Efficient Layer Compression Without Pruning

Jie Wu, Dingshun Zhu, Leyuan Fang, *Senior Member, IEEE*, Yue Deng, and Zhun Zhong

***Abstract*— Network pruning is one of the chief means for improving the computational efficiency of Deep Neural Networks (DNNs). Pruning-based methods generally discard network kernels, channels, or layers, which however inevitably will disrupt original well-learned network correlation and thus lead to performance degeneration. In this work, we propose an Efficient Layer Compression (ELC) approach to efficiently compress serial layers by decoupling and merging rather than pruning. Specifically, we first propose a novel decoupling module to decouple the layers, enabling us readily merge serial layers that include both nonlinear and convolutional layers. Then, the decoupled network is losslessly merged based on the equivalent conversion of the parameters. In this way, our ELC can effectively reduce the depth of the network without destroying the correlation of the convolutional layers. To our best knowledge, we are the first to exploit the mergeability of serial convolutional layers for lossless network layer compression. Experimental results conducted on two datasets demonstrate that our method retains superior performance with a FLOPs reduction of 74.1% for VGG-16 and 54.6% for ResNet-56, respectively. In addition, our ELC improves the inference speed by 2× on Jetson AGX Xavier edge device.**

***Index Terms*— Deep neural networks, layer compression, pruning, image classification.**

I. Introduction

DEEP neural networks (DNNs) have achieved significant development and been widely applied to various tasks, such as object detection [1], [2], image classification [3], [4], semantic segmentation [5], and so on. The improvement of the accuracy of DNNs is accompanied by an increase in the number of network parameters. For instance, the SENet [6] with 1.2×10^8 parameters requires more than 2×10^{10} FLOPs when inferencing an image with resolution 224×224. However, edge devices, i.e., mobile phones, robotics, and drones, have limited computational resources. For example, the computational power of the Snapdragon 865 mobile platform is 1.4 TFLOPS, which is almost 200 times smaller than the RTX 3090 graphics card. High demand for computational power of DNNs limits the deployment on resource-constrained edge devices.

Recent works [7], [8], [9] have demonstrated that the over-parameterized DNNs have a large number of redundant weights which can be removed (i.e., set to zero). Model compression methods emerge to reduce the computational complexity and speed up the execution of DNNs, such as quantization [10], [11], pruning [12], [13], lightweight structure design [2], [14], [15], knowledge distillation [16], [17], [18], and so on. Among them, network pruning is widely applied in various fields and achieves promising performance [19], [20], [21], which involves removing redundant parameters and connections to improve the inference efficiency of the network.

Generally, network pruning methods can be categorized into channel (filter) pruning methods [22], [23] and layer pruning methods [12], [21], [24], [25]. Channel (filter) pruning methods aim to find and prune unimportant kernels on a channel or filter level, thereby reducing the computational complexity and storage memory of the network. Since the convolution operations of different filters in the same layer are usually performed in parallel [24], channel pruning does not achieve considerable acceleration on hardware platforms, and also cannot greatly reduce run-time memory. Therefore, channel (filter) pruning is hard to achieve significant actual acceleration. Layer pruning methods [12] remove convolutional layers and corresponding nonlinear activation layers and Batch Normalization (BN) layers [26]. Different from channel (filter) pruning methods, that reduce the width of convolutional layers, layer pruning methods decrease the depth of the network. Layer pruning does not disrupt the structure of the network and is more friendly to deploy on edge devices. However, discarding the entire layers will damage the correlation of layers, causing severe loss of accuracy.

In general, the process of current network pruning methods [21], [27] can be divided into three steps (as shown in Figure 1): 1) training the original network; 2) pruning unimportant channels or layers; and 3) finetuning the pruned network. Although the last finetuning process can improve the accuracy of the pruned network to some extent, the pruning operation that disrupts the correlation among layers and filters will inevitably deteriorate the performance [21]. The pruned models may be trapped into bad local-optimum, and sometimes cannot even reach a similar level of accuracy

Manuscript received 3 January 2023; revised 13 July 2023; accepted 30 July 2023. Date of publication 10 August 2023; date of current version 17 August 2023. This work was supported in part by the National Natural Science Fund of China under Grant U22B2014 and Grant 61922029, in part by the Science and Technology Plan Project Fund of Hunan Province under Grant 2022RSC3064, in part by the Key Research and Development Program of Hunan under Grant 2021SK2039, and in part by the Natural Science Fund of Hunan Province under Grant 2021JJ30003. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Nikolaos Mitianoudis. *(Corresponding author: Leyuan Fang.)*

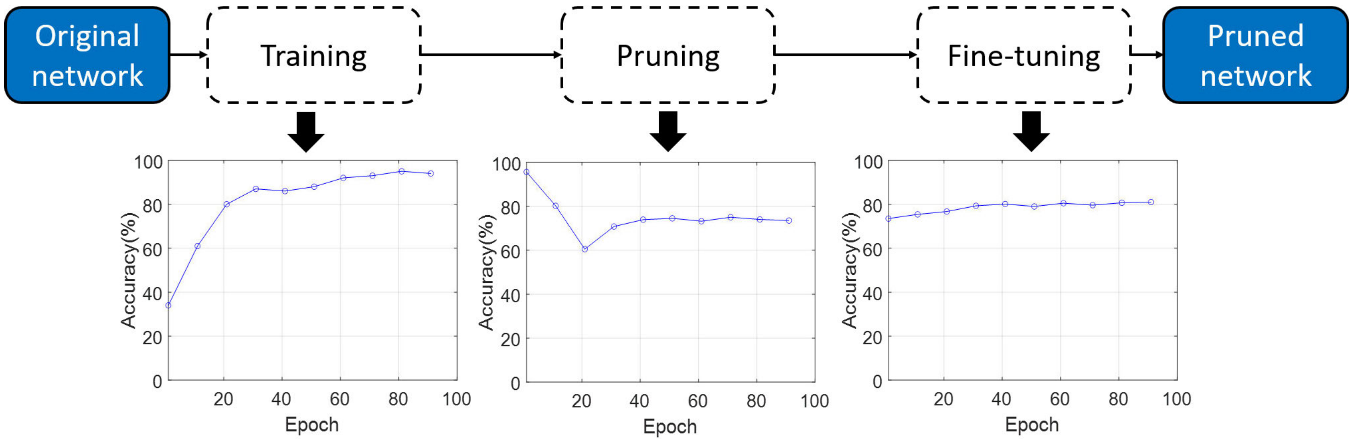
Jie Wu and Dingshun Zhu are with the College of Electrical and Information Engineering, Hunan University, Changsha 410082, China (e-mail: wujie1227@hnu.edu.cn; dszhu@hnu.edu.cn).

Leyuan Fang is with the College of Electrical and Information Engineering, Hunan University, Changsha 410082, China, and also with the Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: fangleyuan@gmail.com).

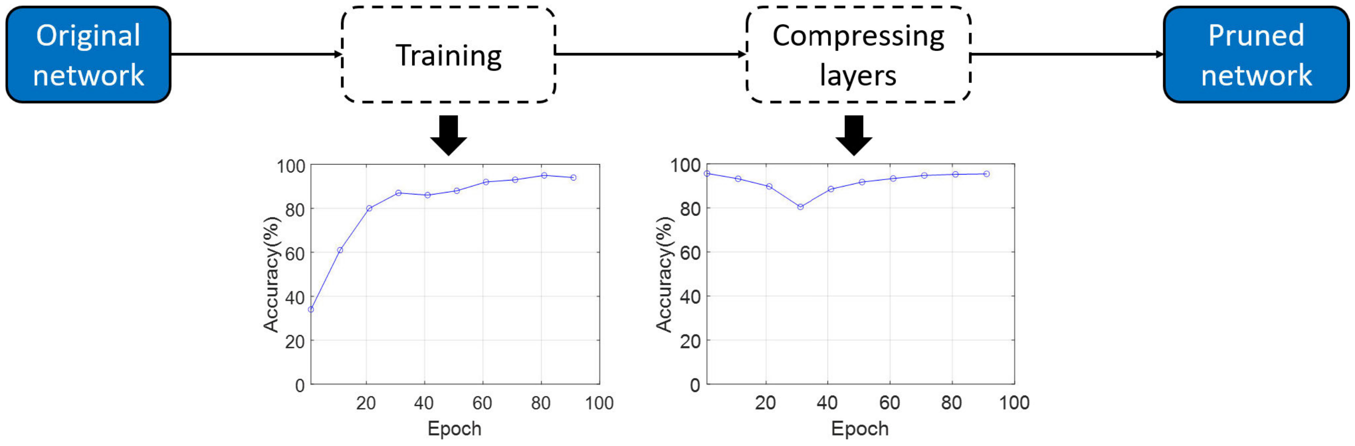
Yue Deng is with the School of Astronautics, Beihang University, Beijing 102206, China, also with the Science and Technology on Space Intelligent Control Laboratory, Beijing 100191, China, and also with the Institute of Artificial Intelligence and the Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100191, China (e-mail: ydeng@buaa.edu.cn).

Zhun Zhong is with the School of Computer Science, University of Nottingham, NG8 1BB Nottingham, U.K. (e-mail: zhunzhong007@gmail.com).

Digital Object Identifier 10.1109/TIP.2023.3302519



(a) The process of existing network pruning methods. In the original training process, the network has a good accuracy. After pruning, the accuracy dropped significantly and the finetuning process is needed to increase it.



(b) The process of our ELC that dose not require a finetuning process and the accuracy drop of compressing layers process is low.

Fig. 1. Process comparison of our ELC method and existing network pruning methods.

with a counterpart of the same structure trained from scratch [13].

To avoid the above-mentioned issue, in this work, we intend to compress the network without pruning any channels or layers. Inspired by the linear mergeability of convolutional layers [28], we propose an Efficient Layer Compression (ELC) method without pruning. Our ECL is based on a layer decoupling and merging strategy. In the decoupling step, we propose a decoupling module to make serial layers that are from different types to be mergeable, in which a removing ReLU module (Rem-ReLU) and a De-coupling convolution module (De-Conv) are designed for nonlinear layers and convolutional layers, respectively. Then, the decoupled network can be losslessly merged to a lightweight network by the equivalent conversion of serial convolutional layers.

Compared with traditional pruning methods, the proposed ELC method has three advantages.

- First, our ELC can largely preserve the representation capabilities of the DNNs, since it does not discard any channels or layers and retain the correlation of channels and layers.
- Second, our ELC does not require a finetuning step, which significantly decreases the training cost.
- Third, the compressed network produced by our ELC is more friendly to edge devices.

Experimental results show that our approach achieves state-of-the-art performance compared with recent channel and layer pruning methods. More specifically, the compressed VGG-16 model on the CIFAR-10 dataset can reduce 74% FLOPs while losing only 0.45% top-1 accuracy. Furthermore, our proposed ELC brings more than $2\times$ speedup ratio on Jetson AGX Xavier edge device.

The rest of this paper is organized as follows. Section II reviews the network pruning and neural architecture search (NAS) methods. Section III introduces the proposed efficient layer compression method without pruning. Section IV shows experimental results on image classification. Section V concludes this paper and suggests some future works.

II. Related Work

A. Network Pruning

Network pruning methods aim to remove unimportant neurons or connections from a DNN. Han [29] firstly proposes the pruning approach that prunes weights below a threshold in the era of deep learning. After that, various weight pruning methods have been proposed [30], [31], [32]. Although weight

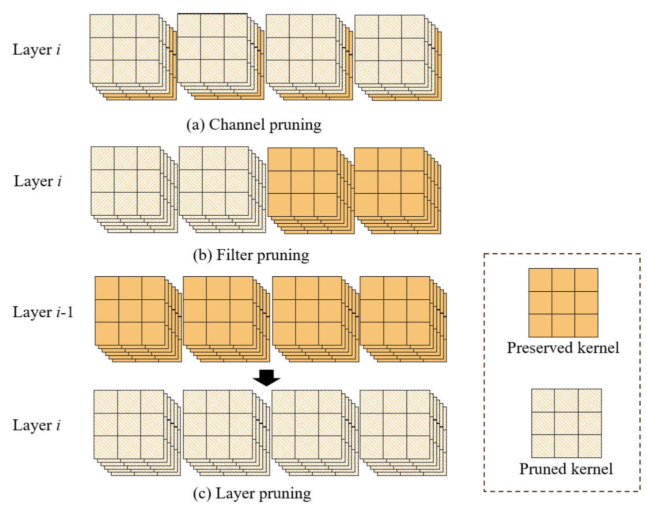


Fig. 2. Different pruning methods.

pruning can reduce the memory footprint, unstructured sparsity of the network makes it difficult to realize actual speedups on common computing frameworks. Compared with weight pruning, channel (filter) and layer pruning methods can remove some whole structures (i.e., residual block [33]), which are more friendly to edge devices. Therefore, we mainly introduce the channel (filter) pruning and layer pruning methods in this section.

1) Channel (Filter) Pruning: Channel pruning and filter pruning remove parameters at the channel level or the filter level. Figure 2 (a) and (b) show the difference between channel pruning and filter pruning. Since they both make the DNNs thinner by removing channels or filters, we consider channel pruning and filter pruning as the same type of pruning method.

Li et al. [34] firstly prune whole filters together with their connecting feature maps from a well-trained model, by calculating the sum of its absolute weights. Work in [35] removes unimportant channels in each layer, by a LASSO regression-based channel selection and least square reconstruction. Work in [8] optimizes the scaling factors in BN layers with L1 regularization and then chooses which channel needed to be removed according to the scaling factors. Ding et al. [21] observe that current channel pruning methods rely on hard pruning strategies, which do not take the correlation among layers and filters into consideration. Therefore, they use long short-term memory (LSTM) to learn the hierarchical features of the network to find where to prune, instead of pruning layer by layer. Liu et al. [36] propose a discrimination-aware pruning method that preserves the channels contributing to the discriminative power by introducing an additional discrimination-aware loss. Liu et al. [37] utilize the meta network to automatically generate weight parameters for any channels, and thus prune the channels with small weights. Chin et al. [38] aim to learn a global ranking of the filters across different layers, which efficiently finds the a sweet-spot between the accuracy and speed. The above pruning methods focus on pruning the least important channels or filters of the network, while Wang et al. [22] show that pruning the structure redundancy can achieve better performance. However, these pruning methods only operate on channel or filter level, which is difficult to achieve high compression rate and actual acceleration.

2) Layer Pruning: Unlike channel (filter) pruning which reduces the width of the network, layer pruning (shown in Figure 2 (c)) prunes entire layers allowing the network shallower. Compared with channel (filter) pruning, layer pruning can reduce the run-time memory more significantly. In addition, since convolutional computations on the same layer are usually operated in parallel, reducing convolutional layers can bring more speedup. Hence, the actual acceleration on edge devices of layer pruning outperforms channel (filter) pruning.

Chen and Zhao [12] propose a layer pruning method based on the feature representation, in which layers with small improvements on feature representations are pruned. Jordao et al. [39] introduce layer pruning utilizing a discriminative importance estimation criterion based on partial least squares projection, and demonstrate that cascading discriminative layer pruning outperforms the methods [12] based on removing parameters from all layers. Wang et al. [40] propose a discrimination based block-level pruning method that prunes redundant blocks based on the discriminative ability of each block. They place a linear classifier after each block to evaluate the discriminative ability and remove the blocks with the least contributions. Work in [24] implements layer pruning on ResNets [33] to obtain a shallow one. They take the characteristics of the residual module into consideration and disconnected redundant residual mappings to achieve the purpose of layer pruning. Xu et al. [41] propose a simple layer pruning method via fusible residual convolution module, which introduces a layer scaling factor behind the BN layer. The fusible residual convolution module can be pruned when the layer scaling factor is small, so that the depth of the network is reduced. As the pruning operation cut down the correlation of layers, the accuracy of the pruned network is severely dropped.

B. Neural Architecture Search

Recently, Neural Architecture Search (NAS) has attracted increasing attention to automating the process of architecture design [42]. The purpose of NAS is to find a well-designed and optimal architecture by choosing the best options and/or connections. Zoph et al. [43] first propose a NAS approach by reinforcement learning to boost performance. NAO [44] proposed by Luo et al. maps a neural network architecture into a continuous representation, and then the gradient optimization can be performed in the continuous space to find the optimal architecture. Zhang et al. [45] propose a One-shot NAS method, aiming to overcome the multi-model forgetting and avoid the performance degradation of the supernet. Guo et al. [46] cast the optimization of NAS to the Markov decision process, which optimizes the operations inside an architecture to enhance the performance. Next, they propose a NAT++ method that further enlarges the search space to conduct architecture optimization in a finer manner [47]. Lin et al. [48] formulates the search for optimal pruned structure as an optimization problem based on the artificial bee colony algorithm. Network pruning refers to finding an optimal

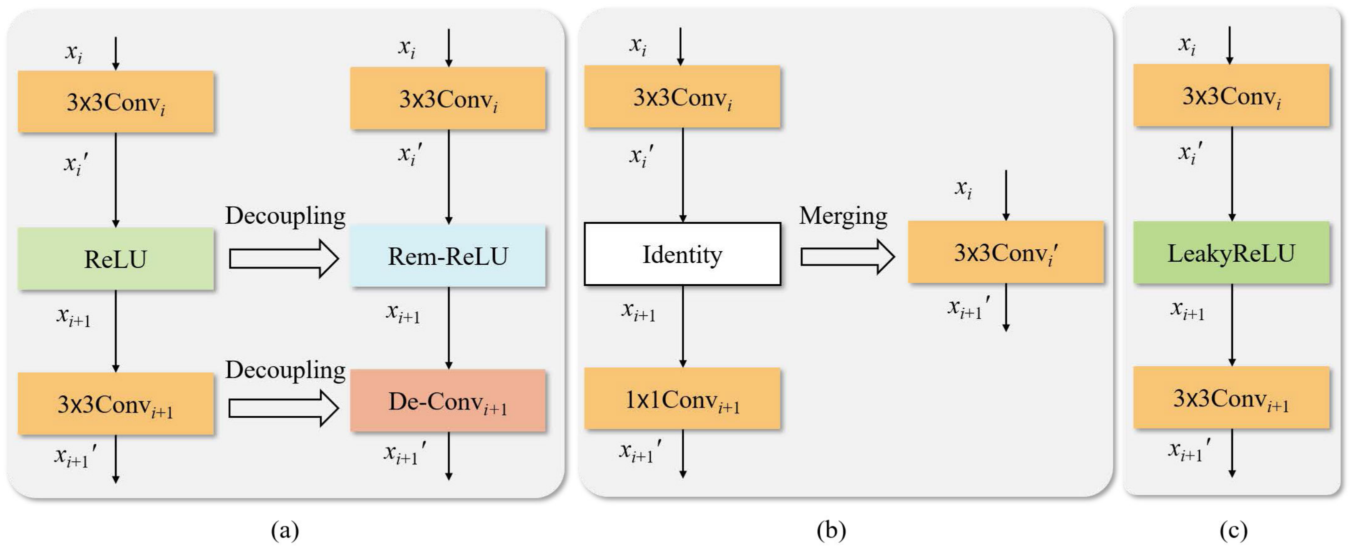


Fig. 3. The proposed layer compression method. (a) Layer decoupling module (replacing nonlinear and convoultional layers with Rem-ReLU and De-Conv) to enable us readily merge serial layers. (b) Equivalent conversion of parameters to losslessly merge the decoupled network into a shallow network. (c) When the layers cannot be merged, Rem-ReLU is equivalently converted to LeakyReLU and De-Conv is re-parameterized into a new vanilla convolutional layer.

substructure from the given unpruned network, which can be seen as a simple NAS in channel/layer dimension.

III. Proposed Efficient Layer Compression

As described in section II, channel (filter) pruning cannot achieve satisfactory acceleration when the pruned network is deployed on edge devices, and layer pruning will cut down the correlation of layers deteriorating the accuracy. Since convolution operation is linear, we propose an efficient layer compression method (ELC) without pruning, which intends to merge layers rather than pruning them. This is the first work to exploit the mergeability of serial convolutional layers on network compression to achieve efficient layer compression. However, in convolutional neural networks, there are nonlinear activation layers between the convolutional layers. The nonlinearity introduced by nonlinear activation layers (i.e., ReLU [49], PReLU [50], and so on) blocks the serial merging of convolutional layers. Besides, the serial merging of 3×3 convolutional layers which are usually used in DNNs will increase the computational complexity and the number of parameters. For instance, the operation of two serial 3×3 convolutional layers equals a 5×5 convolutional layer, but the number of parameters is increased from 18 to 25. To address the non-mergeability in serial convolutional layers, we intend to decouple activation layers and convolutional layers. Firstly, we replace the ReLU with the proposed Rem-ReLU to discard redundant nonlinear activation layers. Secondly, the De-Conv is introduced to replace the convolutional layers, aiming to solve the problem of increased computational complexity and the number of parameters when merging the serial convolutional layers. Since the operation of convolution is linear, we utilize the equivalent conversion of parameters to losslessly merge the decoupled network into a shallow network. The framework of the ELC is shown in Figure 3.

Section III-A introduces the layer decoupling module. In Section III-B, we present the lossless mergeability of the decoupled network. In Section III-C, we introduce the gradient penalty.

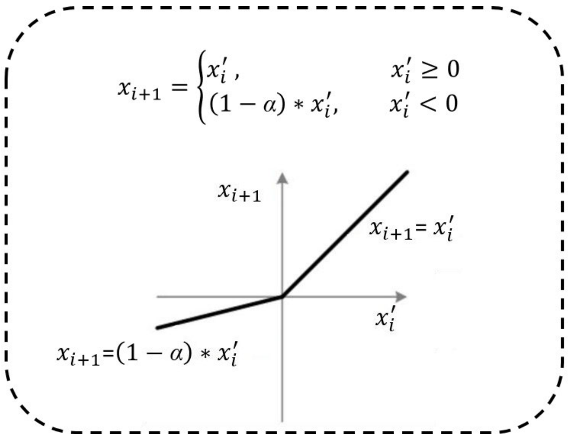
A. Layer Decoupling

We propose two components in the decoupling module: Rem-ReLU to decouple activation layers and De-Conv to decouple convolutional layers, as shown in Figure 4.

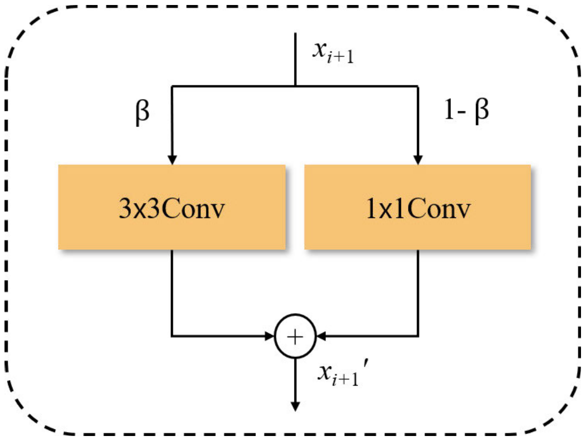
1) Rem-ReLU: Nonlinear activation layers are introduced to enhance the representation ability of the network. But ConvNeXt [51] demonstrates that there is a large amount of redundancy in the nonlinearity of the network. If there is a non-linear activation function (i.e., ReLU) between two convolutional layers, these two layers cannot be merged losslessly. We intend to remove the non-linear activation functions in the original network and then the serial convolutional layers can be merged into one convolutional layer losslessly. To remove the useless nonlinear activation layers, we propose a Rem-ReLU to replace the ReLU of the network. The Rem-ReLU is defined as:

$$x_{i+1} = \sigma(x_i') = \begin{cases} {x'}_i, & {x'}_i \geq 0 \\ (1-\alpha_i){x'}_i, & {x'}_i < 0, \end{cases} \quad (1)$$

where x_{i+1} is the output feature of the i-th activation layer and the input feature of the $(i+1)$-th convolutional layer, $\sigma(\cdot)$ is the Rem-ReLU activation function, and α_i is a learnable parameter that is utilized to adjust the slope of the negative semi-axis of the i-th activation layer and is initialized to zero. When $\alpha_i = 0$, formula 1 can be expressed as $x_{i+1} = \sigma(x_i') = x_i'$. Then, the activation layer does not have nonlinearity and can be removed directly. When $\alpha_i \neq 0$, the activation layer is equivalently converted into an ordinary activation layer (LeakyReLU) (shown in Figure 3 (c)). If the activation layer has nonlinearity, its adjacent convolutional layers cannot be merged. Compared to PReLU [50], our Rem-ReLU is specifically designed to transform the non-linear


$x_{i+1} = \begin{cases} x_i', & x_i' \geq 0 \\ (1-\alpha) * x_i', & x_i' < 0 \end{cases}$
x_{i+1}
$x_{i+1} = x_i'$
x_i'
$x_{i+1} = (1-\alpha) * x_i'$

(a)



(b)

Fig. 4. The proposed (a) Rem-ReLU and (b) De-Conv.

activation function to an identity mapping through the use of gradient penalty. As a result, Rem-ReLU allows us to remove the non-linear activation functions within the network, enabling the seamless integration of two serial layers that includes a non-linear activation function between them.

2) De-Conv: Two convolutional layers with kernel sizes > 1 can be merged into a convolutional layer with larger kernel size, while the parameters and computational complexity of the merged convolutional layer will increase. We propose a De-Conv to decouple serial convolutional layers, that can address the issues of increased parameters and computational complexity of the merged convolutional layers. Given the input feature x_{i+1}, the De-Conv is formulated as:

$$x_{i+1}' = \beta_i \cdot w_{i+1}^{3\times3} * x_{i+1} + (1-\beta_i) \cdot w_{i+1}^{1\times1} * x_{i+1}, \quad (2)$$

where x_{i+1}' is the output feature of the $(i+1)$-th De-Conv. $w_{i+1}^{3\times3}$ and $w_{i+1}^{1\times1}$ are the 3×3 and 1×1 convolutional layers, where $w_{i+1}^{3\times3}$ is initialized with the same weights as the original network to be pruned and $w_{i+1}^{1\times1}$ is initialized to an identity matrix. β_i is a learnable parameter initialized with an identity matrix which controls the weight of the two parallel convolutional layers. Our purpose is to let β_i be 0 by adding the gradient penalty, which will be described in detail in Section III-C. Then, formula 2 can be regarded as $x_{i+1}' = w_{i+1}^{1\times1} * x_{i+1}$. The 1×1 convolutional layer can be merged with the previous convolutional layer without changing the structure of the previous layer. If $\beta_i \neq 0$, the De-Conv can be equivalently re-parameterized into a new vanilla convolutional layer (shown in Figure 3 (c)), and 2 can be equivalently represented as:

$$x_{i+1}' = (\beta_i \cdot w_{i+1}^{3\times3} + (1-\beta_i) \cdot w_{i+1}^{1\times1}) * x_{i+1} = w_{i+1}' * x_{i+1}, \quad (3)$$

where w_{i+1}' is the weights of the updated $(i+1)$-th convolutional layer.

B. Equivalent Merging of the Layer Decoupled Network

When $(\alpha_i, \beta_i) = 0$, shown in Figure 3 (b), the $(i+1)$-th layer consists of a 1×1 convolutional layer and an identity mapping, represented as:

$$x_{i+1}' = x_{i+1} * w_{i+1}^{1\times1}. \quad (4)$$

Based on the linear mergeability of convolutional operations, the $(i+1)$-th layer can be merged with the previous convolutional layer. The calculation process of equivalently mergeability is formulated as:

$$x_{i+1}' = (x_i * w_i^{3\times3}) * w_{i+1}^{1\times1}. \quad (5)$$

Since the convolution operation is linear, Eq. 5 can be represented as:

$$x_{i+1}' = x_i * (w_i^{3\times3} * w_{i+1}^{1\times1}) = x_i * w_i'', \quad (6)$$

w_i'' is the merged convolutional layer. Hence, the calculation of two serial convolutional layers equals a convolution layer, achieving the layer compression.

C. Gradient Penalty

To make (α_i, β_i) decay to zero, we apply a gradient penalty on these parameters. Given an uncompressed pre-trained model F_u, we replace the convolutional layers and ReLU activation layers with De-Conv and Rem-ReLU, obtaining the decoupled network F_{de}. The α_i, β_i parameters are initialized as 1, and then the F_{de} is equivalent to F_u. We collect the parameters of adjacent De-Conv and Rem-ReLU to form (α_i, β_i) pair, and let $M = \{(\alpha_i, \beta_i), i \in N\}$, where N is the number of layers. Next, we retrain F_{de} with a loss function that is the same as used in the training of the uncompressed model. K is the number of compression layers. The first K smallest α_i, β_i pairs are selected to form the set M'. We apply gradient penalty to M', which makes M' gradually decay to 0. The update of the parameters is expressed as:

$$W' = \begin{cases} W - l \cdot G_0, & (\alpha_i, \beta_i) \in M' \\ W - l \cdot G + \lambda \cdot sign(W), & (\alpha_i, \beta_i) \notin M', \end{cases} \quad (7)$$

where W is the parameters of F_{de}, W' is the updated parameters, l is the learning rate, and G is the gradient of the network. When $(\alpha_i, \beta_i) \in M'$, G_0 is a fixed number so that (α_i, β_i) will gradually decrease to 0. When $(\alpha_i, \beta_i) \notin M'$, we apply an additional gradient penalty decay to improve compression efficiency, where λ is a weight parameter and $sign(\cdot)$ is a symbolic function:

$$sign(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0. \end{cases} \tag{8}$$

IV. Experiments

In this section, we first introduce the datasets, models, and experiment settings. Then, we show the comparison between our proposed method and other state-of-the-art compression methods on the image classification task. Thirdly, we present the speedup performance when the compressed models are deployed on edge devices. Lastly, we conduct ablation studies to demonstrate the effectiveness of the proposed ELC to compress network layers and maintain high accuracy.

A. Datasets, Models and Experimental Settings

1) Datasets: We conduct experiments on three widely used datasets (The CIFAR-10 [60], CIFAR-100, and ImageNet [61]). CIFAR-10 dataset has 10 classes, containing 50,000 images for training and 10,000 images for validation. CIFAR-100 has the same images as CIFAR-10, but CIFAR-100 is divided into 100 classes. ImageNet dataset consists of 1.28 million training images and 50,000 testing images with 1000 classes.

2) Models: For the image classification task, we adopt three models: VGG [62], ConvNeXt [51], and ResNet [33]. Specifically, we use VGG-16, ResNet-56, ResNet-110, and ConvNeXt on CIFAR-10 dataset, VGG-19 on CIFAR-100, and ResNet-34 on ImageNet dataset.

3) Experimental Settings: We train all networks from scratch to get the baseline models and then apply our ELC method to these baseline models. During training our layer compression method, the experimental settings are shown as follows. For models on CIFAR-10 and CIFAR-100 datasets, we train these models with 300 epochs, batch size of 128, $\lambda = 0.0001$, and stepped decay learning rate that is initialized as 0.01 and multiplied 0.1 at epoches 140 and 240. For ResNet-34, we adopt these hyper-parameters: the batch size of 256, $\lambda = 0.0001$, an initial learning rate of 0.01 with cosine annealing for 240 epochs.

B. Compression Results on Image Classification

1) Compression Results on CIFAR-10: To investigate the effectiveness of our proposed ELC method, some experiments are taken on VGG-16, ResNet-56, and ResNet-110. We report the results in Table I. Besides, we conduct experiments on a lightweight structure (ConvNeXt) to validate the generalization of our proposed ELC (seen in Table II).

On VGG-16, we include three channel pruning methods (GAL [52], SSS [53], and Hrank [54]) and one layer pruning method (FRLP [12]) for results comparison. In Table I, we show two different layer compression results of our proposed ECL, which are the ELC-0.5 with 52.5% FLOPs drop and ECL-0.7 with 74.0% FLOPs drop. All the convolutional layers and non-linear activation layers are replaced with De-Conv and Rem-ReLU when training ELC-0.5. We observe that the proposed ELC-0.5 has the least accuracy drop (0.31%) with the 52.5% FLOPs drop. In training ELC-0.7, we decouple the convolutional layers and non-linear activation layers whose previous layer is not max-pooling layer. The accuracy drop of our ELC-0.7 (0.45%) exceeds 1.17% than Hranks (1.62%), while the FLOPs drop of ELC-0.7 is 8.6% higher than that of Hrank. The ELC-0.7 totally has a reduction of 4 convolutional layers in the VGG-16 network.

We verify the feasibility of the proposed ELC on ResNet-56 and ResNet-110 (shown in Table I). On ResNet-56, we compare our proposed ELC with eight channel pruning methods (GAL [52], Hrank [54], LFPC [57], LSTM [21], NISP [55], FPGM [56]), ResRep [13], and DCP [36]) and four layer pruning methods (FRLP [12], DBP [40], DLP [39], and LPSR [24]). Compared with these methods, our ELC achieves relatively superior performance. For example, the LSTM prunes ResNet-56 by 47.5% of FLOPs, with 0.87% drop in accuracy. In contrast, our proposed ELC-0.5 reduces FLOPs by 53.4%, but improves accuracy by 0.21%. We still can observe that the accuracy of our ELC-0.5 is over 0.54% higher than that of DBP when the FLOPs drop of these two models are the same (53.4%). Our ELC-0.5 has a similar accuracy drop with LPSR (-0.21% vs -0.19%) but the FLOPs drop of our ELC-0.5 is higher 1.5% than that of LPSR. The FLOPs drop of our proposed ELC-0.5 (53.4%) is higher than those of DCP [39] (49.8%) and ResRep [13] (52.9%). The ELC has a lower accuracy drop (-0.21%) than DCP (0.02%) and ResRep (0.0%). Besides, our ELC-0.6 reaches 63.4% FLOPs drop, with only 0.37% accuracy decreasing. Compared with DBP which has a similar accuracy drop, the FLOPs of our ELC-0.6 is lower by 12.6M. For further investigation, we plot the FLOPs with respect to each residual block on the original ResNet-56, ELC-0.5, and ELC-0.6. As shown in Figure 5, when the residual block index is 1, 2, 24, and 25, the FLOPs of ELC-0.5 and ELC-0.6 is zero.

Since some methods do not have results on ResNet-110, we only make comparisons on GAL [52], NISP [55], Hrank [54], DBP [40], FPGM [56], ResRep [13], SAP [59], and DLP [39] methods. As shown in Table I, the GAL-0.5 reduces the FLOPs of ResNet-110 by 48.5%, but our ELC achieves 54.5% FLOPs drop ratio and increases accuracy by 0.76%. While we achieve similar accuracy with Hrank, our FLOPs drop ratio is 13.3% higher than that of Hrank. Compared with LFPC with 60.3% FLOPs drop, our ELC-0.6 achieves the gain of 1.08% than that of LFPC. Our ELC-0.6 has lower FLOPs (92.3M) than ResRep and even gains 0.47% increase in accuracy, while the accuracy of ResRep drops 0.02%. The ELC-0.6 outperforms the SAP-0.6 in terms of both accuracy and FLOPs drop. Both accuracy drop and FLOPs drop ratio of our ELC are better than two-layer pruning methods: DBP and DLP.

For ConvNeXt [51], the baseline structure we utilized has three stages, where the channels of different stages are [32, 64, 128] and each stage has 4 blocks. We present 3 pruned models with different compression ratios in Table II. ELC-0.4 with 45.4% FLOPs drop obtains 93.27% Top-1 accuracy, which is only less 0.01% than the baseline model. ELC-0.5 has 0.64%

TABLE I

PERFORMANCE COMPARISONS ON CIFAR-10. "ACC." DENOTES THE TOP-1 ACCURACY. "-" DENOTES RESULTS ARE NOT REPORTED. THE BEST RESULTS IN TERMS OF ACCURACY DROP ARE IN **BOLD**. THE SECOND-BEST RESULTS ARE UNDERLINED

Model	Methods	Type	Baseline ACC.%	Pruned ACC. ↑%	ACC. Drop ↓%	FLOPs ↓M	FLOPs Drop ↓%
VGG-16	FRLP [12] (TPAMI2019)	Layer	93.50	93.00	0.50	191.7	38.9
	GAL-0.05 [52](CVPR2019)	Channel	93.96	92.03	1.93	189.5	39.6
	SSS [53] (ECCV2018)	Channel	93.96	93.02	0.94	183.1	41.6
	ELC-0.5 (Ours)	Layer	93.56	93.25	**0.31**	**144.9**	**52.5**
	Hranks [54] (CVPR2020)	Channel	93.96	92.34	1.62	108.6	65.4
	ELC-0.7 (Ours)	Layer	93.56	93.11	0.45	81.7	74.0
ResNet-56	FRLP [12] (TPAMI2019)	Layer	93.03	92.32	0.71	80.5	34.8
	GAL-0.6 [52] (CVPR2019)	Channel	93.26	92.98	0.28	78.3	37.6
	NISP [55](CVPR2018)	Channel	-	-	0.03	70.5	43.6
	DLP [39] (JSTSP2020)	Layer	-	-	0.82	65.8	47.4
	LSTM [21] (TIP2021)	Channel	93.04	92.93	0.11	65.6	47.5
	DCP [36] (TPAMI2022)	Channel	93.74	93.72	0.02	62.8	49.8
	Hrank [54] (CVPR2020)	Channel	93.26	93.17	0.09	62.7	50.0
	LPSR [24] (SPL2022)	Layer	93.21	93.40	-0.19	60.1	51.9
	FPGM [56] (CVPR2019)	Channel	93.59	93.26	0.33	59.4	52.4
	LFPC [57] (CVPR2020)	Channel	93.59	93.24	0.35	59.2	52.9
	ResRep [13] (ICCV2021)	Channel	93.71	93.71	0.00	59.2	52.9
	DBP [40] (Arxiv2019)	Layer	93.72	93.39	0.33	58.3	53.4
	ELC-0.5 (Ours)	Layer	93.45	93.66	**-0.21**	**58.3**	**53.4**
	ELC-0.6 (Ours)	Layer	93.45	93.08	0.37	45.3	63.8
	IFM [58] (ICPR 2021)	-	93.59	91.28	2.31	42.3	66.2
	ELC-0.7 (Ours)	Layer	93.45	91.57	1.88	29.8	76.2
ResNet-110	Hrank [54] (CVPR2020)	Channel	93.5	94.23	-0.73	148.7	41.2
	NISP [55] (CVPR2018)	Channel	-	-	0.18	126.6	43.7
	DBP [40] (Arxiv2019)	Layer	93.97	93.61	0.36	141.9	43.9
	GAL-0.5 [52] (CVPR2019)	Channel	93.50	92.55	0.95	130.2	48.5
	DLP [39] (JSTSP2020)	Layer	-	-	0.25	129.7	48.7
	FPGM [56] (CVPR2019)	Channel	93.68	93.74	-0.16	120.6	52.4
	ELC-0.5 (Ours)	Layer	93.60	94.36	**-0.76**	**115.0**	**54.5**
	SAP-0.5 [59] (ICLR2023)	Weight	-	94.19	-	114.2	54.8
	ResRep [13] (ICCV2021)	Channel	94.64	94.62	0.02	105.3	58.2
	LFPC [57] (CVPR2020)	Channel	93.68	93.07	0.61	101.8	60.3
	SAP-0.6 [59] (ICLR2023)	Weight	-	93.85	-	94.9	62.4
	ELC-0.6 (Ours)	Layer	93.60	94.07	-0.47	92.3	63.5

TABLE II

RESULTS OF CONVNEXT [51] ON CIFAR-10 DATASET

Method	ACC. (%)	ACC. Drop (%)	FLOPs (M)	FLOPs Drop (%)
Baseline	93.28	-	127.9	-
ELC-0.4	93.27	0.01	69.8	45.4
ELC-0.5	92.64	0.64	62.5	51.1
ELC-0.8	92.13	1.15	47.8	62.6

TABLE III

PERFORMANCE COMPARISON ON CIFAR-100. "ACC." DENOTES THE TOP-1 ACCURACY. THE BEST RESULTS ARE IN **BOLD**

VGG-19	ACC. (%)	ACC. Drop (%)	FLOPs(M)	FLOPs Drop (%)
Baseline	73.26	-	399.3	-
Slimming [8]	70.92	2.34	127.0	68.1
Baseline (Ours)	71.28	-	399.3	-
ELC (Ours)	70.03	**1.25**	**124.5**	**68.8**

accuracy drop and reduces 45.4% FLOPs. ELC-0.6 only has 47.8M FLOPs and obtains an accuracy of 92.13%. We can figure that our ELC can still perform well on lightweight-designed models.

2) Compression Results on CIFAR-100: We conduct experiments on the VGG-19 model to validate the effectiveness of proposed ELC. We train the VGG-19 on cifar-100 and get the baseline model, whose accuracy is 71.28%. And then, we compress the VGG-19 with our proposed ELC. We get the compressed VGG-19 network whose FLOPs are reduced by 68.8%, with 1.25% accuracy drop. As can be seen from Table III, our method outperforms the slimming method (above 1.09%), and the FLOPs drop is also higher than that of the slimming method [8].

3) Compression Results on ImageNet: We compare the compression results of our proposed ELC on ImageNet with

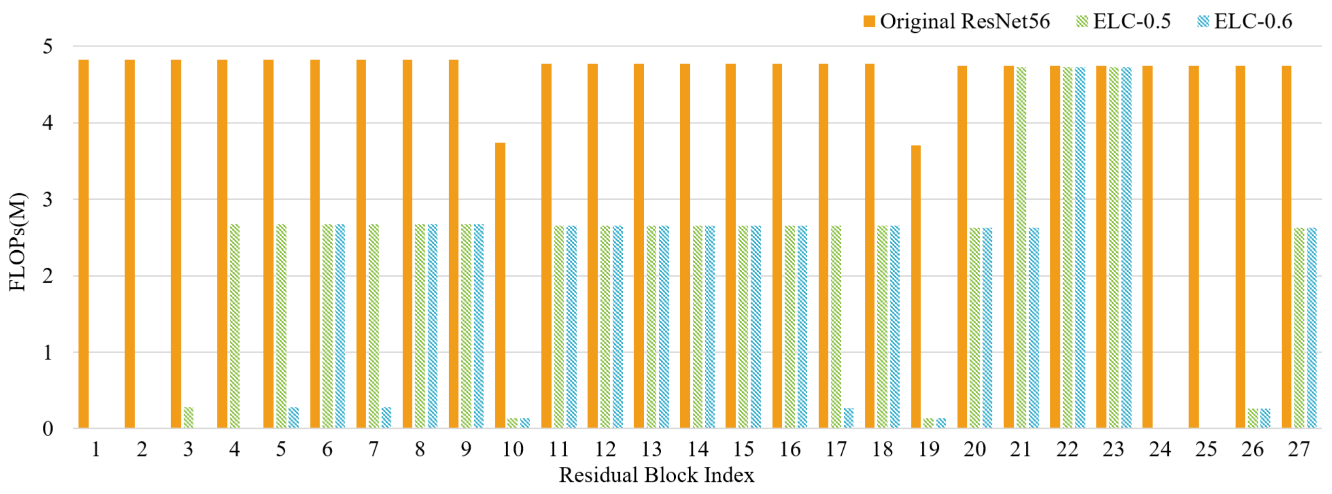


Fig. 5. Number of FLOPs with respect to each residual block in the original ResNet-56 and our compressed ResNet-56 (ELC-0.5 with 53.6% FLOPs drop and ELC-0.6 with 63.5% FLOPs drop).

TABLE IV

Performance Comparisons on ImageNet. "ACC." Denotes the Top-1 Accuracy. The Best Results in Terms of Accuracy Drop Are in **Bold**. The Second-Best Results Are underlined

Model	Methods	Type	Baseline ACC.%	Pruned ACC. ↑%	ACC. Drop ↓%	FLOPs ↓B	FLOPs Drop ↓%
ResNet-34	Taylor [63] (CVPR2019)	Channel	73.31	72.83	0.48	2.83	23.1
	NISP [55] (CVPR2018)	Channel	73.31	73.04	0.28	2.68	27.3
	LPSR-0.3 [24] (SPL2022)	Layer	73.31	72.63	0.68	2.52	31.5
	ELC-0.3 (Ours)	Layer	74.02	73.79	**0.23**	**2.43**	**34.0**
	SFP [64] (IJCAI2018)	Channel	73.92	71.83	2.09	2.16	41.1
	FPGM [56] (CVPR2019)	Channel	73.92	72.63	1.29	2.16	41.1
	ELC-0.4 (Ours)	Layer	74.02	72.95	1.07	2.14	41.8
	DMC [65] (CVPR2020)	Channel	73.30	72.57	0.73	2.10	42.9
	ELC-0.5 (Ours)	Layer	74.02	71.16	2.86	1.84	50.0

five pruning methods, namely, Taylor expansion (Taylor) [63], layer pruning for shallow ResNets (LPSR) [24], neuron importance score propagation (NISP) [55], soft filter pruning (SFP) [64], discrete model compression (DMC) [65], and filter pruning via geometric median (FPGM) [56]. Table IV shows results of ResNet-34 on ImageNet. Our proposed ELC-0.3 with 34.0% FLOPs reduction only has 0.23% accuracy drop and obtains the best accuracy (73.79%). Compared to LPSR-0.3 with a similar drop in FLOPs, our ELC-0.3 achieves 1.05% higher accuracy than LPSR-0.3. As the FLOPs drop increases to 41.8%, our proposed method still achieves a promising performance (1.06% accuracy drop) compared with recent approaches (SFP with 2.09% accuracy drop and FPGM with 1.29% accuracy drop). These results verify the effectiveness of our approach on layer compression. We also plot the FLOPs with respect to each residual block on the original ResNet-34, ELC-0.3, ELC-0.4, and ELC-0.5. We show these results in Figure 6. It can be seen that the when residual block index is 1, 2, 24, and 25, the FLOPs of ELC-0.5 and ELC-0.6 is zero. We also report the FLOPs with respect to each residual block on the original ResNet-34, ELC-0.3, and ELC-0.5 (shown in Figure 6).

C. Training Cost Comparison on ImageNet

Our proposed ELC does not require a finetuning process to increase accuracy, so our ELC has low training cost. To validate that our ELC has lower training costs, we compare the training epochs of some pruning methods with our ELC method. Since many methods do not provide training epochs, we only take Hrank [54], DMC [65], PFP [66], and LSTM [21] for comparison. The training cost comparison on ImageNet is shown in Table V. It can be seen that Hrank and DMC need more than 400 training epochs. PFP and LSTM still require 270 and 290 epochs to converge. The training epochs of our ELC is only 240, which is less than these pruning methods.

D. Model Deployment on Edge Devices

To validate the acceleration performance of our proposed layer compression method on edge devices, we deploy the compressed models on NVIDIA Jetson AGX Xavier, which is an embedding device including an integrated Volta GPU with Tensor Cores, octal-core NVIDIA Carmel ARMv8.2 CPU, 32GB 256-bit LPDDR4x with 137GB/s of memory bandwidth. We test every model 1000 times on Jetson AGX, and averaged all time-consuming results to get the execution time per

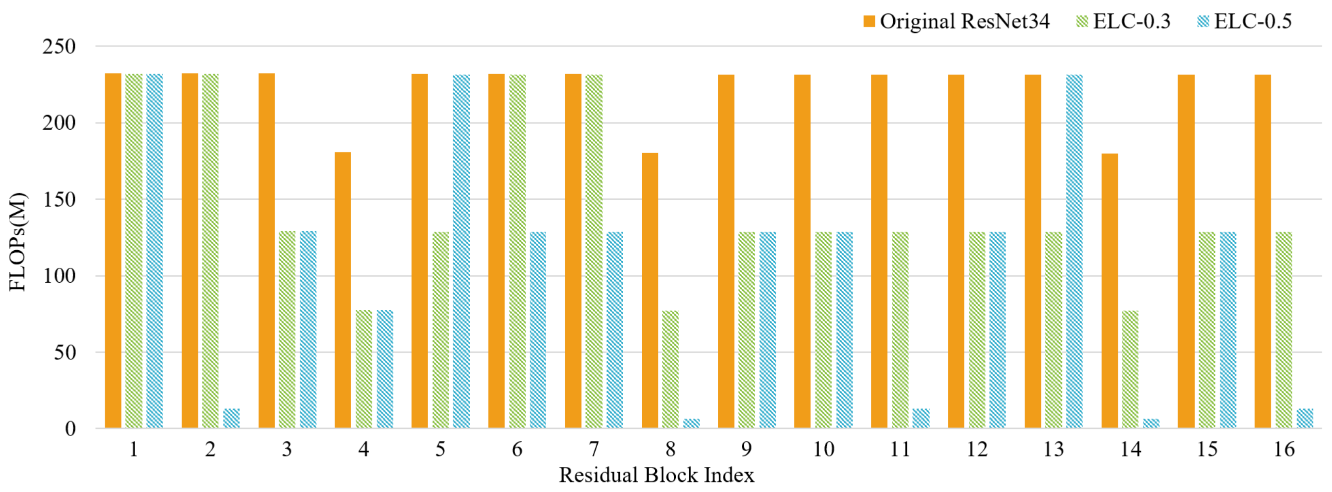


Fig. 6. Number of FLOPs with respect to each residual block in the original ResNet-34 and our compressed ResNet-34 (ELC-0.3 with 34.0% FLOPs drop and ELC-0.5 with 50.0% FLOPs drop).

TABLE V
TRAINING COST COMPARISON OF DIFFERENT PRUNING METHODS AND OUR ELC METHOD ON IMAGENET

Methods	Type	Epochs
Hrank [54]	Channel	570
DMC [65]	Channel	490
LSTM [21]	Channel	290
PFP [66]	Channel	270
ELC (Ours)	Layer	240

TABLE VI
INFERENCE ACCELERATION OF THE COMPRESSED MODELS ON NIVIDIA JETSON AGX XAVIER

Models	Methods	FLOPs	Execution Time	Speedup Ratio
ResNet-56	Baseline	125.0M	2.62ms	1.0
	Hrank [54]	62.7M	1.40ms	1.8
	FPGM [56]	59.4M	1.62ms	1.6
	ELC (Ours)	**58.3M**	**1.26ms**	**2.1**
ResNet-34	Baseline	3.67B	8.73ms	1.0
	SFP [64]	2.16B	7.67ms	1.1
	ELC (Ours)	**1.84B**	**4.26ms**	**2.0**

model. We report the results in Table VI. For ResNet-56, the time-consuming of the compressed model reduces to 1.36ms, compared with the original ResNet-56 network. ResNet-34 with the 50% FLOPs drop has a speedup ratio of 2.0×. Compared with Hrank [54], FPGM [64], and SFP [64], the proposed ELC has competitive speedup results. These experiments demonstrate that our proposed layer compression method has the ability to compact the model and reduce the execution time on edge devices.

E. Ablation Studies

In this section, we analyze the effectiveness of the proposed ELC in three aspects: 1) effectiveness of ELC on preserving layer correlations. 2) effect of the Rem-ReLU. 3) the generalization at different pruning ratios. 4) the comparison with reparameterization.

TABLE VII
COMPARISONS ON PRESERVING LAYER CORRELATIONS ON CIFAR-10. ELC-0.7 IS THE COMPRESSED VGG-16 NETWORK WITH FOUR LAYERS REDUCTION. ELC-CUT IS A DIRECTLY CUT VGG-16 NETWORK

Methods	Acc. (%)	FLOPs (M)
ELC-Cut	93.01	190.8
ELC-0.7	**93.11**	**81.7**

1) Effectiveness of ELC on Preserving Layer Correlations: To investigate the effectiveness of our proposed ELC in preserving layer correlations, we conduct a set of experiments. Since the ELC-0.7 on VGG-16 cut 4 layers (layer 2, layer 7, layer 9 layer 13), we directly cut these layers and finetune the cut network (named as ELC-Cut) on CIFAR-10. The hyper-parameters are set the same as in Section IV-A. The results are tabulated in Table VII. We can observe that the proposed ELC-0.7 outperforms the ELC-Cut by 0.10% in terms of accuracy. Besides, the proposed ELC replaces some 3×3 convolutional layers with 1×1, and so the FLOPs drop of ELC exceeds the directly cut network by 34.8%.

2) Effectiveness of the Rem-ReLU: We design an experiment to study the effects of the Rem-ReLU (ResNet-56 without and with Rem-ReLU) on CIFAR-10. The model without Rem-ReLU means that all activation functions of the model are classical ReLU functions. Convolutional layers of both models are all replaced with De-Conv. The experimental settings of these two models are the same. As shown in Table VIII, it is clear that the model with Rem-ReLU has better FLOPs drop and accuracy than the model without Rem-ReLU (FLOPs drop increases 1.9% and accuracy increases 0.74%). Hence, the proposed Rem-ReLU can increase the compression ratio and promote better results.

3) The Generalization at Different Pruning Ratios: We demonstrate that our proposed ELC can converge well at

TABLE VIII
EFFECT OF THE REM-RELU ON CIFAR-10. BASELINE IS THE ORIGINAL RESNET-56

Model	ACC. (%)	FLOPs (M)	FLOPs Drop (%)
Baseline	93.45	125.0	-
Without Rem-ReLU	92.79	60.3	51.8
With Rem-ReLU	**93.53**	**57.9**	**53.7**

TABLE IX
RESULTS OF RESNET-56 AT DIFFERENT COMPRESSION RATES ON CIFAR-10

Methods	ACC. (%)	ACC. Drop (%)	FLOPs (M)	FLOPs Drop (%)
Baseline	93.45	-	125	-
ELC-0.3	94.29	-0.84	81.8	34.6
ELC-0.5	93.66	-0.21	58.3	53.4
ELC-0.6	93.12	0.33	45.3	63.8
ELC-0.7	91.57	1.88	29.8	76.2
ELC-0.8	89.06	4.39	17.2	86.2

TABLE X
COMPARISONS RESULTS ABOUT REPVGG [28] ON CIFAR-10 DATASET

Model	Method	ACC. (%)	FLOPs (M)	FLOPs Drop (%)
VGG16	Baseline	93.56	313.7	-
	Baseline + RepVGG	**94.19**	313.7	-
	ELC-0.5	93.25	144.9	52.5
	ELC-0.7	93.11	**81.7**	**74.0**
ResNet56	Baseline	93.45	125.0	-
	Baseline + RepVGG	93.28	125.0	-
	ELC-0.5	**93.66**	**58.3**	**53.4**
	ELC-0.6	93.12	45.3	63.8

different ratios in this part. We obtain five pruned ResNet-56 models with different pruning ratios and show the results in Table IX. When the pruning ratio is 34.6%, the pruned network gets an accuracy increase of 0.84% than the baseline model. ELC-0.6 reduces 63.8% FLOPs and only decreases 0.33% in terms of accuracy. ELC-0.8 with FLOPs of 17.2M still has an accuracy of 89.06%. As the pruning ratio increases, our proposed ELC still converges well.

4) The Comparison With the Reparameterization Strategy: We compare our proposed method with the reparameterization strategy from RepVGG [28]. The reparameterization strategy can be seen as a model enhancement strategy but cannot reduce the network FLOPs. We have trained the VGG16 and ResNet56 with the training strategy of RepVGG. We observe that the RepVGG does enhance the performance of the VGG structure. But for ResNet, the model trained with the reparameterization strategy has a lower accuracy than the baseline model. Compared with RepVGG, our proposed method can significantly reduce the FLOPs and generalizes well on different models. The specific experimental settings and results are as follows.

Specifically, we add a new parallel 1×1 convolutional layer to the 3×3 convolutional layer on VGG16 and ResNet56 networks as the RepVGG does. The comparison results are shown in Table X. The VGG16 model trained with RepVGG strategy has the accuracy of 94.19%. When VGG16 is pruned by our proposed ELC, it drops 52.5% FLOPs and has a reduction of accuracy of 0.94%. For ResNet56, the baseline model trained with RepVGG strategy has an accuracy of 93.28%. We prune 53.4% FLOPs of ResNet56 model but increase the accuracy of 0.38%. The ELC-0.6 on ResNet56 with a pruning ratio of 63.8% only drops 0.16% in terms of accuracy.

V. CONCLUSION

In this paper, we have proposed an efficient layer compression method without pruning. Specifically, we first introduced De-Conv and Rem-ReLU to decouple convolutional layers and non-linear activation layers, making the network mergeable. Then, based on the property of linear mergeability of convolutional layers, we losslessly merged the decoupled layers achieving efficient layer compression. Compared with current pruning methods, our proposed ELC did not prune any channels or layers, preserving the network correlation. Experimental results demonstrated that the proposed ELC can reach high FLOPs drop and maintain high accuracy. Besides, the proposed ELC was more friendly to edge devices, where the inference time of the compressed model was reduced by $2\times$ on the Jetson AGX Xavier edge device.

In our future work, we will use quantization method on ELC model to achieve more efficient acceleration on edge devices. In addition, we will apply the proposed model compression method for object detection, image segmentation, and other visual tasks. Furthermore, we will study the structure of depth-wise and pointwise convolutions and design elaborate modules to compress these lightweight-designed modules efficiently.

ACKNOWLEDGMENT

The authors deepest gratitude goes to the editors and reviewers for their careful work and thoughtful suggestions that have helped improve this article substantially.

REFERENCES

[1] Y. Zhu, C. Zhao, H. Guo, J. Wang, X. Zhao, and H. Lu, "Attention CoupleNet: Fully convolutional attention coupling network for object detection," *IEEE Trans. Image Process.*, vol. 28, no. 1, pp. 113–126, Jan. 2019.
[2] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10778–10787.
[3] Y. Pan, Y. Xia, and D. Shen, "Foreground Fisher vector: Encoding class-relevant foreground to improve image classification," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4716–4729, Oct. 2019.
[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.
[5] L. Jing, Y. Chen, and Y. Tian, "Coarse-to-fine semantic segmentation from image-level labels," *IEEE Trans. Image Process.*, vol. 29, pp. 225–236, 2020.
[6] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[7] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.
[8] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2755–2763.

[9] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, “Pruning and quantization for deep neural network acceleration: A survey,” *Neurocomputing*, vol. 461, pp. 370–403, Oct. 2021.

[10] E. Doutsi, L. Fillatre, M. Antonini, and P. Tsakalides, “Dynamic image quantization using leaky integrate-and-fire neurons,” *IEEE Trans. Image Process.*, vol. 30, pp. 4305–4315, 2021.

[11] X. Yan, Y. Fan, K. Chen, X. Yu, and X. Zeng, “QNet: An adaptive quantization table generator based on convolutional neural network,” *IEEE Trans. Image Process.*, vol. 29, pp. 9654–9664, 2020.

[12] S. Chen and Q. Zhao, “Shallowing deep networks: Layer-wise pruning based on feature representations,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 12, pp. 3048–3056, Dec. 2019.

[13] X. Ding et al., “ResRep: Lossless CNN pruning via decoupling remembering and forgetting,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 4490–4500.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5MB model size,” 2016, *arXiv:1602.07360*.

[15] 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.

[16] J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge distillation: A survey,” *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.

[17] Z. Tu, X. Liu, and X. Xiao, “A general dynamic knowledge distillation method for visual analytics,” *IEEE Trans. Image Process.*, vol. 31, pp. 6517–6531, 2022.

[18] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, “Decoupled knowledge distillation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11943–11952.

[19] Y.-J. Zheng, S.-B. Chen, C. H. Q. Ding, and B. Luo, “Model compression based on differentiable network channel pruning,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 15, 2022, doi: 10.1109/TNNLS.2022.3165123.

[20] S. Gao, F. Huang, W. Cai, and H. Huang, “Network pruning via performance maximization,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 9266–9276.

[21] G. Ding, S. Zhang, Z. Jia, J. Zhong, and J. Han, “Where to prune: Using LSTM to guide data-dependent soft pruning,” *IEEE Trans. Image Process.*, vol. 30, pp. 293–304, 2021.

[22] Z. Wang, C. Li, and X. Wang, “Convolutional neural network pruning with structural redundancy reduction,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14908–14917.

[23] J. Guo, W. Zhang, W. Ouyang, and D. Xu, “Model compression using progressive channel pruning,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 3, pp. 1114–1124, Mar. 2021.

[24] K. Zhang and G. Liu, “Layer pruning for obtaining shallower ResNets,” *IEEE Signal Process. Lett.*, vol. 29, pp. 1172–1176, 2022.

[25] G. Li, C. Qian, C. Jiang, X. Lu, and K. Tang, “Optimization based layer-wise magnitude-based pruning for DNN compression,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2383–2389.

[26] S. Elkerdawy, M. Elhoushi, A. Singh, H. Zhang, and N. Ray, “One-shot layer-wise accuracy approximation for layer pruning,” in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 2940–2944.

[27] Z. Liu, X. Zhang, Z. Shen, Y. Wei, K.-T. Cheng, and J. Sun, “Joint multi-dimension pruning via numerical gradient update,” *IEEE Trans. Image Process.*, vol. 30, pp. 8034–8045, 2021.

[28] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun, “RepVGG: Making VGG-style ConvNets great again,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13728–13737.

[29] S. Han, J. Pool, J. Tran, and W. Dally, “Learning both weights and connections for efficient neural network,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1–9.

[30] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, “Compressing neural networks with the hashing trick,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2285–2294.

[31] T. Zhang et al., “A systematic DNN weight pruning framework using alternating direction method of multipliers,” in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 184–199.

[32] X. Ma et al., “PCONV: The missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices,” in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5117–5124.

[33] 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.

[34] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient ConvNets,” in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.

[35] 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. 1398–1406.

[36] J. Liu et al., “Discrimination-aware network pruning for deep model compression,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4035–4051, Aug. 2022.

[37] Z. Liu et al., “MetaPruning: Meta learning for automatic neural network channel pruning,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3295–3304.

[38] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, “Towards efficient model compression via learned global ranking,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1515–1525.

[39] A. Jordao, M. Lie, and W. R. Schwartz, “Discriminative layer pruning for convolutional neural networks,” *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 4, pp. 828–837, May 2020.

[40] W. Wang, S. Zhao, M. Chen, J. Hu, D. Cai, and H. Liu, “DBP: Discrimination based block-level pruning for deep model acceleration,” 2019, *arXiv:1912.10178*.

[41] P. Xu et al., “Layer pruning via fusible residual convolutional block for deep neural networks,” *Beijing Da Xue Xue Bao*, vol. 58, no. 5, pp. 801–807, 2022.

[42] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–18.

[43] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–16.

[44] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, “Neural architecture optimization,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7827–7838.

[45] M. Zhang, H. Li, S. Pan, X. Chang, and S. Su, “Overcoming multi-model forgetting in one-shot NAS with diversity maximization,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 7806–7815.

[46] Y. Guo et al., “NAT: Neural architecture transformer for accurate and compact architectures,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 737–748.

[47] Y. Guo et al., “Towards accurate and compact architectures via neural architecture transformer,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6501–6516, Oct. 2022.

[48] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, “Channel pruning via automatic structure search,” in *Proc. Int. Jt. Conf. Artif. Intell.*, 2021, pp. 673–679.

[49] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proc. Int. Conf. Artif. Intell. Stat.*, 2011, pp. 315–323.

[50] 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.

[51] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, “A ConvNet for the 2020s,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 11966–11976.

[52] S. Lin et al., “Towards optimal structured CNN pruning via generative adversarial learning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2785–2794.

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

[54] M. Lin et al., “HRank: Filter pruning using high-rank feature map,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1526–1535.

[55] R. Yu et al., “NISP: Pruning networks using neuron importance score propagation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[56] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter pruning via geometric median for deep convolutional neural networks acceleration,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4335–4344.

[57] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, “Learning filter pruning criteria for deep convolutional neural networks acceleration,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2006–2015.

[58] M. Soltani, S. Wu, J. Ding, R. Ravier, and V. Tarokh, “On the information of feature maps and pruning of deep neural networks,” in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 6988–6995.
[59] E. Diao et al., “Pruning deep neural networks from a sparsity perspective,” in *Proc. Int. Conf. Learn. Represent.*, 2022, pp. 1–28.
[60] A. Krizhevsky et al., “Learning multiple layers of features from tiny images,” M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.
[61] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[62] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014, *arXiv:1409.1556*.
[63] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, “Importance estimation for neural network pruning,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11256–11264.
[64] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2234–2240.
[65] S. Gao, F. Huang, J. Pei, and H. Huang, “Discrete model compression with resource constraint for deep neural networks,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1896–1905.
[66] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, “Provable filter pruning for efficient neural networks,” in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–29.

Jie Wu received the M.S. degree in electrical automation and information engineering from Tianjin University, Tianjin, China, in 2019. She is currently pursuing the Ph.D. degree in control science and engineering with Hunan University, Changsha, China.

Her research interests include model compression, remote sensing image processing, and deep learning

Dingshun Zhu received the B.S. degree from the South Central University for Nationalities, Wuhan, China, in 2020. He is currently pursuing the M.S. degree with the College of Electrical and Information Engineering, Hunan University, Changsha, China.

His research interests include remote sensing image processing, object detection, and lightweight models.

Leyuan Fang (Senior Member, IEEE) received the Ph.D. degree from the College of Electrical and Information Engineering, Hunan University, Changsha, China, in 2015.

From September 2011 to September 2012, he was a Visiting Ph.D. Student with the Department of Ophthalmology, Duke University, Durham, NC, USA, supported by the China Scholarship Council. From August 2016 to September 2017, he was a Postdoctoral Researcher with the Department of Biomedical Engineering, Duke University. He is currently a Professor with the College of Electrical and Information Engineering, Hunan University, and an Adjunct Researcher with the Peng Cheng Laboratory, Shenzhen, China. His research interests include sparse representation and multiresolution analysis in remote sensing and medical image processing.

Dr. Fang was a recipient of one 2nd-Grade National Award from the Nature and Science Progress of China in 2019. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and *Neurocomputing*.

Yue Deng received the B.E. degree (Hons.) in automatic control from Southeast University, Nanjing, China, in 2008, and the Ph.D. degree (Hons.) in control science and engineering from the Department of Automation, Tsinghua University, Beijing, China, in 2013.

He is currently a Faculty Member with the School of Astronautics, Beihang University, Beijing. His current research interests include machine learning, signal processing, and computational biology.

Zhun Zhong received the Ph.D. degree in computer science and technology from Xiamen University, China, in 2019.

He was a Joint Ph.D. Student with The University of Technology Sydney. He is currently with the School of Computer Science, University of Nottingham, U.K. His research interests include person re-identification and domain adaptation.