IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 6, 2019, accepted November 28, 2019, date of publication December 2, 2019,
date of current version December 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2957203

Pruning Blocks for CNN Compression and Acceleration via Online Ensemble Distillation

ZONGYUE WANG[1], SHAOHUI LIN[2], (Member, IEEE), JIAO XIE[3], AND YANGBIN LIN[1]

[1]Computer Engineering College, Jimei University, Xiamen 361021, China
[2]Department of Computer Science, National University of Singapore, Singapore 117417
[3]Department of Automation, Xiamen University, Xiamen 361005, China

Corresponding author: Shaohui Lin (shaohuilin007@gmail.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFC0502902, in part by the National Natural Science Foundation of China under Grant 61701191, in part by the Key Technical Project of Fujian Province under Grant 2017H6015, in part by the Natural Science Foundation of Fujian Province under Grant 2018J05108, and in part by the Foundation of Xiamen Science and Technology Bureau under Grant 3502Z20183032.

ABSTRACT In this paper, we propose an online ensemble distillation (OED) method to automatically prune blocks/layers of a target network by transferring the knowledge from a strong teacher in an end-to-end manner. To accomplish this, we first introduce a soft mask to scale the output of each block in the target network and enforce the sparsity of the mask by sparsity regularization. Then, a strong teacher network is constructed online by replicating the same target networks and ensembling the discriminative features from each target as its new features. Cooperative learning between multiple target networks and the teacher network is further conducted in a closed-loop form, which improves their performance. To solve the optimization problem in an end-to-end manner, we employ the fast iterative shrinkage-thresholding algorithm to fast and reliably remove the redundant blocks, in which the corresponding soft masks are equal to zero. Compared to other structured pruning methods with iterative fine-tuning, the proposed OED is trained more efficiently in one training cycle. Extensive experiments demonstrate the effectiveness of OED, which can not only simultaneously compress and accelerate a variety of CNN architectures but also enhance the robustness of the pruned networks.

INDEX TERMS Fast iterative shrinkage-thresholding algorithm, model compression and acceleration, network pruning, online ensemble distillation.

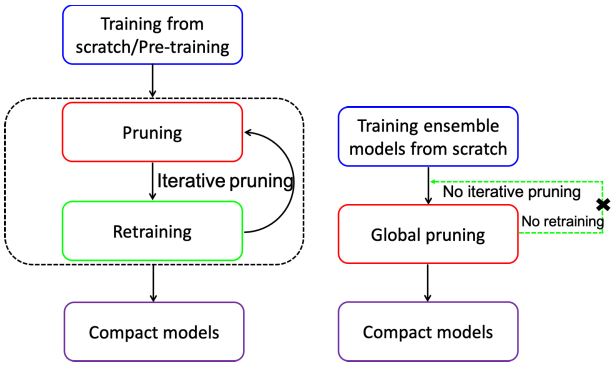
I. INTRODUCTION

In recent years, convolutional neural networks (CNNs) have achieved remarkable success in many computer vision tasks, for instance image recognition [24], [39], [70], [72], object detection [13], [65], semantic segmentation [69], *etc*. They usually have very deep and/or wide architectures with a large number of parameters [24], [77], [82]. For instance, ResNet-152 [24] classifies one color image of a resolution size 224×224, which requires a 230 MB storage cost and 11B FLOPs.[1] This restricts their usage on resource-limited devices, such as mobile phones. To address this problem, several CNN compression techniques have been proposed to achieve compact yet accurate models including network pruning [21], [44], parameter quantization [8], [9], [15], [64], and low-rank decomposition [11], [35], [36]. Thus, model compression has attracted increasing attention in both academia and industry.

Network pruning is a simple and effective method for reducing the network complexity and can be categorized as either non-structured or structured. Non-structured pruning [21] aims to prune the unimportant weight connections with small magnitudes in the pre-trained neural networks, but it may generate sparse CNNs with irregular convolutional filters that require special software/hardware accelerators for inference speedup [19], [55]. In contrast, structured pruning [27], [44], [56], [58], [75] aims to prune structured weights, including 2D kernels, filters or blocks, by which the pruned networks do not require special packages for fast inference. Therefore, it is very efficient and friendly in the deployment platform and implementation. In this paper, we focus on structured pruning, especially on pruning residual blocks with skip-connections. This is due to the residual

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

[1]FLOPs denotes the number of floating-point operations.



(a) Multi-stage pruning (b) Our one-shot pruning

FIGURE 1. Multi-stage pruning *v.s.* one-shot pruning. Multi-stage pruning produces compact networks by iteratively pruning and retraining, while one-shot pruning removes the redundant structures at once in a global way to obtain compact ones without any iterative pruning and retraining.

blocks are widely used in ResNets [24], ResNeXts [77], and compact networks (*e.g.*, MobileNet V2 [68] and ShuffleNet [59], [85]), which have strong track records for computer vision tasks [10], [23], [24] and their tolerance to the removal of blocks [31], [74]. Actually, these skip-connections make CNNs skip the computation of specific layers without cutting off the information flow and behave like ensembles of many shallow networks. Veit *et al.* [74] found that the removal of a single residual block generally has a minor impact on performance. However, the accuracy drops significantly if too many residual blocks are removed. To reduce the decrease in accuracy, learning-based structured pruning is proposed to train the networks from scratch, with sparse constraints on the weights [2], [75] or the scaling factors [31], [56], [78], using supervised class labels. It provides limited knowledge for learning the sparse networks, resulting in limited improvement of the performance. Moreover, the existing structured pruning methods typically adopt iterative pruning and retraining with multi-stage optimizations, as shown in Fig. 1. It is extremely inefficient to prune a very deep and wide network.

How can we acquire more knowledge to effectively learn the structured sparse networks in a one-shot manner? Motivated by knowledge distillation [28], a small student network can mimic a powerful teacher network by obtaining the knowledge from the teacher's soft predictions [28] or feature representations [66], [83]. However, this distillation process requires a static pre-trained teacher model and relies on multi-stage training, which are commercially unattractive with the complex training strategy [3].

To this end, we propose an *online ensemble distillation* (OED) method to automatically prune redundant structures at once in a global way without any iterative pruning and retraining, as shown in Fig. 1(b). Our key innovation lies in the online construction of a strong teacher network formed by ensembling multi-student and automatically learning the sparse structures in a one-shot manner. Fig. 2 demonstrates the workflow of the proposed method. We first introduce and initialize a soft mask randomly after each block[2] in the target (*a.k.a.* student or branch in Fig. 2) network, and then enforce the sparsity of the mask with ℓ_1-regularization. Our strong teacher network is constructed online by replicating the same target network with a different initial condition, in which the last convolutional features from each branch are concatenated to be the new features of the teacher network. Furthermore, cooperative learning between each student network and teacher network is conducted to improve their performance in a closed-loop form. Actually, it merges the training processes of the student and teacher models and uses the peer student network to provide knowledge. On the one hand, knowledge from the strong teacher (*i.e.*, soft output in the green box of Fig. 2) can be transferred to improve the accuracy of each target network. On the other hand, each improved target network can learn more discriminative features, which increases the diversity of the teacher's feature representation. Thus, it improves the accuracy of the teacher network. By forcing more scaling factors in the soft mask to be zero and pruning the target network in a one-shot manner, we can leverage the fast iterative shrinkage-thresholding algorithm (FISTA) [5], [14] to fast and reliably remove the redundant blocks.

The proposed OED is evaluated on a variety of network architectures, including ResNets [24], MobileNet V2 [68] and ResNeXts [77]. Compared to the state-of-the-art structured pruning methods [27], [31], [44], [53], [56], [58], [76], [81], [84], the proposed OED achieves a superior performance. For example, on CIFAR-10, the pruned ResNet-56 and ResNet-110 achieve increases in the classification error of only 0.68% and 0.5%, with FLOPs savings of 41.4% and 54.1% and parameters savings of 43.5% and 48.8%, respectively. On ImageNet ILSVRC 2012, the pruned ResNet-50 achieves a 9.97% Top-5 error and results in a factor of 1.9× compression. Moreover, we also evaluate the robustness of the pruned networks using OED. Compared to the original networks, the pruned networks achieve more robustness against adversarial attacks.

The rest of this paper is organized as follows. Related works are briefly summarized in Section II. The proposed online ensemble distillation method is introduced in Section III. Section IV presents the experimental results. Finally, conclusions are drawn in Section V.

II. RELATED WORK

A. NETWORK PRUNING

In early time, network pruning mainly focused on non-structured pruning, which removes weights independently to achieve a highly sparse network. For instance, the work [22], [43] proposed a saliency measurement to remove redundant weights determined by the second-derivative information of weights. Han *et al.* [20], [21] iteratively pruned the unimportant weights with small absolute values and retrained

[2]This soft mask can also be added after other structures, such as the channel/filter. For simplicity and better discussion, we only consider courser-grained pruning by removing the residual blocks.

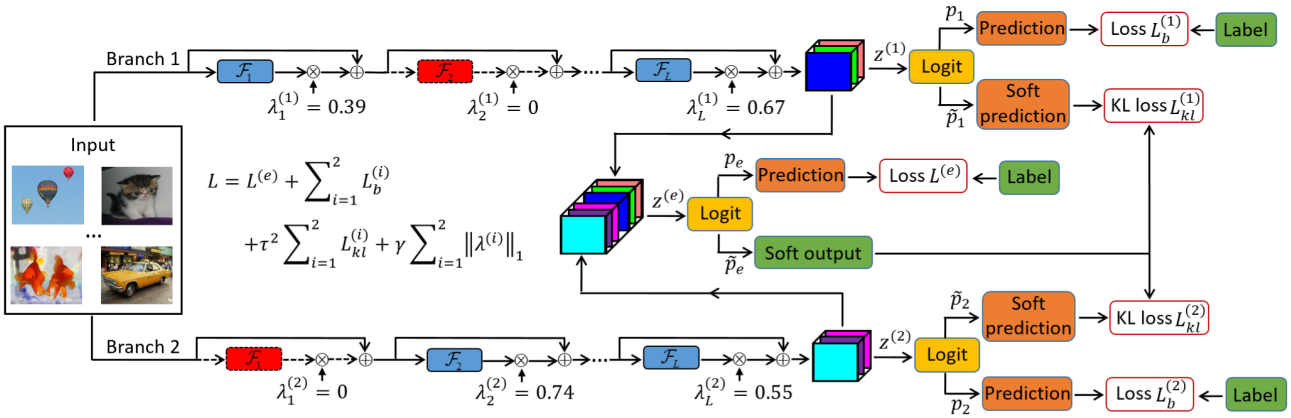


FIGURE 2. An illustration of OED. For simplicity, we describe two branches here. Each target network is a branch with the same structure but a different initial condition, in which we add a soft mask λ after each block. A strong teacher network is constructed online by concatenating the last convolutional features of all branches and adding one fully-connected layer and classifier. Furthermore, cooperative learning between each student network and teacher network is conducted in a closed-loop manner to improve their performance. On the one hand, the knowledge from the strong teacher can be distilled back to enhance each target model learning. On the other hand, each improved target network can learn more discriminative features, which increases the diversity of the teacher's feature representation. Finally, we train all target and teacher models from scratch and employ the fast iterative shrinkage-thresholding algorithm to fast and reliably remove the redundant blocks, the scaling factors of which in the soft mask are 0 (*i.e.*, red dotted block). In the test, we select a branch with the best trade-off between accuracy and parameter/FLOPs reduction for evaluation.

the sparse network. Guo *et al.* [18] proposed dynamic network surgery to reduce the model size by pruning and splicing the connections. As introduced in Section I, non-structured pruning generates irregular sparse CNNs, in which the special software/hardware accelerators must be required for fast network inference. In contrast, structured pruning is more friendly and efficient on various off-the-shelf deployment platforms, simultaneously speeding up network inference and reducing the memory overhead of CNNs. It can be further categorized into greedy-based pruning [25], [27], [30], [44], [52], [58], [60], [81], search-based pruning [17], [26], [54], [57], dynamic pruning [7], [12], [47], [63], [73], [76], and sparsity regularization-based pruning [31], [42], [45], [46], [51], [53], [56], [75], [78], [80], [84].

For greedy-based pruning, the work in [44] proposed a magnitude-based pruning to remove filters and their corresponding feature maps by measuring the ℓ_1-norm of filters in a layer-wise manner. He *et al.* [25] proposed to calculate the saliency of filters with their ℓ_2-norm. Hu *et al.* [30] proposed the sparsity of the feature map to determine the importance of a filter. Luo *et al.* [58] and He *et al.* [27] utilized statistics computed from the next layer to guide a greedy layer-wise pruning. A Taylor-expansion based criterion was proposed in [60] to iteratively prune one filter and then retrain the pruned network. Recently, a neuron importance score propagation (NISP) [81] criterion is proposed to propagate the importance score of each neuron from the high-level layers to the low-level layers. Lin *et al.* [52] proposed a global and dynamic pruning scheme to reduce the number of redundant filters by greedy alternative updating. All these greedy-based pruning methods iteratively prune each filter or layer and retrain the remaining models in a multi-stage manner, which is prohibitively costly when compressing deeper networks. Different from them, the proposed OED method globally prunes the redundant blocks at once without iterative pruning and retraining, which is significantly efficient when pruning deeper networks.

For search-based pruning, AutoML for model compression is used to automatically find the compression rate of each layer by reinforcement learning [26] or the evolutionary algorithm [57]. Motivated by MobileNet, [29] using the fixed and same width multiplier to reduce the widths of all channels, Gordon *et al.* [17] presented MorphNet to iteratively shrink the target network via a sparsity regularization and expand it via a uniform width multiplier, which is searched to satisfy the constraints on the computational complexity or the model size. Recently, Lin *et al.* [54] proposed the macroblock scaling method to find the optimal width multiplier for each convolutional layer. Compared to these search-based pruning methods with two separated steps for searching for the optimal pruning rate and the subsequent pruning, our method is much simpler to obtain a thinner network by only training the ensemble network from scratch.

Dynamic pruning reduces model inference complexity by selectively using only parts of the model conditioned on each input. Generally, an additional gate network [7], [12], [47], [63], [73], [76] is introduced to select activating structures based on each image for network inference. For example, Rao *et al.* [63] proposed a generic runtime network routing (RNR) framework to dynamically select an optimal path/branch based on each input for fast inference. The optimal path selection primarily resorts to the simple decision/gate network with the encoder-RNN-decoder structure, which can be represented as a Markov decision process and solved by reinforcement learning. Although dynamic pruning can achieve a better trade-off between accuracy and speedup,

it is difficult to reduce the model size, which restricts its usage on memory-limited devices. Different from them, our method prunes the redundant structures in a permanent manner, simultaneously speeding up the computation and reducing the memory overhead of CNNs.

In line with our work, sparsity regularization was proposed to penalize unimportant parameters and prune redundant filters. Group sparsity regularization on filters (*e.g.*, $\ell_{2,1}$-regularization [42] and $\ell_{2,0}$-regularization [51]) or multiple structures (*e.g.*, filter shapes and layers [75], combination of group and exclusive sparsity [80] and out-in-channel sparsity regularization [45]) has been proposed to sparsify them during training. The sparse scaling parameters [56], [78] in batch normalization (BN) are introduced to guide channel pruning by a threshold. However, all these methods prune redundant structures by iterative pruning and retraining, which is inefficient in offline pruning. To address these problems, scaling factors after the specific structures [31], [53] with ℓ_1-regularization are added to train by the accelerated proximal gradient algorithm, which forces the scaling factors to exact zeros. However, only the knowledge from label [31] or the output of the pre-trained model [53] is limited to improve the performance of the target model. Different from them, our method proposes a strong teacher network online without pre-training, providing more knowledge to improve the performance of the target network.

B. KNOWLEDGE DISTILLATION

The proposed online ensemble distillation is related to knowledge distillation (KD) [4], [6], [28]. Hinton *et al.* [28] transfers knowledge from a large pre-trained model to a small student network, which uses extra supervision provided by the softened final output of a pre-trained teacher. The extra supervision extracted from a pre-trained teacher model is often in the form of class posterior probabilities [28], feature representations [37], [49], [66], [83], a distribution of intermediate activations [1], or inter-layer flow [79]). These distillation methods require at least two-stage training, including pre-training the teacher network and training the student network with extra supervision, which is computationally expensive during training. Recently, deep mutual learning [87] has been proposed to conduct online distillation in one-stage training between two peer student networks. This online distillation uses each student as the opposing teacher, which is not a powerful "teacher" that limits the ability of knowledge discovery. We overcome this limitation by designing a new online ensemble distillation, in which the strong teacher network is constructed online by ensembling all the same target networks and cooperative learning between the teacher and the student model is used to improve their performance.

C. OTHER ORTHOGONAL METHODS

There are some other CNN compression methods (*e.g.*, compact network design [29], [32], [59], [68], [85], low-rank decomposition [11], [36], [41], [50], [86], [88], and parameter quantization [8], [9], [34], [64]), which are orthogonal to our method. For more details, the reader is referred to a survey [71]. The above orthogonal methods can be integrated into our approach to achieve higher compression and speedup rates.

III. PROPOSED METHOD

In this section, we first describe the notations and preliminaries. Then, we present our online ensemble distillation to prune the residual blocks. Finally, the fast iterative shrinkage-thresholding algorithm is introduced to solve the corresponding optimization problem.

A. NOTATIONS AND PRELIMINARIES

Consider a CNN model consisting of L blocks,[3] which are interlaced with ReLU [61], BN [33] and pooling. For convenience, we take ResNet-50 as an example. Each residual block contains bottleneck structures with three convolutional layers (followed by both batch normalization and ReLU) and shortcut connections. In the convolutional layer, the convolution transforms an input $\mathbf{a} \in H \times W \times N$ into an output $\mathbf{o} \in H' \times W' \times N'$ by the following linear mapping:

$$\mathbf{o}_{h',w',n'} = \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{n=1}^{N}\mathbf{K}_{i,j,n,n'}\mathbf{a}_{h_i,w_j,n}, \tag{1}$$

where $\mathbf{K} \in D \times D \times N \times N'$ represents convolutional filters. Here, $D \times D$ are the spatial dimensions, while N and N' are the numbers of input and output channels, respectively. The height and width of the input are denoted as $h_i = h' + i - 1$ and $w_j = w' + j - 1$, respectively. For simplicity, we assume a unit stride without zero-padding and skip biases. Therefore, the residual block can be formulated as:

$$\mathbf{a}_{l+1} = \mathcal{F}\big(\mathbf{a}_l, \{\mathcal{W}_l\}\big) + \mathbf{a}_l, \tag{2}$$

where $\mathbf{a}_l$ and $\mathbf{a}_{l+1}$ are the input and output of the l-th block, respectively. $\mathcal{F}$ is a residual mapping that includes three convolutional layers in Eq. (1), and $\{\mathcal{W}_l\}$ is the set of weights in the l-th block. We further give a set of training datasets $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_i^n$ with n samples, where $\mathbf{x}_i$ and y_i belong to an input and a target output to one of C classes $y_i \in \mathcal{Y} = \{1, 2, \cdots, C\}$, respectively. The network with all weights $\mathcal{W}$ outputs a probabilistic class posterior $p(c|\mathbf{x}_i, \mathcal{W})$ for a sample $\mathbf{x}_i$ over a class c as:

$$p(c|\mathbf{x}_i, \mathcal{W}) = \text{softmax}(\mathbf{z}_i) = \frac{exp(\mathbf{z}_i^c)}{\sum_{j=1}^{C} exp(\mathbf{z}_i^j)}, \tag{3}$$

where $\mathbf{z}_i$ is the logits or the final output before the "softmax" operator based on all network weights $\mathcal{W}$ and the input $\mathbf{x}_i$. To train the network $\mathcal{W}$ with multi-class classification, the cross-entropy (CE) loss between the predictions and ground-truth label distributions is typically adopted by:

$$\mathcal{L}_{CE}(\mathcal{W}) = -\frac{1}{B}\sum_{i=1}^{B}\sum_{c=1}^{C} I_{c,y_i} \log p(c|\mathbf{x}_i, \mathcal{W}), \tag{4}$$

[3] A block is equivalent to one layer for plain networks (*e.g.*, AlexNet [39] or VGGNets [70]), while a block contains multiple layers for multi-path networks (*e.g.*, ResNets [24] or GoogLeNet [72]).

where I_{c,y_i} is the indicator function, which returns 1 if c equals to the ground-truth label y_i and 0 otherwise. B is the number of mini-batches.

Generally, we solve the objective function in Eq. (4) through the stochastic gradient descent (SGD). Each block weight $\mathcal{W}_l$ can be updated by:

$$\mathcal{W}_l = \mathcal{W}_l - \eta \frac{\partial \mathcal{L}_{CE}(\mathcal{W})}{\partial \mathcal{W}_l}, \quad l = 1, 2, \cdots, L, \tag{5}$$

where the partial $\mathcal{L}_{CE}(\mathcal{W})$ with respect to $\mathcal{W}_l$ can be calculated by back-propagation and η is the learning rate.

B. ONLINE ENSEMBLE DISTILLATION FOR RESIDUAL BLOCK PRUNING

Aiming to prune residual blocks, we add the soft mask after each block to determine its importance and propose online ensemble distillation to acquire more knowledge to improve the accuracy of the pruned network. As illustrated in Fig. 2, the proposed online ensemble distillation (OED) consists of three key components:

1) SOFT MASKS IN A MULTI-BRANCH NETWORK

To construct a strong teacher, we first copy a target/branch network to form m branches. Each branch has the same configuration, which serves as an independent classification model. The soft mask is added after each block as an auxiliary parameters to determine the importance of the block. Compared to the number of network weights, the number of parameters in the soft mask is negligible, due to each block having only one parameter. Therefore, the new residual block obtained after adding the soft mask is reformulated as:

$$\mathbf{a}_{l+1}^{(i)} = \lambda_l^{(i)} \mathcal{F}\big(\mathbf{a}_l^{(i)}, \{\mathcal{W}_l^{(i)}\}\big) + \mathbf{a}_l^{(i)}, \tag{6}$$

where $\mathbf{a}_l^{(i)}$ and $\mathbf{a}_{l+1}^{(i)}$ are the input and output of the l-th block at the i-th branch, respectively. $\lambda_l^{(i)}$ is the scaling factor of the soft mask at the l-th block of the i-th branch. If $\lambda_l^{(i)} = 0$, we can reliably prune the corresponding block, as its corresponding output has no contribution to the subsequent computation.

The loss function of each branch $\mathcal{L}_{b^{(i)}}$ for a sample $\mathbf{x}_i$ over a class c can be further formulated as:

$$\begin{aligned}
&\mathcal{L}_{b^{(i)}}(\lambda^{(i)}, \mathcal{W}^{(i)}) \\
&\quad = -\frac{1}{B}\sum_{j=1}^{B}\sum_{c=1}^{C} I_{c,y_j} \log\left(\text{softmax}(\mathbf{z}_j^{(i)})\right) \\
&\quad = -\frac{1}{B}\sum_{j=1}^{B}\sum_{c=1}^{C} I_{c,y_j} \log p^{(i)}(c|\mathbf{x}_j, \lambda^{(i)}, \mathcal{W}^{(i)}),
\end{aligned} \tag{7}$$

where $\mathbf{z}_j^{(i)}$ is the logits at the i-th branch based on the input $\mathbf{x}_j$, the soft mask $\lambda^{(i)}$ and the branch weights $\mathcal{W}^{(i)}$. To learn a sparse soft mask, we employ the ℓ_1-regularization on the soft mask. Therefore, the overall loss based on all branches is reformulated as:

$$\mathcal{L}_b = \sum_{i=1}^{m} \mathcal{L}_{b^{(i)}}(\lambda^{(i)}, \mathcal{W}^{(i)}) + \gamma \sum_{i=1}^{m} \|\lambda^{(i)}\|_1, \tag{8}$$

where γ is a hyperparameter used to control the number of pruned blocks, which will be discussed in the experiments.

2) A STRONG TEACHER IS CONSTRUCTED ONLINE

As depicted in Fig. 2, we construct a strong teacher online by concatenating all the features from the last convolutional outputs of m branches followed by the BN, ReLU, global average pooling (GAP) [48] and one fully-connected (FC) layer. Therefore, the logits $\mathbf{z}^{(e)}$ in the strong network are formulated as:

$$\mathbf{z}^{(e)} = FC\left(g\left(\left[\mathbf{a}_L^{(1)}, \mathbf{a}_L^{(2)}, \cdots, \mathbf{a}_L^{(m)}\right]\right), \theta\right), \tag{9}$$

where $\mathbf{a}_L^{(i)}, i = 1, 2, \cdots, m$ is the last convolutional output of the i-th branch. $g(\cdot)$ represents a set of operators including the BN, ReLU and GAP. The construction based on a multi-branch design has two merits for model training. On the one hand, a strong teacher network is created online, which can be simultaneously trained with several student networks in a one-shot manner. On the other hand, we can avoid a complex asynchronous update between multiple networks.

After obtaining the strong teacher network, we also adopt the CE loss, $\mathcal{L}_e$, between the predictions of the teacher and the ground-truth class label distributions to improve its performance. This can be reformulated by:

$$\mathcal{L}_e = -\frac{1}{B}\sum_{j=1}^{B}\sum_{c=1}^{C} I_{c,y_j} \log\left(\text{softmax}(\mathbf{z}_j^{(e)})\right). \tag{10}$$

3) THE OVERALL LOSS FUNCTION

Given the teacher's logits on each training example, we distill the knowledge from the teacher back to all the branches. Followed by knowledge distillation [28], the soft probability distributions between the teacher and each branch are aligned to improve both the performance of the teacher and that of the student models. Therefore, we first compute the distributions with a temperature of τ for the teacher and individual branch as follows:

$$\begin{aligned}
&\hat{p}^{(e)}(c|\mathbf{x}_j, \lambda, \mathcal{W}) \\
&\quad = \text{softmax}(\mathbf{z}_j^{(e)}/\tau), c \in \mathcal{Y},
\end{aligned} \tag{11}$$

$$\begin{aligned}
&\hat{p}^{(i)}(c|\mathbf{x}_j, \lambda^{(i)}, \mathcal{W}^{(i)}) \\
&\quad = \text{softmax}(\mathbf{z}_j^{(i)}/\tau), c \in \mathcal{Y}, \quad i \in [1, \cdots, m],
\end{aligned} \tag{12}$$

where $\lambda = \cup_{i=1}^{m}\lambda^{(i)}, \mathcal{W} = \{\cup_{i=1}^{m}\mathcal{W}^{(i)}, \theta\}$ are the soft mask and weights in the teacher model, respectively. The higher the values of τ, the softer the distributions are. The detailed setups are discussed in Section IV. Then, the loss based on the Kullback Leibler divergence (KL-divergence) between the softened distributions from the teacher and branches is

Algorithm 1 FISTA to Minimize Eq. (14)

Input: Training data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_i^n$ with n samples, sparsity factor γ, temperature τ, branch number m, mini-batch size number B, learning rate η, maximum iterations T.

Output: The weights $\mathcal{W}$ and their soft mask λ.

// Training

1: Randomly initialize $\mathcal{W}$, $\lambda \sim \mathcal{N}(1, 0.1)$, and $t = 1$.
2: **repeat**
3: $t = t + 1$
4: Randomly sample a batch size of B samples from $\mathbf{x}$.
5: **Forward Pass:** Compute the predictions $p^{(e)}$ and $p^{(i)}$ with $\tau = 1$ by Eq. (11) and Eq. (12) and also the softened predictions $\hat{p}^{(e)}$ and $\hat{p}^{(i)}$ with $\tau > 1$ by Eq. (11) and Eq. (12).
6: **Backward Pass:** Compute the gradients of weights $\mathcal{W}$ and λ via back-propagation.
Update: Alternatively update $\mathcal{W}$ and λ. We first update $\mathcal{W}$ by:

$$\mathcal{W} = \mathcal{W} - \eta \frac{\partial \mathcal{L}}{\partial \mathcal{W}}. \tag{15}$$

For λ, we update it by Eq. (20).
7: **until** convergence or t reaches the maximum iterations T.

// Testing

8: Select the single model $(\mathcal{W}^{(i)}, \lambda^{(i)})$ with the best trade-off between accuracy and speedup/compression rate.

introduced to quantify their alignment, which is written as:

$$\mathcal{L}_{kl} = -\frac{1}{B}\sum_{i=1}^{m}\sum_{j=1}^{B}\sum_{c=1}^{C}\hat{p}^{(e)}(c|\mathbf{x}_j, \lambda, \mathcal{W}) \times \log\frac{\hat{p}^{(e)}(c|\mathbf{x}_j, \lambda, \mathcal{W})}{\hat{p}^{(i)}(c|\mathbf{x}_j, \lambda^{(i)}, \mathcal{W}^{(i)})}. \tag{13}$$

After we obtain $\mathcal{L}_b$, $\mathcal{L}_e$ and $\mathcal{L}_{kl}$, the overall loss function can be constructed for online distillation training by:

$$\mathcal{L} = \mathcal{L}_b + \mathcal{L}_e + \tau^2\mathcal{L}_{kl}. \tag{14}$$

Note that we multiply $\mathcal{L}_{kl}$ by a factor of τ^2 to ensure the relative contributions of the ground truth and that the probability distributions of the teacher remain roughly unchanged. The optimization achieved by minimizing the overall loss function in Eq. (14) is proposed in Section III-C.

C. OPTIMIZATION

Stochastic gradient descent (SGD) is a popular method for directly solving the optimization problem in Eq. (14). However, SGD is less efficient in convergence. Moreover, by using SGD, we have observed non-exact zero scaling factors in the soft mask λ, as they are always jittering during SGD training. We therefore need a threshold to remove the corresponding structures, the scaling factors of which are lower than the threshold. By doing so, the accuracy of the target network becomes significantly lower than the baseline, and retraining is further required to improve the accuracy. It is less efficient in pruning the deeper networks. To address this problem, we introduce FISTA [5], [14] to effectively solve the optimization problem of Eq. (14) via two alternating steps. Algorithm 1 presents the optimization process.

First, we compute the predictions $p^{(e)}$, $p^{(i)}$, $\hat{p}^{(e)}$ and $\hat{p}^{(i)}$ by forward-propagation. The gradients of weights $\mathcal{W}$ and λ are then computed by back-propagation. We then update $\mathcal{W}$ and λ alternatively. For updating, we use SGD to update the weights $\mathcal{W}$ via Eq. (15). For better illustration, to update λ, we shorten the loss with weights $\mathcal{W}$ of Eq. (14) as $\mathcal{H}(\mathcal{W}, \lambda)$, *i.e.*,

$$\mathcal{H}(\mathcal{W}, \lambda) = \sum_{i=1}^{m}\mathcal{L}_{b^{(i)}}(\lambda^{(i)}, \mathcal{W}^{(i)}) + \mathcal{L}_e(\mathcal{W}, \lambda) + \tau^2\mathcal{L}_{kl}(\mathcal{W}, \lambda). \tag{16}$$

Therefore, we have:

$$\mathcal{L}(\mathcal{W}, \lambda) = \mathcal{H}(\mathcal{W}, \lambda) + \gamma\|\lambda\|_1. \tag{17}$$

λ is then updated by FISTA with the initialization of $\alpha_{(1)} = 1$:

$$\alpha_{(t+1)} = \frac{1}{2}\left(1 + \sqrt{1 + 4\alpha_{(t)}^2}\right), \tag{18}$$

$$\mu_{(t+1)} = \lambda_{(t)} + \frac{\alpha_{(t)} - 1}{\alpha_{(t+1)}}\left(\lambda_{(t)} - \lambda_{(t-1)}\right), \tag{19}$$

$$\lambda_{(t+1)} = prox_{\eta_{(t+1)}\gamma\|\cdot\|_1}\left(\mu_{(t+1)} - \eta_{(t+1)}\frac{\partial\mathcal{H}(\mathcal{W}, \mu_{(t+1)})}{\partial\mu_{(t+1)}}\right), \tag{20}$$

where $\eta_{(t+1)}$ is the learning rate at iteration $t + 1$ and $prox_{\eta_{(t+1)}\gamma\|\cdot\|_1}(\mathbf{v}_i) = \text{sign}(\mathbf{v}_i) \circ (|\mathbf{v}_i| - \eta_{(t+1)}\gamma)_+$. In practice, we update the learning rate η with a fixed step. Details of the setup are discussed in Section IV.

IV. EXPERIMENTS

To evaluate the performance of the proposed OED method, we conduct comprehensive experiments on three widely used datasets, CIFAR-10 [38], CIFAR-100 [38] and ImageNet [67]. CIFAR-10/100 contains 50,000 training images and 10,000 test images with a size of 32 × 32 from 10/100 classes. ImageNet ILSVRC 2012 contains 1.28M training images from 1,000 classes, as well as 50,000 validation images. For ImageNet, training images are rescaled to a size of 256 × 256, with a 224 × 224 crop randomly sampled from each image and its horizontal flip. We test the model on the validation set using the single-view testing as the classification error. We also evaluate the performance on different kinds of network architectures: ResNets [24], MobileNet V2 [68] and ResNeXts [77].

Implementation: We implement the proposed OED by PyTorch [62]. We minimize the overall objective function by running on four NVIDIA GTX 1080Ti GPUs. The weight decay is set to 0.0002, and the momentum is set to 0.9. The hyperparameter γ is selected by cross-validation in the range [0.001, 0.1] for block pruning. On CIFAR-10 and

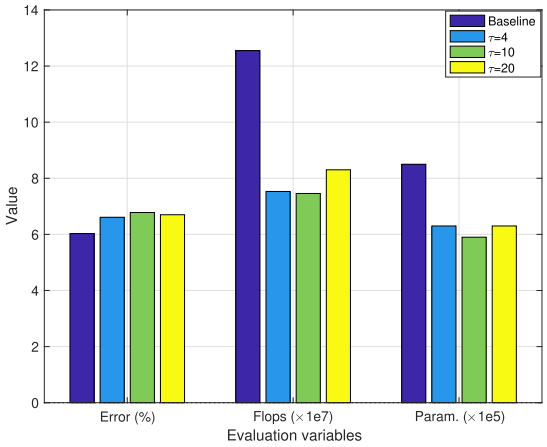


FIGURE 3. Comparison of different temperatures by OED on CIFAR-10. Baseline presents the original ResNet-56 for 300 epochs training, which achieves an error of 6.03% when the number of parameters is 8.5 × 1e5 and FLOPs is 12.549 × 1e7. $\tau = \alpha$ denotes OED training by setting temperature τ to α. Value means the different evaluation values, including the error, FLOPs and number of Parameters.

CIFAR-100, we use the same training parameters, in which the mini-batch size is set to 128 for 300 epochs and the initial learning rate is set to 0.01. The learning rate is scaled by 0.1 at 50% and 75% of the total number of training epochs. On ImageNet, the mini-batch size is set to 64 for 90 epochs. The learning rate is set to 0.01 initially, and is divided by 10 over 20 epochs. For the branch number m and temperature τ, we discuss their setups in Section IV-A.

A. ABLATION STUDY

In this section, we select ResNet-56 for an ablation study to obtain the suitable branch number and temperature, and also evaluate the effectiveness of the FISTA optimizer. We train ResNet-56 from scratch for 300 epochs as a baseline and obtain an error of 6.03% when the number of parameters is 0.85M and FLOPs is 125.49M.

1) EFFECT ON THE TEMPERATURES

To evaluate the effect of the temperatures, we first fix the branch number m and γ to 3 and 0.02, respectively. With the same training setting, we train the OED method at different temperatures: 4, 10 and 20. As shown in Fig. 3, the temperature has an effect on the performance of OED, especially on the FLOPs and parameter number. Compared to temperatures τ of 4 and 10, the number of FLOPs significantly increases when the temperature is τ 20, while its corresponding accuracy is just on par with that of the result when the temperature τ is 10. This result is due to the fact that high temperatures affect the softened distributions of the teacher and branches, providing limited knowledge that allows learning all networks in OED. In fact, OED with a temperature of 4 achieves a comparable result to that at a temperature of 10. To reduce the increase in classification error and simplify the experiments, we set the temperature to 4 in all the following experiments.

TABLE 1. Pruning results for different branch numbers. In all tables and figures, PR represents the pruning rate, T/S refers to the model belonging to the teacher/student and M/B means million/billion.

Model	T/S	Error %	FLOPs(PR)	#Param.(PR)
ResNet-56	-	6.03	125.49M(0%)	0.85M(0%)
$m = 1$	S	7.50	68.86M(45.1%)	0.59M(30.6%)
$m = 2$	S	7.33	59.43M(52.6%)	0.56M(34.1%)
	S	7.43	68.86M(45.1%)	0.60M(29.4%)
	T	6.04	-	-
$m = 3$	S	7.25	68.86M(45.1%)	0.60M(29.4%)
	S	7.33	59.42M(52.7%)	0.45M(47.1%)
	S	8.15	46.45M(63.0%)	0.48M(43.5%)
	T	6.01	-	-
$m = 4$	S	7.11	54.71M(56.4%)	0.43M(49.4%)
	S	7.29	68.86M(45.1%)	0.48M(43.5%)
	S	7.38	59.43M(52.6%)	0.40M(52.9%)
	S	7.74	49.99M(60.2%)	0.48M(43.5%)
	T	5.67	-	-

2) EFFECT ON THE BRANCH NUMBER

To evaluate the effect of the branch number, we fix γ to 0.03 and train OED with 4 groups of branch numbers: 1, 2, 3 and 4. The teacher model is the same as the student model when the branch number is set to 1. In other words, we prune the network without knowledge distillation. As shown in Table 1, with the increase in branch number, the error of the teacher model gradually decreases, becoming lower than the baseline (see the model where m is set to 3 or 4). We can also see that OED achieves better student models as the number of branches increases. Thus, OED achieves the best performance when the branch number is set to 4. For example, compared to the result when m is set to 1, 2 and 3, the model with m set to 4 achieves the highest number of FLOPs and parameter pruning rates of 56.4% and 49.4% and the lowest error of 7.11%. As discussed above, do these results indicate that we can increase the number of branches endlessly to pursue the higher performance? Actually, increasing the number of branches will slow down the training speed and increase the GPU RAM overhead. With the limited number of GPUs in our implementation platform, it is impossible to increase the number of branches endlessly. To obtain the best trade-off between training speed and performance, we set the branch number to 4 in the following experiments, which also matches the number of GPUs in our heterogeneous computing platform.

3) OED vs. DARK KNOWLEDGE AND ATTENTION TRANSFER

Dark knowledge (DK) [28] and attention transfer (AT) [83] use class posterior probabilities and feature representation of the hidden layers from a pre-trained teacher model to train the student model, respectively. The soft mask is also added after each residual block of the target network in both DK and AT. Therefore, we can employ FISTA to prune the redundant blocks at once. For a fair comparison, we choose ResNet-56 as their teachers. Moreover, we train OED, DK and AT with the same training settings, including temperature $\tau = 4$ and hyperparameter $\gamma = 0.04$. As shown in Table 2, our OED achieves the best performance compared to DK and AT.

TABLE 2. Comparison of different knowledge distillation methods on CIFAR-10. DK*/AT* is the result of method DK/AT based on our implementation, which uses ResNet-56 as its teacher, while ResNet-110-DK*/ResNet-110-AT* is the result of method DK/AT based on our implementation where the pruned network structure of the OED method is regarded as the student and ResNet-110 is regarded as the teacher.

Method	Error %	FLOPs(PR)	#Param.(PR)
ResNet-56	6.03	125.49M(0%)	0.85M(0%)
DK* [28]	7.49	87.74M(30.1%)	0.55M(35.3%)
AT* [83]	6.73	87.74M(30.1%)	0.53M(37.7%)
ResNet-110-DK* [28]	6.95	73.58M(41.4%)	0.48M(43.5%)
ResNet-110-AT* [83]	6.62	73.58M(41.4%)	0.48M(43.5%)
OED	6.71	73.58M(41.4%)	0.48M(43.5%)

TABLE 3. Results of the different optimizers. PN/PN-FT denotes the pruned networks without/with fine-tuning.

Model	Error/PN/PN-FT %	FLOPs(PR)	#Param.(PR)
ResNet-56	6.03/-/-	125.49M(0%)	0.85M(0%)
SGD	7.74/85.37/7.86	55.45M(55.81%)	0.53M(37.6%)
FISTA	7.11/7.11/-	54.71M(56.4%)	0.43M(49.4%)

This outcome is due to the cooperative learning between the teacher and individual student models in OED, which is used to improve their performances. Furthermore, we also evaluate the performance of DK and AT by distilling the knowledge from the deeper pre-trained ResNet-110 to the pruned network structure of OED. As shown in Table 2, AT achieves a slightly lower error than that of OED by pre-training ResNet-110 for 300 epochs and distilling the knowledge back to the student network for 300 epochs. However, it requires a significantly large number of training epochs with multi-stage training. In contrast, we can train both the teacher and student network by cooperative learning in a one-shot manner, which significantly reduces the number of training epochs. Compared to DK, OED achieves a better performance with fewer training epochs.

4) FISTA vs. SGD

To evaluate the effectiveness of the FISTA optimizer in OED, we compare it with the SGD optimizer. As discussed in Section III-B, we cannot obtain the soft mask with an exact scaling factor of 0 by SGD. Therefore, we introduce a threshold to guide the pruning after stable training. We set the threshold to 0.001 in our experiments. As shown in Table 3, the accuracy drops significantly after pruning with thresholding by SGD (see the Error and PN columns), as the small near-zero scaling factors in the soft mask might have a large impact on the final network output. After fine-tuning the pruned network, the accuracy can be improved substantially. Advantageously, our FISTA can safely remove the redundant residual blocks during training and achieves better performance without any fine-tuning than that of SGD.

B. COMPARISON WITH THE STATE-OF-THE-ART

1) CIFAR-10

We evaluate the performance of the proposed OED on CIFAR-10 in four widely used networks: ResNet-56 [24], ResNet-110 [24], ResNeXt-29 [77] and MobileNet V2 [68].

TABLE 4. Results for pruning ResNet-56 and ResNet-110 on CIFAR-10. In all tables, OED$-\gamma$ refers to the model where OED is trained with the sparsity factor γ.

Model	Error %	FLOPs(PR)	#Param.(PR)
ResNet-56	6.03	125.49M(0%)	0.85M(0%)
He *et al.* [27]	8.20	62M(50.6%)	-
L1 [44]	6.94	90.9M(27.6%)	0.73M(14.1%)
NISP [81]	6.99	81M(35.5%)	0.49M(42.4%)
GAL [53]	6.62	78.30M(37.6%)	0.75M(11.8%)
	8.42	49.99M(60.2%)	0.29M(65.9%)
OED-0.03	7.11	54.71M(56.4%)	0.43M(49.4%)
OED-0.04	6.71	73.58M(41.4%)	0.48M(43.5%)
	7.71	40.55M(67.7%)	0.34M(60.0%)
ResNet-110	5.9	252.89M(0%)	1.72M(0%)
L1 [44]	6.45	213M(15.8%)	1.68M(2.3%)
	6.7	155M(38.7%)	1.16M(32.6%)
GAL [53]	6.41	205.7M(18.7%)	1.65M(4.1%)
	7.26	130.2M(48.5%)	0.95M(44.8%)
BlokDrop [76]	6.4	173M(31.6%)	-
OED-0.005	6.39	134.92M(46.7%)	1.13M(34.3%)
OED-0.01	6.4	116.05M(54.1%)	0.88M(48.8%)

TABLE 5. Results for pruning ResNeXt-29 on CIFAR-10.

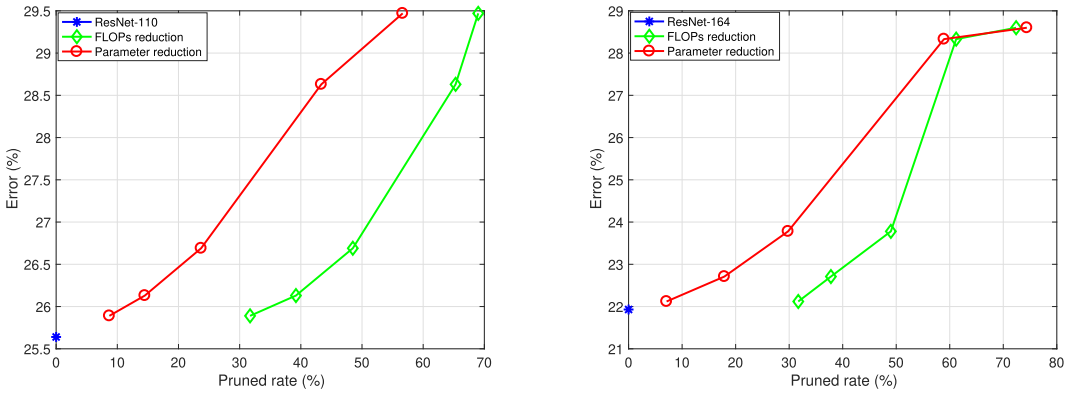
Method	Error %	FLOPs(PR)	#Param.(PR)
ResNeXt-29	3.95	5387.27M(0%)	34.37M(0%)
Edge-level [84]	4.11	2405.03M(55.4%)	24.61M(28.4%)
OED-0.05	4.08	2400.92M(55.4%)	14.25M(58.5%)

a: ResNet-56 and ResNet-110

ResNet-56 and ResNet-110 have 27 and 54 residual blocks, respectively. Each block contains two 3×3 convolutional layers. We prune the residual block in both ResNet-56 and ResNet-110, which are summarized in Table 4. For ResNet-56, when γ is set to 0.04, we obtain two of the four student models, with the best trade-off between the classification error and FLOP pruning rate. For example, compared to L1 [44] and NISP [81], either one of them with OED-0.04 achieves the lowest error of 6.71% by removing 11 residual blocks with the highest FLOP pruning rates (*i.e.*, 41.4% *vs.* 27.6% in L1 and 35.5% in NISP). Furthermore, compared to He *et al.* [27] and GAL [53], the other thinner model pruned by OED-0.04 achieves the highest FLOP pruning rate of 67.7% by removing 18 residual blocks and achieves the lowest classification error of 7.71%. For ResNet-110, BlockDrop [76] dynamically prunes the network based on each image, achieving a low error of 6.4% with a higher average FLOP savings of 31.6% compared those of L1. However, for BlockDrop, the model size is difficult to reduce, as the blocks are not pruned permanently. Compared to all baselines, OED also achieves the best result after pruning 29 residual blocks when γ is set to 0.01. Compared to L1, GAL and BlockDrop, OED-0.01 achieves an error of 6.4% with the highest savings of 54.1% and 48.8% in terms of the FLOPs and parameters, respectively.

b: ResNeXt-29

We use ResNeXt-29 (8×64d) to evaluate the performance of OED. It has 9 residual blocks with a cardinality of 8 and



(a) ResNet-110 on CIFAR-100

(b) ResNet-164 on CIFAR-100

FIGURE 4. Error *vs.* pruning rate of FLOPs and parameters by OED for ResNet-110 and ResNet-164 on CIFAR-100.

TABLE 6. Modified MobileNet V2 on CIFAR-10. We modify only the stride *s* to adapt to CIFAR-10: the other settings (*t*, *c*, *n*) are same as those in the original one on ImageNet. For a detailed introduction, refer to [68].

Input	Operator	t	c	n	s
$32^2 \times 3$	conv2d	-	32	1	1
$32^2 \times 32$	bottleneck	1	16	1	1
$32^2 \times 16$	bottleneck	6	24	2	1
$32^2 \times 24$	bottleneck	6	32	3	1
$32^2 \times 32$	bottleneck	6	64	4	2
$16^2 \times 64$	bottleneck	6	96	3	1
$16^2 \times 96$	bottleneck	6	160	3	2
$8^2 \times 160$	bottleneck	6	320	1	1
$8^2 \times 320$	conv2d 1×1	-	1280	1	1
$8^2 \times 1280$	avgpool 8×8	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1×1	-	10	-	

bottleneck width of 64. The pruning results of ResNeXt-29 are shown in Table 5. By adding edge sparsity regularization, edge-level pruning [84] achieves an increase in error of 0.16% with 55.4% and 28.4% pruning rates in terms of the FLOPs and parameters, respectively. Compared to edge-level pruning, our OED removes 5 out of 9 residual blocks, achieving a higher parameter pruning rate of 58.5% (*vs.* 28.4%), with a slightly lower classification error of 4.08% (*vs.* 4.11%).

c: MobileNet V2

We also evaluate the effectiveness of OED in pruning a compact network, MobileNet V2 [68], used in mobile or embedded devices. We simply modify the original MobileNet V2 on ImageNet to adapt to CIFAR-10 by reducing the stride, as their input image sizes are different, *i.e.,* 224 × 224 on ImageNet *vs.* 32 × 32 on CIFAR-10. The modified MobileNet V2 architecture on CIFAR-10 is summarized in Table 6. We report two selected student models pruned by training them on OED with the same γ, as shown in Table 7. Compared to GAL [53], with the same pruning rate, both of them with OED achieve a lower error of 6.3% (*vs.* 6.93%), which is also lower than that of MobileNet V2. This result

TABLE 7. Results of pruning MobileNet V2 on CIFAR-10. GAL* is the result based on our implementation.

Method	Error %	FLOPs(PR)	#Param.(PR)
MobileNet V2	6.6	296.47M(0%)	2.20M(0%)
GAL* [53]	6.93	173.73M(41.4%)	1.96M(10.7%)
OED-0.02	6.3	173.73M(41.4%)	1.96M(10.7%)
	6.65	153.57M(48.2%)	1.76M(19.9%)

indicates that there are redundancies in inverted residual blocks on MobileNet V2. Compared to MobileNet V2, the other modified one with OED achieves a negligible error increase, and higher pruning rates of 48.2% and 19.9% in terms of FLOPs and parameters, respectively. This outcome is due to the strong teacher achieved by online ensembling multiple student networks.

2) CIFAR-100

We further evaluate the performance of OED on CIFAR-100 in two popular networks: ResNet-110 [24] and ResNet-164 [24].

a: ResNet-110 and ResNet-164

Both ResNet-110 and ResNet-164 consist of 54 residual blocks. Each block in ResNet-110 has two convolutional layers, while each block in ResNet-164 has a bottleneck structure with three convolutional layers. We first evaluate the different performances for five groups of γ, as shown in Fig. 4. We can see that OED achieves a slight increase in error when the parameter pruning rate is lower than 40% on both ResNet-110 and ResNet-164. When the parameter pruning rate is higher than 40% (especially above 50%), the error significantly increases. This outcome is due to the excessive block pruning decreasing the accuracy of the teacher model, which in turn affects the learning of each student model.

We further compare OED to different state-of-the-art structured pruning methods, such as BlockDrop [76], Liu *et al.* [56] and SSS [31]. Table 8 and Table 9 summarize the comparisons of ResNet-110 and ResNet-164 on CIFAR-100,

TABLE 8. Comparison for pruning ResNet-110 on CIFAR-100.

Method	Error %	FLOPs(PR)	#Param.(PR)	#Block usage
ResNet-110	25.64	252.89M(0%)	1.73M(0%)	54
BlockDrop [76]	26.3	-	-	18
OED-0.04	29.47	78.31M(69.0%)	0.75M(56.7%)	17

TABLE 9. Comparison for pruning ResNet-164 on CIFAR-100.

Method	Error %	FLOPs(PR)	#Param.(PR)
ResNet-164	21.93	247.65M(0%)	1.68M(0%)
Liu *et al.* [56]	22.87	-(33.3%)	-(15.5%)
	23.91	-(50.6%)	-(29.7%)
SSS [31]	25.5	100M(59.6%)	1.3M(22.6%)
OED-0.005	22.71	154.08M(37.8%)	1.38M(17.9%)
OED-0.01	23.78	126.30M(49.0%)	1.18M(29.8%)

respectively. For ResNet-110, compared to BlockDrop [76], our OED use a smaller number of residual blocks (17 *vs.* 18 in BlockDrop). Although BlockDrop achieves a lower error, it prunes the blocks dynamically based on each input image without permanently removing them, which leads to a large number of blocks still being kept in offline memory storage. In contrast, OED permanently removes the redundant blocks to reduce both the computation and memory cost. For ResNet-164, compared to Liu *et al.*, our OED with setting γ to 0.005 achieves higher pruning rates both in terms of the FLOPs (37.8% *vs.* 33.3%) and the parameters (17.9% *vs.* 15.5%), and a lower error (22.71% *vs.* 22.87%). Moreover, compared to SSS with high parameter reduction, OED-0.01 achieves a lower error of 23.87% (*vs.* 25.5%) with a higher parameter pruning rate of 29.8% (*vs.* 22.6%).

3) IMAGENET ILSVRC 2012

We also evaluate OED on ImageNet using ResNet-50. The results of four branches pruned by training them on OED are presented in Fig. 5. In Fig. 5(a) and Fig. 5(b), we can see that OED learns multiple student models and a teacher model with online ensembling together, which significantly decreases the error of all networks. The teacher model however achieves almost the same error as that of ResNet-50. Note that the second branch achieves the highest error in all models, which is due to the partially important blocks being pruned during training. In Fig. 5(c), after 25 epochs, the number of blocks pruned away remains generally stable, which means that all networks aim at decreasing the classification error in the following epochs. We also found that the third branch prunes the largest number of blocks.

We further compare OED with several state-of-the-art pruning methods, as shown in Table 10. Compared to He *et al.* [27], the first branch achieves a better performance with a lower Top-1 error of 26.45% (*vs.* 27.7%) and lower computational cost (2.56B FLOPs *vs.* 2.73B FLOPs in [27]). Additionally, compared to the block pruning methods (*i.e.*, SSS [31] and GAL [53]), the third branch achieves the highest compression rate of 1.9× yet with a comparable accuracy. The third branch also achieves a comparable result to that

TABLE 10. Pruning results for ResNet-50 on ImageNet.

Model	T/S	Top-1 %	Top-5 %	FLOPs	#Param.
ResNet-50	-	23.85	7.13	4.09B	25.5M
ThiNet-50 [58]	-	28.99	9.98	1.71B	12.38M
ThiNet-30 [58]	-	31.58	11.70	1.10B	8.66M
He *et al.* [27]	-	27.70	9.20	2.73B	-
GDP-0.6 [52]	-	28.81	9.29	1.88B	-
GDP-0.5 [52]	-	30.42	9.86	1.57B	-
SSS-32 [31]	-	25.82	8.09	2.82B	18.6M
SSS-26 [31]	-	28.18	9.21	2.33B	15.6M
GAL-0.5 [53]	-	28.05	9.06	2.33B	21.2M
GAL-1 [53]	-	30.12	10.25	1.58B	14.67M
OED-0.05	S	26.45	8.80	2.56B	18.95M
	S	33.00	14.76	2.24B	15.89M
	S	28.78	9.97	2.07B	13.42M
	S	25.65	8.19	2.99B	19.52M
	T	23.88	7.27	-	-

TABLE 11. Training and testing time for ResNet-50 and OED. OED-0.05-all/b3 represents all the ensemble models or the third branch.

Model	Top-1 %	Top-5 %	Train GPU	Test	
				GPU	CPU
ResNet-50	23.85	7.13	-	138ms	9,873ms
OED-0.05-all	23.88	7.27	$\sim$ 306h	-	-
OED-0.05-b3	28.78	9.97	-	85ms	6,172ms

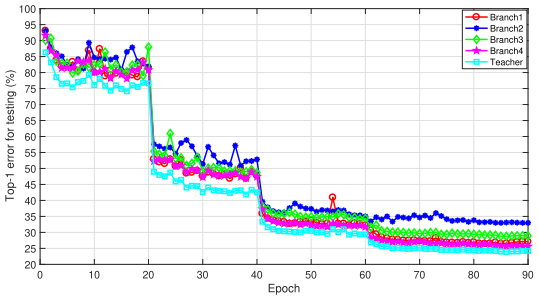
of ThiNet-50 [58] and GDP-0.6 [52]. In fact, OED is established based on simultaneous multi-branch training without iterative training and fine-tuning, which is significantly efficient against filter pruning methods with multi-stage training, such as the iteratively layer-wise pruning and retraining in ThiNet and the two-stage pruning in GDP. Furthermore, it is convenient that the corresponding branch can be selected in OED for practical applications, also including a strong teacher model.

We further measure the training and testing time of the original ResNet-50 and OED, as shown in Table 11. For training, we use four GTX 1080Ti GPUs with a batch size of 64. For testing, we use Intel i7-6900K CPU and one GTX 1080Ti GPU both with the batch sizes of 64 to measure the realistic running time of different models on CPU and GPU, respectively. We need about 306 hours to train all the ensemble models including all the branches and the teacher model. Compared to ResNet-50, the third branch by OED (*i.e.*, OED-0.05-b3) achieves 1.62× and 1.6× actual speedup in GPU and CPU for model inference. It exist the gap between theoretical and realistic model, which may come from and the limitation of IO delay, buffer switch and efficiency of BLAS libraries.

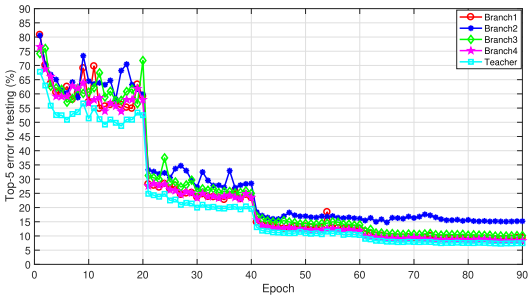
C. ROBUSTNESS AGAINST ADVERSARIAL ATTACKS

OED has demonstrated its effectiveness to simultaneously accelerate and compress CNNs on CIFAR-10/100 and ImageNet classification tasks. We further investigate the robustness of the pruned networks in resisting adversarial attacks.

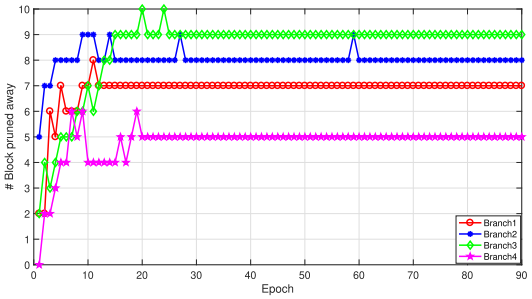
We select ResNet-56 and its model pruned by OED with γ setting to 0.04 on CIFAR-10 as the two models for evalu-



(a) Epoch *vs.* Top-1 error



(b) Epoch *vs.* Top-5 error



(c) Epoch *vs.* the number of block pruned away

FIGURE 5. Error *vs.* Top-1 error, Top-5 error and the number of blocks pruned away by OED for ResNet-50 on ImageNet.

TABLE 12. Attack results of ResNet-56 and OED-0.04 on CIFAR-10. α pixel represents the change in the number of α pixels from the initial value for each original testing image.

Method	Model	Error %										
		Ori.	1 pixel	2 pixels	3 pixels	4 pixels	5 pixels	6 pixels	7 pixels	8 pixels	9 pixels	10 pixels
FGSM [16]	ResNet-56	6.03	50.00	66.40	73.10	76.38	78.20	79.29	80.05	80.61	81.16	81.74
	OED-0.04	6.71	34.04	49.05	58.35	65.76	69.47	72.22	74.61	76.50	77.16	78.54
One-Step-LL [40]	ResNet-56	6.03	31.08	55.92	54.25	59.99	64.02	67.69	70.57	72.77	74.70	76.30
	OED-0.04	6.71	22.32	32.72	39.19	46.14	49.60	53.04	56.27	59.43	62.62	65.83
Iter-LL [40]	ResNet-56	6.03	50.00	84.83	96.14	99.79	100	-	-	-	-	-
	OED-0.04	6.71	34.04	64.91	84.11	96.81	99.12	-	-	-	-	-

ation. To generate adversarial examples, we consider three algorithms: fast gradient sign method (FGSM) [16], single-step least-likely class method (One-Step-LL) [40] and iterative attack (Iter-LL) [40]. The results are summarized in Table 12. Compared to ResNet-56, with the same change in the number of pixel in the three adversarial attacking methods, the network pruned using OED achieves a lower classification error. This result indicates that the pruned network is more robust to adversarial attacks.

V. CONCLUSION

In this paper, we developed an online ensemble distillation (OED) approach to effectively prune the blocks of CNNs without any iterative pruning and retraining. We first introduced a soft mask to scale the output of each block in the target network, upon which ℓ_1-regularization on the mask is designed to enforce its sparsity. To improve the performance of the target network, we constructed online a strong teacher network by replicating the target network to form a multi-branch network, in which each branch presents a target network. The cooperative learning in a closed loop between the teacher network and multiple target networks was then proposed to enhance their discriminability. Moreover, by forcing more scaling factors in the soft mask to zero and pruning the target network in a one-shot manner, we leveraged the fast iterative shrinkage-thresholding algorithm to fast and reliably remove the redundant blocks. We have comprehensively evaluated the performance of OED on a variety of CNN architectures over different datasets, which not only demonstrated the superior performance gains over the state-of-the-art methods but also enhanced the robustness of the pruned networks.

ACKNOWLEDGMENT

We would like to thank Prof. Linkai Luo at Xiamen University for his helpful feedback and discussions.

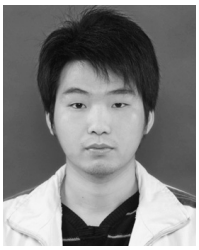
REFERENCES

[1] S. Ahn, S. X. Hu, A. Damianou, N. D. Lawrence, and Z. Dai, "Variational information distillation for knowledge transfer," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 9163–9171.

[2] J. M. Alvarez and M. Salzmann, "Learning the number of neurons in deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2270–2278.
[3] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.
[4] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.
[5] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[6] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2006, pp. 535–541.
[7] Z. Chen, Y. Li, S. Bengio, and S. Si, "You look twice: GaterNet for dynamic filter selection in CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 9172–9180.
[8] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, 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
[9] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[10] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.
[11] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.
[12] X. Gao, Y. Zhao, L. Dudziak, R. Mullins, and C.-Z. Xu, "Dynamic channel pruning: Feature boosting and suppression," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–14.
[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.
[14] T. Goldstein, C. Studer, and R. Baraniuk, "A field guide to forward-backward splitting with a FASTA implementation," 2014, *arXiv:1411.3406*. [Online]. Available: https://arxiv.org/abs/1411.3406
[15] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*. [Online]. Available: https://arxiv.org/abs/1412.6115
[16] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: https://arxiv.org/abs/1412.6572
[17] A. Gordon, E. Eban, O. Nachum, B. Chen, H. Wu, T.-J. Yang, and E. Choi, "MorphNet: Fast & simple resource-constrained structure learning of deep networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1586–1595.
[18] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1379–1387.
[19] 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. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 243–254.
[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.
[21] 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.*, 2015, pp. 1135–1143.
[22] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 164–171.
[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2980–2988.
[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[25] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2234–2240.
[26] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 784–800.
[27] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 1389–1397.
[28] 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
[29] 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
[30] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," 2016, *arXiv:1607.03250*. [Online]. Available: https://arxiv.org/abs/1607.03250
[31] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 304–320.
[32] 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< 0.5 MB model size," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[33] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167
[34] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[35] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," 2014, *arXiv:1405.3866*. [Online]. Available: https://arxiv.org/abs/1405.3866
[36] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–15.
[37] A. Koratana, D. Kang, P. Bailis, and M. Zaharia, "LIT: Learned intermediate representation training for model compression," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, pp. 3509–3518.
[38] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009 2009.
[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[40] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv preprint arXiv:1607.02533*.
[41] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CP-decomposition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–10.
[42] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2554–2564.
[43] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 598–605.
[44] 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. 1–9.
[45] J. Li, Q. Qi, J. Wang, C. Ge, Y. Li, Z. Yue, and H. Sun, "OICSR: Out-in-channel sparsity regularization for compact deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 7046–7055.
[46] L. Liang, L. Deng, Y. Zeng, X. Hu, Y. Ji, X. Ma, G. Li, and Y. Xie, "Crossbar-aware neural network pruning," *IEEE Access*, vol. 6, pp. 58324–58337, 2018.
[47] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 2181–2191.
[48] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013, *arXiv:1312.4400*. [Online]. Available: https://arxiv.org/abs/1312.4400
[49] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, "Holistic CNN compression via low-rank decomposition with knowledge transfer," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 12, pp. 2889–2905, Dec. 2019.

[50] S. Lin, R. Ji, X. Guo, and X. Li, "Towards convolutional neural networks compression via global error reconstruction," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1753–1759.
[51] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published.
[52] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, "Accelerating convolutional networks via global & dynamic filter pruning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2018, pp. 2425–2432.
[53] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2790–2799.
[54] Y.-H. Lin, C.-N. Chou, and E. Y. Chang, "MBS: Macroblock scaling for CNN model reduction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 9117–9125.
[55] B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Pensky, "Sparse convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 806–814.
[56] 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.*, Oct. 2017, pp. 2755–2763.
[57] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, T. K.-T. Cheng, and J. Sun, "Metapruning: Meta learning for automatic neural network channel pruning," 2019, *arXiv:1903.10258*. [Online]. Available: https://arxiv.org/abs/1903.10258
[58] J. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 2790–2799.
[59] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient cnn architecture design," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 116–131.
[60] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–17.
[61] V. Nair and G. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[62] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *Proc. Adv. Neural Inf. Process. Syst. Workshops*, 2017, pp. 1–4.
[63] Y. Rao, J. Lu, J. Lin, and J. Zhou, "Runtime network routing for efficient image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2291–2304, Oct. 2019.
[64] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.
[65] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[66] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[67] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[68] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[69] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 640–651, Apr. 2017.
[70] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–10.
[71] 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.
[72] 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. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[73] A. Veit and S. Belongie, "Convolutional networks with adaptive inference graphs," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 3–18.
[74] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 550–558.
[75] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.
[76] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8817–8826.
[77] S. Xie, R. Girshick, and P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5987–5995.
[78] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–11.
[79] J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 4133–4141.
[80] J. Yoon and S. J. Hwang, "Combined group and exclusive sparsity for deep neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 3958–3966.
[81] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.
[82] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: https://arxiv.org/abs/1605.07146
[83] S. Zagoruyko and N. Komodakis, "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 1–13.
[84] Q. Zhang, M. Zhang, M. Wang, W. Sui, C. Meng, J. Yang, W. Kong, X. Cui, and W. Lin, "Efficient deep learning inference based on model compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1695–1702.
[85] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2074–2082.
[86] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun, "Efficient and accurate approximations of nonlinear convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1984–1992.
[87] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4320–4328.
[88] 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.

ZONGYUE WANG received the B.E. and M.S. degrees in computer science from the Wuhan University of Technology, Wuhan, China, and the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan. He is currently an Associate Professor with the School of Computer Engineering, Jimei University, Xiamen, China. His research interests include computer vision, pattern recognition, and remote sensing data processing.



SHAOHUI LIN received the Ph.D. degree from Xiamen University, Xiamen, China, in 2019. He is currently a Research Fellow in computer science with the National University of Singapore, Singapore. He has authored about ten scientific articles at top venues, including IEEE TPAMI, TNNLS, CVPR, IJCAI, and AAAI. His research interests include machine learning and computer vision. He serves as a Reviewer for IEEE TNNLS, IJCV, TMM, and PR.

JIAO XIE received the M.S. degree from Jimei University, Fujian, China, in 2014. She is currently pursuing the Ph.D. degree with the School of Aerospace Engineering, Xiamen University. Her research interests include computer vision and deep learning.

YANGBIN LIN received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Xiamen University, Xiamen, China, in 2008, 2011, and 2016, respectively. He was a Research Assistant with Hong Kong City University, in 2010 and was also a Software Engineer with Google Company (Shanghai), from 2011 to 2012. He is currently a Lecturer with the Computer Engineering College, Jimei University, Xiamen, China. His current research interests include point cloud, graphics, and optimization.