JCST
Vol.37 No.3 May 2022

ISSN 1000-9000(Print)
/1860-4749(Online)
CODEN JCTEEM

Journal of Computer Science & Technology

SPONSORED BY INSTITUTE OF COMPUTING TECHNOLOGY
THE CHINESE ACADEMY OF SCIENCES &
CHINA COMPUTER FEDERATION
SUPPORTED BY NSFC
CO-PUBLISHED BY SCIENCE PRESS &
SPRINGER

COMPUTER

Zhao HL, Shi KJ, Jin XG *et al.* Probability-based channel pruning for depthwise separable convolutional networks. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 37(3): 584–600 May 2022. DOI 10.1007/s11390-022-2131-8

Probability-Based Channel Pruning for Depthwise Separable Convolutional Networks

Han-Li Zhao[1] (赵汉理), *Senior Member*, *CCF*, Kai-Jie Shi[1] (史开杰)
Xiao-Gang Jin[2] (金小刚), *Distinguished Member*, *CCF*, Ming-Liang Xu[3] (徐明亮), *Member*, *CCF*
Hui Huang[1] (黄　辉), *Senior Member*, *CCF*, Wang-Long Lu[1,4] (卢望龙), and Ying Liu[1] (刘　影)

[1] *College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou 325035, China*
[2] *State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310058, China*
[3] *School of Information Engineering, Zhengzhou University, Zhengzhou 450000, China*
[4] *Department of Computer Science, Memorial University of Newfoundland, St. John's, NL A1B 3X5, Canada*

E-mail: hanlizhao@wzu.edu.cn; 194511981396@stu.wzu.edu.cn; jin@cad.zju.edu.cn; iexumingliang@zzu.edu.cn
huanghui@wzu.edu.cn; wanglongl@mun.ca; 194511981400@stu.wzu.edu.cn

Received January 1, 2022; accepted May 6, 2022.

Abstract Channel pruning can reduce memory consumption and running time with least performance damage, and is one of the most important techniques in network compression. However, existing channel pruning methods mainly focus on the pruning of standard convolutional networks, and they rely intensively on time-consuming fine-tuning to achieve the performance improvement. To this end, we present a novel efficient probability-based channel pruning method for depthwise separable convolutional networks. Our method leverages a new simple yet effective probability-based channel pruning criterion by taking the scaling and shifting factors of batch normalization layers into consideration. A novel shifting factor fusion technique is further developed to improve the performance of the pruned networks without requiring extra time-consuming fine-tuning. We apply the proposed method to five representative deep learning networks, namely MobileNetV1, MobileNetV2, ShuffleNetV1, ShuffleNetV2, and GhostNet, to demonstrate the efficiency of our pruning method. Extensive experimental results and comparisons on publicly available CIFAR10, CIFAR100, and ImageNet datasets validate the feasibility of the proposed method.

Keywords network compression, channel pruning, depthwise separable convolution, batch normalization

1 Introduction

With the tremendous development of deep learning, network compression [1] has been becoming a hot research topic for a small memory footprint and low runtime latency with good performance. As one of commonly-used compression techniques, channel pruning [2, 3] compresses the network model by removing redundant structures and parameters. It boosts the development of artificial intelligence applications in our daily life, such as driverless cars, robotics and augmented reality [4].

Most channel pruning algorithms [2, 5] consist of the following three phases: pre-training, pruning, and fine-tuning. The pre-training phase produces the network model with some regularization items while the pruning phase prunes the pre-trained model by certain pruning schemes. Usually, the model after the pruning phase has a smaller size than the pre-trained model at the cost of accuracy loss. The last phase is then used to recover the accuracy by iterative parameter fine-tuning. However, the fine-tuning phase makes the whole pipeline of network pruning very time-consuming [5]. Some algorithms [3, 6] try to keep the accuracy performance

Regular Paper
Special Section of CVM 2022

This work was supported by the National Natural Science Foundation of China under Grant Nos. 62036010 and 62072340, the Zhejiang Provincial Natural Science Foundation of China under Grant Nos. LZ21F020001 and LSZ19F020001, and the Open Project Program of the State Key Laboratory of CAD&CG, Zhejiang University under Grant No. A2220.

©Institute of Computing Technology, Chinese Academy of Sciences 2022

while removing the time-consuming fine-tuning phase. However, the reduction of network parameters and FLOPs is still limited and has room for improvement.

Many applications equip with only limited computational resources and low-power batteries while requiring instant response. Recent light-weight neural networks [7–11] are built on the block of depthwise separable convolutions to achieve a balance between the resource and accuracy for mobile and embedded vision applications. The resource consumption of a pruned depthwise separable convolutional network can be further reduced and thus can be deployed in more resource-limited devices. However, most of existing network pruning algorithms are designed to prune redundant channels for standard neural convolutions [3, 5, 6]. Only a few algorithms [12] focus on depthwise separable convolutions pruning. However, they require additional time-consuming fine-tuning or retraining. We observe that depthwise convolution applies a single filter to each channel [13] and thus does not change the number of channels. This motivates us to develop an efficient channel pruning algorithm for depthwise separable convolutional networks.

To this end, we present a novel efficient probability-based channel pruning method for depthwise separable convolutional networks. Our method takes full advantage of the properties of batch normalization (BN) [14] and rectified linear unit (ReLU) [15] which are continuous in the depthwise separable convolution. If the output of BN is less than or equal to zero, ReLU will return zero. This observation gives us the intuition to determine unimportant channels in which most of BN's outputs are below zero. Consequently, we can prune these channels by developing a novel probability-based pruning criterion by considering the scaling and shifting factors of BN layers. If the output of a BN layer is less than or equal to zero with a high probability, the corresponding channel is viewed as an unimportant channel and can be pruned effectively. Since the channel number of input to depthwise convolution is identical to that of output, we propose to consider four cases based on the proposed pruning criterion to guarantee the channel consistency. In order to avoid large errors introduced in channel pruning, we further develop a sophisticated channel pruning algorithm by performing a novel shifting factor fusion technique. We test the efficiency of our new method using MobileNetV1 [7], MobileNetV2 [8], ShuffleNetV1 [9], ShuffleNetV2 [10], and GhostNet [11] networks on publicly available CIFAR10 [16], CIFAR100 [16], and ImageNet [17] datasets. The experimental results on the above representative networks show that the proposed method is able to achieve a high accuracy at low resource consumption.

In summary, our paper makes the following contributions.

- A simple yet effective probability-based channel pruning criterion is developed by considering the scaling and shifting factors of BN.
- An efficient probability-based pruning algorithm without requiring extra time-consuming fine-tuning is proposed for depthwise separable convolutional networks by using a novel shifting factor fusion technique.
- The feasibility of our method is validated through extensive experiments, and the results show that our method outperforms the state-of-the-art on performance.

2 Related Work

A number of network pruning algorithms have been carried out in the past years. In this section, we will review most related work in this topic.

Many algorithms consist of the pre-training, pruning, and fine-tuning phases for effective network pruning [3, 18–20]. The pre-training phase is used to produce clues for the pruning phase. It trains a network by adding some extra constraints, such as group-lasso [21, 22], L1 regularization [2, 3, 23] and polarization regularizer [24]. The pruning phase usually prunes weights, filters, channels or layers via various pruning criteria [25], respectively. Some methods [3, 26] prune unimportant channels with scaling factors below zero in the BN layer and other methods prune network channels by minimizing the least square reconstruction error on output feature maps [20]. Hu *et al.* [18] introduced a network trimming method by pruning unimportant channels with a high average percentage of zeros after the ReLU mapping and retrained the trimmed network to enhance the performance. Yang *et al.* [27] chose a sub-network that has a higher accuracy and lower resource consumption by removing different filters from one layer. Since the performance of the pruned network model is usually not so good as the pre-trained one, the fine-tuning phase is further required to train the pruned model [20, 21, 28, 29]. Zhang *et al.* [12] first pruned channels of the depthwise separable convolution unit based on information gain and then restored the performance with fine-tuning. However, these algorithms usually require an additional time-consuming fine-tuning or retraining

step on the pruned network in order to achieve comparable performance to the unpruned one. Recently, Liu *et al.*[5] conducted a number of experiments to indicate that the benefits gained from pruning are attributed to the architecture of the pruned network rather than the fine-tuned weights. Moreover, the pruning criterion based on a single scaling factor may prune some important channels since both the scaling and the shifting factors contribute to the BN layer. Different from them, we investigate a new pruning criterion by considering both the scaling and the shifting factors in the BN layer.

Some researchers focus on efficient network pruning algorithms without fine-tuning or retraining. The NFP algorithm[6] first prunes channels based on the scaling factor and then compensates the contribution of pruned convolutional channels to the next convolutional filters. The compensation can produce a pruned model whose accuracy is almost the same as that of the unpruned one. He *et al.*[30] dynamically determined whether a certain channel is pruned or not in a soft manner while He *et al.*[31] removed filters via geometric median which minimizes the sum of Euclidean distances. Kang and Han[32] incorporated training and soft channel pruning by introducing learnable differentiable masks. Our method is different from these methods in that our pruning criterion considers both scaling and shifting factors of BN, and no mask is required.

Our channel pruning algorithm is also related to the neural architecture search which provides a violence search method to discover the compressed model structure. A shared network[33] is trained with switchable batch normalization and can adjust the network's width on the fly instead of downloading and offloading different models. A slimmable network[34] is used to approximate the network's accuracy of different channels and is then greedily slimmed for minimal accuracy drop. He *et al.*[35] got the model compression policy by reinforcement learning while Liu *et al.*[36] searched a good-performing pruned network by the evolutionary procedure. However, these methods require much training time and GPU resources to search for efficient structures.

3 Preliminaries

Depthwise separable convolutions, initially introduced in [13], are effective to reduce the neural network's computation. As shown in Fig. 1, a standard convolution filters and combines input channels into a new set of output channels in one step while a depthwise separable convolution operation employs a depthwise convolution for filtering and a pointwise convolution for combining. The depthwise convolution uses a lightweight convolutional filter per input channel and the number of output channels is the same as the one of input channels. The pointwise convolution applies a 1×1 convolution to combine all channels produced by the depthwise convolution for building new feature channels. The depthwise separable convolution has much less computation than the standard convolution at only a small reduction in accuracy[7].

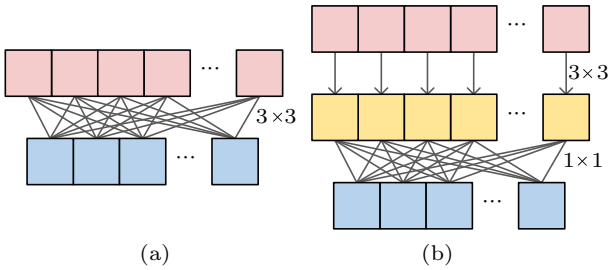


Fig.1. Illustration of (a) standard convolution and (b) depthwise separable convolution. The data are flowing from top to down.

MobileNetV1[7] and MobileNetV2[8] are representative networks which use depthwise separable convolutions as basic convolutional blocks. As illustrated in Fig.2(a) and Fig.2(b), MobileNetV1 is a single-branch network while MobileNetV2 is a multi-branch network. Both MobileNetV1 and MobileNetV2 make heavy use of BN and ReLU nonlinearity.

ShuffleNetV1[9] and ShuffleNetV2[10] employ the channel shuffle and depthwise separable convolutions to reduce computation. As illustrated in Fig. 2(c), ShuffleNetV1 inserts a channel shuffle operation between pointwise convolution (PWConv) and depthwise convolution (DWConv). As illustrated in Fig. 2(d), ShuffleNetV2 first splits the input features into two components, then concatenates them after convolutions, and finally uses the channel shuffle operation to obtain the output.

GhostNet[11] uses a novel Ghost module to generate more ghost feature maps from cheap operations. As illustrated in Fig.2(e) and Fig.2(f), GhostNet is designed by stacking Ghost bottlenecks with Ghost modules as the building block. Ghost modules make use of depthwise separable convolutions and the shortcut connection.

BN[14] is a linear transformation layer plugged between the convolutional layer and the activation function layer. It whitens and transforms each channel

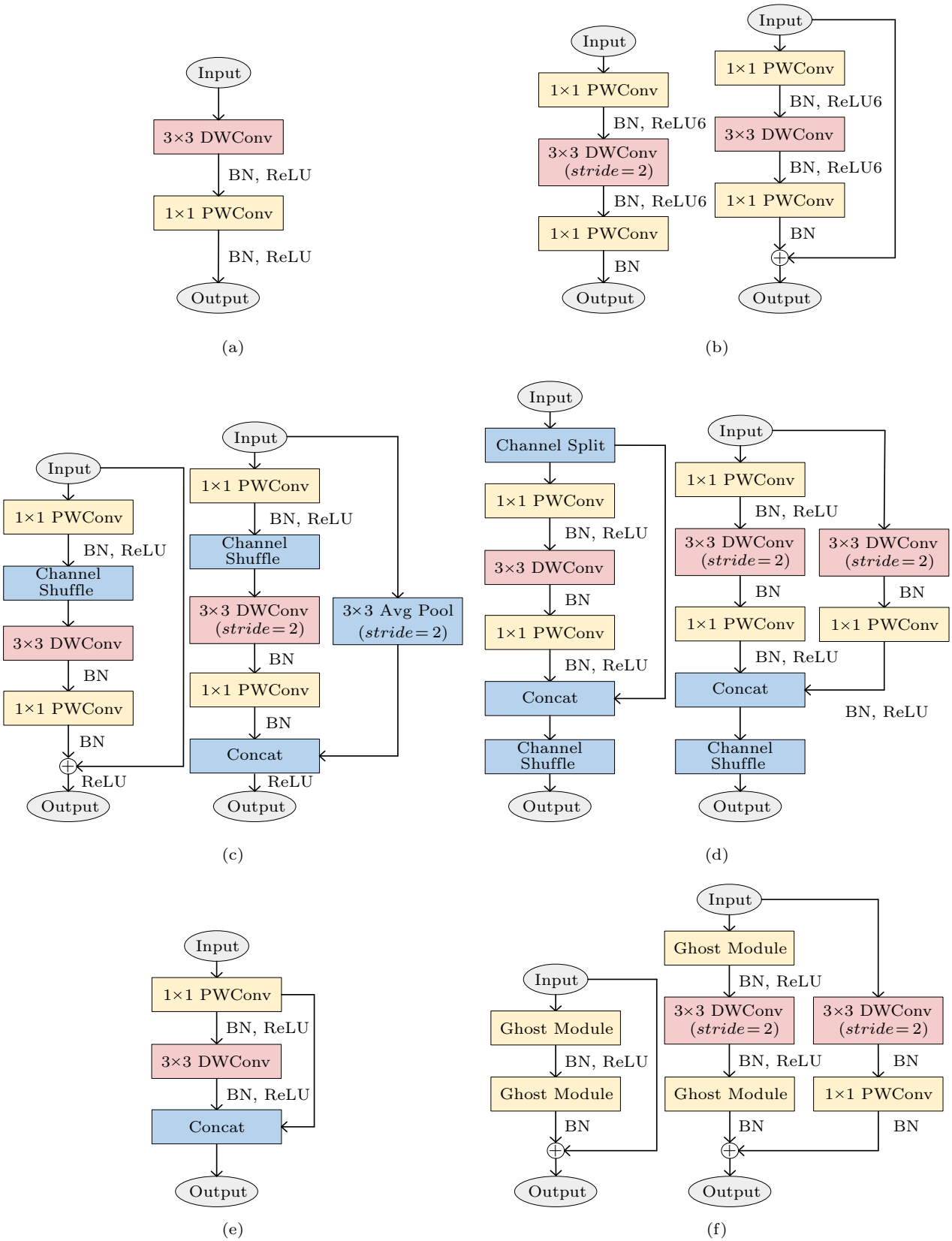


Fig.2. Illustration of structures of basic blocks in (a) MobileNetV1, (b) MobileNetV2, (c) ShuffleNetV1, (d) ShuffleNetV2, (e) Ghost module, and (f) GhostNet.

across different samples. Let x and $\hat{y}$ be an input and corresponding output of the BN layer, respectively. The operation of BN is defined by normalizing x and then performing an affine transformation as follows:

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}, \tag{1}$$

$$\hat{y} = BN(x) = \gamma \times \hat{x} + \beta, \tag{2}$$

where $\epsilon = 10^{-5}$ is a parameter avoiding division by 0, $E[x]$ and $Var[x]$ represent the mean and the variance of x respectively, and γ and β are the learnable scaling factor and shifting factor respectively. The numbers of γ and β are the same as the number of convolutional filters. Notice that the values of $E[x]$ and $Var[x]$ are calculated across mini-batches during training and are fixed during inference [14].

ReLU [15] is the rectified linear unit allowing a network to easily obtain sparse representations. Formally, ReLU is defined as:

$$y = ReLU(\hat{y}) = \begin{cases} 0, & \text{if } \hat{y} \leqslant 0, \\ \hat{y}, & \text{if } \hat{y} > 0. \end{cases} \tag{3}$$

MobileNetV2 uses a variant unit ReLU6 [8], which has the same property when $\hat{y} \leqslant 0$. For simplicity, we always use ReLU instead of ReLU6 in the rest of this paper.

From (3) we can see that the output value is 0 with the input $\hat{y} \leqslant 0$ for ReLU. Therefore, if $\hat{y} \leqslant 0$ holds for a certain channel, we can prune this channel. This observation allows us to develop an effective pruning criterion for depthwise separable convolutions by taking advantage of BN and ReLU.

4 Proposed Method

4.1 Pre-Training

To obtain a pre-trained model for pruning, one more regularization item is added to the objective function for network pruning [3, 19, 24], which is different from the normal training of the original network.

We pre-train the depthwise separable convolutional network with Kaiming initialization [37]. In addition, we use L1 regularization on scaling factors in BN as in [3]. Specifically, our pre-training objective function is defined as:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i) + \lambda L_1(\gamma_p),$$

where N denotes the number of train samples, x_i and y_i denote the train input and the target respectively, θ denotes trainable weights, the first sum term represents the normal training loss, λ is a hyperparameter, γ_p denotes the scaling factors in BN, and L_1 represents L1 regularization.

4.2 Probability-Based Pruning Criterion

If the input of the ReLU layer is less than or equal to zero for a certain channel, the channel will have no impact on the following convolutions and thus can be pruned. Many deep depthwise separable convolutional networks [7–11] employ multiple basic blocks that contain a BN layer followed by a ReLU layer. Therefore, if we know the output of a BN layer is below zero, we can prune the corresponding channel.

We assume that $\hat{x}$ in (1) follows a normal distribution by normalizing the input x from a large number of input samples [32]. Therefore, the output $\hat{y}$ of BN in (2) is normally distributed with mean β and variance γ^2, that is, $\hat{y} \sim N(\beta, \gamma^2)$. Let z be a predefined standard score of the standard normal distribution $N(0, 1)$, and then the normal distribution $N(\beta, \gamma^2)$ has the upper confidence limit:

$$Z(z) = \beta + z \times |\gamma|. \tag{4}$$

Under the assumption, the standard score z corresponds to a probability of the standard normal distribution:

$$\begin{aligned} & P(\hat{y} \leqslant Z(z)) \\ = \ & P\left(\frac{\hat{y} - \beta}{|\gamma|} \leqslant z\right) \\ = \ & P(\hat{x} \leqslant z) \\ = \ & \int_{-\infty}^{z} \frac{\mathrm{e}^{-\frac{\hat{x}^2}{2}}}{\sqrt{2\pi}} \mathrm{d}\hat{x}. \end{aligned}$$

Given a normal distribution with a high probability P, if the upper confidence limit for the output of BN is less than or equal to zero, that is, $Z = \beta + z \times |\gamma| \leqslant 0$, there is a high probability to have $\hat{y} \leqslant 0$ to be true, as illustrated in Fig. 3(a). On the other hand, if $\beta + z \times |\gamma| > 0$, then it is probably not true for $\hat{y} \leqslant 0$, as illustrated in Fig.3(b). If $\beta + z \times |\gamma| \leqslant 0$ holds, the output of ReLU is likely to be equal to zero according to the definition of the ReLU operation.

Therefore, we propose a novel probability-based pruning criterion by taking advantage of the standard score: if $\beta + z \times |\gamma| \leqslant 0$ for a BN layer, the corresponding channel is regarded as an unimportant channel and can be pruned effectively. The standard score z can

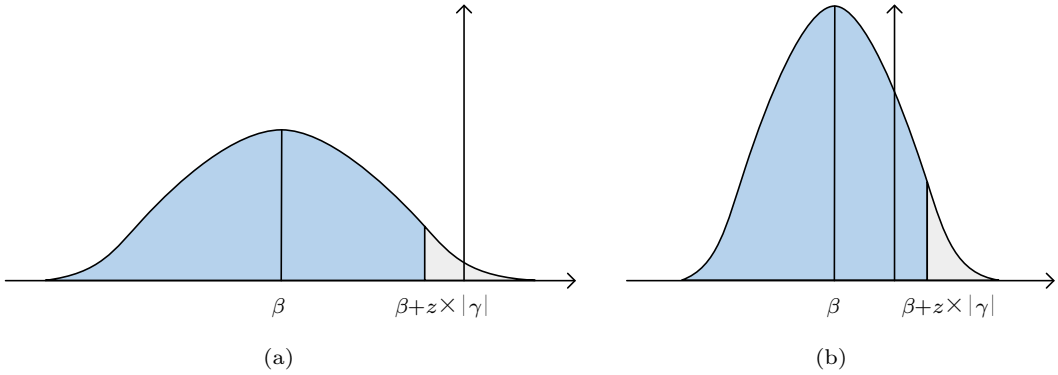


Fig.3. Illustration of normal distributions①: (a) $\beta + z \times |\gamma| \leqslant 0$ and (b) $\beta + z \times |\gamma| > 0$. Areas in blue represent the accumulative probabilities for $\beta + z \times |\gamma|$. All values below zero will be rectified to zero in the ReLU layer.

be used to adjust the probability threshold. A larger z corresponds to a higher probability and vice versa. Actually, this is true even for general distributions other than normal distributions between $\hat{x}$ and $\hat{y}$. Consequently, less channels are pruned with a larger z and more channels are pruned with a smaller z. The performance and the number of parameters of the pruned model decrease monotonically with the decrease of z. Therefore, the value of z is selected for a trade-off between the accuracy and the size.

Many existing algorithms [3, 6, 24] prune channels corresponding to the scaling factor γ in (2) below a threshold. In comparison, both the scaling factor γ and the shifting factor β are considered in our probability-based pruning criterion. We illustrate the difference between the widely used γ-based pruning criterion and the proposed pruning criterion in Fig. 4. We can find that channels with small γ but large β are also pruned in the γ-based pruning criterion. Moreover, the γ-based pruning criterion cannot prune channels with great $|\gamma|$ but $\beta + z \times |\gamma| \leqslant 0$ because the criterion views these channels as important features. In comparison, we prune channels based on the combination of both parameters (β and γ), $\beta + z \times |\gamma|$, no matter which value $|\gamma|$ is. As a result, our criterion can effectively prune channels with great $|\gamma|$ but $\beta + z \times |\gamma| \leqslant 0$.

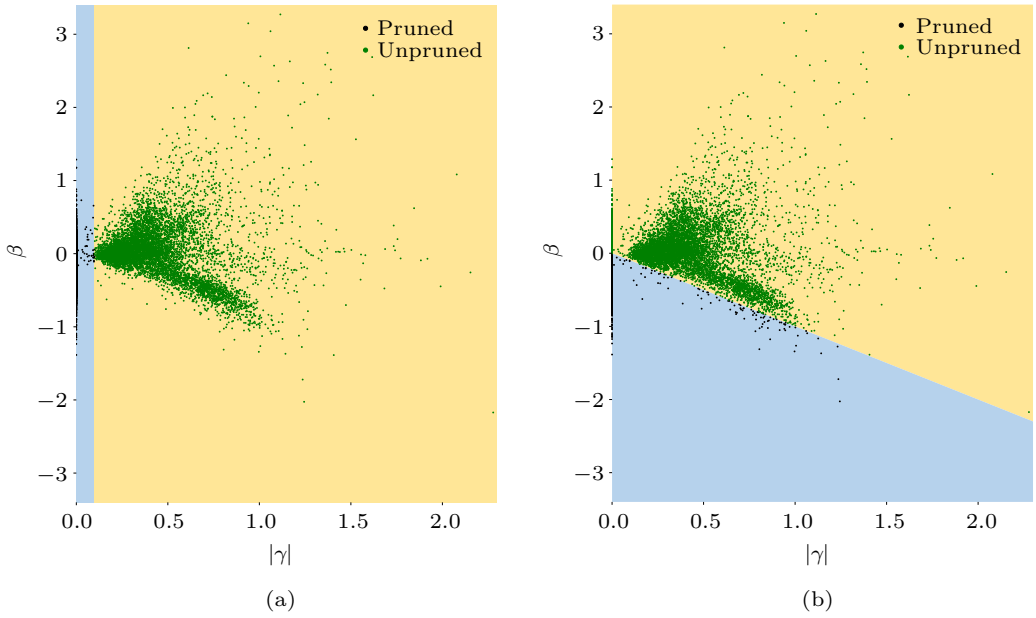


Fig.4. Illustration of the difference between (a) the γ-based criterion and (b) the proposed criterion based on both γ and β. Black points represent pruned channels while green points represent unpruned channels, respectively.

①https://www.calculator.net/z-score-calculator.html, Oct. 2021.

4.3 Channel Pruning with Shifting Factor Fusion

Depthwise separable convolutional networks are built upon multiple identical basic blocks with depthwise separable convolution. In each basic block, a depthwise convolution (DWConv) layer is followed by a pointwise convolution (PWConv) layer. As illustrated in Fig. 2, the bottleneck is the basic building block for MobileNetV2 [8], ShuffleNetV1 [9], ShuffleNetV2 [10], and GhostNet [11]. Each bottleneck block consists of two 1×1 PWConv layers and a 3×3 DWConv layer. The 1×1 layers are responsible for reducing and then increasing (restoring) the dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions [38]. In addition, BN and ReLU layers are used between two convolutional layers in each basic block. In our pruning method, channels are pruned based on BN layers before and after the DWConv layer, as illustrated in Fig.5. Therefore, we perform the channel pruning in the same manner for the simple case of using a PWConv layer and a DWConv layer and the bottleneck case of using two PWConv layers and a DWConv layer.

As illustrated in Fig. 5, there are three BN layers (denoted as the $(i-1)$-th, the i-th, and the $(i+1)$-th BN layer) before and after each DWConv layer and PWConv layer. The numbers of channels in the $(i-1)$-th and the i-th BN layer are the same since depthwise convolution applies a single filter to each channel.

In order to guarantee the channel consistency, we first introduce a naive channel pruning algorithm by considering four cases based on the proposed pruning criterion. As shown in Table 1, there are four cases for a given channel with the channel index k in DWConv between the $(i-1)$-th and the i-th BN layer.

Case 1. Both the $(i-1)$-th and the i-th BN layers do not conform to the pruning criterion, and then both BN layers and corresponding ReLU and DWConv layers for the k-th channel are unpruned, as illustrated in the topmost channel in Fig.5.

Case 2. The $(i-1)$-th BN layer does not conform to the pruning criterion but the i-th BN layer does, and then both BN layers and corresponding ReLU and DWConv layers for the k-th channel are pruned, as illustrated in the second channel in Fig.5.

Case 3. The $(i-1)$-th BN layer conforms to the pruning criterion but the i-th BN layer does not, and then both BN layers and corresponding ReLU and DWConv layers for the k-th channel are pruned, as illustrated in the third channel in Fig.5.

Case 4. Both the $(i-1)$-th and the i-th BN layers conform to the pruning criterion, and then both BN layers and corresponding ReLU and DWConv layers for the k-th channel are pruned, as illustrated in the fourth channel in Fig.5.

Although we can prune depthwise separable convolutional networks efficiently by using the proposed pruning criterion, the naive pruning algorithm will have performance penalty for case 3. Let us take the channel highlighted with the red contour in Fig.5 as an example. For case 3, the k-th channel of the $(i-1)$-th BN

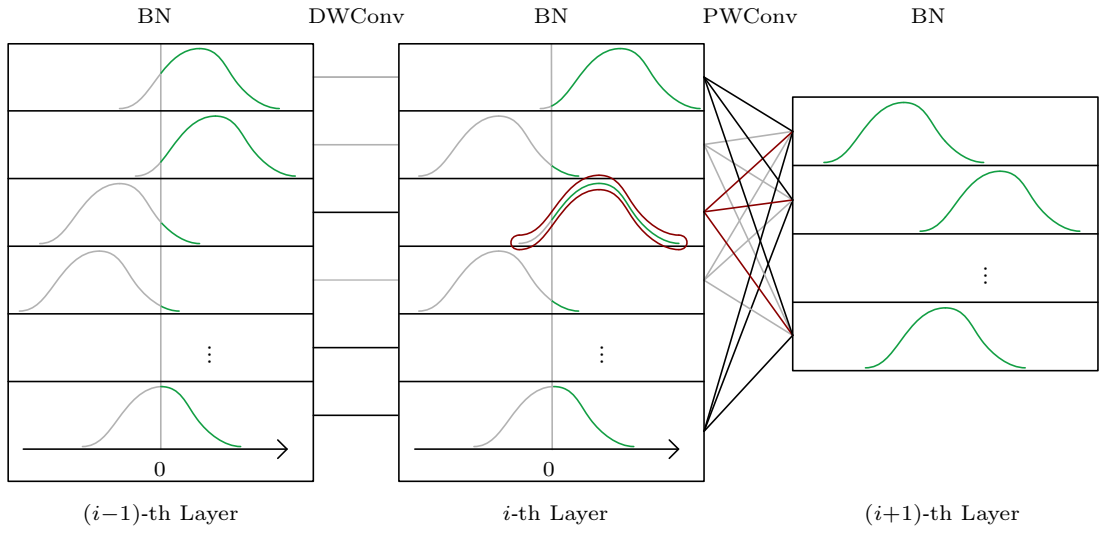


Fig.5. Illustration of the pruning process. Each curve represents a channel where light gray represents zero value outputted by ReLU. The channel highlighted with the red contour should be further processed.

is pruned according to the proposed pruning criterion and the impact of x_k^i on the k-th channel of the i-th BN is also pruned, i.e., $x_k^i = 0$. The channel numbers of the input and output for DWConv should be consistent. Therefore, we should also prune the corresponding channel for the i-th BN. For cases 2–4, the k-th channel of the i-th BN is pruned and it is equivalent to truncating $y_k^i = 0$. Based on the probability-based pruning criterion, it is probably true for case 2 and case 4 but not true for case 3. Note that the truncation error for case 3 will be accumulated with the increase of network layers. As a result, the network performance decreases with the naive pruning algorithm. This motivates us to investigate a sophisticated channel pruning algorithm to get rid of such issues.

Table 1. Channel Pruning Scheme for Different Cases in DWConv Between the $(i-1)$-th and the i-th BN Layers

Case	$(i-1)$-th BN	i-th BN	Pruning	Fusion
1	$Z^{i-1} > 0$	$Z^i > 0$	×	×
2	$Z^{i-1} > 0$	$Z^i \leqslant 0$	✓	×
3	$Z^{i-1} \leqslant 0$	$Z^i > 0$	✓	✓
4	$Z^{i-1} \leqslant 0$	$Z^i \leqslant 0$	✓	×

Note: We suppose $Z^{i-1} = \beta_k^{i-1} + z \times |\gamma|_k^{i-1}$ and $Z^i = \beta_k^i + z \times |\gamma|_k^i$ for the $(i-1)$-th and the i-th BN layers, respectively.

For case 3, we first calculate the impact t_k^i on the i-th BN according to definitions of BN and ReLU:

$$\begin{aligned} t_k^i &= ReLU(BN(x_k^i)) \\ &= ReLU\left(\gamma_k^i \times \frac{x_k^i - E[x_k^i]}{\sqrt{Var[x_k^i] + \epsilon}} + \beta_k^i\right) \\ &= ReLU\left(\gamma_k^i \times \frac{-E[x_k^i]}{\sqrt{Var[x_k^i] + \epsilon}} + \beta_k^i\right), \end{aligned}$$

where there is no ReLU after DWConv in the basic block of ShuffleNets, as illustrated in Fig.2(c) and Fig.2(d).

Let K_1 and K_3 be the index sets of channels for case 1 and case 3, respectively. Then the output x^{i+1} of the PWConv layer can be calculated as follows:

$$x^{i+1} = \sum_{k \in K_1} w_k^i y_k^i + \sum_{k \in K_3} w_k^i t_k^i,$$

where w_k^i denotes the PWConv weight. We can see that the second sum term is constant, since trainable parameters are fixed for a pre-trained network model.

The calculation of the $(i+1)$-th BN can be derived as follows:

$$\begin{aligned} & BN_{\gamma^{i+1},\beta^{i+1}}^{i+1}(x^{i+1}) \\ &= BN_{\gamma^{i+1},\beta^{i+1}}^{i+1}\left(\sum_{k \in K_1} w_k^i y_k^i + \sum_{k \in K_3} w_k^i t_k^i\right) \\ &= \gamma^{i+1} \times \frac{\sum_{k \in K_1} w_k^i y_k^i - E[x^{i+1}]}{\sqrt{Var[x^{i+1}] + \epsilon}} + \\ & \quad \gamma^{i+1} \times \sum_{k \in K_3} \frac{w_k^i t_k^i}{\sqrt{Var[x^{i+1}] + \epsilon}} + \beta^{i+1}. \end{aligned}$$

Now we have a constant $\beta_{\text{fusion}}^{i+1}$ defined as follows:

$$\beta_{\text{fusion}}^{i+1} = \gamma^{i+1} \times \sum_{k \in K_3} \frac{w_k^i t_k^i}{\sqrt{Var[x^{i+1}] + \epsilon}} + \beta^{i+1}. \quad (5)$$

Finally, we have the following updated formula by fusing the truncation error to the $(i+1)$-th BN layer:

$$BN_{\gamma^{i+1},\beta^{i+1}}^{i+1}(x^{i+1}) = BN_{\gamma^{i+1},\beta_{\text{fusion}}^{i+1}}^{i+1}(x^{i+1}).$$

Therefore, we can fuse the involved learnable parameters of pruned channels into $\beta_{\text{fusion}}^{i+1}$ for case 3 using (5) which we call it as shifting factor fusion. We propose the sophisticated channel pruning algorithm by replacing the shifting factor β^{i+1} with $\beta_{\text{fusion}}^{i+1}$ in the $(i+1)$-th BN layer for case 3. As a consequence, we can prune all corresponding layers robustly for case 3. We visualize the fusion offsets between β values before and after shifting factor fusion in Fig.6. We can see that the proposed shifting factor fusion technique effectively applies the calculated offsets to β for the $(i+1)$-th BN layer. The effectiveness of our pruning method with fusion offsets is demonstrated in Section 5 and we can see that the novel shifting factor fusion technique effectively recovers the network performance.

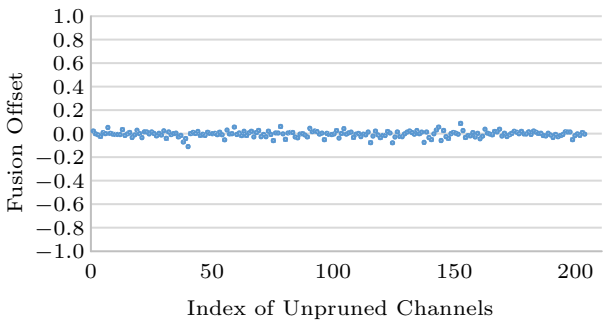


Fig.6. Visualization of the fusion offset between β values before and after shifting factor fusion. The data are sampled from the 11th BN layer of the pruned MobileNetV1 trained on CIFAR100.

4.4 Additional Processing

Residual learning [38] is widely used to ease the training of deep networks. Residual blocks with shortcut connections are also used in multi-branch depthwise separable convolution networks [8–11], and a residual block is defined as the elementwise addition [38]. Let x be the input and $F(x)$ be its residual mapping, and the dimensions of x and $F(x)$ must be equal. If some channels of $F(x)$ are pruned, the channel number of $F(x)$ is different from that of x.

Similar to [39], a normal elementwise addition is applied to unpruned channels while channels of x are used directly for pruned channels:

$$y_k = \begin{cases} x_k + F(x_k), & \text{if } k \in K_1, \\ x_k, & \text{otherwise.} \end{cases}$$

The proposed probability-based method prunes channels efficiently based on BN layers followed by ReLU. For each BN layer after the PWConv layer, if there is no ReLU layer, we add an associated gate as in [23].

4.5 Pseudocode

We provide the pseudocode of our probability-based channel pruning for depthwise separable convolution networks in Algorithm 1.

Algorithm 1. Our Probability-Based Channel Pruning

Input: pre-trained model M_0

for each basic block B in M_0 **do** ▷ Traversal
  Obtain BN^{i-1}, BN^i, BN^{i+1} in B;
  for each channel k in BN^{i-1} **do** ▷ Pruning
    $Z^{i-1} = BN^{i-1}[k].\beta + z \times BN^{i-1}[k].|\gamma|$;
    $Z^i = BN^i[k].\beta + z \times BN^i[k].|\gamma|$;
    if $Z^{i-1} > 0$ and $Z^i > 0$ **then** ▷ Case 1
      Mark k as unpruned;
    else if $Z^{i-1} > 0$ and $Z^i \leqslant 0$ **then** ▷ Case 2
      Mark k as pruned;
    else if $Z^{i-1} \leqslant 0$ and $Z^i > 0$ **then** ▷ Case 3
      Mark k as pruned;
      $K_3^i.append(k)$;
    else if $Z^{i-1} \leqslant 0$ and $Z^i \leqslant 0$ **then** ▷ Case 4
      Mark k as pruned;
for each basic block B in M_0 **do** ▷ Traversal
  Obtain BN^{i-1}, BN^i, BN^{i+1} in B;
  if $K_3^i.size() > 0$ **then**
    for each channel k in BN^{i+1} **do** ▷ Fusion
      Update $BN^{i+1}[k].\beta$ using (5);
Clone unpruned channels of M_0 to M_1;
return pruned model M_1

The proposed channel pruning algorithm takes the pre-trained model as input. We first prune unimportant channels by using the novel probability pruning criterion. Then, we perform the shifting factor fusion technique for pruned channels. The channel pruning process is implemented by creating a new channel pruned model and cloning the corresponding weights of unpruned channels from the pre-trained model. The proposed channel pruning algorithm traverses the pre-trained network twice. In the first traversal, the probability-based pruning criterion is applied to determine whether a channel should be pruned. In the second traversal, the shifting factor fusion technique is employed to recover the network performance.

The proposed algorithm is easy to implement on deep learning frameworks. In Algorithm 1, we can see that the time complexity of the proposed pruning method is determined by the network complexity and is irrelevant to input images. Note that our pruned model is able to achieve competitive performance compared with the original network without extra time-consuming fine-tuning.

5 Experimental Results and Discussions

5.1 Settings

The proposed pruning method is implemented using PyTorch. All experiments are conducted on CIFAR10 [16], CIFAR100 [16], and ImageNet [17] datasets.

The CIFAR10 and CIFAR100 datasets are with 10 and 100 classes, and consist of a training set and a testing set containing 50 000 and 10 000 images, respectively. The image size is 32×32. During training, the stochastic gradient descent optimizer is adopted for the minimization of the objective function. We train the models for 160 epochs with a mini-batch size of 64. We set the initial learning rate as 0.1 and drop it by 10x at 50% and 75% of all epochs, respectively. The hyper-parameter λ is set as 10^{-4}. As the image size is small, for MobileNetV1, we reduce the network to four down-sampling layers by replacing the first down-sampling layer with $stride = 1$. For other networks, we only use three down-sampling layers as in [34].

The ImageNet dataset is with 1 000 classes and consists of a training set and a validation set containing about 1 200 000 and 50 000 images, respectively. The image size is randomly resized and cropped to 224×224 during data augmentation. During training, the stochastic gradient descent optimizer is adopted for the minimization of objective function. We train the models for 150 epochs with a mini-batch size of 128.

We set the initial learning rate as 0.05 and use the cosine learning rate decay as in [40]. The hyperparameter λ is set as 10^{-5}. We also employ label smoothing [41] for better generalization.

5.2 Comparison with State-of-the-Art Methods

In this subsection, we compare the proposed method with state-of-the-art methods by pruning MobileNets [7, 8], ShuffleNets [9, 10], and GhostNet [11] on CIFAR10, CIFAR100, and ImageNet datasets respectively. The compared results of PFS (Pruning From Scratch) [23], AMC [35], MetaPruning [36], NetAdapt [27], and NLSP (Neuron-Level Structured Pruning) [24] are collected from the authors' papers, respectively. The results of Slimming [3] and NFP [6] are obtained with our own implementations as the authors do not provide related codes or experimental results. For slimming with fine-tuning, a learning rate of 10^{-3} and 30 epochs are adopted on CIFAR100 while a learning rate of 10^{-5} and 30 epochs are adopted on ImageNet.

Table 2 shows the comparison statistics on the CIFAR100 dataset of MobileNets. NFP improves the network slimming with no fine-tuning. Our model with fusion is much better than network slimming [3] without fine-tuning in both the accuracy and the size. Our model with fusion is also better than network slimming with fine-tuning and NFP [6]. However, fine-tuning involves additional training of model weights, which is quite time-consuming. Both network slimming and NFP prune channels based on the scaling factor of BN. In comparison, the proposed method takes into account both shifting and scaling factors of BN, resulting in a better performance in terms of the accuracy and size.

Table 3 shows the comparison statistics on the ImageNet dataset of MobileNets. Our model with fusion reduces 40% of parameters and 40% of FLOPs compared with the MobileNetV1 baseline [7] but still

Table 2. Comparison of Proposed Method with State-of-the-Art Methods for Pruning MobileNetV1 and MobileNetV2 Models on CIFAR100

Method	CIFAR100					
	MobileNetV1			MobileNetV2		
	Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Top-1 (%)	Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Top-1 (%)
Original model [7, 8]	3.31	46.47	71.09	2.32	88.10	75.26
Slimming w/o fine-tuning [3]	1.34	26.36	69.49	1.25	39.70	71.69
Slimming w fine-tuning [3]	1.34	26.36	71.25	1.25	39.70	74.83
NFP [6]	1.34	26.36	70.78	1.25	39.70	74.55
Ours w/o fusion	1.33	26.30	69.49	1.23	39.66	71.42
Ours w fusion	1.33	26.30	71.27	1.23	39.66	75.37

Note: "w" and "w/o" are short for the words "with" and "without", respectively.

Table 3. Comparison of Proposed Method with State-of-the-Art Methods for Pruning MobileNetV1 and MobileNetV2 Models on ImageNet

Method	ImageNet							
	MobileNetV1				MobileNetV2			
	Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Baseline (%)	Top-1 (%)	Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Baseline (%)	Top-1 (%)
Original model [7, 8]	4.2	569	70.6	70.6	3.5	300	71.8	71.8
AMC [35]	2.4	285	70.6	70.5	N/A	219	71.8	70.8
MetaPruning [36]	N/A	281	70.6	70.6	N/A	217	72.0	71.2
NetAdapt [27]	N/A	284	70.6	69.1	N/A	N/A	N/A	N/A
NLSP [24]	N/A	N/A	N/A	N/A	N/A	216	72.0	71.8
PFS [23]	4.0	567	70.9	71.6	3.5	300	71.8	72.1
Slimming w/o fine-tuning [3]	2.6	338	70.6	70.4	3.0	238	71.8	71.4
Slimming w fine-tuning [3]	2.6	338	70.6	70.5	3.0	238	71.8	71.6
NFP [6]	2.6	338	70.6	71.2	3.0	238	71.8	71.8
Ours w/o fusion	2.5	338	70.6	71.3	2.9	210	71.8	71.6
Ours w fusion	2.5	338	70.6	71.6	2.9	210	71.8	71.8

Note: "w" and "w/o" are short for the words "with" and "without", respectively. "N/A" represents the corresponding item is unavailable.

achieves a better top-1 accuracy. Our pruned model prunes 17.1% of parameters and 30.0% of FLOPs compared with the MobileNetV2 baseline [8]. Although our pruned model has greater FLOPs than models of AMC [35], MetaPruning [36], and NetAdapt [27] for pruning MobileNetV1 [7], the top-1 accuracy is significantly higher than theirs. The accuracy of our method is comparable with that of NLSP [24] but the FLOPs value of our method is better. PFS [23] is the only one method that has a higher accuracy than our method. Unfortunately, the reduction of the number of parameters and FLOPs is rather limited with PFS. Moreover, our pruned model with fusion has the least number of parameters and FLOPs even compared with state-of-the-art methods.

In addition, for a fair comparison, we include baselines of state-of-the-art methods collected from the original papers, respectively, as shown in Table 3. We can see that AMC [35], MetaPruning [36], NetAdapt [27], NLSP [24], and Slimming [3] drop the performance in the top-1 accuracy as compared with their respective baseline. In comparison, our method achieves an even higher accuracy for MobileNetV1. Moreover, the pruned models using our method contains less parameters and FLOPs than the ones using PFS [23] and NFP [6] with similar top-1 accuracies. It is noted that PFS [23] increases the top-1 accuracy while reducing few parameters and FLOPs.

Table 4 shows results of ShuffleNetV1, ShuffleNetV2, and GhostNet on CIFAR10, CIFAR100, and ImageNet, respectively. On CIFAR10 and CIFAR100, the pruned ShuffleNetV1 uses less parameters than the pruned ShuffleNetV2 and GhostNet, while the pruned GhostNet has less FLOPs than the pruned ShuffleNetV1 and ShuffleNetV2. The pruned ShuffleNetV2 takes a balance between parameters and FLOPs compared with our pruned ShuffleNetV1 and GhostNet. The results show that our method without fine-tuning is able to achieve better performance than Slimming [3] and NFP [6].

From the above experimental results provided in Tables 2–4, the proposed criterion can reduce more parameters and FLOPs than the γ-based criterion [3, 6] when similar top-1 accuracies are obtained. On the other hand, the proposed criterion using both γ and β achieves a higher accuracy than the γ-based criterion when similar numbers of parameters are pruned. The γ-based pruning criterion cannot prune channels with great $|\gamma|$ but $\beta + z \times |\gamma| \leqslant 0$ because the criterion views these channels as important features. On the contrary, the proposed criterion can effectively prune these channels by taking advantage of both values of γ and β. Therefore, the proposed criterion using both γ and β can effectively improve the pruning performance.

From the experimental results, we can conclude that our novel probability-based pruning method using the shifting factor fusion technique can achieve a satisfactory performance bonus compared with our method without fusion. It should be noted that the proposed novel shifting factor fusion allows a high accuracy with-

Table 4. Comparison of Proposed Method with State-of-the-Art Methods for Pruning ShuffleNetV1, ShuffleNetV2, and GhostNet Models on CIFAR10, CIFAR100, and ImageNet

Network	Method	CIFAR10			CIFAR100			ImageNet		
		Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Top-1 (%)	Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Top-1 (%)	Number of Parameters ($\times 10^6$)	FLOPs ($\times 10^6$)	Top-1 (%)
ShuffleNetV1	Original model [9]	3.50	166.49	93.17	3.67	166.68	75.32	5.4	524	73.7
	Slimming [3]	0.53	40.31	93.15	0.86	49.01	75.28	4.1	391	72.5
	NFP [6]	0.53	40.31	93.17	0.86	49.01	75.32	4.1	391	72.8
	Ours w/o fusion	0.47	36.11	93.13	0.86	48.59	75.31	4.0	387	72.6
	Ours w fusion	0.47	36.11	93.17	0.86	48.59	75.32	4.0	387	72.8
ShuffleNetV2	Original model [10]	2.47	95.17	92.55	2.56	95.27	74.59	3.5	299	72.6
	Slimming [3]	1.28	43.73	92.24	1.65	52.49	74.58	3.4	284	72.6
	NFP [6]	1.28	43.73	92.55	1.65	52.49	74.59	3.4	284	72.6
	Ours w/o fusion	1.14	37.68	92.52	1.57	51.02	74.36	3.3	280	72.5
	Ours w fusion	1.14	37.68	92.55	1.57	51.02	74.59	3.3	280	72.6
GhostNet	Original model [11]	3.63	56.87	91.18	3.99	57.18	74.95	5.2	141	73.9
	Slimming [3]	2.79	25.76	91.05	3.36	31.23	74.94	5.0	122	73.4
	NFP [6]	2.79	25.76	91.18	3.36	31.23	74.95	5.0	122	73.5
	Ours w/o fusion	2.75	22.48	89.92	3.23	26.61	74.88	4.9	120	73.4
	Ours w fusion	2.75	22.48	91.18	3.23	26.61	74.95	4.9	120	73.5

Note: “w” and “w/o” are short for the words “with” and “without”, respectively.

out requiring extra time-consuming fine-tuning. The experimental results show that our novel probability-based pruning method outperforms the state-of-the-art methods in terms of both the accuracy and the size.

5.3 Visualization of Pruned Channels

The number of channels in the pruned model closely relates to the accuracy as well as the speed. In order to demonstrate the performance of the proposed pruning method in pruning channels, we visualize the statistics of numbers of channels of BN layers after each DWConv in Fig.7.

There are four pruning cases for different BN channels in our pruning method. As shown in Fig.7, four cases are visualized with different colors. In order to preserve the channel consistency, two adjacent BN layers before and after the DWConv layer (see the $(i-1)$-th and the i-th layer in Fig.5) share the same pruning scheme in our pruning method. If one BN layer does not conform to the criterion but another BN layer does, both BN layers and corresponding ReLU and DWConv layers are effectively pruned with our pruning method. Therefore, every pair of adjacent BN layers before and after the DWConv layer have the same number of channels for each case. An impressive percent of channels conform to case 3 that can take advantage of the proposed shifting factor fusion technique. As illustrated in Fig.2(c) and Fig.2(d), there is no ReLU after DWConv in ShuffleNetV1 and ShuffleNetV2. Therefore, as shown in Fig.7, all pruned channels are classified as case 3, which means that shifting factor fusion should be applied to all pruned channels in ShuffleNetV1 and ShuffleNetV2. Channels conforming to case 1 are unpruned and thus used for the pruned model. As a result, the proposed pruning method is capable of pruning channels efficiently.

5.4 Ablation Study on Fine-Tuning

Here we perform the ablation study on fine-tuning to further validate the effectiveness of the proposed shifting factor fusion technique.

We experiment to fine-tune our pruned model from unpruned weights [3] which are trained in the pre-training phase. We also experiment to fine-tune our pruned model from scratch [5]. The performance comparison of fine-tuning from unpruned weights and fine-tuning from scratch with our pruned models is illustrated in Fig.8. We can see that the top-1 accuracy of fine-tuning from unpruned weights fluctuates over a very small range around the top-1 accuracy without any fine-tuning (i.e., result with zero epoch). Fine-tuning from scratch trains learnable weights as same as the pre-training phase except L1 regularization. Although the top-1 accuracy increases with the increase of fine-tuning epoch, the top-1 accuracy is still slightly lower than that of our pruned model without fine-tuning in 150 epochs.

As a consequence, our pruned model without any fine-tuning has competitive performance compared with the pruned model with unpruned weights and the pruned model from scratch. The experimental results demonstrate that our pruned model with the proposed shifting factor fusion technique does not require additional fine-tuning.

5.5 Ablation Study on the Standard Score

Now we perform the ablation study on the standard score parameter z to study the sensitivity of the proposed pruning method with respect to z.

We show statistics of the top-1 accuracy, FLOPs, and the number of parameters of the pruned models by varying the value of z in Fig. 9. The proposed probability-based pruning criterion is based on the standard score parameter z as defined in (4). A greater standard score corresponds to a high probability and vice versa. We can see that the top-1 accuracy, FLOPs, and the number of parameters all monotonically increase with the increase of z. The curves of these quantities rise significantly when $z < 2$, rise slowly when $z \in [2, 4]$, and become stable when $z > 4$.

Both the accuracy and the speed are important quantities for a pruned model. Therefore, we empirically choose $z \in [2, 4]$ for a trade-off between the accuracy and the size.

6 Conclusions

In this paper, we presented a novel efficient probability-based channel pruning method for depthwise separable convolutional networks. By leveraging the scaling and shifting factors of BN, a simple yet effective probability-based channel pruning criterion was proposed. A novel shifting factor fusion technique was developed to further improve the pruning performance. We validated the efficiency of the proposed channel pruning method on representative depthwise separable convolutional networks including MobileNetV1, MobileNetV2, ShuffleNetV1, ShuffleNetV2, and Ghost-

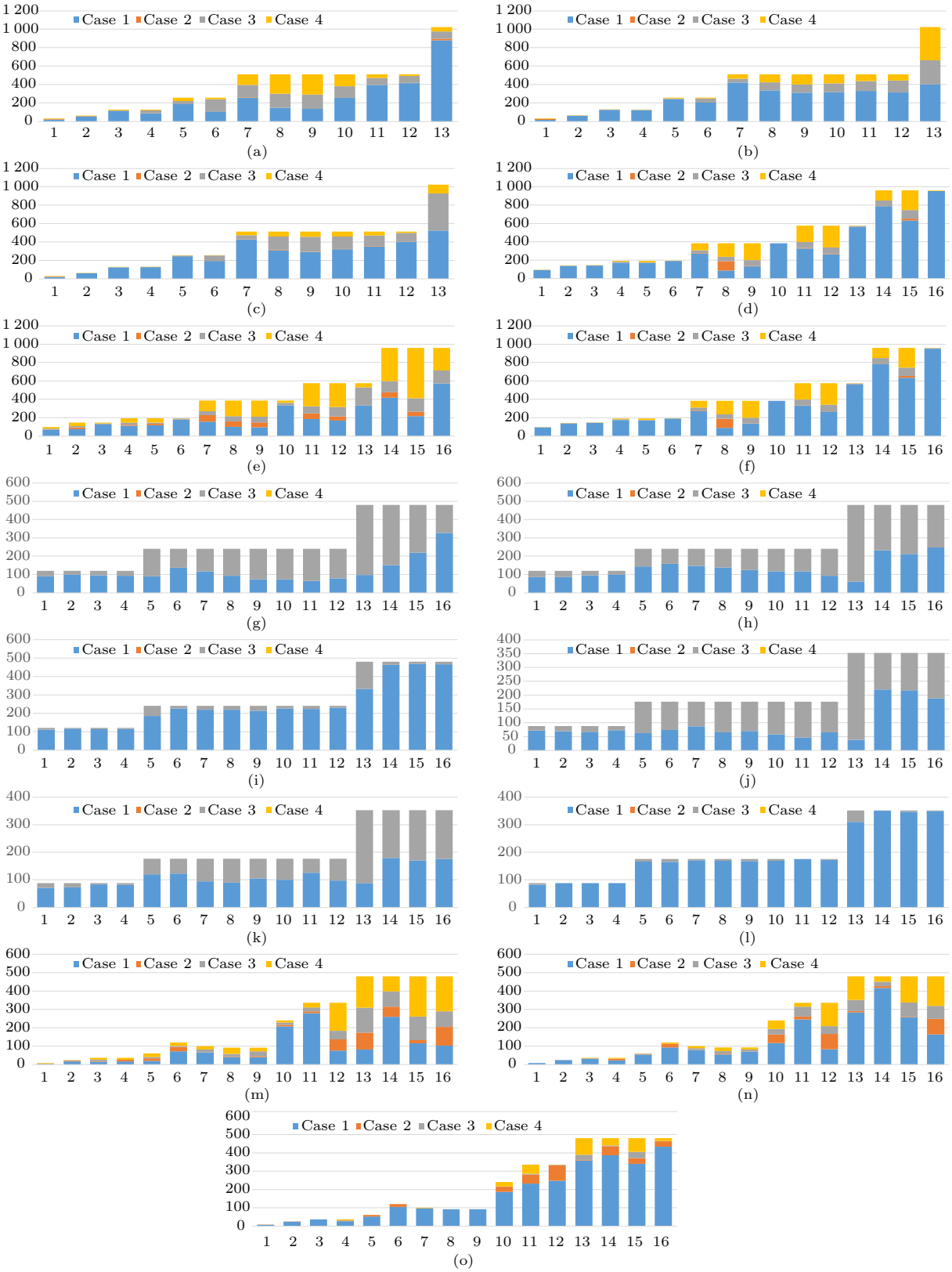


Fig. 7. Statistics of numbers of channels of BN layers after each DWConv. (a) MobileNetV1 on CIFAR10. (b) MobileNetV1 on CIFAR100. (c) MobileNetV1 on ImageNet. (d) MobileNetV2 on CIFAR10. (e) MobileNetV2 on CIFAR100. (f) MobileNetV2 on ImageNet. (g) ShuffleNetV1 on CIFAR10. (h) ShuffleNetV1 on CIFAR100. (i) ShuffleNetV1 on ImageNet. (j) ShuffleNetV2 on CIFAR10. (k) ShuffleNetV2 on CIFAR100. (l) ShuffleNetV2 on ImageNet. (m) GhostNet on CIFAR10. (n) GhostNet on CIFAR100. (o) GhostNet on ImageNet.

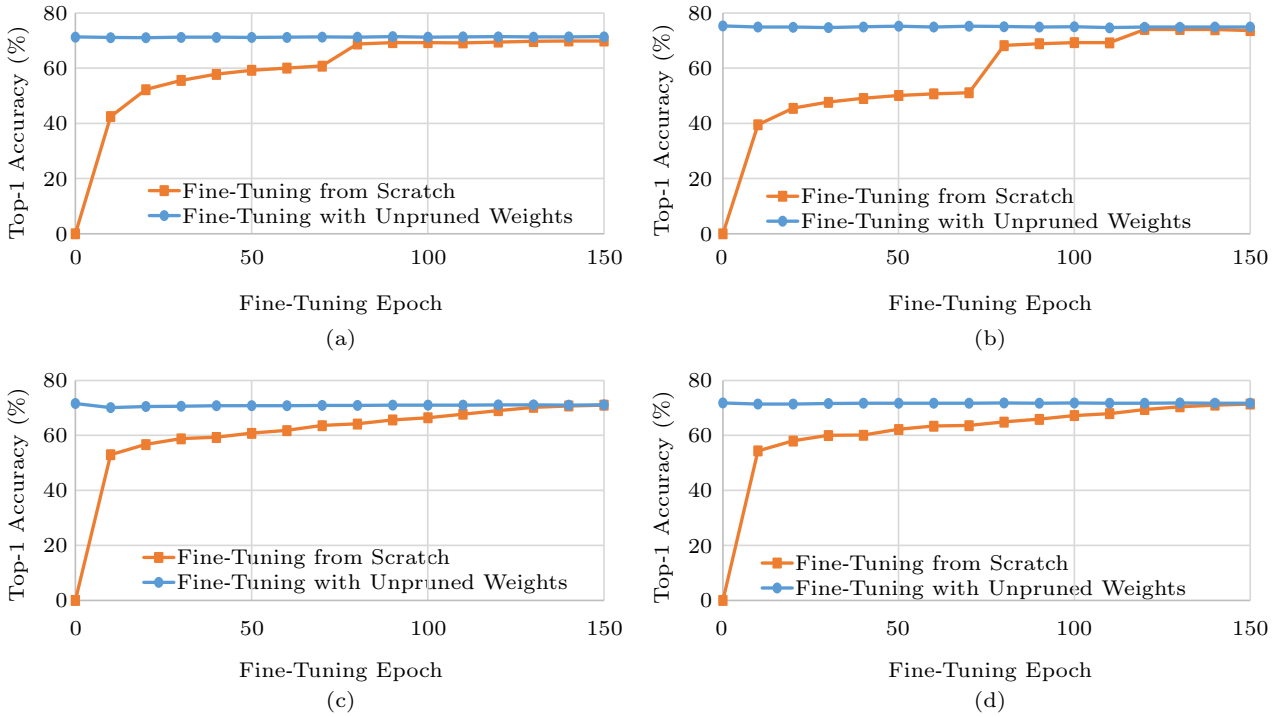


Fig.8. Performance comparison of fine-tuning from unpruned weights and fine-tuning from scratch with our pruned models. (a) Pruned MobileNetV1 on CIFAR100. (b) Pruned MobileNetV2 on CIFAR100. (c) Pruned MobileNetV1 on ImageNet. (d) Pruned MobileNetV2 on ImageNet.

Net. Promising experimental results and comparisons showed the feasibility of the proposed method.

In the future, we would like to apply the proposed pruning method to more depthwise separable convolutional networks. Moreover, we plan to use our pruned model as the backbone for other computer vision tasks, such as vehicle logo recognition [42] and medical image segmentation [43].

References

[1] Cheng Y, Wang D, Zhou P, Zhang T. A survey of model compression and acceleration for deep neural networks. arXiv:1710.09282, 2017. https://arxiv.org/abs/ 1710.09282, Jun. 2021.

[2] Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural network. In *Proc. the 28th International Conference on Neural Information Processing Systems*, Dec. 2015, pp.1135-1143.

[3] Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C. Learning efficient convolutional networks through network slimming. In *Proc. the 2017 IEEE International Conference on Computer Vision*, Oct. 2017, pp.2736-2744. DOI: 10.1109/ICCV.2017.298.

[4] West D M. The Future of Work: Robots, AI, and Automation. Brookings Institution Press, 2018.

[5] Liu Z, Sun M, Zhou T, Huang G, Darrell T. Rethinking the value of network pruning. arXiv:1810.05270, 2018. https://arxiv.org/abs/1810.05270, Mar. 2021.

[6] Liu R, Cao J, Li P, Sun W, Zhang Y, Wang Y. NFP: A no fine-tuning pruning approach for convolutional neural network compression. In *Proc. the 3rd International Conference on Artificial Intelligence and Big Data*, May 2020, pp.74-77. DOI: 10.1109/ICAIBD49809.2020.9137429.

[7] Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861, 2017. https://arxiv.org/abs/ 1704.04861, Apr. 2021.

[8] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp.4510-4520. DOI: 10.1109/CVPR.2018.00474.

[9] Zhang X, Zhou X, Lin M, Sun J. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *Proc. the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp.6848-6856. DOI: 10.1109/CVPR.2018.00716.

[10] Ma N, Zhang X, Zheng H, Sun J. ShuffleNetV2: Practical guidelines for efficient CNN architecture design. In *Proc. the 15th European Conference on Computer Vision*, Sept. 2018, pp.116-131. DOI: 10.1007/978-3-030-01264-9_8.

[11] Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C. GhostNet: More features from cheap operations. In *Proc. the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp.1580-1589. DOI: 10.1109/CVPR42600.2020.00165.

[12] Zhang K, Cheng K, Li J, Peng Y. A channel pruning algorithm based on depth-wise separable convolution unit.

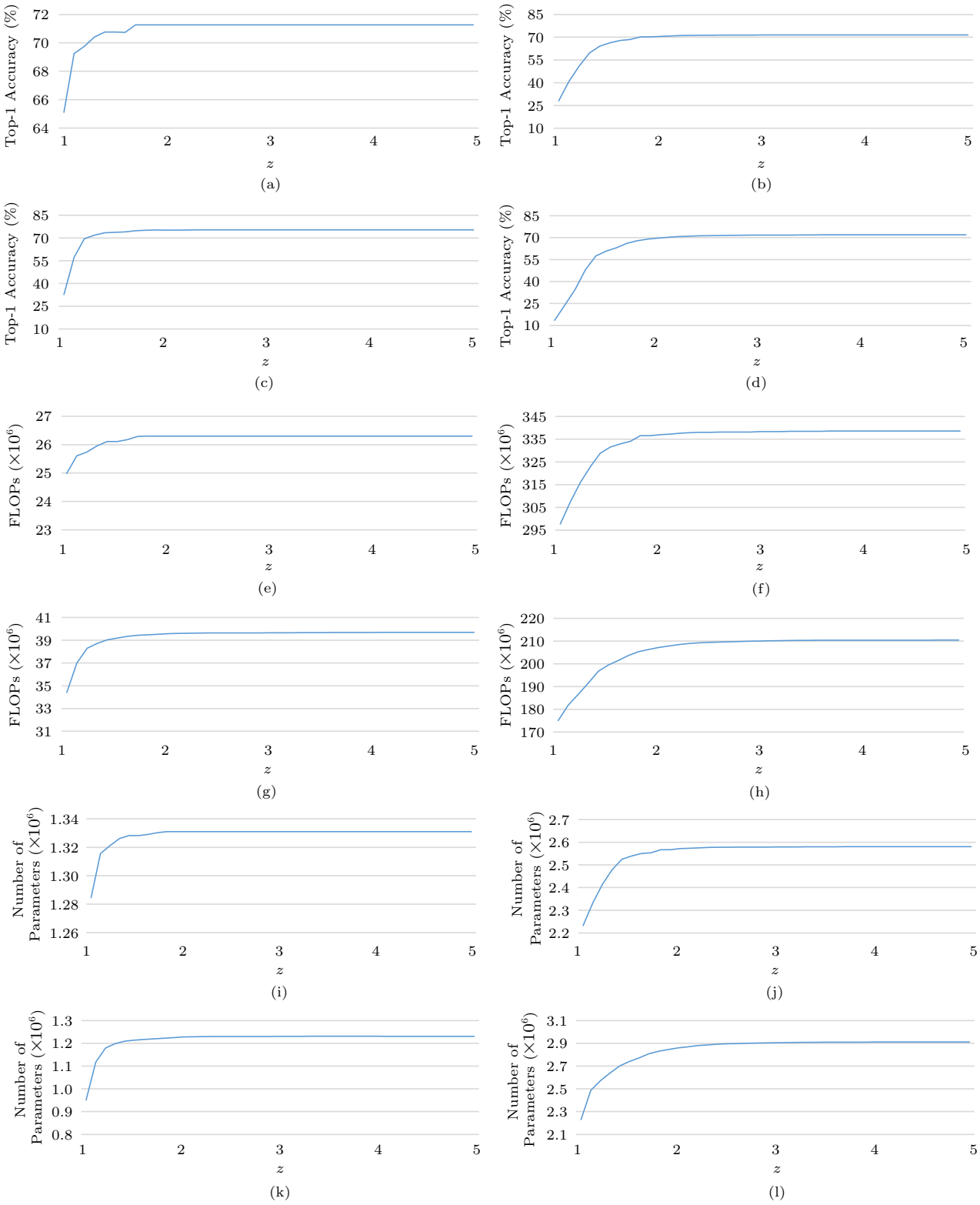


Fig.9. Statistics of the top-1 accuracy, FLOPs and the number of parameters by varying the value of the standard score parameter z. (a) Top-1 accuracy of pruned MobileNetV1 on CIFAR100. (b) Top-1 accuracy of pruned MobileNetV1 on ImageNet. (c) Top-1 accuracy of pruned MobileNetV2 on CIFAR100. (d) Top-1 accuracy of pruned MobileNetV2 on ImageNet. (e) FLOPs of pruned MobileNetV1 on CIFAR100. (f) FLOPs of pruned MobileNetV1 on ImageNet. (g) FLOPs of pruned MobileNetV2 on CIFAR100. (h) FLOPs of pruned MobileNetV2 on ImageNet. (i) Number of parameters of pruned MobileNetV1 on CIFAR100. (j) Number of parameters of pruned MobileNetV1 on ImageNet. (k) Number of parameters of pruned MobileNetV2 on CIFAR100. (l) Number of parameters of pruned MobileNetV2 on ImageNet.

IEEE Access, 2019, 7: 173294-173309. DOI: 10.1109/ACCESS.2019.2956976.

[13] Sifre L, Mallat S. Rigid-motion scattering for texture classification. arXiv:1403.1687, 2014. https://arxiv.org/abs/1403.1687, Mar. 2021.

[14] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. the 32nd International Conference on Machine Learning*, Jul. 2015, pp.448-456.

[15] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. In *Proc. the 14th International Conference on Artificial Intelligence and Statistics*, Apr. 2011, pp.315-323. DOI: 10.1.1.208.6449.

[16] Krizhevsky A. Learning multiple layers of features from tiny images. Technical Report, University of Toronto. http: // www.cs.toronto.edu/∼kriz/learning-features-2009-TR.pdf, June 2022.

[17] Deng J, Dong W, Socher R, Li L, Li K, L F. ImageNet: A large-scale hierarchical image database. In *Proc. the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp.248-255. DOI: 10.1109/CVPR.2009.5206848.

[18] Hu H, Peng R, Tai Y, Tang C. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv:1607.03250, 2016. https://arxiv.org/abs/1607.03250, Jul. 2021.

[19] Han S, Mao H, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv:1510.00149, 2015. https: //arxiv.org/ abs/1510.00149, Feb. 2021.

[20] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In *Proc. the 2017 IEEE International Conference on Computer Vision*, Oct. 2017, pp.1389-1397. DOI: 10.1109/ICCV.2017.155.

[21] Luo J, Wu J, Lin W. ThiNet: A filter level pruning method for deep neural network compression. In *Proc. the 2017 IEEE International Conference on Computer Vision*, Oct. 2017, pp.5058-5066. DOI: 10.1109/ICCV.2017.541.

[22] Lebedev V, Lempitsky V. Fast ConvNets using group-wise brain damage. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.2554-2564. DOI: 10.1109/CVPR.2016.280.

[23] Wang Y, Zhang X, Xie L, Zhou J, Su H, Zhang B, Hu X. Pruning from scratch. In *Proc. the 34th AAAI Conference on Artificial Intelligence*, Feb. 2020, pp.12273-12280. DOI: 10.1609/aaai.v34i07.6910.

[24] Zhuang T, Zhang Z, Huang Y, Zeng X, Shuang K, Li X. Neuron-level structured pruning using polarization regularizer. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2020, pp.9865-9877.

[25] Wen W, Wu C, Wang Y, Chen Y, Li H. Learning structured sparsity in deep neural networks. In *Proc. the Annual Conference on Neural Information Processing Systems*, Dec. 2016, pp.2082-2090.

[26] Ye J, Lu X, Lin Z, Wang J Z. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. arXiv:1802.00124, 2018. https://arxiv.org/abs/1802.00124, Feb. 2021.

[27] Yang T, Howard A, Chen B, Zhang X, Go A, Sandler M, Sze V, Adam H. NetAdapt: Platform-aware neural network adaptation for mobile applications. In *Proc. the 15th European Conference on Computer Vision*, Sept. 2018, pp.285-300. DOI: 10.1007/978-3-030-01249-6_18.

[28] Li H, Kadav A, Durdanovic I, Samet H, Graf H P. Pruning filters for efficient convnets. arXiv:1608.08710, 2016. https://arxiv.org/abs/1608.08710, Mar. 2021.

[29] Huang Z, Wang N. Data-driven sparse structure selection for deep neural networks. In *Proc. the 15th European Conference on Computer Vision*, Sept. 2018, pp.304-320. DOI: 10.1007/978-3-030-01270-0_19.

[30] He Y, Kang G, Dong X, Fu Y, Yang Y. Soft filter pruning for accelerating deep convolutional neural networks. arXiv:1808.06866, 2018. https://arxiv.org/abs/1808.06866, Aug. 2021.

[31] He Y, Liu P, Wang Z, Hu Z, Yang Y. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proc. the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp.4340-4349. DOI: 10.1109/CVPR.2019.00447.

[32] Kang M, Han B. Operation-aware soft channel pruning using differentiable masks. arXiv:2007.03938, 2020. https: //arxiv.org/abs/2007.03938, Jul. 2021.

[33] Yu J, Yang L, Xu N, Yang J, Huang T. Slimmable neural networks. arXiv:1812.08928, 2018. https://arxiv.org/abs/1812.08928, Dec. 2021.

[34] Yu J, Huang T. AutoSlim: Towards one-shot architecture search for channel numbers. arXiv:1903.11728, 2019. https://arxiv.org/abs/1903.11728, Jun. 2021.

[35] He Y, Lin J, Liu Z, Wang H, Li L, Han S. AMC: AutoML for model compression and acceleration on mobile devices. In *Proc. the 15th European Conference on Computer Vision*, Sept. 2018, pp.784-800. DOI: 10.1007/978-3-030-01234-2_48.

[36] Liu Z, Mu H, Zhang X, Guo Z, Yang X, Cheng K, Sun J. MetaPruning: Meta learning for automatic neural network channel pruning. In *Proc. the 2019 IEEE/CVF International Conference on Computer Vision*, Oct. 2019, pp.3296-3305. DOI: 10.1109/ICCV.2019.00339.

[37] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proc. the 2015 IEEE International Conference on Computer Vision*, Dec. 2015, pp.1026-1034. DOI: 10.1109/ICCV.2015.123.

[38] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.770-778. DOI: 10.1109/CVPR.2016.90.

[39] He Y, Dong X, Kang G, Fu Y, Yan C, Yang Y. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Transactions on Cybernetics*, 2019, 50(8): 3594-3604. DOI: 10.1109/TCYB.2019.2933477.

[40] He T, Zhang Z, Zhang H, Zhang Z, Xie J, Li M. Bag of tricks for image classification with convolutional neural networks. In *Proc. the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2019, pp.558-567. DOI: 10.1109/CVPR.2019.00065.

[41] Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. Rethinking the inception architecture for computer vision. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.2818-2826. DOI: 10.1109/CVPR.2016.308.

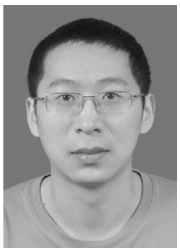
[42] Lu W, Zhao H, He Q, Huang H, Jin X. Category-consistent deep network learning for accurate vehicle logo recognition. *Neurocomputing*, 2021, 463: 623-636. DOI: 10.1016/j.neucom.2021.08.030.

[43] Zhao H, Qiu X, Lu W, Huang H, Jin X. Retinal vessel segmentation using generative adversarial learning with a large receptive field. *International Journal of Imaging Systems and Technology*, 2020, 30(3): 828-842. DOI: 10.1002/ima.22428.

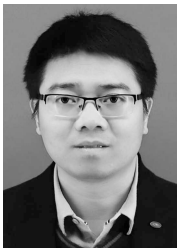
Han-Li Zhao is a professor of College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou. He received his B.Sc. degree in software engineering from Sichuan University, Chengdu, in 2004, and his Ph.D. degree in computer science from the State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, in 2009. His current research interests include computer vision, pattern recognition, medical image analysis, and deep learning. He is a senior member of CCF.

Kai-Jie Shi is a Master student at College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou. He received his B.Sc. degree in geographic information science from Nanjing Xiaozhuang University, Nanjing, in 2019. His research fields include computer vision and deep learning.

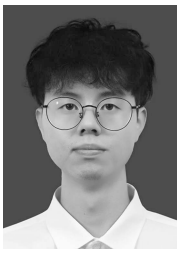
Xiao-Gang Jin is a professor of the State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou. He received his B.Sc. degree in computer science in 1989, and his M.Sc. and Ph.D. degrees in applied mathematics in 1992 and 1995, all from Zhejiang University, Hangzhou. His current research interests include texture synthesis, image completion, virtual try-on, insect swarm simulation, traffic simulation, implicit surface modeling and applications, creative modeling, sketch-based modeling and image processing. He received an ACM Recognition of Service Award in 2015 and the Best Paper Awards from CASA 2017 and CASA 2018. He is a distinguished member of CCF, and a member of ACM and IEEE.



Ming-Liang Xu is a professor with the School of Information Engineering, Zhengzhou University, Zhengzhou. He received his Ph.D. degree in computer science and technology from the State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou. His research interests include computer graphics, multimedia, and artificial intelligence.



Hui Huang is a deputy dean of College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou. He received his Master's degree in computer software and theory from Zhejiang University of Technology, Hangzhou, in 2008. He is now working for his Ph.D. degree in Northwestern Polytechnical University, Xi'an. His research interests include image processing, parallel computing and machine learning.



Wang-Long Lu is a Ph.D. student at Memorial University of Newfoundland, St. John's. He received his B.Sc. degree in digital media technology from Communication University of Zhejiang, Hangzhou, in 2018, and his M.Sc. degree in computer software and theory from Wenzhou University, Wenzhou, in 2021. His research interests include image recognition, object detection, image editing and processing.

Ying Liu is a Master student at College of Computer Science and Artificial Intelligence, Wenzhou University, Wenzhou. He received his B.Sc. degree in software engineering from Tongda College of Nanjing University of Posts & Telecommunications, Nanjing, in 2019. His research fields include computer vision and deep learning.

JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY

Volume 37, Number 3, May 2022

Content

JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY
《计算机科学技术学报》

Volume 37 Number 3 2022 (Bimonthly, Started in 1986)

Indexed in: **SCIE, Ei, INSPEC, JST, AJ, MR, CA, DBLP**

Edited by:
THE EDITORIAL BOARD OF JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY
Zhi-Wei Xu, Editor-in-Chief, P.O. Box 2704, Beijing 100190, P.R. China
Managing Editor: Feng-Di Shu E-mail: jcst@ict.ac.cn http://jcst.ict.ac.cn Tel.: 86-10-62610746

Copyright ©Institute of Computing Technology, Chinese Academy of Sciences 2022

Sponsored by: Institute of Computing Technology, CAS & China Computer Federation
Supervised by: Chinese Academy of Sciences
Undertaken by: Institute of Computing Technology, CAS
Published by: Science Press, Beijing, China
Printed by: Beijing Baochang Color Printing Co. Ltd
Distributed by:
China: All Local Post Offices
Other Countries: Springer Nature Customer Service Center GmbH, Tiergartenstr. 15, 69121 Heidelberg, Germany
Available Online: https://link.springer.com/journal/11390

ISSN 1000-9000 05>

国内统一刊号: CN11-2296/TP 国内邮发代号: 2-578 RMB ¥160.00