optimization problem for each computational constraint respectively, and address how to solve them.

1) OPTIMAL LAYER-WISE PRUNING RATIO FOR RESOURCE MEMORY OCCUPATION CONSTRAINT

First, it is targeted to find the optimal layer-wise pruning ratio $\mathbf{pr}$ that minimize $\prod_i LS_i$ subject to satisfy resource memory occupation constraint of (16). The optimal condition for this optimization problem can be derived as following Theorem 1.

Theorem 1: The optimal layer-wise pruning ratios $\mathbf{pr}$ *that minimize* $\prod_i LS_i$*, while meeting the memory occupation constraint of* (16)*, satisfy the following condition*:

$$\frac{LS_1'(pr_1)}{LS_1(pr_1)} \frac{1}{|\mathbf{x}_1|_0 n_1} = \ldots = \frac{LS_M'(pr_M)}{LS_M(pr_M)} \frac{1}{|\mathbf{x}_M|_0 n_M}, \quad (18)$$

where LS_i' *denotes the derivative.*

Proof: The optimization problem targeted to solve is finding optimal layer-wise pruning ratio $\mathbf{pr}$ that minimize $\prod_i LS_i(pr_i)$ subject to (16). The corresponding dual problem is written as:

$$\mathcal{L}(\mathbf{pr}, \lambda) = \prod_{i=1}^{M} LS_i(pr_i) - \lambda \sum_{i=1}^{M} |\mathbf{x}_i|_0 n_i (1 - pr_i - r_{mem}), \quad (19)$$

where M denotes the total number of layers in the network.

The partial derivatives with regard to the pruning ratio of a certain layer pr_k gives:

$$\frac{\partial \mathcal{L}}{\partial pr_k} = \prod_{i=1}^{M} LS_i(pr_i) \frac{LS_k'(pr_k)}{LS_k(pr_k)} + \lambda |\mathbf{x}_i|_0 n_k = 0, \quad \forall k \in \{1, \ldots, M\}. \quad (20)$$

From (20), it can be expand as:

$$-\lambda = \prod_{i=1}^{M} LS_i(pr_i) \frac{LS_k'(pr_k)}{LS_k(pr_k)} \frac{1}{|\mathbf{x}_i|_0 n_k}, \quad \forall k \in \{1, \ldots, M\}. \quad (21)$$

Therefore, as $\prod_{i=1}^{M} LS_i(pr_i)$ term in (21) has same value for $\forall k$, following equation holds.

$$\frac{LS_1'(pr_1)}{LS_1(pr_1)} \frac{1}{|\mathbf{x}_1|_0 n_1} = \ldots = \frac{LS_M'(pr_M)}{LS_M(pr_M)} \frac{1}{|\mathbf{x}_M|_0 n_M} \quad (22)$$

□

When denoting $f_{mem,i}(pr_i) = \frac{LS_i'(pr_i)}{LS_i(pr_i)} \frac{1}{|\mathbf{x}_i|_0 n_i}$, the optimal pruning ratios can be practically searched by incrementally varying the threshold variable ρ from zero to top until meeting computational constraints at first as described in Algorithm 2, where ϵ denotes incrementing unit. After the

Algorithm 2 Optimal Layer-Wise Pruning Ratio Searching Scheme for Memory Constraint: `Mem-opt()`

Input: Target memory constraint level r_{mem}

$c_j^i \leftarrow 1$ for $\forall i, j$
calculate s_j^i for $\forall i, j$
$\rho \leftarrow 0$
pr $\leftarrow [0, \ldots, 0]$
while $\sum_i |x_i|_0 n_i (1 - pr_i) > r_{mem} \sum_i |x_i|_0 n_i$ **do**
  $\forall i, pr_i \leftarrow f_{mem}^{-1}(\rho)$
  $\rho \leftarrow \rho + \epsilon$
end while
return pr

Algorithm 3 Optimal Layer-Wise Pruning Ratio Searching Scheme for FLOP Constraint: `Flop-opt()`

Input: Target FLOP constraint level r_F

$c_j^i \leftarrow 1$ for $\forall i, j$
calculate s_j^i for $\forall i, j$
pr $\leftarrow [0, \ldots, 0]$
while $\sum_i (1 - pr_{i-1})(1 - pr_i) CO_i > r_F \sum_i CO_i$ **do**
  $\rho \leftarrow f_{FLOP}(pr_1)$
  for i in $\{2, \ldots, M\}$ **do**
    $pr_i \leftarrow f_{FLOP,i}^{-1}(\rho)$
  end for
  $pr_1 \leftarrow pr_1 + \epsilon$
end while
return pr

layer-wise pruning ratio is determined, pruning by channel sensitivity s_j^i is conducted on each layer with its determined pruning ratio pr_i.

Actually, the proposed pruning method for optimizing on activation memory occupation can also be modified to support the optimization of weights memory occupation. When denoting the size (memory occupation) of the weight parameters in i-th layer as w_i, then the size of weight parameter is reduced in proportion to the both pruning ratios (pr_i, pr_{i-1}) of current layer and the previous layer as $(1 - pr_{i-1})(1 - pr_i)w_i$. Accordingly, the weights memory occupation constraints that is in the similar form of (17) is derived, and the optimization can be derived likewise. However, for the conciseness of the paper, only the optimization on memory occupation by the intermediate feature maps is considered in this paper, and the optimization on the weights memory occupation is remained as future work.

2) OPTIMAL LAYER-WISE PRUNING RATIO FOR FLOP CONSTRAINT

For the optimization on computation overhead (FLOP) constraint, the goal is to find optimal layer-wise pruning ratio **pr** that minimize $\prod_i LS_i$ such that satisfies FLOP constraint of (17). Likewise, the optimal condition for this optimization problem for FLOP constraint can be derived as following Theorem 2. In the theorem, pr_0 denotes pruning ratio for input image channel which is fixed to zero.

Theorem 2: The optimal layer-wise pruning ratios **pr** *that minimize* $\prod_i LS_i$*, while meeting the computation overhead constraint of* (17)*, satisfy the following condition*:

$$\frac{LS_1'(pr_1)}{LS_1(pr_1)} \frac{1}{CO_1(1-pr_0)} = \ldots = \frac{LS_M'(pr_M)}{LS_M(pr_M)} \frac{1}{CO_M(1-pr_{M-1})} \quad (23)$$

Proof: The optimization problem targeted to solve is finding optimal layer-wise pruning ratio **pr** that minimize $\prod_i LS_i(pr_i)$ subject to (17). The corresponding dual problem is written as:

$$\mathcal{L}(\mathbf{pr}, \lambda) = \prod_{i=1}^{M} LS_i(pr_i) - \lambda \sum_{i=1}^{M} CO_i((1 - pr_i)(1 - pr_{i-1}) - r_F), \quad (24)$$

where M denotes the total number of layers in the network.

Partial derivatives gives:

$$\frac{\partial \mathcal{L}}{\partial pr_k} = \prod_{i=1}^{M} LS_i(pr_i) \frac{LS_k'(pr_k)}{LS_k(pr_k)} + \lambda CO_k(1 - pr_{k-1}) = 0, \quad \forall k \in \{1, \ldots, M\}. \quad (25)$$

From (25),

$$-\lambda = \prod_{i=1}^{M} LS_i(pr_i) \frac{LS_k'(pr_k)}{LS_k(pr_k)} \frac{1}{CO_k(1 - pr_{k-1})}, \quad \forall k \in \{1, \ldots, M\}. \quad (26)$$

Therefore, as $\prod_{i=1}^{M} LS_i(pr_i)$ term has same value for $\forall k$, following equation holds, where pr_0 denotes pruning ratio of input image channel that is fixed to zero ($pr_0 = 0$):

$$\frac{LS_1'(pr_1)}{LS_1(pr_1)} \frac{1}{CO_1(1 - pr_0)} = \ldots = \frac{LS_M'(pr_M)}{LS_M(pr_M)} \frac{1}{CO_M(1 - pr_{M-1})}. \quad (27)$$

□

From the optimal condition, as $pr_0 = 0$, according to the value of pr_1, the others $(pr_2, \ldots, pr_L)$ are decided deterministically in sequential, and denote $f_{FLOP}(pr_i) = \frac{LS_i'(pr_i)}{LS_i(pr_i)} \frac{1}{CO_i(1-pr_{i-1})}$. The optimal pruning ratios can be practically searched by incrementally varying the pruning ratio of the first layer pr_1 from zero to one until meeting computational constraints at first as described in Algorithm 3. Likewise, after the layer-wise pruning ratio is determined, pruning by channel sensitivity s_j^i is conducted on each layer with its determined pruning ratio.

IV. EVALUATION

In this section, the proposed methods are evaluated in two main aspects: robustness of performance degradation on pruning and feasibility on DL serving. Through the evaluation, the robustness of performance degradation on various test cases is observed, and the acceleration effect on computing resources is measured.

A. EXPERIMENTAL SETTINGS

The proposed methods are compared with conventional channel pruning methods on three cases: VGG-16 [26] with CIFAR-10 dataset [27], wide ResNet(WRN)-18 [1] with Caltech101 dataset [28], and ResNet-101 [29] with UC Merced satellite imagery dataset [30]. The detail settings for training each case are described as follows.

1) VGG-16 WITH CIFAR-10 DATASET

One of cases, the evaluation is conducted on modified VGG-16 architecture where an average pooling layer is attached after the last convolutional layer, and only a single fully connected layer with 512 input channel is connected at the end for CIFAR-10 from the original VGG-16 architecture [26]. The model is trained by using SGD with momentum of 0.9, batch size of 128 and the weight decay rate of 0.0001, and train 160 epochs in total. The initial learning rate is configured to 0.1 and decayed by 0.1 at every 60 epochs. The standard data augmentation (i.e., translation up to 4 pixels for fitting to VGG-16, random horizontal flip and normalization) is applied.

2) WRN-18 WITH Caltech101 DATASET

For WRN-18 with Calteach101 dataset, WRN-18 architecture [1] is applied, where only a single fully connected layer at the end is modified with 101 output channels for Caltech101 dataset. The model is trained by using SGD with momentum of 0.9, batch size of 32 and the weight decay rate of 0.0001, and train 80 epochs in total. The initial learning rate is configured to 0.1 and decayed by 0.1 at 60 epoch. For data augmentation, only resizing of the input data to 224×224 size is applied. In the dataset [28], 90% of total dataset is split to training data and remaining 10% is used for test data.

3) ResNet-101 WITH UC MERCED SATELLITE IMAGERY DATASET

Likewise, ResNet-101 architecture [29] modifying only the last single fully connected layer with 21 output channels for UC Merced satellite imagery dataset is used as third case. The model is trained by using SGD with momentum of 0.9, batch size of 128 and the weight decay rate of 0.0001, and train 160 epochs in total. The initial learning rate is configured to 0.1 and decayed by 0.1 at every 60 epochs. Only resizing the input data to 256×256 size is applied to data augmentation. In the dataset [30], 90% of total dataset is split to training data and remaining 10% is used for test data.

B. ROBUSTNESS OF PERFORMANCE DEGRADATION

The robustness of task performance degradation on the proposed methods are evaluated over aforementioned 3 different cases. The task performance is evaluated on 20 sparsity levels, and the best top-1 test accuracy is measured as the performance of the model trained from re-initialized state of pruned model with respect to evaluation of lottery ticket hypothesis [6]. Aforementioned three proposed pruning methods (`s-ls-global`, `mem-opt`, `flop-opt`) are evaluated by comparing two conventional methods as listed:

- `snip-sum`: Adapting method of SNIP [3], the single-shot based weight pruning, to channel pruning scheme by summing up weight-wise scores linked to target channel [21].
- `LTH-ch`: Channel pruning form of evaluation in lottery ticket hypothesis [6] that prunes channels according to sum on magnitude of weight parameters linked to output channel. To observe comparison on as much similar training overhead as possible, single-step version of lottery ticket hypothesis [6] is applied where the pruning is conducted only once.

For `LTH-ch`, iterative pruning version of lottery ticket hypothesis [6] was also conducted, which requires much more training computation overhead than the single-step version. However, as the results of iterative pruning version show rather earlier layer removal than the single-step version in channel pruning scheme, only the results of single-step version (`LTH-ch`) are presented as representative one in the paper.

To evaluate the robustness of performance degradation on pruning, the shapes of performance (accuracy) with regards to three aspects are observed: *1) the number of total remaining channels*, *2) remaining resource memory occupation size*, and *3) remaining FLOP of pruned model*.

The evaluation on VGG-16 model with CIFAR-10 dataset is shown in Fig. 6. The results show that the proposed methods improves robustness of performance degradation comparing to the conventional methods. Especially, `s-ls-global` shows best performance on reducing the number of output channels, `mem-opt` shows the best performance on reducing resource memory occupation size, and `flop-opt` shows the best performance on reducing computation overhead (FLOP) that corresponds to the intent of each proposed method.

As an ablation study, in order to see the effect of proposed layer-wise sensitivity, further evaluations on other variant methods (`s-local`, `s-global`) are also conducted:

- `s-local`: Pruning each layer with same pruning ratio by channel sensitivity s_j^i in each layer.
- `s-global`: Searching channels to prune by channel sensitivity s_j^i globally in whole network.

As shown in the results (Fig. 6), `s-ls-global` shows more robust to performance degradation than `s-local` and `s-global` with the help of layer-wise sensitivity that regulates excessive pruning on a certain layer. This effect can be explicitly clarified by observing layer-wise pattern of remaining fractions as shown in Fig. 7. The layer-wise sensitivity smoothen the pruning pattern under high global pruning ratio to regulate pruning a certain layer excessively comparing patterns of `s-ls-global` and `s-global`. The proposed `flop-opt` shows tendency of pruning 9, 10-th layers more highly that contains higher FLOP than other layers as observed in Fig. 4. Likewise, `mem-opt` prunes front layers aggressively that contains larger resource memory

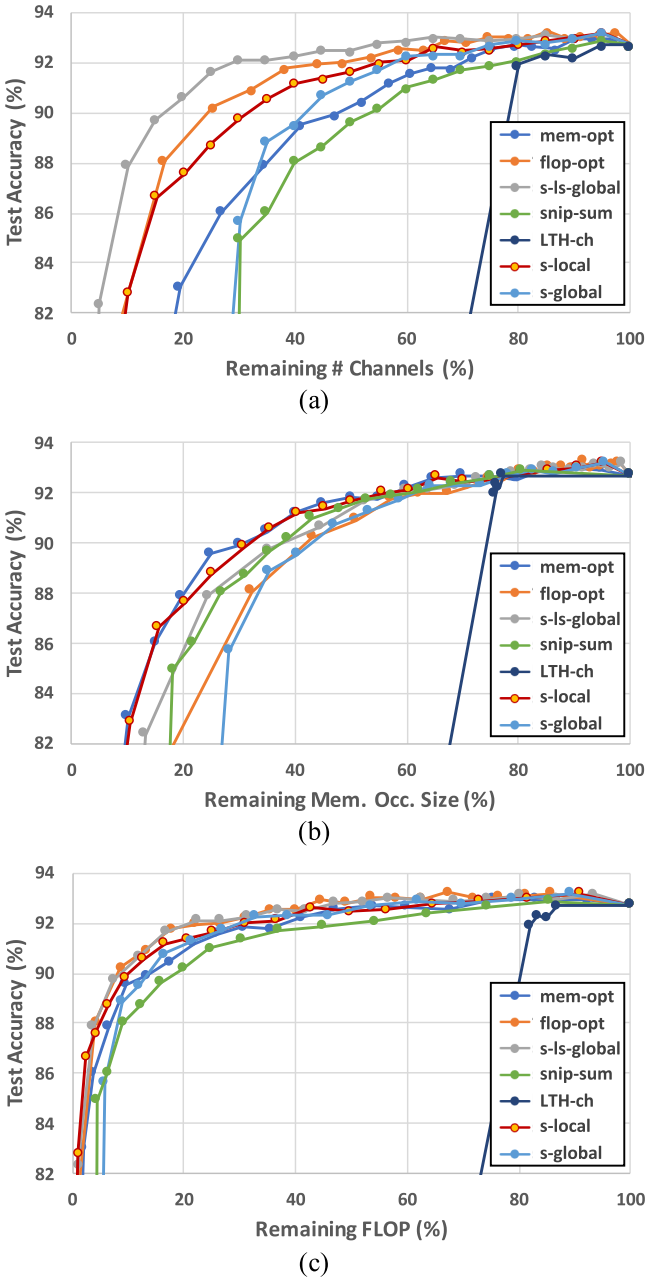

FIGURE 6. Test top-1 accuracy results of pruning methods on VGG-16 for CIFAR-10 with respect to *(a)* the number of remaining channels, *(b)* remaining resource memory occupation size, and *(c)* remaining FLOP of pruned model.

occupation size for intermediate feature maps. In addition, in accuracy results, pruning with channel sensitivity itself also shows higher performance than `snip-sum` and `LTH-ch` with regards to the number of remaining channels, as stated in theoretical validity analysis including Lemma 1.

Moreover, when comparing `mem-opt` and `s-ls-global` on the aspects to the number of remaining channels and resource memory occupation size, `mem-opt` shows lower robustness of performance degradation on reducing the number of channels, but shows higher robustness on reducing resource memory occupation size. This implies that the

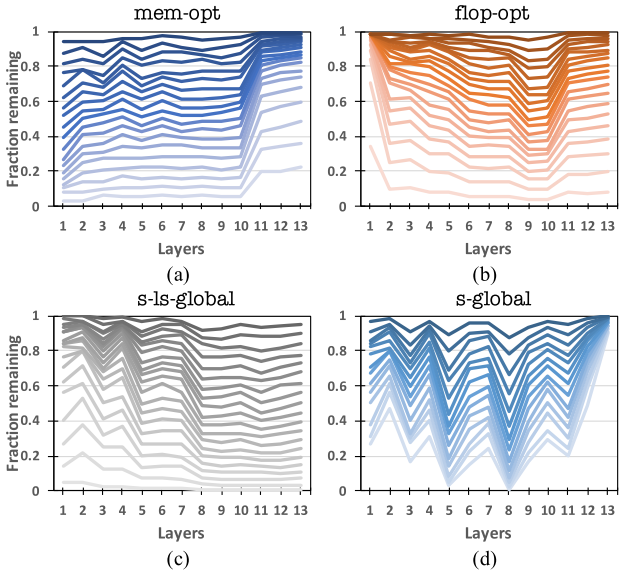

FIGURE 7. Fraction of remaining channels on each layer of VGG-16 for CIFAR-10 pruned by each comparing method ((a) `mem-opt`, (b) `flop-opt`, (c) `s-ls-global`, (d) `s-global`) on several overall degrees of pruning.

proposed `mem-opt` can select efficient channels containing high layer-wise memory occupation reducing effect even if its channel sensitivity is high. The similar effect also corresponds to comparison of `flop-opt` and `s-ls-global`.

Fig. 8 shows results on WRN-18 network with Caltech101 dataset. Likewise, in the results, the proposed methods show higher robustness of performance degradation on reducing the number of channels, resource memory occupation, computation overhead (FLOP). However, for reducing FLOP, `mem-opt` shows more robustness of performance degradation than `flop-opt`. It can be caused by difference on calculating effect of FLOP reduction (23) on convolutional layers at residual links in the network as we apply shared output channel masking on residual links and just calculate FLOP of them using only one pruning ratio of the most fore-headed previous layer among several candidate layers.

Evaluation on deeper residual network (ResNet-101) with satellite imagery dataset is also conducted. As shown in Fig. 9, the proposed methods show higher robustness of performance degradation on all reducing aspects. However, likewise to the case of WRN-18, `mem-opt` and `flop-opt` can not show their optimal decision on high reduction level of computational properties respectively as accumulation of the mis-calculation on each reducing property effect becomes higher on this network with much more residual links. Though, the proposed `s-ls-global` can highly endure performance degradation and can be applied instead of them on such case.

In Fig. 8 and 9, the layer removal occurs at an earlier sparsity than the result of CIFAR-10 when pruning with only the criterion of global channel sensitivity s_j^i (`s-global`), which shows more distinct risk of early layer removal on global channel sensitivity s_j^i. In the results, `s-global` shows comparable robustness of task performance degradation

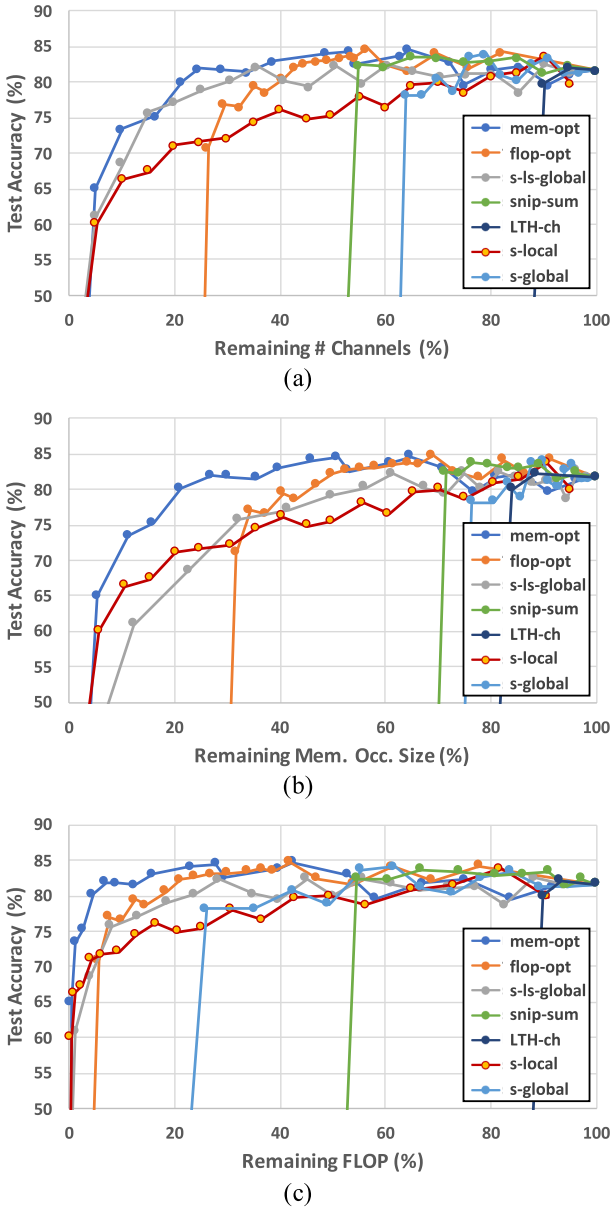

FIGURE 8. Test top-1 accuracy results of pruning methods on WRN-18 for Caltech101 with respect to *(a)* the number of remaining channels, *(b)* remaining resource memory occupation size, and *(c)* remaining FLOP of pruned model.

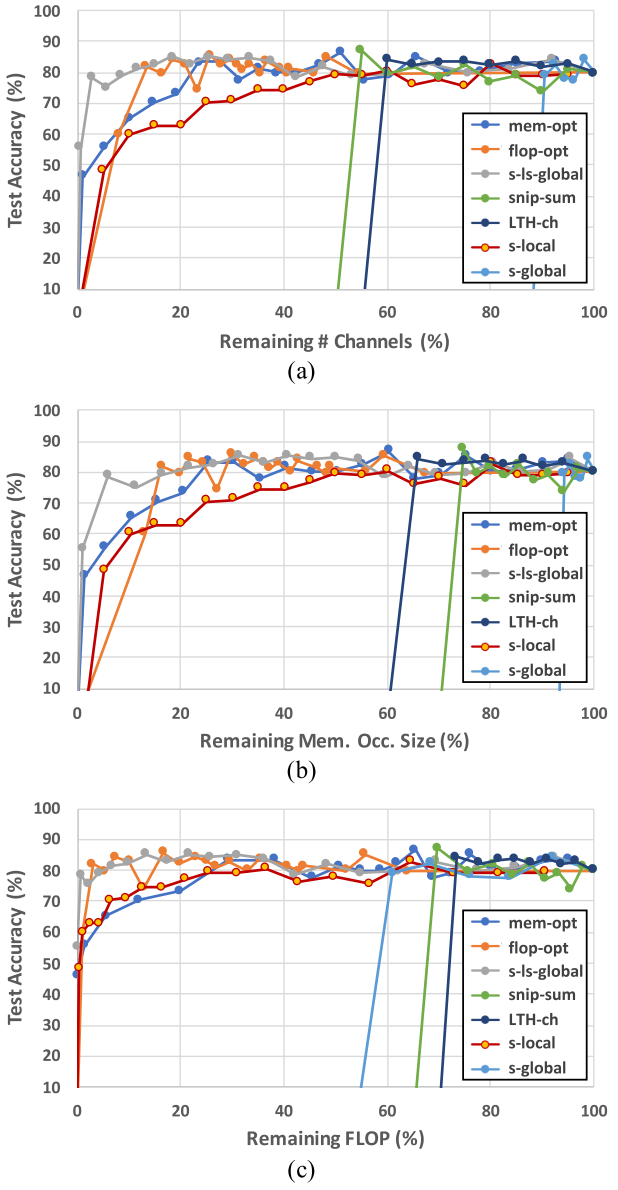

FIGURE 9. Test top-1 accuracy results of pruning methods on ResNet-101 for UC Merced satellite imagery dataset with respect to *(a)* the number of remaining channels, *(b)* remaining resource memory occupation size, and *(c)* remaining FLOP of pruned model.

to the other proposed methods (`mem-opt`, `flop-opt`, `s-ls-global`) until the layer removal occurs, but the layer removal on `s-global` occurs even earlier than `LTH-ch` or `snip-sum`. In contrast to such aspects, through additionally considering the proposed layer-wise sensitivity, the proposed `mem-opt`, `flop-opt` and `s-ls-global` show much more robustness on the task performance degradation by regulating the early layer removal. Moreover, in the case of `s-local`, early layer removal did not occur, but the task performance is mostly degraded more than the proposed `mem-opt`, `flop-opt` and `s-ls-global` for each sparsity level, which implies the importance of considering layer-wise computational characteristics together.

C. FEASIBILITY OF DL SERVING WITH PRUNING

1) PRACTICAL ENVIRONMENT OF ON-BOARD EMBEDDED SYSTEM

In order to evaluate the feasibility of the proposed methods on the restricted computing environments such as satellite on-board computing system [11], the embedded system board [31] in which inference of the deep learning models can be served in low power management is used as a test environment. Fig. 10 shows the hardware prototype of our embedded system board [31], and it consists of NVIDIA Jetson Nano chipset for managing host/GPU and ASIC chip that is designed to process the inference of the deep learning models. In the system, 4GB DDR4 memory is available,

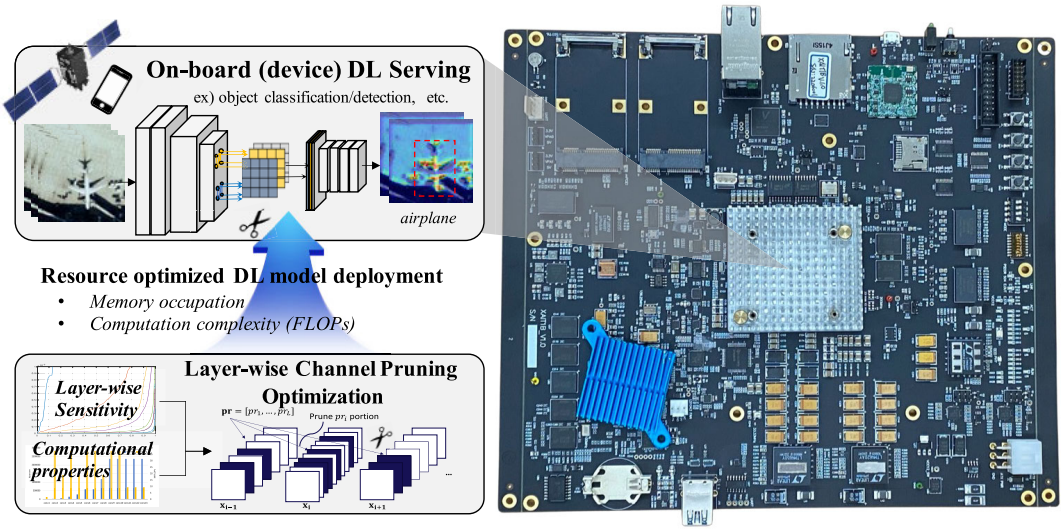

FIGURE 10. Developed prototype of on-board system [31] with proposed layer-wise channel pruning optimization model.

TABLE 2. Effect of acceleration on the inference processing time under same batch size.

	mem	flop	s-ls	LTH-ch	Ori.
Accuracy (%)	83.33	81.90	81.43	84.29	80.00
Mem. occ. (MiB)	2,213	2,813	2,911	5,701	5,767
Latency (ms)	184	63	81	359	523
Speed up	×2.84	×8.27	×6.43	×1.46	×1.00

TABLE 3. Effect of throughput improvement for DL serving.

	mem	flop	s-ls	LTH-ch	Ori.
Max. batch size	160	180	87	32	31
Throughput (Req/s)	85.3	258.8	198.2	44.6	30.7
Improvement	×2.78	×8.42	×6.45	×1.45	×1.00

and ASIC chip is prototyped under Samsung foundry 28-nm CMOS process with 200mW power consumption and minimum 7.5W in the entire on-board system.

The custom ASIC chip [31] mainly consists of shared memory and convolutional/vector processor, and it interworks with the programmable logic controller and the external memory to process the DL inference task. In the programmable logic controller, the task of DL inference is partitioned into subtasks of size that the intrinsic resources can accommodate, and the subtasks are processed sequentially. The partitioned subtasks are prepared in external memory, and then transferred to the shared memory of ASIC chip with the management of programmable logic controller. After that, each subtask is processed by convolutional/vector processor, where the convolutional processor supports various settings of convolutional operations in parallel using the multiple array processing units, and the vector processor supports operations of MaxPool, AvgPool, batch normalization, ReLU, and GEMM. After processing each subtask, programmable logic controller loads the results from the shared memory of ASIC chip to external memory. Further details of the prototype onboard system are provided in [31].

2) ACCELERATION EFFECT OF THE PROPOSED METHODS ON TEST ENVIRONMENT

The feasibility of the proposed methods on DL serving is also conducted. Some methods evaluated on previous subsection are also applied in this evaluation, and their first few abbreviations are denoted as in Table 2 and 3. The test is conducted on ResNet-101 for satellite imagery dataset which is the most practical application, and the latency for processing inference of the pruned model is measured by averaged from 100 trials under the developed on-board system (described in Section IV-C1) with 1024 × 1024 RGB input image size.

First, the effect of acceleration on inference processing by setting same batch size (=16) over comparing methods is evaluated, and the resource memory occupation size for each case is also measured together. In each comparing method, the most highly pruned model that shows equal or greater performance to original model as target deploying model is selected to deploy on the target hardware resource. As shown in Table 2, the proposed `mem-opt` can achieve the lowest resource memory occupation size while maintaining its accuracy, and the proposed `flop-opt` can achieve the highest speed up on inference latency under same batch size setting by largely reducing computation overhead. Although `mem-opt` shows relatively lower speed up on inference latency than `flop-opt` and `s-ls-global` as it is mainly targeted to reduce memory occupation size, it achieves higher speed up than the conventional method (`LTH-ch`) that prunes from pretrained state like lottery ticket hypothesis study [6], and can largely reduce resource memory occupation size keeping the accuracy which is more feasible in the aspect of enabling deployment on restricted computing environments like on-board system that has severe memory occupation constraint.

The effect of the proposed methods on improving throughput for DL serving with the same deploying models used in Table 2 is also evaluated. In each method, the maximum

available batch size that can be deployed into the target hardware resource for DL inference serving is searched by manual trials, which is practically applied in the study of DL serving [32]. Along with the maximum batch size, the throughput (the number of processed request images per second) is also measured on each test case. As shown in Table 3, the proposed methods show higher improvement than conventional pruning method of lottery ticket hypothesis framework, and `flop-opt` achieves highest improvement on throughput by largely reducing inference latency time.

The acceleration effect on training phase is already presented in previous section with Table 1. According to the result, the proposed method (`flop-opt`) shows the smallest computing overhead also on the total training phase, but can not show as much accelerating effect as in inference latency due to training with configured batch size does not fully utilize the resource on our test environment, and can expect further acceleration by enabling larger available batch size with the help of reducing memory occupation.

V. CONCLUSION

As the channel pruning generally shows severe performance degradation than weight pruning, the conventional methods mainly addressed how to efficiently recover the performance degradation from pruning under pretrained state, which requires heavy training overhead in double (pretraining and fine-tuning). In this paper, we propose a new scheme of layer-wise channel pruning that can sophisticatedly reflect each layer's characteristics and can be applied in a single-shot manner to alleviate computational overhead of pretraining by just observing dataset only once. Moreover, in order to improve robustness of performance degradation, we also propose a layer-wise sensitivity for single-shot based pruning scheme, and extend to optimization problems for two main computational constraints in layer-wise pruning decision scheme with proposing practical methods to find optimums. From the empirical evaluation, the proposed methods show robustness on performance degradation in aspects of reducing the number of channels, resource memory occupation, and computation overhead (FLOP) in deploying and serving DL model. The proposed methods also show feasibility on DL serving under constrained computing environments by reducing memory occupation, providing higher acceleration effect on inference latency and throughput improvement while maintaining the accuracy performance, and reduce computing overhead of training.

ACKNOWLEDGMENT

(Minsu Jeon and Taewoo Kim contributed equally to this work.)

REFERENCES

[1] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, 2016, pp. 1–15.
[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.
[3] N. Lee, T. Ajanthan, and P. H. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," in *Proc. ICLR*, 2019, pp. 1–15.
[4] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," in *Proc. NeurIPS*, 2020, pp. 1–13.
[5] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.
[6] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *Proc. ICLR*, 2019, pp. 1–42.
[7] R. Liu, Y. Cao, H. Chen, R. Guo, and M. Yoshikawa, "Flame: Differentially private federated learning in the shuffle model," in *Proc. AAAI*, 2020, pp. 8688–8696.
[8] W.-J. Kim and C.-H. Youn, "Lightweight online profiling-based configuration adaptation for video analytics system in edge computing," *IEEE Access*, vol. 8, pp. 116881–116899, 2020.
[9] D. Zhu, D. Song, Y. Chen, C. Lumezanu, W. Cheng, B. Zong, J. Ni, T. Mizoguchi, T. Yang, and H. Chen, "Deep unsupervised binary coding networks for multivariate time series retrieval," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 1403–1411.
[10] H. Kim, K. Lee, C. Lee, S. Hwang, and C.-H. Youn, "An alternating training method of attention-based adapters for visual explanation of multi-domain satellite images," *IEEE Access*, vol. 9, pp. 62332–62346, 2021.
[11] G. Giuffrida, L. Diana, F. de Gioia, G. Benelli, G. Meoni, M. Donati, and L. Fanucci, "CloudScout: A deep neural network for on-board cloud detection on hyperspectral images," *Remote Sens.*, vol. 12, no. 14, p. 2205, Jul. 2020.
[12] 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.
[13] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.
[14] B. Akin, Z. A. Chishti, and A. R. Alameldeen, "ZCOMP: Reducing DNN cross-layer memory footprint using vector extensions," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchitecture*, Oct. 2019, pp. 126–138.
[15] N. Otterness, M. Yang, S. Rust, E. Park, J. H. Anderson, F. D. Smith, A. Berg, and S. Wang, "An evaluation of the NVIDIA TX1 for supporting real-time computer-vision workloads," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2017, pp. 353–364.
[16] A. Shawahna, S. M. Sait, and A. El-Maleh, "FPGA-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.
[17] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. ICLR*, 2017, pp. 1–17.
[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. ICLR*, 2016, pp. 1–14.
[19] M. A. Rumi, X. Ma, Y. Wang, and P. Jiang, "Accelerating sparse CNN inference on GPUs with performance-aware weight pruning," in *Proc. ACM Int. Conf. Parallel Architectures Compilation Techn.*, Sep. 2020, pp. 267–278.
[20] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.
[21] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. ICLR*, 2017, pp. 1–13.
[22] M. Risso, A. Burrello, F. Conti, L. Lamberti, Y. Chen, L. Benini, E. Macii, M. Poncino, and D. Jahier Pagliari, "Lightweight neural architecture search for temporal convolutional networks at the edge," *IEEE Trans. Comput.*, early access, May 26, 2022, doi: 10.1109/TC.2022.3177955.
[23] A. Wan, X. Dai, P. Zhang, Z. He, Y. Tian, S. Xie, B. Wu, M. Yu, T. Xu, K. Chen, P. Vajda, and J. E. Gonzalez, "FBNetV2: Differentiable neural architecture search for spatial and channel dimensions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12965–12974.
[24] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "MorphNet: Fast & simple resource-constrained structure learning of deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1586–1595.

[25] L. G. Valiant, "A theory of the learnable," *Commun. ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
[26] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.
[27] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[28] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 594–611, Apr. 2006.
[29] 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.
[30] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proc. 18th SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS)*, 2010, pp. 270–279.
[31] T. Kim, M. Jeon, C. Lee, J. Kim, G. Ko, J.-Y. Kim, and C.-H. Youn, "Federated onboard-ground station computing with weakly supervised cascading pyramid attention network for satellite image analysis," *IEEE Access*, vol. 10, pp. 117315–117333, 2022.
[32] H. Shen, L. Chen, Y. Jin, L. Zhao, B. Kong, M. Philipose, A. Krishnamurthy, and R. Sundaram, "Nexus: A GPU cluster engine for accelerating DNN-based video analysis," in *Proc. 27th ACM Symp. Operating Syst. Princ.*, Oct. 2019, pp. 322–337.

MINSU JEON received the B.S. degree in electronic engineering from Sogang University, in 2016, and the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interests include deep learning (DL) application/model, DL model compression, DL serving, and high-performance computing systems.

TAEWOO KIM received the B.S. degree in electrical engineering from Kyungpook National University, Deagu, South Korea, in 2015, and the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interests include the deep learning (DL) framework, GPU computing, and interactive learning.

CHANGHA LEE (Member, IEEE) received the B.S. degree in electronic engineering from Hanyang University, Seoul, South Korea, in 2018, and the M.S. degree in electronic engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2020, where he is currently pursuing the Ph.D. degree. Since 2018, he has been a member of the Network and Computing Laboratory, KAIST. His current research interests include deep learning acceleration platform and high-performance edge-cloud computing systems.

CHAN-HYUN YOUN (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in electronics engineering from Kyungpook National University, Daegu, South Korea, in 1981 and 1985, respectively, and the Ph.D. degree in electrical and communications engineering from Tohoku University, Japan, in 1994. Before joining the University, from 1986 to 1997, he was the Head of the KT Telecommunications Network Research Laboratories, High-Speed Networking Team, where he had been involved in the research and developments of centralized switching maintenance systems, high-speed networking, and ATM networks. Since 1997, he has been a Professor with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He was an Associate Vice-President of office of planning and budgets at KAIST, from 2013 to 2017. He is currently the Director of the Grid Middleware Research Center and XAI Acceleration Technology Research Center, KAIST, where he is developing core technologies that are in the areas of high-performance computing, explainable AI systems, satellite imagery analysis, AI acceleration systems, and others. He wrote a book on *Cloud Broker and Cloudlet for Workflow Scheduling* (Springer, 2017). He served many international conferences as a TPC member. He was the General Chair of the 6th EAI International Conference on Cloud Computing (Cloud Comp 2015), KAIST, in 2015. He was a Guest Editor of IEEE Wireless Communications, in 2016.