# Probability-based channel pruning for depthwise separable convolutional networks

Hanli Zhao
Wenzhou University

Kaijie Shi
Wenzhou University

Xiaogang Jin
Zhejiang University

Mingliang Xu
Zhengzhou University

Hui Huang
Wenzhou University

Wanglong Lu
Wenzhou University

Ying Liu
Wenzhou University

## 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.**

## 1. Introduction

With the tremendous development of deep learning, network compression [1] has been becoming a hot research topic for small memory footprint and low runtime latency with good performance. As one of commonly used compression techniques, channel pruning [6, 24] compresses the network model by removing redundant structures and parameters. It boosts the development of artificial intelligence applications in our daily life, such as driveless cars, robotics and augmented reality [34].

Most channel pruning algorithms [6, 26] 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 [26]. Some algorithms [23, 24] try to keep the accuracy performance 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 [4, 15, 28, 29, 40] are built on the block of depthwise separable convolutions to achieve a balance between 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 [23, 24, 26]. Only a few algorithms [39] 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 [30] 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) [18] and rectified linear unit (ReLU) [3] 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 [15], MobileNetV2 [29], ShuffleNetV1 [40], ShuffleNetV2 [28], and GhostNet [4] networks on publicly available CIFAR10 [20], CIFAR100 [20], and ImageNet [2] datasets. The experimental results on the above representative networks show that the proposed method is able to achieve high accuracy at a low resource consumption.

In summary, our paper makes the following contributions:

- A simple yet effective probability-based channel pruning criterion by considering the scaling and shifting factors of BN.
- An efficient probability-based pruning algorithm without requiring extra time-consuming fine-tuning for depthwise separable convolutional networks by using a novel shifting factor fusion technique.
- We validate the feasibility of our method through extensive experiments, and 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 employ pre-training, pruning, and fine-tuning phases for effective network pruning [5, 14, 16, 24]. The pre-training phase is used to produce clues for pruning phase. It trains a network by adding some extra constraints, such as group-lasso [21, 27], L1 regularization [6, 24, 32] or Polarization regularizer [41]. The pruning phase usually prunes weights, filters, channels or layers via various pruning criteria [33], respectively. Some methods [24, 36] 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 [14]. Hu et al. [16] introduce a network trimming method by pruning unimportant channels with high average percentage of zeros after the ReLU mapping and retrain the trimmed network to enhance the performance. Yang et al. [35] choose a sub-network that has higher accuracy and lower resource consumption by removing different filters from one layer. Since the performance of the pruned network model is usually not as good as the pre-trained one, the fine-tuning phase is further required to train the pruned model [14, 17, 22, 27]. Zhang et al. [39] first prune channels of depthwise separable convolution unit based on information gain and then restore 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. [26] conduct a number of experiments to indicate that the benefits gained from pruning is attributed to the architecture of 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 shifting factors contribute to the BN layer. Different from them, we investigate a new pruning criterion by considering both the scaling and shifting factors in the BN layer.

Some researchers focus on efficient network pruning algorithms without fine-tuning or retraining. The NFP algorithm [23] 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 as same as that of the unpruned one. He et al. [11] dynamically determine whether a certain channel is pruned or not in a soft manner while He et al. [13] remove filters via geometric median which minimizes the sum of Euclidean distances. Kang and Han [19] incorporate 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 neural architecture search which provides a violence search method to discover the compressed model structure. A shared network [38] is trained with switchable batch normalization and can adjust the network's width on the fly rather than downloading and offloading different models. A slimmable network [37] is used to approximate the network's accuracy of different channels and is then greedily slimmed for minimal accuracy drop. He et al. [12] get the model compression policy by reinforcement learning while Liu et al. [25] search a good-performing pruned network by

evolutionary procedure. However, these methods require much training time as well as GPU resources to search for efficient structures.

## 3. Preliminaries

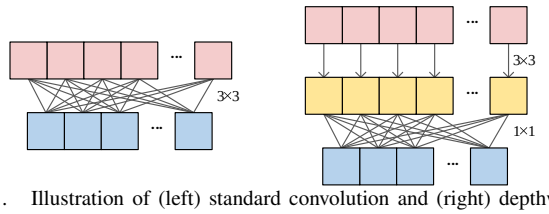


Fig.1. Illustration of (left) standard convolution and (right) depthwise separable convolution. Data are flowing from top to down.

**Depthwise separable convolutions**, initially introduced in [30], are effective to reduce 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 $1 \times 1$ 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 [15].

**MobileNetV1** [15] and **MobileNetV2** [29] 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** [40] and **ShuffleNetV2** [28] 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 and depthwise convolution. 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** [4] uses a novel Ghost module to generate more ghost feature maps from cheap operations. As illustrated in Fig. 2 (e), 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** [18] is a linear transformation layer plugged between the convolutional layer and the activation function layer. It whitens and transforms each channel 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:

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

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

where $\epsilon = 10^{-5}$ is a parameter avoiding dividing by 0, $\mathrm{E}[x]$ and $\mathrm{Var}[x]$ respectively represent the mean and variance of $x$, $\gamma$ and $\beta$ are learnable scaling factor and shifting factor. The numbers of $\gamma$ and $\beta$ are the same as the number of convolutional filters. Notice that the values of $\mathrm{E}[x]$ and $\mathrm{Var}[x]$ are calculated across mini-batches during training and are fixed during inference [18].

**ReLU** [3] is the rectifier activation function allowing a network to easily obtain sparse representations. Formally, ReLU is defined as:

$$y = \mathrm{ReLU}(\hat{y}) = \begin{cases} 0, & \hat{y} \leq 0; \\ \hat{y}, & \hat{y} > 0. \end{cases} \tag{3}$$

MobileNetV2 uses a variant activation function ReLU6 [29], which has the same property when $\hat{y} \leq 0$. For simplicity, we always use ReLU instead of ReLU6 in the rest of this paper.

From Eq. 3 we can see that the output value is 0 with the input $\hat{y} \leq 0$ for ReLU. Therefore, if $\hat{y} \leq 0$ 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. The 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 [5, 24, 41], which is different from the normal training of the original network.

We pre-train the depthwise separable convolutional network with Kaiming initialization [7]. In addition, we use L1 regularization on scaling factors in BN as in [24]. Specifically, our pre-training objective function is defined as:

$$\min_{\theta} \frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \theta), y_i) + \lambda \cdot L_1(\gamma_p), \tag{4}$$

where $N$ denotes the number of train samples, $x_i$ and $y_i$ denote train input and target, $\theta$ denotes trainable weights, the first sum term represents the normal training loss, $\lambda$ is a hyperparameter, $\gamma_p$ denotes the scaling factors in BN, and $L_1$ represents L1 regularization.

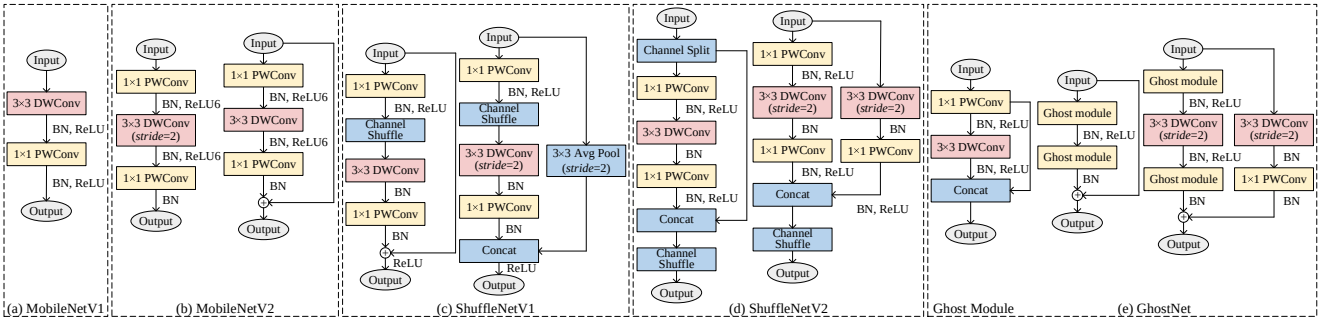


Figure 2. Illustration of structures of basic blocks in (a) MobileNetV1, (b) MobileNetV2, (c) ShuffleNetV1, (d) ShuffleNetV2, and (e) GhostNet.

### 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 to the following convolutions and thus can be pruned. Many deep depthwise separable convolutional networks [4, 15, 28, 29, 40] 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 Eq. 1 follows a normal distribution by normalizing the input $x$ from a large number of input samples [19]. Therefore, the output $\hat{y}$ of BN in Eq. 2 is normally distributed with mean $\beta$ and variance $\gamma^2$, that is, $\hat{y} \sim \mathrm{N}(\beta, \gamma^2)$. Let $z$ be a predefined z-score of the standard normal distribution $\mathrm{N}(0, 1)$, then the normal distribution $\mathrm{N}(\beta, \gamma^2)$ has the upper confidence limit:

$$Z(z) = \beta + z \cdot |\gamma|. \tag{5}$$

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

$$\begin{aligned} P(\hat{y} \le Z(z)) &= P(\frac{\hat{y} - \beta}{|\gamma|} \le z) \\ &= P(\hat{x} \le z) = \int_{-\infty}^{z} \frac{\mathrm{e}^{-\frac{\hat{x}^2}{2}}}{\sqrt{2\pi}} d\hat{x}. \end{aligned} \tag{6}$$

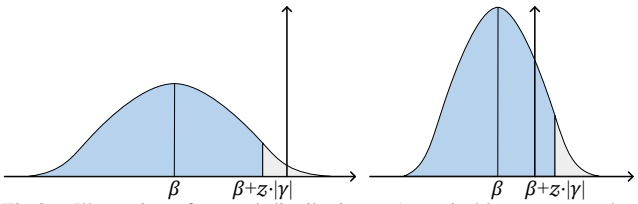


Fig.3. Illustration of normal distributions. Areas in blue represent the accumulative probabilities for $\beta + z \cdot |\gamma|$. All values below zero will be rectified to zero in the ReLU layer.

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, $Z = \beta + z \cdot |\gamma| \le 0$, there is a high probability to have $\hat{y} \le 0$ to be true, as illustrated in Fig. 3 (left). On the other hand, if $\beta + z \cdot |\gamma| > 0$, then it is probably not true for $\hat{y} \le 0$, as illustrated in Fig. 3 (right). If $\beta + z \cdot |\gamma| \le 0$ holds, the output of ReLU is likely to be equal to zero according to the definition of ReLU operation.

Therefore, we propose a novel probability-based pruning criterion by taking advantage of the z-score: if $\beta + z \cdot |\gamma| \le 0$ for a BN layer, the corresponding channel is regarded as an unimportant channel and can be pruned effectively. The z-score $z$ can be used to adjust the probability threshold. A greater $z$ corresponds to a high 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 greater $z$ 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 accuracy and size.

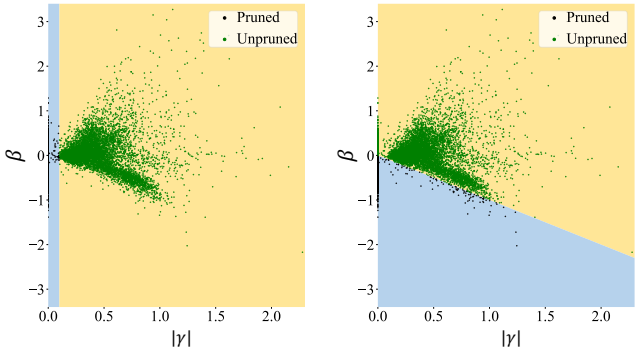


Fig.4. Illustration of the difference between (left) $\gamma$-based criterion and the proposed criterion based on both $\gamma$ and $\beta$. Black points represent pruned channels while green points represent unpruned channels, respectively.

Many existing algorithms [23, 24, 41] prune channels

corresponding to the scaling factor $\gamma$ in Eq. 2 below a threshold. In comparison, not only the scaling factor $\gamma$ but also the shifting factor $\beta$ are considered in our probability-based pruning criterion. We illustrate the difference between the widely used the $\gamma$-based pruning criterion and the proposed pruning criterion in Fig. 4. We can find that channels with small $\gamma$ but great $\beta$ are also pruned in the $\gamma$-based pruning criterion. Moreover, the $\gamma$-based pruning criterion cannot prune channels with great $|\gamma|$ but $\beta + z \cdot |\gamma| \leq 0$ because the criterion views these channels as important features. In comparison, we prune channels based on the combination of both parameters $\beta + z \cdot |\gamma|$ no matter which value $|\gamma|$ is. As a result, our criterion can effectively prune channels with great $|\gamma|$ but $\beta + z \cdot |\gamma| \leq 0$.

### 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 [29], ShuffleNetV1 [40], ShuffleNetV2 [28], and GhostNet [4]. Each bottleneck block consists of two $1 \times 1$ PWConv and a $3 \times 3$ DWConv layers. The $1 \times 1$ layers are responsible for reducing and then increasing (restoring) dimensions, leaving the $3 \times 3$ layer a bottleneck with smaller input/output dimensions [8]. 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 and a DWConv layers and the bottleneck case of using two PWConv and a DWConv layers.

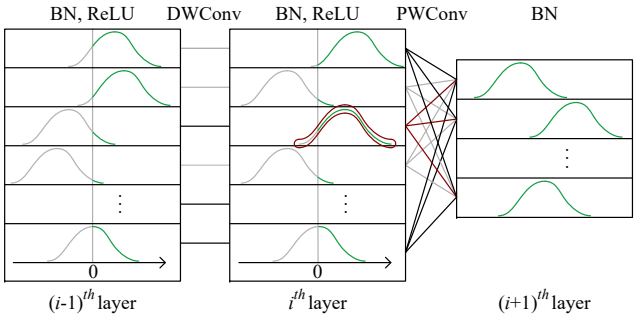

Fig.5. Illustration of pruning process. Each curve represents a channel where light gray represents zero value output by ReLU. The channel highlighted with the red contour should be further processed.

As illustrated in Fig. 5, there are three BN layers (denoted as $(i-1)^{th}$, $i^{th}$, and $(i+1)^{th}$ BN layers) before and after each DWConv and PWConv. The numbers of channels in $(i-1)^{th}$ and $i^{th}$ BN layers 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 $(i-1)^{th}$ and $i^{th}$ BN layers:

**Table 1.** Channel pruning scheme for different cases in DWConv between $(i-1)^{th}$ and $i^{th}$ BN layers. We suppose $Z^{i-1} = \beta_k^{i-1} + z \cdot |\gamma|_k^{i-1}$ and $Z^i = \beta_k^i + z \cdot |\gamma|_k^i$ for $(i-1)^{th}$ and $i^{th}$ BN layers, respectively.

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

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

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

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

**Case 4:** Both $(i-1)^{th}$ and $i^{th}$ BN layers conform to the pruning criterion, then both BN layers and corresponding ReLU and DWConv layers for $k^{th}$ 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. Take the channel highlighted with the red contour in Fig. 5 as an example. For Case 3, $k^{th}$ channel of $(i-1)^{th}$ BN is pruned according to the proposed pruning criterion and the impact of $x_k^i$ on $k^{th}$ channel of $i^{th}$ BN is also pruned, i.e., $x_k^i = 0$. The channel numbers of input and output for DWConv should be consistent. Therefore, we should also prune the corresponding channel for $i^{th}$ BN. For Cases 2∼4, $k^{th}$ channel of $i^{th}$ BN is pruned and it is equivalent to truncate $y_k^i = 0$. Based on the probability-based pruning criterion, it is probably true

for Case 2 and Case 4 but is 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.

For Case 3, we first calculate the impact $t_k^i$ on $i^{th}$ BN according to definitions of BN and ReLU:

$$\begin{aligned} t_k^i &= \mathrm{ReLU}(\mathrm{BN}(x_k^i)) \\ &= \mathrm{ReLU}(\gamma_k^i \cdot \frac{x_k^i - \mathrm{E}[x_k^i]}{\sqrt{\mathrm{Var}[x_k^i] + \epsilon}} + \beta_k^i) \\ &= \mathrm{ReLU}(\gamma_k^i \cdot \frac{-\mathrm{E}[x_k^i]}{\sqrt{\mathrm{Var}[x_k^i] + \epsilon}} + \beta_k^i), \end{aligned} \tag{7}$$

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 PWConv layer can be calculated as follows:

$$x^{i+1} = \sum_{k \in K_1} w_k^i \cdot y_k^i + \sum_{k \in K_3} w_k^i \cdot t_k^i, \tag{8}$$

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.

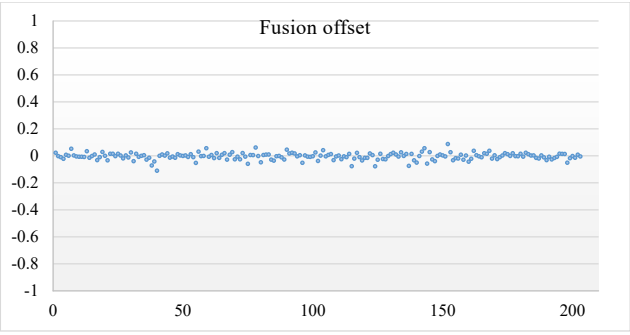

Fig.6. Visualization of the fusion offset between $\beta$ values before and after shifting factor fusion. The horizontal axis represents index of unpruned channels. The data are sampled from $11^{th}$ BN layer of the pruned MobileNetV1 trained on CIFAR100.

The calculation of $(i+1)^{th}$ BN can be derived as follows:

$$\begin{aligned} &BN^{i+1}_{\gamma^{i+1},\beta^{i+1}}(x^{i+1}) \\ &= BN^{i+1}_{\gamma^{i+1},\beta^{i+1}}(\sum_{k \in K_1} w_k^i \cdot y_k^i + \sum_{k \in K_3} w_k^i \cdot t_k^i) \\ &= \gamma^{i+1} \cdot \frac{\sum_{k \in K_1} w_k^i \cdot y_k^i - \mathrm{E}[x^{i+1}]}{\sqrt{\mathrm{Var}[x^{i+1}] + \epsilon}} + \\ &\quad \gamma^{i+1} \cdot \sum_{k \in K_3} \frac{w_k^i \cdot t_k^i}{\sqrt{\mathrm{Var}[x^{i+1}] + \epsilon}} + \beta^{i+1}. \end{aligned} \tag{9}$$

Now we have a constant $\beta^{i+1}_{fusion}$ by defining as follows:

$$\beta^{i+1}_{fusion} = \gamma^{i+1} \cdot \sum_{k \in K_3} \frac{w_k^i \cdot t_k^i}{\sqrt{\mathrm{Var}[x^{i+1}] + \epsilon}} + \beta^{i+1}. \tag{10}$$

Finally, we have the following updated formula by fusing the truncation error to the $i+1^{th}$ BN layer:

$$BN^{i+1}_{\gamma^{i+1},\beta^{i+1}}(x^{i+1}) = BN^{i+1}_{\gamma^{i+1},\beta^{i+1}_{fusion}}(x^{i+1}). \tag{11}$$

Therefore, we can fuse the involved learnable parameters of pruned channels into $\beta^{i+1}_{fusion}$ for Case 3 using Eq. 10 which we call it as shifting factor fusion. We propose the sophisticated channel pruning algorithm by replacing the shifting factor $\beta^{i+1}$ with $\beta^{i+1}_{fusion}$ 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 $\beta$ 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 $\beta$ 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.

### 4.4. Additional processing

Residual learning [8] 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 [4, 28, 29, 40], and a residual block is defined as the elementwise addition [8]. Let $x$ be the input and $F(x)$ be its residual mapping, 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 [10], a normal elementwise addition is applied to unpruned channels while channels of $x$ are used directly for pruned channels:

$$y_k = \begin{cases} x_k + F(x_k), & \text{if } k \in K_1; \\ x_k, & \text{otherwise.} \end{cases} \tag{12}$$

The proposed probability-based method prunes channels efficiently based on BN layers followed by ReLU. For each

**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
    $Z^{i-1} = BN^{i-1}[k].\beta + z \cdot BN^{i-1}[k].|\gamma|$;
    $Z^i = BN^i[k].\beta + z \cdot BN^i[k].|\gamma|$;
    **if** $Z^{i-1} > 0$ and $Z^i > 0$ **then** ▷ Case 1
      Mark $k$ as unpruned;
    **else if** $Z^{i-1} > 0$ and $Z^i \leq 0$ **then** ▷ Case 2
      Mark $k$ as pruned;
    **else if** $Z^{i-1} \leq 0$ and $Z^i > 0$ **then** ▷ Case 3
      Mark $k$ as pruned;
      $K_3^i.append(k)$;
    **else if** $Z^{i-1} \leq 0$ and $Z^i \leq 0$ **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 Eq. 10;
Clone unpruned channels of $M_0$ to $M_1$;
**return** Pruned model $M_1$

BN layer after PWConv layer, if there is no ReLU layer, we add an associated gate as in [32].

### 4.5. Pseudocode

We provide the pseudocode of our probability-based channel pruning for depthwise separable convolution networks in Algorithm 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. From 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 to 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 were conducted on CIFAR10 [20], CIFAR100 [20], and ImageNet [2] 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 $32 \times 32$. During training, the stochastic gradient descent optimizer is adopted for the minimization of 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 $10\times$ at 50% and 75% of all epochs, respectively. The hyperparameter $\lambda$ is set as $10^{-4}$. As the image size is small, for MobileNetV1, we reduce the network to 4 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 [37].

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 $224 \times 224$ 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

Table 2. Comparison of the proposed method to state-of-the-art methods for pruning MobileNetV1 and MobileNetV2 models on CIFAR100. Note that "w" and "w/o" are short for the words "with" and "without", respectively.

| Method | CIFAR100 | | | | | |
|---|---|---|---|---|---|---|
| | MobileNetV1 | | | MobileNetV2 | | |
| | #Params | FLOPs | Top-1 | #Params | FLOPs | Top-1 |
| Original model [15, 29] | 3.31M | 46.47M | 71.09% | 2.32M | 88.10M | 75.26% |
| Slimming w/o fine-tuning [24] | 1.34M | 26.36M | 69.49% | 1.25M | 39.70M | 71.69% |
| Slimming w fine-tuning [24] | 1.34M | 26.36M | 71.25% | 1.25M | 39.70M | 74.83% |
| NFP [23] | 1.34M | 26.36M | 70.78% | 1.25M | 39.70M | 74.55% |
| Ours w/o fusion | 1.33M | 26.30M | 69.49% | 1.23M | 39.66M | 71.42% |
| **Ours w fusion** | **1.33M** | **26.30M** | **71.27%** | **1.23M** | **39.66M** | **75.37%** |

Table 3. Comparison of the proposed method to state-of-the-art methods for pruning MobileNetV1 and MobileNetV2 models on ImageNet. "N/A" represents the corresponding item is unavailable. For a fair comparison, we include baselines of state-of-the-art methods collected from their original papers, respectively.

| Method | ImageNet | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MobileNetV1 | | | | MobileNetV2 | | | |
| | #Params | FLOPs | Baseline | Top-1 | #Params | FLOPs | Baseline | Top-1 |
| Original model [15, 29] | 4.2M | 569M | 70.6% | 70.6% | 3.5M | 300M | 71.8% | 71.8% |
| AMC [12] | 2.4M | 285M | 70.6% | 70.5% | N/A | 219M | 71.8% | 70.8% |
| MetaPruning [25] | N/A | 281M | 70.6% | 70.6% | N/A | 217M | 72.0% | 71.2% |
| NetAdapt [35] | N/A | 284M | 70.6% | 69.1% | N/A | N/A | N/A | N/A |
| NLSP [41] | N/A | N/A | N/A | N/A | N/A | 216M | 72.0% | 71.8% |
| PFS [32] | 4.0M | 567M | 70.9% | 71.6% | 3.5M | 300M | 71.8% | 72.1% |
| Slimming w/o fine-tuning [24] | 2.6M | 338M | 70.6% | 70.4% | 3.0M | 238M | 71.8% | 71.4% |
| Slimming w fine-tuning [24] | 2.6M | 338M | 70.6% | 70.5% | 3.0M | 238M | 71.8% | 71.6% |
| NFP [23] | 2.6M | 338M | 70.6% | 71.2% | 3.0M | 238M | 71.8% | 71.8% |
| Ours w/o fusion | 2.5M | 338M | 70.6% | 71.3% | 2.9M | 210M | 71.8% | 71.6% |
| **Ours w fusion** | **2.5M** | **338M** | **70.6%** | **71.6%** | **2.9M** | **210M** | **71.8%** | **71.8%** |

use the cosine learning rate decay as in [9]. The hyperparameter $\lambda$ is set as $10^{-5}$. We also employ label smoothing [31] for better generalization.

### 5.2. Comparison with state-of-the-art methods

In this subsection, we compare the proposed method to state-of-the-art methods by pruning MobileNets [15, 29], ShuffleNets [28, 40], and GhostNet [4] on CIFAR10, CIFAR100, and ImageNet datasets. The compared results of PFS (Pruning From Scratch) [32], AMC [12], MetaPruning [25], NetAdapt [35], and NLSP (Neuron-Level Structured Pruning) [41] are collected from the authors' papers, respectively. The results of Slimming [24] and NFP [23] 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 [24] without fine-tuning in both accuracy and size. Our model with fusion is also better than network slimming with fine-tuning and NFP [23]. 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 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 to the MobileNetV1 baseline [15] but still achieves a better top-1 accuracy. Our pruned model prunes 17.1% of parameters and 30.0% of FLOPs compared to the MobileNetV2 baseline [29]. Although our pruned model has greater FLOPs than models of AMC [12], MetaPruning [25], and NetAdapt [35] for pruning MobileNetV1 [15], the top-1 accuracy is significantly higher than theirs. The accuracy of our method is comparable with that of NLSP [41] but the FLOPs value of our method is better. PFS [32] is the only one method that has a higher accuracy than our method. Unfortunately, the reduction of 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 to 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 [12], MetaPruning [25], NetAdapt [35], NLSP [41], and Slimming [24] drop the performance in the top-1 accuracy as compared to 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

Table 4. Comparison of the proposed method to state-of-the-art methods for pruning ShuffleNetV1, ShuffleNetV2, and GhostNet models on CIFAR10, CIFAR100, and ImageNet, respectively.

| | Method | CIFAR10 | | | CIFAR100 | | | ImageNet | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | #Params | FLOPs | Top-1 | #Params | FLOPs | Top-1 | #Params | FLOPs | Top-1 |
| ShuffleNetV1 | Original model [40] | 3.50M | 166.49M | 93.17% | 3.67M | 166.68M | 75.32% | 5.4M | 524M | 73.7% |
| | Slimming [24] | 0.53M | 40.31M | 93.15% | 0.86M | 49.01M | 75.28% | 4.1M | 391M | 72.5% |
| | NFP [23] | 0.53M | 40.31M | 93.17% | 0.86M | 49.01M | 75.32% | 4.1M | 391M | 72.8% |
| | Ours w/o fusion | 0.47M | 36.11M | 93.13% | 0.86M | 48.59M | 75.31% | 4.0M | 387M | 72.6% |
| | **Ours w fusion** | **0.47M** | **36.11M** | **93.17%** | **0.86M** | **48.59M** | **75.32%** | **4.0M** | **387M** | **72.8%** |
| ShuffleNetV2 | Original model [28] | 2.47M | 95.17M | 92.55% | 2.56M | 95.27M | 74.59% | 3.5M | 299M | 72.6% |
| | Slimming [24] | 1.28M | 43.73M | 92.24% | 1.65M | 52.49M | 74.58% | 3.4M | 284M | 72.6% |
| | NFP [23] | 1.28M | 43.73M | 92.55% | 1.65M | 52.49M | 74.59% | 3.4M | 284M | 72.6% |
| | Ours w/o fusion | 1.14M | 37.68M | 92.52% | 1.57M | 51.02M | 74.36% | 3.3M | 280M | 72.5% |
| | **Ours w fusion** | **1.14M** | **37.68M** | **92.55%** | **1.57M** | **51.02M** | **74.59%** | **3.3M** | **280M** | **72.6%** |
| GhostNet | Original model [4] | 3.63M | 56.87M | 91.18% | 3.99M | 57.18M | 74.95% | 5.2M | 141M | 73.9% |
| | Slimming [24] | 2.79M | 25.76M | 91.05% | 3.36M | 31.23M | 74.94% | 5.0M | 122M | 73.4% |
| | NFP [23] | 2.79M | 25.76M | 91.18% | 3.36M | 31.23M | 74.95% | 5.0M | 122M | 73.5% |
| | Ours w/o fusion | 2.75M | 22.48M | 89.92% | 3.23M | 26.61M | 74.88% | 4.9M | 120M | 73.4% |
| | **Ours w fusion** | **2.75M** | **22.48M** | **91.18%** | **3.23M** | **26.61M** | **74.95%** | **4.9M** | **120M** | **73.5%** |

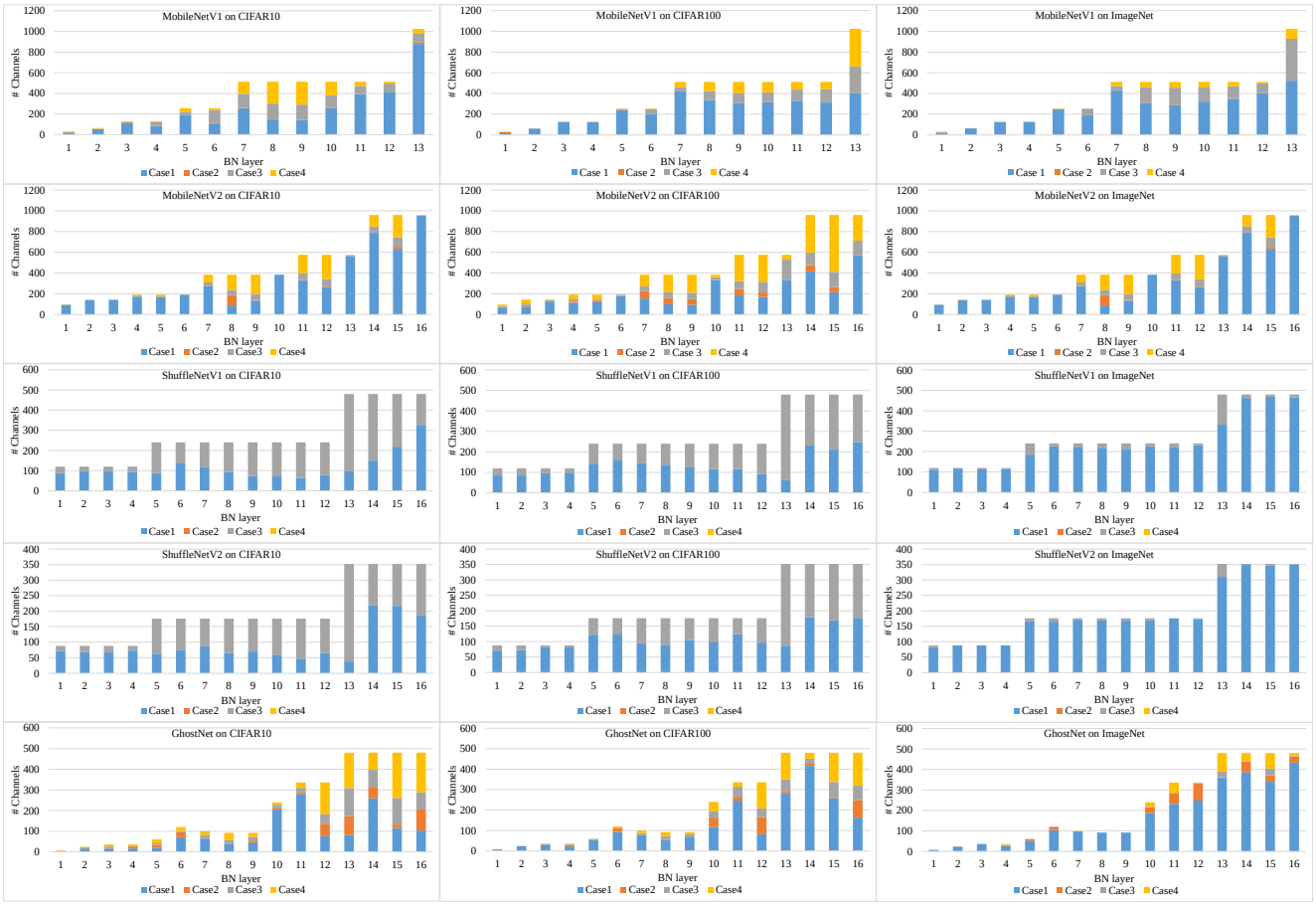


Figure 7. Statistics of numbers of channels of BN layers after each DWConv in MobileNetV1, MobileNetV2, ShuffleNetV1, ShuffleNetV2, and GhostNet, respectively.

PFS [32] and NFP [23] with similar top-1 accuracies. It is noted that PFS [32] 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 parame-

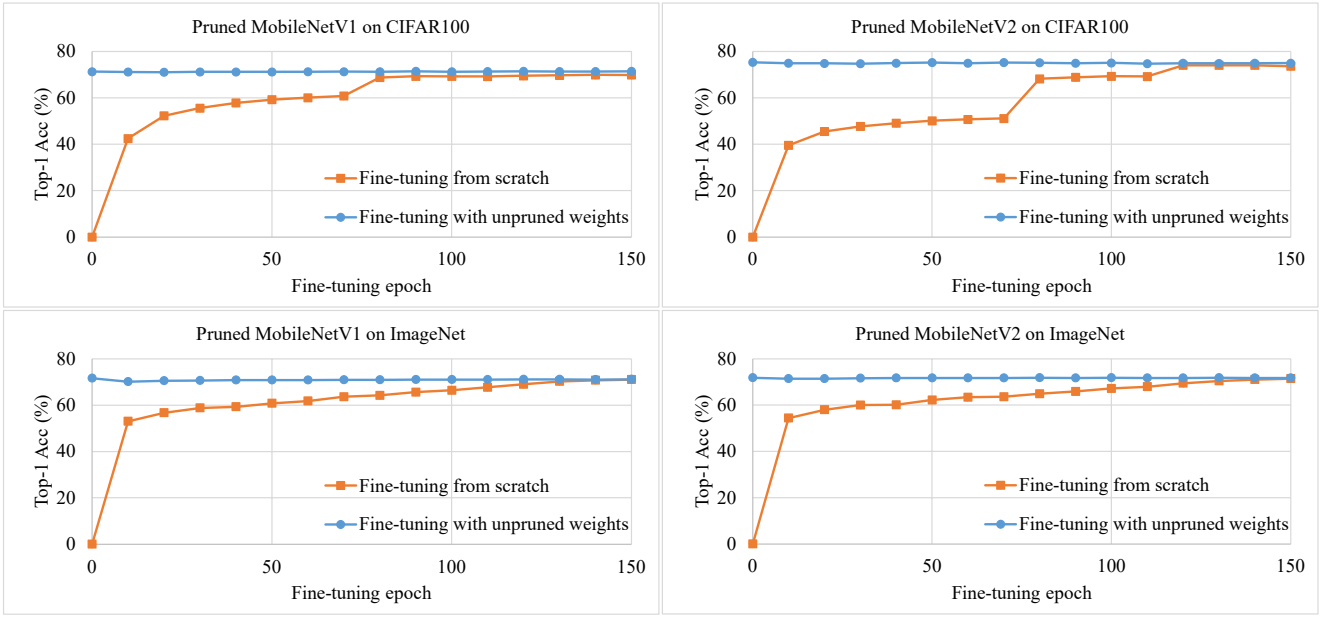


Figure 8. Performance comparison of our pruned models between fine-tuning from unpruned weights and from scratch.

ters and FLOPs compared to our pruned ShuffleNetV1 and GhostNet. The results show that our method without fine-tuning is able to achieve better performance than Slimming [24] and NFP [23].

From the above experimental results provided in Tables 2∼4, the proposed criterion can reduce more parameters and FLOPs than the $\gamma$-based criterion [23, 24] when similar top-1 accuracies are obtained. On the other hand, the proposed criterion using both $\gamma$ and $\beta$ achieves a higher accuracy than the $\gamma$-based criterion when similar numbers of parameters are pruned. The $\gamma$-based pruning criterion cannot prune channels with great $|\gamma|$ but $\beta + z \cdot |\gamma| \leq 0$ 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 $\gamma$ and $\beta$. Therefore, the proposed criterion using both $\gamma$ and $\beta$ 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 to our method without fusion. It should be noted that the proposed novel shifting factor fusion allows a high accuracy without 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 accuracy and 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 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, adjacent two BN layers between each DWConv layer 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 between each DWConv 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 [24] which are trained in the pre-training

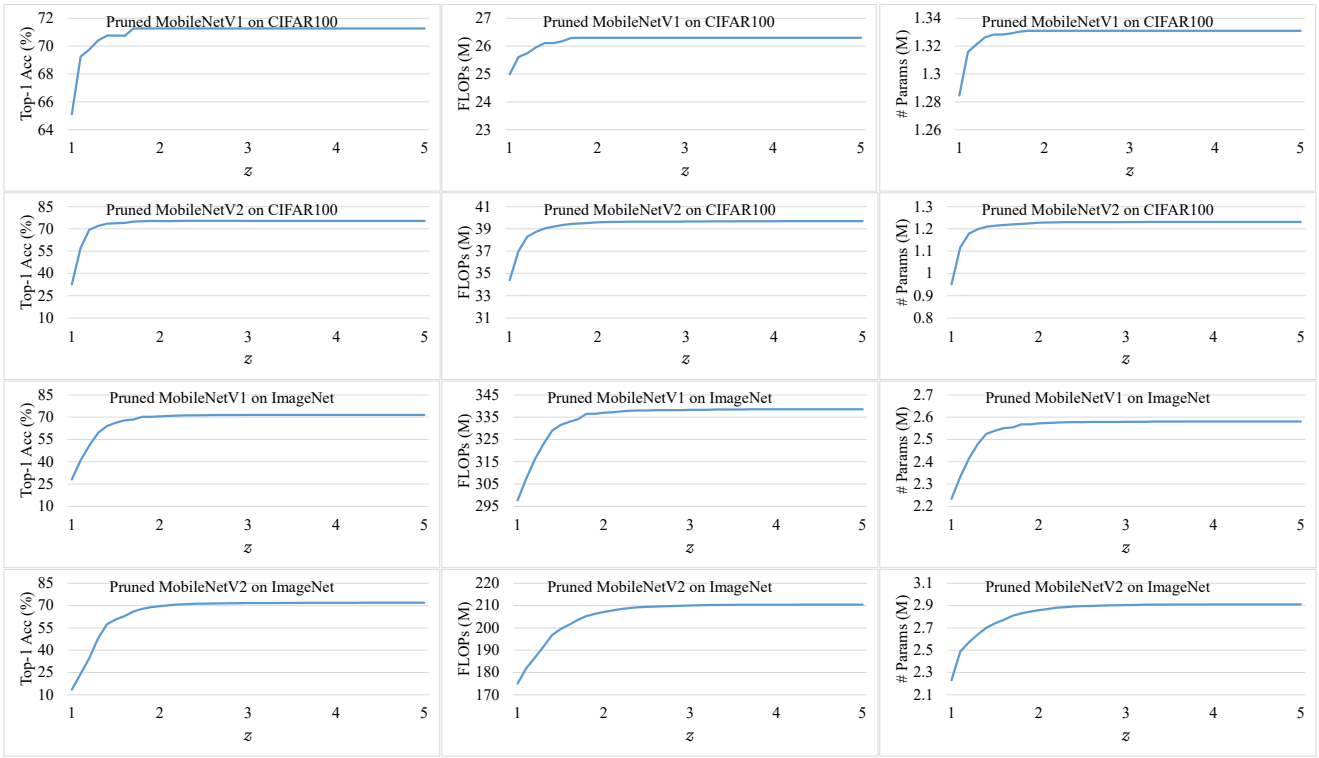


Figure 9. Statistics of the top-1 accuracy, FLOPs and number of parameters of the pruned MobileNetV1 and MobileNetV2 by varying the value of z-score parameter $z$.

phase. We also experiment to fine-tune our pruned model from scratch [26]. The performance comparison of our pruned models between fine-tuning from unpruned weights and from scratch is illustrated in Fig. 8. Please zoom in for better visualization. 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 to 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 z-score parameter $z$

Now we perform the ablation study on the z-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 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 z-score parameter $z$ as defined in Eq. 5. A greater z-score corresponds to a high probability and vice versa. We can see that all of the top-1 accuracy, FLOPs, and number of parameters 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 speed are important quantities for a pruned model. Therefore, we empirically choose $z \in [2, 4]$ for a trade-off between accuracy and size.

## 6. Conclusions

In this paper, we have 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 is proposed. A novel shifting factor fusion technique is developed to further improve the pruning performance. We have validated the efficiency of the proposed channel pruning method on representative depthwise separable convolutional networks including MobileNetV1, MobileNetV2, ShuffleNetV1, ShuffleNetV2, and GhostNet. Promising experimental results and comparisons have shown the feasi-

bility 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 object detection and image segmentation.

## Acknowledgements

The authors would like to thank our anonymous reviewers for their valuable comments to improve this paper. X. Jin was supported by the National Natural Science Foundation of China (No. 62036010) and the Fundamental Research Funds for the Central Universities. H. Zhao was supported by the Zhejiang Provincial Natural Science Foundation of China (No. LZ21F020001). H. Huang was supported by the National Natural Science Foundation of China (No. 62072340) and Zhejiang Provincial Natural Science Foundation of China (No. LSZ19F020001).

## References

[1] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint*, arXiv:1710.09282, 2017. 1

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2, 7

[3] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proc. International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011. 2, 3

[4] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu. Ghostnet: More features from cheap operations. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020. 1, 2, 3, 4, 5, 6, 8, 9

[5] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Proc. International Conference on Learning Representations*, 2016. 2, 3

[6] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Proc. Advances in Neural Information Processing Systems*, volume 28, pages 1135–1143, 2015. 1, 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proc. IEEE International Conference on Computer Vision*, pages 1026–1034, 2015. 3

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 5, 6

[9] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li. Bag of tricks for image classification with convolutional neural networks. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 8

[10] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Transactions on Cybernetics*, 50(8):3594–3604, 2019. 6

[11] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proc. International Joint Conference on Artificial Intelligence*, pages 2234–2240, 2018. 2

[12] 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. European Conference on Computer Vision*, pages 784–800, 2018. 2, 8

[13] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 2

[14] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *Proc. IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 2

[15] 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. *arXiv preprint*, arXiv:1704.04861, 2017. 1, 2, 3, 4, 8

[16] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint*, arXiv:1607.03250, 2016. 2

[17] Z. Huang and N. Wang. Data-driven sparse structure selection for deep neural networks. In *Proc. European Conference on Computer Vision*, pages 304–320, 2018. 2

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. International Conference on Machine Learning*, pages 448–456, 2015. 2, 3

[19] M. Kang and B. Han. Operation-aware soft channel pruning using differentiable masks. In *Proc. International Conference on Machine Learning*, pages 5122–5131, 2020. 2, 4

[20] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *University of Toronto*, Technical Report TR-2009, 2009. 2, 7

[21] V. Lebedev and V. Lempitsky. Fast convnets using group-wise brain damage. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2554–2564, 2016. 2

[22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *arXiv preprint*, arXiv:1608.08710, 2016. 2

[23] R. Liu, J. Cao, P. Li, W. Sun, Y. Zhang, and Y. Wang. Nfp: A no fine-tuning pruning approach for convolutional neural network compression. In *Proc. International Conference on Artificial Intelligence and Big Data*, pages 74–77, 2020. 1, 2, 4, 8, 9, 10

[24] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *Proc. IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 1, 2, 3, 4, 8, 9, 10

[25] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proc. IEEE/CVF International*

*Conference on Computer Vision*, pages 3296–3305, 2019. 2, 8

[26] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. In *Proc. International Conference on Learning Representations*, 2018. 1, 2, 11

[27] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proc. IEEE International Conference on Computer Vision*, pages 5058–5066, 2017. 2

[28] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. 1, 2, 3, 4, 5, 6, 8, 9

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018. 1, 2, 3, 4, 5, 6, 8

[30] L. Sifre and S. Mallat. Rigid-motion scattering for texture classification. *arXiv preprint*, arXiv:1403.1687, 2014. 1, 3

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016. 8

[32] Y. Wang, X. Zhang, L. Xie, J. Zhou, H. Su, B. Zhang, and X. Hu. Pruning from scratch. In *Proc. AAAI Conference on Artificial Intelligence*, pages 12273–12280, 2020. 2, 7, 8, 9

[33] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *Proc. Advances in Neural Information Processing Systems*, volume 29, pages 2074–2082, 2016. 2

[34] D. M. West. *The future of work: Robots, AI, and automation*. Brookings Institution Press, 2018. 1

[35] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *Proc. European Conference on Computer Vision*, pages 285–300, 2018. 2, 8

[36] 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. International Conference on Learning Representations*, 2018. 2

[37] J. Yu and T. Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint*, arXiv:1903.11728, 2019. 2, 7

[38] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang. Slimmable neural networks. In *Proc. International Conference on Learning Representations*, 2019. 2

[39] K. Zhang, K. Cheng, J. Li, and Y. Peng. A channel pruning algorithm based on depth-wise separable convolution unit. *IEEE Access*, 7:173294–173309, 2019. 1, 2

[40] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 1, 2, 3, 4, 5, 6, 8, 9

[41] T. Zhuang, Z. Zhang, Y. Huang, X. Zeng, K. Shuang, and X. Li. Neuron-level structured pruning using polarization regularizer. In *Proc. Advances in Neural Information Processing Systems*, volume 33, pages 9865–9877, 2020. 2, 3, 4, 8