IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 9, 2019, accepted October 26, 2019, date of publication November 11, 2019, date of current version December 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952615

Incremental Layers Resection: A Novel Method to Compress Neural Networks

XIANG LIU[1], LI-NA WANG[2], WENXUE LIU[2], AND GUOQIANG ZHONG[2], (Member, IEEE)
[1]Department of Electrical Engineering, Ocean University of China, Qingdao 266100, China
[2]Department of Computer Science and Technology, Ocean University of China, Qingdao 266100, China

Corresponding author: Guoqiang Zhong (gqzhong@ouc.edu.cn)

This work was supported in part by the Major Project for New Generation of AI under Grant 2018AAA0100400, in part by the National Natural Science Foundation of China (NSFC) under Grant 41706010, in part by the Science and Technology Program of Qingdao under Grant 17-3-3-20-nsh, in part by the CERNET Innovation Project under Grant NGII20170416, in part by the Joint Fund of the Equipments Pre-Research and Ministry of Education of China under Grand 6141A020337, and in part by the Fundamental Research Funds for the Central Universities of China.

ABSTRACT In recent years, deep neural networks (DNNs) have been widely applied in many areas, such as computer vision and pattern recognition. However, we observe that most of the DNNs include redundant layers. Hence, in this paper, we introduce a novel method named incremental layers resection (ILR) to resect the redundant layers in DNNs, while preserving their learning performances. ILR uses a multistage learning strategy to incrementally resect the inconsequential layers. In each stage, it preserves the data representations learned by the original network, while connecting the two nearby layers of each resected one. Particularly, based on a teacher-student knowledge transfer framework, we have designed the layer-level learning and overall learning procedures to enforce the resected network performing similarly with the original one. Extensive experiments demonstrate that, compared to the original networks, the compressed ones by ILR need only about half of the storage space and have higher inference speed. More importantly, they even deliver higher classification accuracy than the original networks.

INDEX TERMS Network compression, acceleration, multistage learning, knowledge transfer, layers resection.

I. INTRODUCTION

In recent years, deep neural networks (DNNs) have attracted much attention in the areas related to artificial intelligence, such as computer vision and pattern recognition. Particularly, deep convolutional neural networks (CNNs) have obtained great successes in the applications of image classification and object detection, since AlexNet [2] won the champion of the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012. Following AlexNet, many excellent deep CNNs, such as GoogleNet [3], VGGNet [4] and ResNet [5], are proposed to achieve higher performances. However, the deeper and wider the neural networks are, the more storage space and running time they generally need. For the classification of 32*32*3 images, AlexNet with 8 layers occupies 30.4M disk space, whereas VGGNet with 16 layers needs 70M. To the end, many deep neural networks can only be deployed on the burdensome and expensive GPU servers. Under this circumstance, making DNNs lightweight to allow them running on the low-powered devices becomes a crucial and urgent challenge.

In the literature, many methods for compressing and accelerating DNNs have been proposed [6]–[16]. Among others, weight pruning approaches focus on removing redundant connections in the DNNs [7], while knowledge distillation (KD) methods are aimed to guide the training of a student network based on an existing teacher network [15], [16]. However, weight pruning approaches generally need to manually set the sensitivity of each connection, and can hardly improve the inference efficiency of the original network. Additionally, traditional KD methods can only be used to networks with the softmax classifier, and the capabilities of the teacher network and student network may vary widely.

In our work, we observe that although DNNs can obtain good results in many applications, they include redundant

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

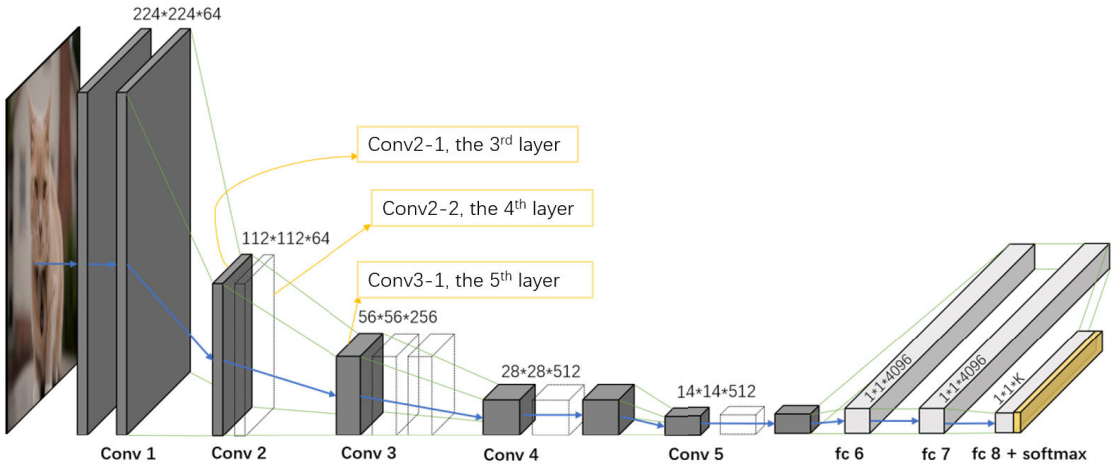


FIGURE 1. Layers resection on the VGGNet-16, which includes 5 convolutional blocks (conv 1-5) and 3 fully connected (fc) layers. The blank boxes denote the resected layers, while the gray boxes are those kept. The blue arrows indicate the flow path of the image representations. Assume the 4^{th} layer (Conv2-2) is a redundant layer. After resecting it, the outputs of the 3^{rd} layer (conv2-1) are directly fed into the 5^{th} layer (conv3-1).

layers. Some layers within the DNNs are inconsequential and the DNNs need not to be so deep as they are. Therefore, in this paper, we propose a new network compression method called incremental layers resection (ILR), to remove the redundant layers in DNNs. ILR combines the ideas of weight pruning and KD. It removes the inconsequential layers, and meanwhile, transfers the knowledge of the original network to the compressed one. In most cases, ILR can even improve the performances of the networks, as shown in Section V about the experimental results.

Fig. 1 illustrates the layers resection on the VGGNet-16. In the original network, the forward propagation from the 3^{rd} layer to the 5^{th} layer is as follows:

$$\begin{aligned} a_4^j &= g(W_4^j * a_3 + b_4^j), \\ a_5^j &= g(W_5^j * a_4 + b_5^j), \end{aligned} \tag{1}$$

where a_i^j denotes the activation output of the j-th channel in the i-th layer, W_i^j and b_i^j are the parameters, and g is the nonlinear activation function. That is, the output of the 3^{rd} layer are first sent into the 4^{th} layer, and then, the outputs of the 4^{th} layer are sent into the 5^{th} layer. Assume that the 4^{th} layer (Conv2-2) is an inconsequential layer. We resect it and connect the 3^{th} (conv2-1) and the 5^{th} (conv3-1) layers. To the end, the new forward propagation from the 3^{th} layer to the 5^{th} layer becomes

$$\hat{a}_5^j = g(\hat{W}_5^j * a_3 + \hat{b}_5^j), \tag{2}$$

where $\hat{a}$, $\hat{W}$, and $\hat{b}$ denote the updated values after the resection of the original 4^{th} layer. With the layers resection, the inference time and the size of the parameters can be reduced, as the networks becomes shallower than before. In order to preserve the performance of the networks, we propose a multistage learning strategy to fine tune the resected networks. From the perspective of teacher-student knowledge transfer, we can consider the original network as the teacher network, and the resected network as the student network. However, unlike KD [15], the student network here need to learn both the data representations at the resected layer and the final outputs of the teacher network. Fig. 2 shows the ILR learning process, which is detailedly introduced in Section III. As there may be many inconsequential layers, ILR can be applied iteratively until all of them are removed from the original network.

The contributions of this work can be summarized as follows.

1) Compared with previous weight pruning approaches, which mainly remove the redundant connections, ILR focuses on layer-level resection. The compressed networks by ILR in general have higher inference speed and less storage consumption than the original networks.
2) Compared with previous KD approaches, where the structures of the teacher and student networks may quite different, the student network inherits the structure of the teacher network and only inconsequential layers are removed during the ILR learning process.
3) In order to preserve the capacity of the original network, we propose a multistage learning strategy. After selecting an inconsequential layer, the resected network recovers the performance of the original one by alternating the layer-level learning and overall learning procedures. As the redundant layers are removed step by step, the original network is incrementally compressed, without any hurt to its performance.

In the following section, we review some related work about network compression. In Section III, we introduce the proposed ILR method in detail. In Section IV, we extend ILR to neural blocks resection. Section V presents the experimental results with comparison to related approaches. Section VI concludes this paper.

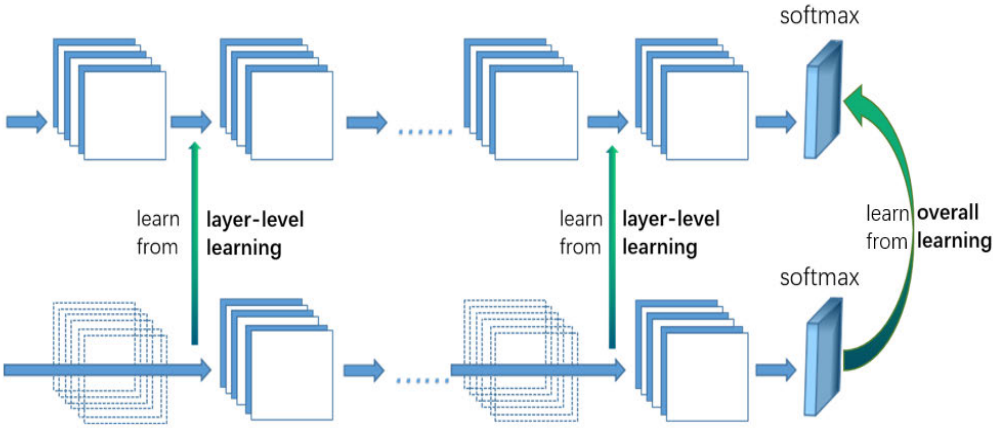


FIGURE 2. The learning process of ILR. The upper part shows the original (teacher) network, while the lower part shows the resected (student) network. After selecting one (or more) inconsequential layer(s), we alternate the layer-level learning and overall learning per epoch to train the student network. For concreteness, in one epoch, we update the student network with the data representations at the resected layer of the teacher network (layer-level learning), and in the next epoch, we fine-tune it with the outputs of the teacher network (overall learning). As a multistage learning strategy, the layer-level learning and overall learning procedures are repeated until reaching the maximum learning epoch. The whole learning process is terminated when there are no redundant layers.

II. RELATED WORK

In recent years, numerous network compression methods have been proposed. Among them, weight pruning, knowledge distillation and incremental network quantization are quite relevant to this work. We review them in the following.

A. WEIGHTS PRUNING

Weights pruning is to remove the redundant connections in a well-trained network [7], [17], [18]. In general, the weights with small absolute values are directly set to zero. Weight pruning can greatly reduce the storage requirement of the neural networks without affecting their classification performance. Particularly, similar neurons merging [19] can be considered as a special weight pruning approach, which merges similar neurons in the same layer based on clustering analysis. Moreover, weight pruning can also be applied on filters, which results in channel pruning [20]–[22]. In most cases, deactivating the redundant filters (setting their values to zero permanently) has little impact on the final outputs. In this work, we focus on pruning the inconsequential layers in DNNs. The proposed ILR can speed up the inference of the DNNs and reduce their storage requirement but without any hurt to their performances.

B. KNOWLEDGE DISTILLATION

Knowledge distillation (KD) is first proposed by Hinton and his collaborators in 2015 [15]. The main idea of KD is to train a student network with the guide of the outputs of a teacher network, so that it can generalize in the same way as the teacher network. KD has been proved to be an effective technique to improve the student network's performance in a teacher-student knowledge transfer manner. In particular, it has been applied in many tasks to deploy DNNs on devices with limited computational resources, such as text detection [25] and synthetic aperture radar (SAR) targets recognition [26]. Some papers propose to train the student network using not only the teacher network's outputs but also its intermediate information [27]–[30]. However, these work still attempts to teach an initial student network from scratch. The architectures of the teacher and student networks are distinct from each other. In contrast, the architecture of the student network in ILR is inherits from the teacher network. To preserve the capability of the teacher network, we adopt a multistage learning strategy to train the resected (student) network with both the data representations at the resected layer and the final outputs of the original (teacher) network.

C. INCREMENTAL NETWORK QUANTIZATION

In 2017, Zhou et al. propose a network compression method, called incremental network quantization (INQ) [23]. One of its core ideas is weights partition: a part of the weights are fixed to be either powers of two or zero, i.e. $\{0, \pm 2^0, \pm 2^{-1}, \cdots, \pm 2^{-n}\}$. This inevitably leads to accuracy loss, so that the other part of the weights are be updated to recover the classification accuracy. The ratio of the fixed parameters increases incrementally, and finally achieves 100%, i.e. the network is completely quantified. The methodology to learn incrementally is extensively applied in the machine learning research, such as online learning. In this paper, to preserve the capacity of the original network, we incrementally resect the redundant layers until all the redundant layers are removed.

III. INCREMENTAL LAYERS RESECTION

Fig. 2 illustrates the learning process of the proposed method, incremental layers resection (ILR). In the implementation of ILR, we first test the layers to find out the inconsequential ones, and then resect them from the network incrementally. In order to preserve the capability of the original network, we adopt a multistage learning strategy. After resecting an

inconsequential layer, we fine-tune the resected network with layer-level learning and overall learning alternately until it recovers the performance of the original network. We repeat the above process continuously until there are no inconsequential layers. In the following, we introduce the learning process of ILR in detail.

A. SELECTING INCONSEQUENTIAL LAYERS

Inconsequential layers are those we can remove them from a DNN without any hurt to the performance of the network. Based on our observations, the inconsequential layers usually play a role similar to identity mapping among DNNs. An ideal tool to measure how the weight matrix approximates to an identity one is based on its non-zero singular values. The closer the matrix approaches to an identity one, the closer its non-zero singular values are to 1. However, the weight matrix composed of the filters (represented as column vectors in the weight matrix) at a convolutional layer is generally too large and sparse to conduct the singular value decomposition (SVD). In this case, we propose a more practical method based on the network test and accuracy threshold, which can also be applied to the inconsequential layers not doing the identity mapping.

In order to select the inconsequential layers, we first set an accuracy threshold and split the original training set into a training set and a validation set. And then, we resect each layer independently from the first hidden layer to the last one, and obtain the layer resulting in the highest validation accuracy with the remainder network. If the highest validation accuracy is larger than the accuracy threshold, we consider the removed layer to be inconsequential. Otherwise, the removed layer is not inconsequential, we terminate the layers resection process and recover the resected layer back. Moreover, if an inconsequential layer is obtained, we remove it and test other layers until the validation accuracy of the resected network is lower than the accuracy threshold. For the accuracy threshold, we set it as a value based on the primary validation accuracy of the teacher network (Acc_T) and the number of the classes (C): $\frac{\alpha \cdot Acc\_T}{log_{10}C}$. Here, α is a hyperparameter, which we set it to 0.5 in our experiments. Additionally, considering the influence of C to the error tolerance of the resected networks, we normalize the accuracy threshold by $log_{10}C$.

Please note that, the first and last convolutional layers are very important for the representation learning: the first convolutional layer extracts primary features from the images, while the last convolutional layer connects to the fully connected layers and transforms the image features to vectorized representations. Therefore, we don't consider to resect these two layers in our method.

B. LAYER-LEVEL LEARNING

As shown in Fig. 1, suppose that the 4^{th} layer is an inconsequential layer. After resecting it, the 3^{rd} layer should be connected to the new 4^{th} layer directly. Here, the key problem is how to maintain the performance of the original network, letting the outputs of the 3^{rd} layer and that of the original 4^{th} layer as similar as possible. In other words, we need to enforce the student network to learn the layer-level knowledge of the teacher network.

Firstly, we assume the l^{th} layer is an inconsequential one. We use the Kullback-Leibler (KL) divergence to measure the similarity between the distributions of the outputs of the $(l-1)^{th}$ and l^{th} layers. It is expected that, after resecting the l^{th} layer and fine-tuning, the outputs of the $(l-1)^{th}$ layer approximate that of the original l^{th} layer. The KL divergence is defined as

$$\begin{aligned}\mathcal{L}_{KL} &= D_{KL}(P \parallel Q) \\ &= -\sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} P(x) log\left(\frac{Q(y)}{P(x)}\right).\end{aligned} \tag{3}$$

In our work, x represents the activation output of the $(l-1)^{th}$ layer of the student network, y represents the activation output of the l^{th} layer of the teacher network, $\mathcal{X}$ denotes the output set of the $(l-1)^{th}$ layer of the student network, $\mathcal{Y}$ denotes the output set of the l^{th} layer of the teacher network, $P(x)$ is the probability estimated by the occurrence frequency of x in $\mathcal{X}$, and $Q(y)$ is the probability estimated by the occurrence frequency of y in $\mathcal{Y}$. The KL divergence is always a non-negative value. Its gradient with respect to x can be computed as

$$\begin{aligned}\frac{\partial}{\partial x}\mathcal{L}_{KL} &= \frac{\partial}{\partial x} D_{KL}(P \parallel Q) \\ &= -\sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} \frac{\partial}{\partial x} P(x)\left(log\frac{Q(y)}{P(x)} - 1\right).\end{aligned} \tag{4}$$

However, as KL divergence measures the holistic difference between two distributions, it cannot enforce the element-wise approximation between the outputs of the $(l-1)^{th}$ layer of the student network and that of the l^{th} layer of the teacher network. In this case, we apply an element-wise regularization to encourage the knowledge transfer from the teacher network to the student network:

$$\mathcal{L}_{ew} = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} log(1 + \|x-y\|_1). \tag{5}$$

Note that minimizing $\mathcal{L}_{ew}$ can lead to sparse $x - y$, which corresponds to that x is as close as possible to y. To the end, the learned representations of the $(l-1)^{th}$ layer of the student network are very similar to that of the l^{th} layer of the teacher network. The gradient of $\mathcal{L}_{ew}$ with respect to x can be calculated as

$$\frac{\partial}{\partial x}\mathcal{L}_{ew} = \sum_{\substack{x\in\mathcal{X}\\ y\in\mathcal{Y}}} \frac{sign(x-y)}{1 + \|x-y\|_1}. \tag{6}$$

To this end, the loss function for the layer-level learning can be written as

$$\mathcal{L}oss^{L} = \lambda\mathcal{L}_{KL} + (1-\lambda)\mathcal{L}_{ew}, \tag{7}$$

where λ is a trade-off hyperparameter between the KL divergence loss and the element-wise loss. We set it to 0.8 in our experiments.

Note that as $\mathcal{L}oss^L$ includes both the distribution level and element-wise losses, minimizing it can result in that the $(l-1)^{th}$ layer of the student network learns similar data representations with the l^{th} layer of the teacher network. This means that the inconsequential layer in the original network can be safely resected.

C. OVERALL LEARNING

The layer-level learning introduced in Sec. III-B enforces the outputs of the layer in front of the resected one (in the student network) to be as same as that of the resected one (in the teacher network). However, it cannot directly affect the final performance of the student networks. Motivated by the idea of KD, we apply an overall learning to the student network and encourage it performing similarly to the teacher network. The overall learning uses both of the soft and hard targets:

$$\mathcal{L}oss_{hard} = -\frac{1}{N}\sum_{n=1}^{N}[\, y_n \log(\hat{y}_n)+(1-y_n)\log(1-\hat{y}_n)], \quad (8)$$

$$\mathcal{L}oss_{soft} = -\frac{1}{N}\sum_{n=1}^{N}[\, p_n \log(q_n)+(1-p_n)\log(1-q_n)]. \quad (9)$$

In Eq. (8) and Eq. (9), y_n and p_n are targets that the student network learns from. For concreteness, y_n is the hard targets (true label) represented as a one-hot vector, and the value of its element can only be 0 or 1; whereas, p_n is the soft target, which is the softmax output calculated by the teacher network. Moreover, $\hat{y}_n$ and q_n are the final hard and soft outputs of the student network.

The loss function of the overall learning can be defined as:

$$\mathcal{L}oss^O = \xi\mathcal{L}oss_{soft} + (1-\xi)\mathcal{L}oss_{hard}. \quad (10)$$

Note that in ILR, ξ is not a fixed number, but a function $\xi = tanh(\frac{epoch}{\nu})$, where ν is a hyperparameter larger than 1. We suggest to set it as the half of the total epochs. That is, at the beginning to train the student network, we expect it to learn more from the soft targets, and when its training is relatively stable, it can accept more knowledge from the hard targets.

Here, $\mathcal{L}oss^O$ measures the difference between the teacher and student networks. The smaller the $\mathcal{L}oss^O$ is, the more similar the final outputs of the teacher and student networks are. Therefore, minimizing the $\mathcal{L}oss^O$ enforces the student network to achieve competitive performance with the teacher network.

As mentioned in Sec. II-C, layers resection is an incremental process. We first select the inconsequential layers step by step, resect them, and then fine-tune the resected network with $\mathcal{L}oss^L$ and $\mathcal{L}oss^O$, alternately. This procedure is continuously repeated until there are no inconsequential layers. The incremental layers resection algorithm is summarized in *Algorithm 1*.

Algorithm 1 Incremental Layers Resection

Input: The original teacher network T.
Output: The student network S after layers resection.
Repeat:
  Find the inconsequential layer(s) h
  $S <= T$ %% Copy T to S
  Resect h from S
  $S := \mathcal{L}oss^L \& \mathcal{L}oss^O$ (alternately)
  %% Update S with $\mathcal{L}oss^L$ and $\mathcal{L}oss^O$ alternately
  $T <= S$
Until no inconsequential layers
Return S

IV. BLOCKS RESECTION

In this section, we extend the layers resection to blocks resection. To facilitate the construction of deep neural networks, recent deep architectures are in general designed with neural blocks, such as VGGNet [4] and ResNet [5]. In the blocks resection, we select the inconsequential blocks in a deep neural network, and then remove them from the network just like resecting the inconsequential layers from a network. The resection of a block from a network is illustrated in Fig. 3.

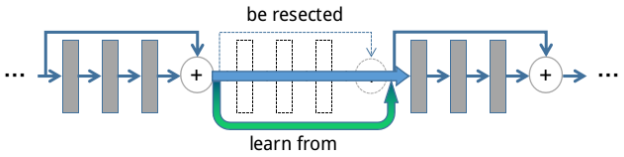


FIGURE 3. Resection of a block.

Here, it's unnecessary to worry about that resecting many layers (in a block) at the same time may damage the information transmission of the network, as the block is indeed inconsequential and it only approximately performs as an identity mapping, especially for the residual blocks in ResNet [5]. Furthermore, for blocks with bottleneck architecture, we do not suggest to resect a convolutional layer alone from the entire block, since dimensions of data representations vary among the layers and resecting one layer may result in loss of important information.

V. EXPERIMENTS

In order to test the effectiveness and universality of ILR on network compression, we conducted experiments on four data sets, i.e. the CIFAR-10/100[1], SVHN[2], and CalTech-101[3] data sets, and applied ILR on various deep architectures, such as VGGNet [4], ResNet [5], ResNeXt [39] and SENet [40]. In the following, we introduce the used data sets, the applied models, and report the experimental results with comparison to related approaches.

[1] http://www.cs.toronto.edu/ kriz/cifar.html
[2] http://ufldl.stanford.edu/housenumbers/
[3] http://www.vision.caltech.edu/Image_Datasets/Caltech101/

FIGURE 4. Some example images in the four used data sets. (a) CIFAR-10/CIFAR-100; (b) SVHN; (c) CALTECH-101.

A. THE USED DATA SETS

In our experiments, we used four data sets to evaluate the effectiveness of ILR. They were the CIFAR-10/100, SVHN, and CalTech-101 data sets. Some example images from these data sets are shown in Fig. 4. As below, we introduce them briefly.

1) CIFAR-10/100

The CIFAR-10 and CIFAR-100 data sets are established by Krizhevsky el al. for testing deep learning algorithms. Both of them consist of 60,000 of color images of size 32*32*3, 50,000 in which are used for training(contained the 1k validation sets), and the remainder 10,000 for test. The difference between them is the number of classes, as CIFAR-100 further splits the classes of CIFAR-10.

2) SVHN

The Street View House Numbers (SVHN) data set is a color image data set for digits recognition. We cropped and chose 14879 single number images from this data set, and resized them to 32*32. Among them, 11903 images (about 80%) were used for training, and 2976 images (about 20%) for test. Specifically, 1/4 of the training set were used for the validation in our experiments.

3) CalTech-101

CalTech-101 is established by Li et al.. It consists of 9145 images in 102 categories (including a category of background), each images in which has roughly 200*300*3 pixels. In our experiments, 7316 images (about 80%) were used for training, and the rest 1829 images (about 20%) for test. Among the training set, 1/4 of the training set were used for validation. All of the images were resized to 224*224*3, when they were fed into the deep networks.

B. THE APPLIED MODELS

In our experiments, we applied ILR to various DNNs, including the VGGNet [4], ResNet [5], ResNeXt [39], and Squeeze-and-excitation networks (SENet) [40], to test its universality. In the following, we introduce these models briefly.

1) VGGNet

VGGNet was proposed by Simonyan and Zisserman in 2014 [4]. It demonstrated that using small filters and increasing the depth of networks can enhance the performance of deep CNNs. Specifically, VGGNet has high generalization ability on image classification tasks. In the original VGGNet, it includes 13 convolutional layers and 3 full connected (''fc'' for short) layers. In our experiments, since we attempted to resect the convolutional layers, we only used one fc layer in VGGNet to reduce its influence on the performance of the network.

2) ResNet

ResNet won both the ImageNet Large Scale Visual Recognition Competition (ILSVRC) and Microsoft Common Objects in Context (COCO) Competition [38] in 2015 [5]. It can go as deep as more than 1,000 layers without gradient vanishing and explosion, due to the usage of residual learning, as introduced in Section IV. Here, the problem is that lots of redundant computation is brought in this deep architecture.

3) ResNeXt

ResNeXt was proposed by Xie et al. in 2017, which was an enhanced version of ResNet [39]. ResNeXt improved the performance of traditional ResNet by replacing its primary convolutional blocks with aggregated transformations. However, although ResNeXt has achieved high classification accuracy in many image classification tasks, it has the same problem as ResNet that it includes inconsequential computation.

4) SENet

Squeeze-and-excitation networks (SENet) won the last champion of ILSVRC in 2017, attaining 5.73% top-5 error on the ImageNet-1K object recognition task [40]. In SENet, the authors designed the squeeze-and-excitation module, which can be embedded in the ResNet and GoogLeNet. In our experiments, we used the SE-ResNet-v2 given in [40]. As a result, our experiments indicated the SENet can also be compressed and improved.

TABLE 1. Experimental results obtained by ILR and the compared approaches on the CIFAR-10 and SVHN data sets. The "Reconstruction" model has the same architecture as the resected networks by ILR, but is retrained from scratch. The inference time is calculated on the CIFAR-10 data set. On both of the data sets, the object recognition accuracy is reported.

Networks	# of params	Storage	Compression ratio	Inference time	Speedup rate	GFLOPS	CIFAR-10	SVHN
Original VGGNet	14.73M	57M	1 ×	7.45 ± 0.3 s	1 ×	0.31	94.04%	93.57%
Han's [7]	14.73M	424K	137.66 ×	7.45 ± 0.3 s	1 ×	0.31	93.60%	92.37%
Reconstruction	6.32M	25M	2.28 ×	5.8 ± 0.3 s	1.28 ×	0.12	92.92%	92.90%
KD [15]	6.32M	25M	2.28 ×	5.8 ± 0.3 s	1.28 ×	0.12	92.92%	92.99%
ILR	6.32M	25M	2.28 ×	5.8 ± 0.3 s	1.28 ×	0.12	**94.06%**	**93.71%**
ILR + Han's [7]	6.32M	**232K**	**251.59** ×	5.8 ± 0.3 s	1.28 ×	0.12	93.66%	92.84%
Original ResNet	2.26M	8.8M	1 ×	4.93 ± 0.2 s	1 ×	0.18	92.50%	92.23%
Han's [7]	2.26M	1.6M	5.50 ×	4.93 ± 0.2 s	1 ×	0.18	91.96%	92.01%
Reconstruction	1.08M	4.2M	2.09 ×	3.24 ± 0.1 s	1.52 ×	0.09	91.24%	91.36%
KD [15]	1.08M	4.2M	2.09 ×	3.24 ± 0.1 s	1.52 ×	0.09	91.68%	91.92%
ILR	1.08M	4.2M	2.09 ×	3.24 ± 0.1 s	1.52 ×	0.09	**92.70%**	**92.44%**
ILR + Han's [7]	1.08M	**804K**	**11.21** ×	3.24 ± 0.1 s	1.52 ×	0.09	92.30%	92.12%
Original ResNeXt	2.26M	8.9M	1 ×	4.82 ± 0.2 s	1 ×	0.20	93.05%	92.66%
Han's [7]	2.26M	1.6M	5.56 ×	4.82 ± 0.2 s	1 ×	0.20	91.90%	92.02%
Reconstruction	1.07M	4.15M	2.14 ×	3.0 ± 0.1 s	1.61 ×	0.10	91.93%	91.85%
KD [15]	1.07M	4.15M	2.14 ×	3.0 ± 0.1 s	1.61 ×	0.10	92.44%	92.30%
ILR	1.07M	4.15M	2.14 ×	3.0 ± 0.1 s	1.61 ×	0.10	**93.15%**	**92.66%**
ILR + Han's [7]	1.07M	**685K**	**13.3** ×	3.0 ± 0.10 s	1.61 ×	0.10	92.24%	92.12%
Original SENet	4.56M	17.76M	1 ×	8.2 ± 0.1 s	1 ×	0.23	94.06%	93.30%
Han's [7]	4.56M	1.4M	12.68 ×	8.2 ± 0.1 s	1 ×	0.23	93.40%	92.24%
Reconstruction	2.04M	7.94M	2.24 ×	5.6 ± 0.15 s	1.57 ×	0.105	92.88%	92.05%
KD [15]	2.04M	7.94M	2.24 ×	5.6 ± 0.15 s	1.57 ×	0.105	93.68%	92.98%
ILR	2.04M	7.94M	2.24 ×	5.6 ± 0.15 s	1.57 ×	0.105	**94.10%**	**93.41%**
ILR + Han's [7]	2.04M	**690K**	**26.36** ×	5.6 ± 0.15 s	1.57 ×	0.105	93.45%	92.57%

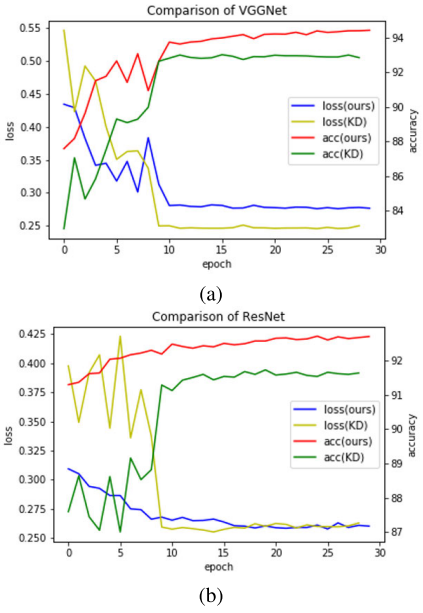


FIGURE 5. Comparison of accuracy and loss curves between knowledge distillation and ours during the last fine-tuning. (a) is the representation of VGGNet on CIFAR-10, and (b) is the representation of ResNet on CIFAR-10.

C. EXPERIMENTAL RESULTS

To show the redundancy of the layers in deep neural networks, we sequentially resected the inconsequential layers in the used ResNet on the CIFAR-10 data set. Table 3 shows the validation accuracy obtained by the resected ResNet. Note that, we directly resected the blocks, which include 3 layers, in our experiments. As we can see, some inconsequential blocks perform approximate to an identity mapping, such as blocks 16, 17 and 6. We can safely remove them without much hurt to the performance of the network and the validation accuracy can be easily recovered after fine-tuning. Moreover, for other inconsequential layers, ILR can find and remove them also to effectively compress the networks.

In order to evaluate the performance of ILR for network compression, we compared it with Han's deep compression method [7], the reconstructed model that shared the same architecture with the resected networks by ILR but was retrained from scratch, and the knowledge distillation method [15]. Moreover, we extended ILR by combining it with Han's deep compression method [7] to further compress the redundant weights in the networks. The results are also reported. Additionally, we compared ILR and the related approaches in many aspects, such as the classification performance of the resected networks, the compression rate, the giga floating-point operations per second (GFLOPS), and the speedup rate. Specifically, in order to accurately compare the inference time of the networks, we tested the networks on a low-powered GTX950M GPU. It consumes much more inference time compared to the data input/output time.

Table 1 shows the experimental results on the CIFAR-10 and SVHN data sets, both of which are 10-class classifica-

TABLE 2. Experimental results obtained by ILR and the compared approaches on the CIFAR-100 and CalTech-101 data sets. The "Reconstruction" model has the same architecture as the resected networks by ILR, but is retrained from scratch. The inference time is calculated on the CalTech-101 data set. On both of the data sets, the object recognition accuracy is reported.

Networks	# of params	Storage	Compression ratio	Inference time	Speedup rate	GFLOPS	CIFAR-100	CalTech-101
Original VGGNet	14.77M	57.5M	1 ×	4.2 ± 0.3 s	1 ×	0.34	73.09%	86.60%
Han's [7]	14.77M	455K	129.41 ×	4.2 ± 0.3 s	1 ×	0.34	71.86%	86.09%
Reconstruction	7.10M	27.6M	2.08 ×	3.3 ± 0.2 s	1.27 ×	0.13	72.41%	85.19%
KD [15]	7.10M	27.6M	2.08 ×	3.3 ± 0.2 s	1.27 ×	0.13	72.83%	86.24%
ILR	7.10M	27.6M	2.08 ×	3.3 ± 0.2 s	1.27 ×	0.13	**73.28%**	**86.83%**
ILR + Han's [7]	7.10M	**260K**	**226.46**×	3.3 ± 0.2 s	1.27 ×	0.13	72.44%	86.44%
Original ResNet	2.31M	9.1M	1 ×	5.2 ± 0.2 s	1 ×	0.21	72.72%	85.94%
Han's [7]	2.31M	1.32M	6.89 ×	5.2 ± 0.2 s	1 ×	0.21	71.84%	84.87%
Reconstruction	1.31M	4.97M	1.83 ×	3.4 ± 0.1 s	1.53 ×	0.12	71.96%	84.31%
KD [15]	1.31M	4.97M	1.83 ×	3.4 ± 0.1 s	1.53 ×	0.12	72.60%	85.65%
ILR	1.31M	4.97M	1.83 ×	3.4 ± 0.1 s	1.53 ×	0.12	**72.70%**	**86.22%**
ILR + Han's [7]	1.31M	**933K**	**9.99** ×	3.4 ± 0.1 s	1.53 ×	0.12	72.05%	85.42%
Original ResNeXt	2.31M	9.1M	1 ×	4.9 ± 0.2 s	1 ×	0.20	73.65%	87.33%
Han's [7]	2.31M	1.6M	5.69 ×	4.9 ± 0.2 s	1 ×	0.20	73.05%	86.12%
Reconstruction	1.17M	4.6M	1.97 ×	2.8 ± 0. s	1.75 ×	0.11	73.08%	85.15%
KD [15]	1.17M	4.6M	1.97 ×	2.8 ± 0. s	1.75 ×	0.11	73.40%	86.62%
ILR	1.17M	4.6M	1.97 ×	2.8 ± 0. s	1.75 ×	0.11	**74.06%**	**87.45%**
ILR + Han's [7]	1.17M	**710K**	**13.12** ×	2.8 ± 0. s	1.75 ×	0.11	73.66%	86.90%
Original SENet	4.62M	18.0M	1 ×	5.5 ± 0.2 s	1 ×	0.23	75.50%	88.23%
Han's [7]	4.62M	1.6M	11.25 ×	5.5 ± 0.2 s	1 ×	0.23	74.56%	87.00%
Reconstruction	2.52M	9.8M	1.84 ×	3.5 ± 0.2 s	1.57 ×	0.12	75.04%	86.32%
KD [15]	2.52M	9.8M	1.84 ×	3.5 ± 0.2 s	1.57 ×	0.12	74.24%	87.93%
ILR	2.52M	9.8M	1.84 ×	3.5 ± 0.2 s	1.57 ×	0.12	**75.65%**	**88.43%**
ILR + Han's [7]	2.52M	**745K**	**24.74** ×	3.5 ± 0.2 s	1.57 ×	0.12	75.08%	87.26%

TABLE 3. Validation accuracy obtained by the resected ResNet on the CIFAR-10 data set. Here, "R" and "F-T" are short for "removing" and "fine-tuning". The resection was a sequential process, so that the accuracy before removing block 6 was that after fine-tuning block 17.

	block 16	block 17	block 6	···	block 11	block 4
Before R	92.50%	92.64%	92.99%	···	93.12%	92.60%
After R	91.68%	91.03%	91.16%	···	81.58%	71.87%
After F-T	92.64%	92.99%	92.83%	···	92.60%	**92.65%**

TABLE 4. Comparison of ILR with and without the element-wise loss on the CIFAT-10 data set. λ was set to 0.8, for ILR with the element-wise loss. "Original VGGNet" had 13 convolutional layers and 1 fully connected layer; The "Reconstruction (*)" model shared the same architecture with the resected network by ILR but trained from scratch; "Without ew (*)" and "With ew (*)" resected 5 convolutional layers from the original VGGNet by ILR; and "Without ew (**)" and "With ew (**)" resected 8 convolutional layers from the original ResNet by ILR.

Networks	# of params	CIFAR-10 accuracy
Original VGGNet	14.73M	94.04%
Reconstruction (*)	6.32M	92.92%
Without ew (*)	6.32M	92.68%
With ew (*)	**6.32M**	**94.06%**
Original ResNet	8.8M	92.50%
Reconstruction (**)	4.2M	91.24%
Without ew (**)	4.2M	91.84%
With ew (**)	**4.2M**	**92.65%**

tion problems. Alternatively, Table 2 shows the experimental results on the CIFAR-100 and CalTech-101 data sets, both of which include about 100 classes of objects to recognize. From Table 1 and Table 2, we can see that ILR achieves higher classification accuracy than the compared approaches on all the four data sets. Particularly, the resected networks by ILR consistently ourperforms the original networks, with about half of the storage space and higher inference speed. This indicates the effectiveness of the proposed method. In addition, for VGGNet, the extension model can even achieve more than two hundreds of compression ratio.

For the resected networks, the original VGGNet has 13 convolutional layers, while the resected networks have only 8 convolutional layers for all the four object recognition tasks; The original ResNet has 17 convolutional blocks, while the resected networks have 7 convolutional blocks for the CIFAR-10 and SVHN tasks and 9 convolutional blocks for the CIFAR-100 and CalTech-101 tasks; The original ResNeXt has 17 convolutional blocks, while the resected networks have 7 convolutional blocks for the CIFAR-10 and SVHN tasks and 8 convolutional blocks for the CIFAR-100 and CalTech-101 tasks; The original SENet has 17 SE blocks, while the resected networks have 7 SE blocks for the CIFAR-10 and SVHN tasks and 9 SE blocks for the CIFAR-100 and CalTech-101 tasks. Meanwhile, considering the results shown in Table 1 and Table 2, it is easy to see that ILR can greatly compress the original deep architecture with no hurt to their performances.

Among the related approaches, KD is the closest one to ILR. In order to analyze the reason for that why ILR performs better than KD on the four object recognition tasks, we conducted the ablation study about KD and ILR. We illustrate the accuracy and loss curves of KD and ILR during the last 30 epochs of the overall learning (fine-tuning). For KD, the temperature was set to $T = 9.9$ and the ratio of soft/hard targets was set to $\xi = 0.9$ as in [15]. Fig. 5 shows the accuracy and loss curves on the CIFAR-10 data set when KD and ILR are applied on the VGGNet and ResNet, respectively. Similarly, Fig. 6 shows that on the CalTech-101 data set when KD and ILR are applied on the ResNeXt and SENet, respectively. As we can see from these groups of figures, ILR consistently performs better than KD during the training process. As a result, it outperforms KD on the test sets of the four tasks.

In the layer-level learning, we applied an element-wise loss to enforce the $(l-1)^{th}$ layer of the student network to learn similar data representations with the l^{th} layer of the teacher network. To analyze the effect of this element-wise loss, we compared ILR with and without the element-wise loss on the CIFAR-10 data set. Table 4 shows the comparison results. As we can see that, although the element-wise loss cannot be used to reduce the number of parameters for ILR, it improves the generalization ability of ILR. For both the VGGNet and ResNet, it delivers the highest classification accuracy among the compared approaches.

In the overall learning of ILR, we suggest to use a variable value for the ratio ξ between the soft and hard targets. In our experiments, for the invariable ratio ξ, we set it to 0.9 following [15], while for the variable ratio, we set it to the *tanh* function as mentioned in Sec. III-C. Table 5 shows the obtained results. As we can see, using the variable ratio, ILR outperforms all the compared approaches.

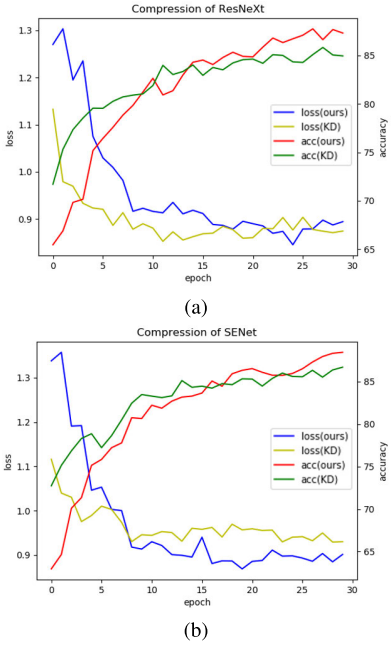


FIGURE 6. Comparison of accuracy and loss curves between knowledge distillation and ours during the last fine-tuning. (a) is the representation of ResNeXt on CALTECH-101, and (b) is the representation of SENet on CalTech-101.

TABLE 5. Comparison between using invariable and variable ratio for the soft and hard targets. ILR was applied on the VGGNet and ResNet, respectively. The CIFAR-10 data set was used.

Networks	# of params	CIFAR-10 accuracy
Original VGGNet	14.73M	94.04%
Reconstruction	6.32M	92.92%
Invariable ratio	6.32M	92.39%
Variable ratio	**6.32M**	**94.06%**
Original ResNet	8.8M	92.50%
Reconstruction	4.2M	91.24%
Invariable ratio	4.2M	92.45%
Variable ratio	**4.2M**	**92.65%**

VI. CONCLUSION AND DISCUSSION

In this paper, we introduce a novel method called ILR to compress the deep architectures, which manages to resect the inconsequential middle layers from DNNs. The motivation is based on our observations that some layers in DNNs plays a role like the identity mapping. To the end, we consider to remove the redundant layers while preserving the performance of the DNNs. We adopt a multistage learning strategy for ILR. ILR removes the inconsequential layers incrementally. After resecting one or more inconsequential layers, ILR uses the layer-level learning to recover the representation learning capability of the resected network. In order to enhance the performance of the student (resected) network, we apply the overall learning to encourage it performing similarly with the teacher (original) network. With this multistage learning strategy, ILR can effectively compress the DNNs without any hurt to their performances. Experimental results on four object recognition tasks and four deep architectures demonstrate the superiority of the proposed method.

In our experiments, we find that the compression ratio can still be improved. For deep CNNs, the convolutional units occupy the majority of computations, but have a low-density parameter distribution, whereas fully connected layers occupy the majority of storage space with relatively massive parameters. Since there are generally few fully connected layers in deep CNNs and they are very important to the classification performance, ILR rarely resects the fully connected layers. However, as indicated in previous work [7], [15], the neurons in the fully connected layers may be redundant.

In [19], a similar neurons merging algorithm has been proposed to merge the neurons with similar performance in the same layer. This algorithm can be applied to either the fully connected layers or the convolutional layers. In the future work, we plan to combine it with ILR, to compress deep architectures and obtain subnetworks thinner and shallower simultaneously.

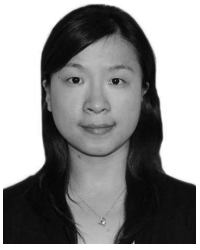
ACKNOWLEDGMENT

The Titan X GPU used for this research was donated by the NVIDIA Corporation. *Xiang Liu and Li-Na Wang contributed equally to this work.*

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, Jun. 2015, pp. 1–9.
[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[6] W. Zhou, Y. Niu, and G. Zhang, "Sensitivity-oriented layer-wise acceleration and compression for convolutional neural network," *IEEE Access*,, vol. 7, pp. 38264–38272, 2019.
[7] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: https://arxiv.org/abs/1510.00149
[8] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. NIPS*, 2015, pp. 3123–3131.
[9] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or −1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830
[10] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. ECCV*. Cham, Switzerland: Springer, 2016, pp. 525–542.
[11] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," 2016, *arXiv:1605.04711*. [Online]. Available: https://arxiv.org/abs/1605.04711
[12] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, *arXiv:1612.01064*. [Online]. Available: https://arxiv.org/abs/1612.01064
[13] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5MB model size," 2016, *arXiv:1602.07360*. [Online]. Available: https://arxiv.org/abs/1602.07360
[14] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861
[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*. [Online]. Available: https://arxiv.org/abs/1503.02531
[16] B. B. Sau and V. N. Balasubramanian, "Deep model compression: Distilling knowledge from noisy teachers," 2016, *arXiv:1610.09650*. [Online]. Available: https://arxiv.org/abs/1610.09650
[17] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. NIPS*, 2015, pp. 1135–1143.
[18] 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," in *Proc. ISCA*, Jun. 2016, pp. 243–254.
[19] G. Zhong, H. Yao, and H. Zhou, "Merging neurons for structure compression of deep networks," in *Proc. ICPR*, Aug. 2018, pp. 1462–1467.
[20] A. Polyak and L. Wolf, "Channel-level acceleration of deep face representations," *IEEE Access*, vol. 3, pp. 2163–2175, 2015.
[21] J. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. ICCV*, Oct. 2017, pp. 5058–5066.
[22] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. ICCV*, Oct. 2017, pp. 1389–1397.
[23] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," 2017, *arXiv:1702.03044*. [Online]. Available: https://arxiv.org/abs/1702.03044
[24] T. Chen, J. Goodfellow, and J. Shlens, "Net2Net: Accelerating learning via knowledge transfer," in *Proc. ICLR*, 2016, pp. 1–10.
[25] P. Yang, F. Zhang, and G. Yang, "A fast scene text detector using knowledge distillation," *IEEE Access*, vol. 7, pp. 22588–22598, 2019.
[26] R. Min, H. Lan, Z. Cao, and Z. Cui, "A gradually distilled CNN for SAR target recognition," *IEEE Access*, vol. 7, pp. 42190–42200, 2019.
[27] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "FitNets: Hints for thin deep nets," 2014, *arXiv:1412.6550*. [Online]. Available: https://arxiv.org/abs/1412.6550
[28] Z. Zhang, G. Ning, and Z. He, "Knowledge projection for deep neural networks," 2017, *arXiv:1710.09505*. [Online]. Available: https://arxiv.org/abs/1710.09505
[29] Z. Huang and N. Wang, "Like what you like: Knowledge distill via neuron selectivity transfer," 2017, *arXiv:1707.01219*. [Online]. Available: https://arxiv.org/abs/1707.01219
[30] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. CVPR*, Jul. 2017, pp. 4133–4141.
[31] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with admm," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3466–3473.
[32] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. ECCV*. Cham, Switzerland: Springer, Sep. 2018, pp. 63–79.
[33] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. CVPR*. Salt Lake City, UT, USA: IEEE, Jun. 2018, pp. 6848–6856.
[34] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. CVPR*, Jul. 2017, pp. 1251–1258.
[35] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*. [Online]. Available: https://arxiv.org/abs/1611.01578
[36] Y. Weng, T. Zhou, L. Liu, and C. Xia, "Automatic convolutional neural architecture search for image classification under different scenes," *IEEE Access*, vol. 7, pp. 38495–38506, 2019.
[37] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. NIPS*, 2014, pp. 2654–2662.
[38] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.
[39] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. CVPR*, Jul. 2017, pp. 1492–1500.
[40] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. CVPR*, Jun. 2018, pp. 7132–7141.

XIANG LIU is currently pursuing the B.S. degree from the Department of Electronic Engineering, Ocean University of China, Qingdao, China. His research interests include neural network compression and acceleration, calculation optimization method of neural networks, multiple object tracking, and underwater vision systems.



LI-NA WANG received the M.S. degree in applied mathematics from the Qingdao University of Science and Technology, Qingdao, China, in 2008. She is currently pursuing the Ph.D. degree in computer technology with the Ocean University of China, Qingdao. Her research interests include machine learning, neural networks, and computer vision.

WENXUE LIU received the B.E. degree in network engineering from Shandong Agricultural University, Taian, China, in 2017. She is currently pursuing the M.E. degree in software engineering with the Ocean University of China, Qingdao. Her research interests include neural network compression and reinforcement learning.

GUOQIANG ZHONG received the B.S. degree in mathematics from Hebei Normal University, Shijiazhuang, China, the M.S. degree in operations research and cybernetics from the Beijing University of Technology (BJUT), Beijing, China, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, in 2004, 2007, and 2011, respectively. From October 2011 to July 2013, he was a Postdoctoral Fellow with the Synchromedia Laboratory for Multimedia Communication in Telepresence, École de Technologie Supérieure (ETS), University of Montreal, Montreal, QC, Canada. Since March 2014, he has been an Associate Professor with the Department of Computer Science and Technology, Ocean University of China, Qingdao, China. He has published three books, four book chapters, and more than 60 technical articles in the areas of artificial intelligence, pattern recognition, machine learning, and data mining. His research interests include pattern recognition, machine learning, and image processing. He has served as a PC Member/Reviewer for many international conferences and top journals, such as the IEEE TNNLS, the IEEE TKDE, the IEEE TCSVT, *Pattern Recognition*, *Knowledge-Based Systems*, *Neurocomputing*, ACM TKDD, ICPR, and ICDAR. He is a member of ACM, IAPR, and a professional committee member of CAAI-PR, CAA-PRMI, and CSIG-DIAR. He has been awarded outstanding reviewer by several journals, such as *Pattern Recognition*, *Neurocomputing*, and *Cognitive Systems Research*.