Network Pruning via Feature Shift Minimization

Yuanzhi Duan[1], Yue Zhou[1], Peng He[1], Qiang Liu[2], Shukai Duan[1], and Xiaofang Hu[1]⋆[0000−0003−3764−2640]

[1] College of Artificial Intelligence, Southwest University, Chongqing, China
[2] Harbin Institute of Technology, Harbin, China
{huxf,duansk}@swu.edu.cn, {swuyzhid,hepeng5}@email.swu.edu.cn,
zhouyuenju@163.com, 18b933041@stu.hit.edu.cn

Abstract. Channel pruning is widely used to reduce the complexity of deep network models. Recent pruning methods usually identify which parts of the network to discard by proposing a channel importance criterion. However, recent studies have shown that these criteria do not work well in all conditions. In this paper, we propose a novel Feature Shift Minimization (FSM) method to compress CNN models, which evaluates the feature shift by converging the information of both features and filters. Specifically, we first investigate the compression efficiency with some prevalent methods in different layer-depths and then propose the feature shift concept. Then, we introduce an approximation method to estimate the magnitude of the feature shift, since it is difficult to compute it directly. Besides, we present a distribution-optimization algorithm to compensate for the accuracy loss and improve the network compression efficiency. The proposed method yields state-of-the-art performance on various benchmark networks and datasets, verified by extensive experiments. Our codes are available at: https://github.com/lscgx/FSM.

1 Introduction

The rapid development of convolutional neural networks (CNN) has obtained remarkable success in a wide range of computer vision applications, such as image classification [11, 18, 24], video analysis [5, 21, 30], object detection [6, 8, 39], semantic segmentation [1, 2, 41], etc. The combination of CNN models and IoT devices yields significant economic and social benefits in the real world. However, better performance for a CNN model usually means higher computational complexity and a greater number of parameters, limiting its application in resource-constrained devices. Therefore, model compression techniques are required.

To this end, compressing the existing network is a popular strategy, including tensor decomposition [38], parameter quantification [32], weight pruning [9, 25], structural pruning [28], etc. Moreover, another strategy is to create a new small network directly, including knowledge distillation [16] and compact network design [17]. Among these compression techniques, structural pruning has

⋆ Corresponding author

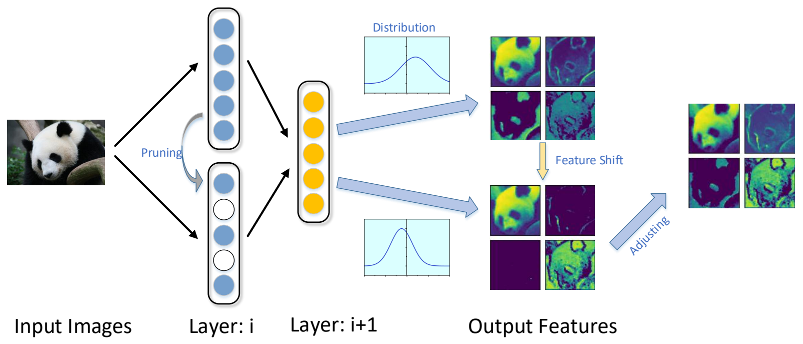


Fig. 1: Diagram of the proposed Feature Shift Minimization method (FSM).

significant performance and is usable for various network architectures. In this paper, we present a channel pruning method, belonging to structural pruning, for model compression.

In the process of channel pruning, some channels that are considered unimportant or redundant are discarded and get a sub-network of the original network. The core of channel pruning lies in the design of filter or feature selection criteria. One idea is to directly use the inherent properties of filters or features, such as statistical properties, matrix properties, etc. For example, Li *et al.* [14, 25] sorting the filters by l_1 norm of filters, and Lin *et al.* [28] prune channels by calculating the rank of features. Another idea is to evaluate the impact of removing a channel on the next layer or the accuracy loss [34]. These criteria, which are based on different properties, succeeded in model compression, but the performance faded in some conditions. Because it is difficult for a single criterion to take into account all factors such as feature size or feature dimension at the same time. So, many pruning criteria are suboptimal because they only work well locally, not globally. As shown in Fig. 2, in some cases, random pruning can even outperform well-designed methods.

To compensate for the limitations of a single criterion, some multi-criteria approaches have been proposed recently. For example, in [27], a collaborative compression method was proposed for channel pruning, which uses the information of compression sensitivity for each layer. Similarly, in [46], the structural redundancy information of each layer is used to guide pruning. Although these methods take into account the influence of other factors on pruning efficiency, they make no improvements to the criteria and require additional computational consumption.

In this paper, we propose a new channel pruning method by minimizing feature shift. We define feature shift as the changes in the distribution of features in a layer due to pruning. As shown in Fig. 1, the feature details changed or disappeared due to pruning, but when the distribution of the features is properly adjusted, the disappeared feature details emerge again. This experiment was done on ImageNet, using the ResNet-50 model. This suggests that the loss of feature detail, caused by pruning some channels, may not have really disappeared, which is related to the feature activation state. Even those channels that are considered unimportant may cause changes in the distribution of features, which in turn lead to under-activation or over-activation of features. Therefore, it is reasonable to minimize the feature shift for model compression. Moreover, we mathematically demonstrate that feature shift is an important factor for channel pruning and analyze the effect of the activation functions, in Section 3.1.

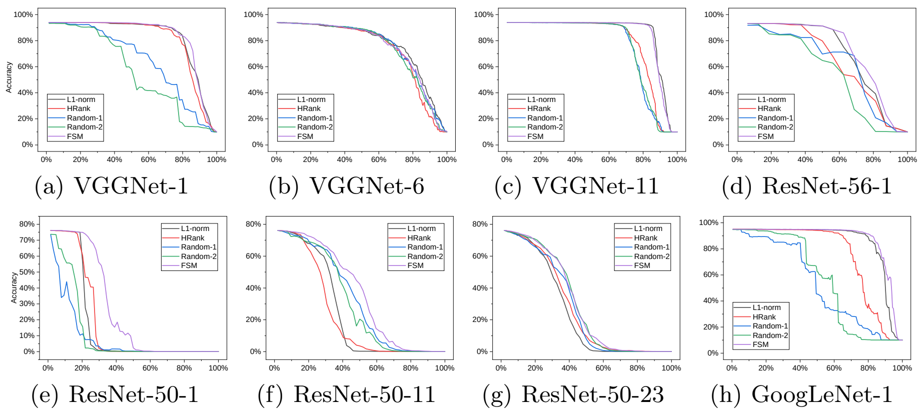
Computing feature shift directly is not an easy task, as it requires traversing the entire training dataset. To address this issue, we propose an evaluation method to measure the feature shift, which combines information from both filters and features. Based on this, we compress models by evaluating the feature shift when each channel is removed, which performs well in different feature dimensions. In particular, the proposed pruning method does not require sampling the features and uses only the parameters of the pre-trained models. In addition, to prove that the feature shift occurs in the pruning process, we design a distribution-optimization algorithm to adjust the pruned feature distribution, which significantly recovers the accuracy loss, as will be discussed in Section 3.4.

To summarize, our main contributions are as follows:

1) We investigate the relationship between compression rate, layer depth, and model accuracy with different pruning methods. It reveals that the feature shift is an important factor affecting pruning efficiency.
2) We propose a novel channel pruning method, Feature Shift Minimization (FSM), which combines information from both features and filters. Moreover, a distribution-optimization algorithm is designed to accelerate network compression.
3) Extensive experiments on CIFAR-10 and ImageNet, using VGGNet, MobileNet, GoogLeNet, and ResNet, demonstrate the effectiveness of the proposed FSM.

2 Related Work

Channel pruning. Channel pruning has been successfully applied to the compression of various network architectures, which lies in the selection of filters or features. Some previous methods use the intrinsic properties of filters to compress models. For example, [14, 25] prune filters according to their l_i norms. In [28], the rank of the features is used to measure the information richness of the features and retain the features with high information content. Some other approaches go beyond the limits of a single layer. For example, Chin *et al.* [3] removes filters by calculating the global ranking. Liu *et al.*[33] analyzes the relationship between the filters and BN layers and designs a scale factor to represent the importance. In addition, computing the statistical information of the next layer and minimizing the feature reconstruction error is also a popular idea [15, 34]. Different from methods that are based on only a single filter or feature, [43,



(a) VGGNet-1 (b) VGGNet-6 (c) VGGNet-11 (d) ResNet-56-1

(e) ResNet-50-1 (f) ResNet-50-11 (g) ResNet-50-23 (h) GoogLeNet-1

Fig. 2: Analysis of the relationship between compression rate, layer depth, accuracy, and methods. Names such as "ResNet-50-1" mean experiment on the 1st layer of ResNet-50. For each subfigure, the horizontal axis is the compression rate, and the vertical axis is the accuracy.

46, 48] remove redundant parts of a model by measuring the difference between the filters or features.

Effect of Layer-Depth. For different layers, the feature size, dimensions, and redundancy are different. Recent works have found that different layers of a model have different sensitivities to the compression rate. Wang *et al.* [46] proposes that the structural redundancy of each layer are different, which plays a key role in the pruning process. In layers with high structural redundancy, more filters can be safely discarded with little loss of accuracy. Li *et al.* [27] represents that the compression sensitivity of each layer is distinct, and it is used to guide pruning. In addition, He *et al.* [12] propose that a pruning criterion does not always work well in all layers, and they achieve better performance by using different pruning criteria on different layer-depths. Earlier methods [13, 19, 51] required iterative tuning of the compression ratio to achieve better model accuracy. It is tedious and time-consuming. Considering the difference between different layers can greatly improve the pruning efficiency.

We investigate the trend of accuracy when adopting different compression rates in different layer-depths. Two methods (L1 and HRank) are investigated, which sort filters and features, respectively. Based on this, we propose a new perspective to explore the redundancy of neural networks.

3 Methodology

In this section, we analyze the process of filter pruning and point out that the distribution of features is changed after pruning. Then we propose a novel pruning criterion that minimizes the feature shift. Unlike some heuristic pruning criteria, there is much theoretical analysis to support the proposed pruning algorithm. Moreover, a distribution optimization algorithm is presented to restore the model accuracy.

3.1 Filter Pruning Analysis

Filter pruning aims to remove those relatively unimportant filters, which evaluate the importance of the filter or feature by a specific importance criterion. However, we found that these criteria do not work well in all layers. We conduct experiments on CIFAR-10 with $l1$-norm and HRank, which sort filters and features, respectively. In addition, we also test the performance of random pruning in different layers. As shown in Fig. 2, two phenomena are obvious: a)As the compression rate increases, the drop in accuracy becomes more obvious. When the compression rate exceeds a critical value, the accuracy declines dramatically. b) For high-dimensional features, sorting filters or feature maps by these criteria isn't substantially better than random pruning.

The phenomena reveal that: a) Exist other factors damage the accuracy, and their impact grows with the compression rate. b) For low-dimensional features, pruning criteria are effective. However, for high-dimensional features, random pruning also performs well. This indicates that the gap between the importance scores of filters/features becomes smaller as the dimensionality increases.

To clarify why these phenomena occur, we mathematically analyze the changes that occur in one feature map before and after pruning. A typical inference process in one layer is Input-Conv-BN[20]-ReLU[7]-Output.

Many benchmark network architectures, such as ResNet, DenseNet, and GoogLeNet, are based on it. In particular, we let $X_i = (x_i^{(1)} \cdots x_i^{(d_i)})$ represent the input for i-th layer of one CNN model, where d_i stand for the number of the dimensions. Firstly, we normalize each dimension on BN layer

$$\widehat{x}_i^{(k)} = \frac{x_i^{(k)} - E[x_i^{(k)}]}{\sqrt{Var[x_i^{(k)}]}},\ y_i^{(k)} = \gamma_i^{(k)} \widehat{x}_i^{(k)} + \beta_i^{(k)}, \tag{1}$$

where E is the expectation, Var is the variance, and k stand for the k-th dimension. We let $Y_i = (y_i^{(1)} \cdots y_i^{(d_i)})$ denotes the output of the BN layer. The output of the k-th dimension has a mean of $\beta_i^{(k)}$ and a standard deviation of $\gamma_i^{(k)}$, which can be directly obtained by the pre-trained models. Then, Y_i is passed on to the ReLU layer. For each of the values in Y_i, the ReLU activation function is applied to it to get thresholded values. The output can be formulated as:

$$\widehat{y}_i^{(k)} = ReLU(y_i^{(k)}) = max(0, y_i^{(k)}). \tag{2}$$

Obviously, the distribution of $\widehat{y}_i^{(k)}$ changes after passing through the ReLU layer. The output expectation

$$E[\widehat{y}_i^{(k)}] = E[max(0, y_i^{(k)})], \tag{3}$$

and the input of the $(i+1)$-th layer

$$x_{i+1}^{(k)} = (\widehat{y}_i^{(1)} \cdots \widehat{y}_i^{(d_i)}) \cdot w_{i+1} \neq (\widehat{y}_i^{(1)} \cdots \widehat{y}_i^{(\widehat{d}_i)}) \cdot w_{i+1},\ s.t.\ \widehat{d}_i \leq d_i, \tag{4}$$

where w represents the weights and $\widehat{d}_i$ stands for the i-th dimension after pruning. As can be seen, the distribution of input x of one channel will shift when pruning some channels in the previous layer. We assume that the shifted expectation of $x_i^{(k)}$ as $\widehat{E}[x_i^{(k)}]$, and the shifted variance as $\widehat{Var}[x_i^{(k)}]$. More, we let $\varDelta_E = E[x_i^{(k)}] - \widehat{E}[x_i^{(k)}]$. So we can reformulate the Eq. (1) as:

$$y_i^{(k)} = \gamma_i^{(k)} \widehat{x}_i^{(k)} + \beta_i^{(k)} = \gamma_i^{(k)} \frac{x_i^{(k)} - E[x_i^{(k)}]}{\sqrt{Var[x_i^{(k)}]}} + \beta_i^{(k)} \tag{5}$$

$$= \gamma_i^{(k)} \frac{x_i^{(k)} - (\widehat{E}[x_i^{(k)}] + \varDelta_E)}{\sqrt{Var[x_i^{(k)}]}} + \beta_i^{(k)} \tag{6}$$

$$= \frac{\sqrt{\widehat{Var}[x_i^{(k)}]}}{\sqrt{Var[x_i^{(k)}]}} \gamma_i^{(k)} \frac{x_i^{(k)} - \widehat{E}[x_i^{(k)}]}{\sqrt{\widehat{Var}[x_i^{(k)}]}} + \beta_i^{(k)} - \gamma_i^{(k)} \frac{\varDelta_E}{\sqrt{Var[x_i^{(k)}]}} \tag{7}$$

The mean of $y_i^{(k)}$ is $\beta_i^{(k)} - \gamma_i^{(k)} \frac{\varDelta_E}{\sqrt{Var[x_i^{(k)}]}}$, and the standard deviation shifted to $\frac{\sqrt{\widehat{Var}[x_i^{(k)}]}}{\sqrt{Var[x_i^{(k)}]}} \gamma_i^{(k)}$.

As a result, the expectation $E[\widehat{y}_i^{(k)}]$ is shifted after pruning, which directly affects the input to the next layer. At the same time, due to the ReLU activation function, some values that were not activated before being activated, or the activated values are deactivated. We refer to the change in the distributions after the ReLU layer in the process of pruning as *feature shift*.

Table 1: Experiments on ResNet-50-23. FSM-R means pruning in reverse order.

Rate	20%	30%	40%
FSM(%)	70.42	56.50	32.20
FSM-R(%)	32.89	8.128	1.62

Predictably, at high compression rates, the magnitude of the feature shift may be greater compared to low rates, and the drop in accuracy may also be greater. To prove it, we consider the feature shift of each channel as the selection criterion in pruning. We prune channels by their expectation of the feature shift. The channels with the least shift are discarded first. As shown in Fig. 2, many experiments show that the decreasing trend of the accuracy of the proposed method is less pronounced than other prevalent methods. Our method can maintain higher accuracy compared to others at the same compression rate (e.g., 11.3%/L1 vs. 8.79%/HRank vs. 66.52%/FSM with a ratio 40% on ResNet-50-11). It reveals that the feature shift is one of the critical factors for performance degradation. In particular, when we prune the model in reverse order, the accuracy catastrophically decreases, as shown in Table 1. It demonstrates that the greater the magnitude of the feature shift, the greater the loss of accuracy. The decrease in accuracy has a positive correlation with the feature shift. So, it is reasonable to use the feature shift to guide pruning. Features that exhibit less feature shift are relatively unimportant.

3.2 Evaluating the Feature Shift

As demonstrated in Eq. (7), we need to calculate $\widehat{E}[x_i^{(k)}]$, which stands for the expectation of the k-th dimension in the i-th layer. It is difficult and time-consuming to calculate it directly over the entire training dataset. We present an approximation method to estimate the expectation of features and prove the feasibility in Section 5.2. Specifically, we first expand the $\widehat{E}[x_i^{(k)}]$ as:

$$\begin{aligned}\widehat{E}[x_i^{(k)}] &= \widehat{E}[(\widehat{y}_{i-1}^{(1)} \cdots \widehat{y}_{i-1}^{(\widehat{d}_{i-1})}) \cdot w_i^{(k)}] &(8)\\ &= \widehat{E}[\widehat{y}_{i-1}^{(1)}] * w_i^{(k,1)} + \cdots + \widehat{E}[\widehat{y}_{i-1}^{(\widehat{d}_{i-1})}] * w_i^{(k,\widehat{d}_{i-1})}, &(9)\end{aligned}$$

where $\widehat{d}_{i-1}$ stands for the $(i-1)$-th dimension after pruning, and $\widehat{d}_{i-1} \leq d_{i-1}$. Moreover, according to Eq. (1), we get

$$\widehat{E}[\widehat{y}_{i-1}^{(k)}] = E[max(0, y_{i-1}^{(k)})] \approx \int_0^{\infty} x \frac{1}{\gamma_{i-1}^{(k)}\sqrt{(2\pi)}} e^{-\frac{(x-\beta_{i-1}^{(k)})^2}{2(\gamma_{i-1}^{(k)})^2}} dx. \quad (10)$$

Without computing the expectation over the entire train dataset, the approximation of $\widehat{E}[x_i^{(k)}]$ can be obtained by combining Eq. (8) and Eq. (10). Then, according to Eq. (7), we get the shifted expectation and the standard deviation of the features.

3.3 Filter Selection Strategy

The pruning strategy of a layer can be regarded as minimizing the sum of the feature shift of all channels of the pruned model. For a CNN model with N layers, the optimal pruning strategy can be expressed as an optimization problem:

$$arg\,min \sum_{i=1}^{N} \sum_{k=1}^{\widehat{d}_i} \left| \Delta_{E_i^{(k)}} \right| = arg\,min \sum_{i=1}^{N} \sum_{k=1}^{\widehat{d}_i} \left| E[x_i^{(k)}] - \widehat{E}[x_i^{(k)}] \right|, \quad (11)$$

where $\widehat{d}_i$ is the dimension of each layer after pruning. During the pruning of a layer, we evaluate the effect of each feature for the feature shift of the next layer, as follows:

$$\delta(x_i^{(k)}) = \sum_{j=1}^{d_{i+1}} \left| w_{i+1}^{(j,k)} * E[y_i^{(k)}] \right|, \quad (12)$$

where $w_{i+1}^{(j,k)}$ denotes the k-th vector of the $(i+1)$-th layer's j-th dimension. $\delta(x_i^{(k)})$ represents the sum of feature shift of one output feature to all channels in the $(i+1)$-th layer. Then, we sort all features according to the values $\delta(\cdot)$. Features with low values are considered unimportant and will be discarded in preference.

3.4 Distribution Optimization

For a pruned model, we present a simple distribution optimization method to recover accuracy. As presented in Eq. (7), the distribution of $y_i^{(k)}$ was changed due to the shift of $E[x_i^{(k)}]$ and $Var[x_i^{(k)}]$ after pruning. Hence, we can adjust them to $\widehat{E}[x_i^{(k)}]$ and $\widehat{Var}[x_i^{(k)}]$ reduce the impact of the shift.

In Eq. (8), we present a evaluation method for $\widehat{E}[x_i^{(k)}]$. Let λ represent the evaluation error, $\lambda = \widehat{E}[x_i^{(k)}] \,/\, E[x_i^{(k)}]$, where the λ is computed on unpruned model. After pruning, we set

$$E[x_i^{(k)}] = \widehat{E}[x_i^{(k)}] \,/\, \lambda \tag{13}$$

For $\widehat{Var}[x_i^{(k)}]$, it is difficult to evaluate, or the error is large. We experimentally show that set

$$Var[x_i^{(k)}] = \frac{\widehat{d}_i}{d_i} \times Var[x_i^{(k)}], \tag{14}$$

performs well in most layers, especially in the deeper layers.

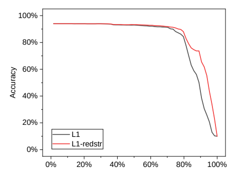
One possible explanation is that the differences between high-dimensional features are not obvious, and the statistical properties are approximate. Random pruning in deep layers produces great results, as shown in Fig. 2, supporting this explanation. Furthermore, as shown in Fig. 3, extensive experiments on different architectures reveal that the proposed distribution optimization strategy is effective and proves that feature shift occurs during network pruning. Obviously, the algorithm is plug-and-play and may also be combined with other pruning methods.

3.5 Pruning Procedure

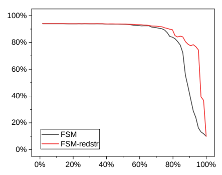
The pruning procedure is summarized as follows:

1) For each channel $x_i^{(k)}$ of one layer, we first calculate its output expectation $E[y_i^{(k)}]$.
2) Then, we calculate the feature shift $\delta(x_i^{(k)})$ caused by channel $x_i^{(k)}$, according to Eq. (12).
3) Sort all channels through $\delta(\cdot)$ and discard those channels with small values.
4) After pruning, using Eq. (13) and Eq. (14) to recover accuracy.
5) Fine-tuning the model one epoch, and back to the first step to prune the next layer.

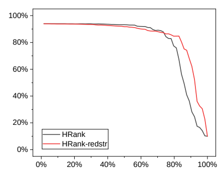
After all layers have been pruned, we train the pruned model for some epochs.



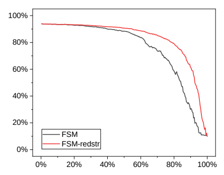
(a) $l1$-norm-1



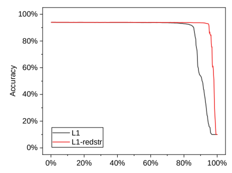
(b) FSM-1



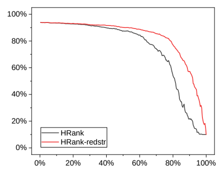
(c) HRank-1



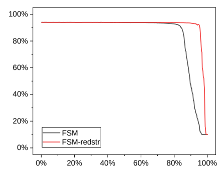
(d) FSM-6



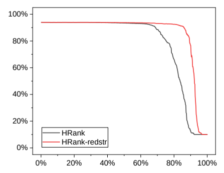
(e) $l1$-norm-11



(f) HRank-6



(g) FSM-11



(h) HRank-11

Fig. 3: Comparison of the accuracy before and after using distribution optimization on VGGNet. Similarly, the horizontal axis represents the compression rate, and the vertical axis is the accuracy. Names such as "HRank-1" mean to adopt the HRank method on the 1st layer of VGGNet. The results suggest that the feature shift is a critical factor that damages model accuracy.

4 Experiments

In this section, in order to verify the effectiveness of the proposed FSM, we conduct extensive experiments on CIFAR-10 [23] and ImageNet [24]. Prevalent models, such as ResNet[11], GoogLeNet[44], MobileNet-V2[40], and VGGNet[42], are adopted to test the performance. All experiments are running on Pytorch 1.8 [36] under Intel i7-8700K CPU @3.70GHz and NVIDIA GTX 1080Ti GPU.

4.1 Implementation Details

For all models, the pruning process is performed in two steps:1) pruning the filters layer by layer and fine-tune 1 epoch for each layer. 2) After pruning, we train the model for some epochs using the stochastic gradient descent algorithm (SGD) with momentum 0.9. For VGGNet and GoogLeNet on CIFAR-10, we train the model for 200 epochs, in which the initial learning rate, batch size, and weight decay are set to 0.01, 128, and 0, respectively. The learning rate is divided by 10 at epochs 100 and 150. For ResNet-56 on CIFAR-10, we train 300 epochs and set the weight decay to 0.0005. The learning rate is divided by 10 at epochs 150 and 225. For ResNet-50, we train the model for 120 epochs with an initial learning rate of 0.01, and the learning rate is divided by 10 at epochs 30, 60, and 90. The batch size is set to 64. For MobileNet-V2, we fine-tune the pruned model for 150 epochs, and the learning rate is decayed by the cosine annealing scheduler with an initial learning rate of 0.01. The weight decay is set to 4×10^{-5}, following the original MobileNet-V2 paper setup. We train all the models three times and report the mean.

Table 2: Comparison with other prevalent pruning methods on CIFAR-10

Model	Method	Baseline Acc.	Top-1 Acc.	FLOPs ↓	Param. ↓
ResNet-56	SFP[13]	93.59%	93.35%	52.6%	-
	CCP[37]	93.50%	93.42%	52.6%	-
	HRank[28]	93.26%	92.17%	50.0%	42.4%
	NPPM[4]	93.04%	93.40%	50.0%	-
	DHP[26]	-	93.58%	49.0%	41.6%
	SCP[22]	93.69	93.23%	51.5%	-
	FSM(ours)	**93.26**%	**93.63**%	**51.2**%	**43.6**%
	FSM(ours)	**93.26**%	**92.76**%	**68.2**%	**68.5**%
VGGNet	L1[25]	93.25%	93.40%	34.2%	63.3%
	GAL[29]	93.96%	92.03%	39.6%	77.2%
	HRank[28]	93.96%	93.42%	53.7%	82.9%
	EEMC[50]	93.36%	93.63%	56.6%	-
	FSM(ours)	**93.96**%	**93.73**%	**66.0**%	**86.3**%
	FSM(ours)	**93.96**%	**92.86**%	**81.0**%	**90.6**%
GoogLeNet	L1[25]	-	94.54%	32.9%	42.9%
	GAL[29]	95.05%	93.93%	38.2%	49.3%
	HRank[28]	95.05%	94.53%	54.6%	55.4%
	FSM(ours)	**95.05**%	**94.72**%	**62.9**%	**55.5**%
	FSM(ours)	**95.05**%	**94.29**%	**75.4**%	**64.6**%

We set the compression rate of each layer according to its accuracy drop curve, which maintains the model accuracy to the greatest extent possible. The details are provided in the supplementary.

4.2 Results and Analysis

CIFAR-10. In Table 2, we compare the proposed FSM method with other prevalent algorithms on VGGNet, GoogLeNet, and ResNet-56. Our method achieves a significantly compression efficiency on reductions of FLOPs and parameters, but with higher accuracy. For example, compared with HRank, FSM yields a better accuracy (93.73% vs. 93.42%) under a greater FLOPs reduction (66.0% vs. 53.7%) and parameters reduction (86.3% vs. 82.9%). For ResNet-56 on CIFAR-10, FSM prunes 51.2% of FLOPs and 43.6% of parameters, but the accuracy improved by 0.37%. NPPM and DHP have been proposed recently and they have a great performance in network compression. Compared with NPPM, our method shows a better performance, and with a higher FLOPs reduction. Compared with DHP, under similar accuracy, FSM performs better at the reduction of FLOPs (51.2% vs. 49.0%) and parameters (43.6% vs. 41.6%). For GoogLeNet, FSM outperforms HRank and GAL in all aspects. With over 62% FLOPs reduction, FSM still maintains 94.72% accuracy.

Furthermore, we verified the performance of FSM at high compression rates. For instance, with more than 90% of the parameters discarded, FSM reduces

Table 3: Comparison with other prevalent pruning methods on ImageNet

Model	Method	Top-1 Acc.	Top-5 Acc.	*Δ* Top-1	*Δ* Top-5	FLOPs ↓
ResNet-50	DCP[51]	74.95%	92.32%	-1.06%	-0.61%	55.6%
	CCP[37]	75.21%	92.42%	-0.94%	-0.45%	54.1%
	Meta[31]	75.40%	-	-1.20%	-	51.2%
	GBN[49]	75.18%	92.41%	-0.67%	-0.26%	55.1%
	BNFI[35]	75.02%	-	-1.29%	-	52.8%
	HRank[28]	74.98%	92.44%	-1.17%	-0.54%	43.8%
	SCP[22]	75.27%	92.30%	-0.62%	-0.68%	54.3%
	SRR-GR[46]	75.11%	92.35%	-1.02%	-0.51%	55.1%
	GReg[45]	75.16%	-	-0.97%	-	56.7%
	FSM(ours)	**75.43%**	**92.45%**	**-0.66%**	**-0.53%**	**57.2%**
MobileNet-V2	CC[47]	70.91%	-	-0.89%	-	30.7%
	BNFI[35]	70.97%	-	-1.22%	-	30.0%
	FSM(ours)	**71.18%**	**89.81%**	**-0.70%**	**-0.48%**	**30.6%**

the accuracy by only 1.1% on VGGNet. The same results can be observed on ResNet-56 and GoogLeNet.

ImageNet 2012. ImageNet 2012 has over 1.28 million training images and 50,000 validation images divided into 1,000 categories. ResNet-50, compared to VGGNet and ResNet-56, has a larger feature size. To verify the applicability of the proposed FSM, on ResNet-50, we test the performance at different compression rates and different layer-depth, by comparing it with L1 and HRank, as shown in Fig. 2. The results show that FSM successfully picks out the more important channels, even for large-size features. In contrast, L1 and HRank performed poorly, even less well than the random method. In most cases, the proposed FSM shows the best performance.

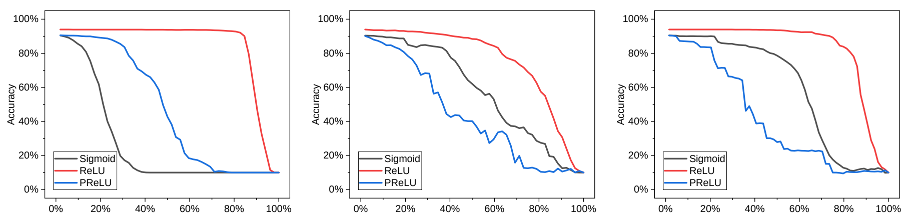
In Table 3, we compare the proposed FSM method with other prevalent methods on ResNet-50. Our method achieves significant performance, reducing the FLOPs by more than 57% with only 0.66% loss in Top-1 accuracy. Compared with HRank, our method shows a better performance (75.43% vs. 74.98%), and with a higher FLOPs reduction (57.2% vs. 43.8%). Compared with GBN and DCP, under similar FLOPs reduction of about 55%, FSM obtains better accuracy. Our method reduces FLOPs by 6% more than MetaPruning under similar Top-1 accuracy. More comparison details can be found in Table 3.

MobileNet-V2 is a memory-efficient model that is suitable for mobile devices. However, to further compress it is challenging while maintaining the model accuracy. When pruning around 30% of FLOPs, FSM achieves better performance (71.18%) than CC (70.91%) and BNFI (70.97%) on Top-1 accuracy. The results demonstrate that FSM has excellent performance on the large-scale ImageNet dataset and shows higher applicability.

The Speedup on GPU. To show the practical speedup of our method in real scenarios, we measure the inference time by time/images on the NVIDIA

Table 4: The practical speedup for the pruned model over the unpruned model. "T": inference time. "S": practical speedup on GPU (NVIDIA 1080Ti).

Model	FLOPs(%) ↓	Param.(%) ↓	T(ms)	S(×)
GoogLeNet	62.9	55.5	5.03(±0.01)	1.47×
	75.4	64.7	4.58(±0.02)	1.62×
ResNet-50	57.2	42.8	13.81(±0.04)	1.30×



(a) layer-1 (b) layer-6 (c) layer-11

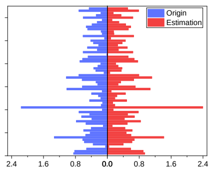
Fig. 4: The effect of different activation functions on channel pruning for VGG. For each subfigure, the horizontal axis is the compression rate, and the vertical axis is the accuracy.

1080Ti GPU. As shown in Table 4, the proposed FSM achieves 1.62× and 1.30× speedup with batch size 64 on GoogLeNet and ResNet-50, respectively.

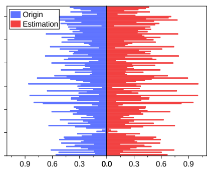
5 Ablation Study

5.1 Effect of the Activation Function

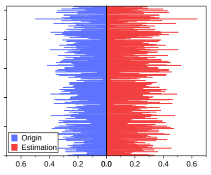
In Section 3.1, we illustrate that the activation function affects the distribution of the features. In channel pruning, the threshold property of the ReLU activation function results in changes in the feature activation state. In this section, we experiment with the performance of three popular activation functions (Sigmoid, ReLU, and PReLU [10]) on different layers in VGGNet. As shown in Fig. 4, the trend of accuracy with Sigmoid and PReLU changes smoother than ReLU. Although they do not have the problem of gradient plunges, there is a large loss of accuracy at small compression rates. In contrast, ReLU achieves better performance at most rates. In particular, $PReLU(x) = max(0, x) + a \times min(0, x)$, and $Sigmoid(x) = \sigma(x) = \frac{1}{1+e^{-x}}$. Different from the ReLU, their gradient is greater than 0 when $x < 0$. Some values should reach 0 under ReLU, but return a negative value under PReLU, which leads to more changes in features. This explains why PReLU loses more accuracy when the feature shift occurs compared to Sigmoid and ReLU. In general, models with ReLU activation functions are more tolerable for channel pruning than Sigmoid and PReLU.



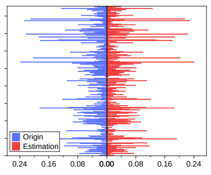
(a) VGGNet-1



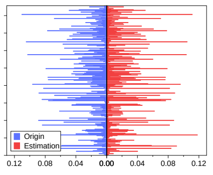
(b) VGGNet-3



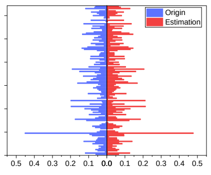
(c) VGGNet-6



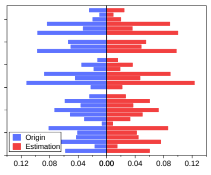
(d) ResNet-50-3



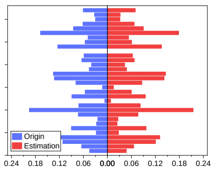
(e) ResNet-50-7



(f) GoogLeNet-3

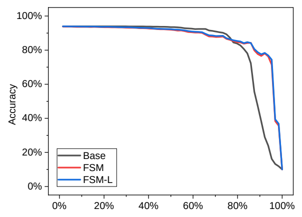


(g) GoogLeNet-5

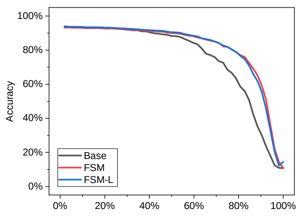


(h) GoogLeNet-6

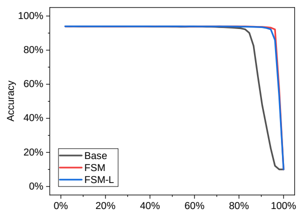
Fig. 5: Comparison of the original and estimated distributions. For each subfigure, the horizontal axis is the expectation, and the vertical axis represents different features. The red indicates the distribution estimation, and the blue is the original distribution. More comparisons are provided in the supplementary.



(a) layer-1



(b) layer-6



(c) layer-11

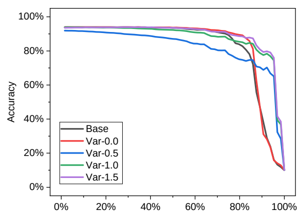
Fig. 6: The effect of evaluation error λ for channel pruning on VGGNet. For each subfigure, the FSM-L means that the expectation evaluation values are amended with λ.

5.2 Error of the Feature Shift Evaluation

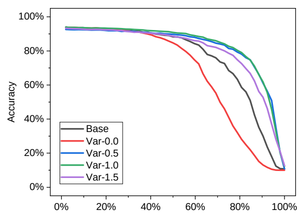
The error of the feature distribution evaluation is shown in Fig. 5. The differences between the proposed evaluation method and the true distribution are compared at different layer depths. It can be seen that the evaluation error is in an acceptable range. In addition, our experiments show that expectation and layer depth are negatively correlated. For very small expectation values, our method can still estimate them accurately.

5.3 Effect of the Evaluation Error λ

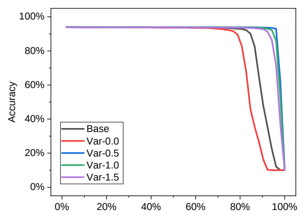
In Section 3.4, we present the estimation error λ to rectify the evaluated distribution. As shown in Fig. 6, the method using λ generates a better performance



(a) layer-1



(b) layer-6



(c) layer-11

Fig. 7: Comparisons of the variance estimation with different magnitude coefficients on VGGNet. Names such as "Var-1.5" stand for reduce the variance $Var[x_i^{(k)}]$ by $(1.5 \times \frac{\widehat{d_i}}{d_i})\%$. Similarly, the horizontal axis represents the compression rate, and the vertical axis is the accuracy.

at low compression rates. Also, no additional accuracy loss is caused at high compression rates.

5.4 Effect of the Variance Adjustment Coefficients

In Section 3.4, we empirically reduce the variance $Var[x_i^{(k)}]$ by $\frac{\widehat{d_i}}{d_i}\%$ because it is difficult to calculate it directly. As shown in Fig. 7, we adopt different magnitude coefficients to adjust the variances of pruned models. The results show that the VAR-1.0, adopted in our work, achieves the best performance in almost all cases.

6 Conclusions

In this paper, we propose a novel channel pruning method by minimizing feature shift. We first prove the existence of feature shift mathematically, inspired by the investigation of accuracy curves in channel pruning. The feature shift is used to explain why the accuracy plummets when the compression rate exceeds a critical value. Based on this, an efficient channel pruning method (FSM) is proposed, which performs well, especially at high compression rates. Then, we present a feature shift evaluation method that does not traverse the training data set. In addition, a distribution optimization method is designed to improve the efficiency of model compression and is plug-and-play. Extensive experiments and rigorous ablation studies demonstrate the effectiveness of the proposed FSM for channel pruning. In the future, we will further research the impact of multi-branch architecture on the FSM. Also, the combination with other pruning methods is worth trying.

Acknowledgements The work was supported by National Natural Science Foundation of China (Grant Nos. 61976246 and U20A20227), Natural Science Foundation of Chongqing (Grant No. cstc2020jcyj-msxm X0385)

References

1. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K.P., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**, 834–848 (2018)
2. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. ArXiv **abs/1802.02611** (2018)
3. Chin, T.W., Ding, R., Zhang, C., Marculescu, D.: Towards efficient model compression via learned global ranking. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1515–1525 (2020)
4. Gao, S., Huang, F., Cai, W.T., Huang, H.: Network pruning via performance maximization. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 9266–9276 (2021)
5. Girdhar, R., Tran, D., Torresani, L., Ramanan, D.: Distinit: Learning video representations without a single labeled video. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 852–861 (2019)
6. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition pp. 580–587 (2014)
7. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks. In: AISTATS (2011)
8. Guo, J., Han, K., Wang, Y., Zhang, C., Yang, Z., Wu, H., Chen, X., Xu, C.: Hit-detector: Hierarchical trinity architecture search for object detection. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 11402–11411 (2020)
9. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. ArXiv **abs/1506.02626** (2015)
10. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2015 IEEE International Conference on Computer Vision (ICCV) pp. 1026–1034 (2015)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016)
12. He, Y., Ding, Y., Liu, P., Zhu, L., Zhang, H., Yang, Y.: Learning filter pruning criteria for deep convolutional neural networks acceleration. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2006–2015 (2020)
13. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. ArXiv **abs/1808.06866** (2018)
14. He, Y., Liu, P., Wang, Z., Yang, Y.: Pruning filter via geometric median for deep convolutional neural networks acceleration. ArXiv **abs/1811.00250** (2018)
15. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 1398–1406 (2017)
16. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. ArXiv **abs/1503.02531** (2015)
17. 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 **abs/1704.04861** (2017)

18. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2261–2269 (2017)
19. Huang, Z., Wang, N.: Data-driven sparse structure selection for deep neural networks. ArXiv **abs/1707.01213** (2018)
20. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. ArXiv **abs/1502.03167** (2015)
21. Jiang, L., Xu, M., Liu, T., Qiao, M., Wang, Z.: Deepvs: A deep learning based video saliency prediction approach. In: ECCV (2018)
22. Kang, M., Han, B.: Operation-aware soft channel pruning using differentiable masks. In: ICML (2020)
23. Krizhevsky, A.: Learning multiple layers of features from tiny images (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**, 84 – 90 (2012)
25. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. ArXiv **abs/1608.08710** (2017)
26. Li, Y., Gu, S., Zhang, K., Gool, L.V., Timofte, R.: Dhp: Differentiable meta pruning via hypernetworks. ArXiv **abs/2003.13683** (2020)
27. Li, Y., Lin, S., Liu, J., Ye, Q., Wang, M., Chao, F., Yang, F., Ma, J., Tian, Q., Ji, R.: Towards compact cnns via collaborative compression. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6434–6443 (2021)
28. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1526–1535 (2020)
29. Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., Doermann, D.S.: Towards optimal structured cnn pruning via generative adversarial learning. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2785–2794 (2019)
30. Lin, T., Liu, X., Li, X., Ding, E., Wen, S.: Bmn: Boundary-matching network for temporal action proposal generation. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 3888–3897 (2019)
31. Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K., Sun, J.: Metapruning: Meta learning for automatic neural network channel pruning. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 3295–3304 (2019)
32. Liu, Z., Shen, Z., Savvides, M., Cheng, K.T.: Reactnet: Towards precise binary neural network with generalized activation functions. ArXiv **abs/2003.03488** (2020)
33. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 2755–2763 (2017)
34. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 5068–5076 (2017)
35. Oh, J., Kim, H., Baik, S., Hong, C., Lee, K.M.: Batch normalization tells you which filter is important. 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) pp. 3351–3360 (2022)
36. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
37. Peng, H., Wu, J., Chen, S., Huang, J.: Collaborative channel pruning for deep networks. In: ICML (2019)

38. Raja, K.B., Raghavendra, R., Busch, C.: Obtaining stable iris codes exploiting low-rank tensor space and spatial structure aware refinement for better iris recognition. 2019 International Conference on Biometrics (ICB) pp. 1–8 (2019)
39. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**, 1137–1149 (2015)
40. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 4510–4520 (2018)
41. Shelhamer, E., Long, J., Darrell, T.: Fully convolutional networks for semantic segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**, 640–651 (2017)
42. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2015)
43. Singh, P., Verma, V.K., Rai, P., Namboodiri, V.P.: Leveraging filter correlations for deep model compression. 2020 IEEE Winter Conference on Applications of Computer Vision (WACV) pp. 824–833 (2020)
44. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1–9 (2015)
45. Wang, H., Qin, C., Zhang, Y., Fu, Y.: Neural pruning via growing regularization. In: International Conference on Learning Representations (2020)
46. Wang, Z., Li, C., Wang, X.: Convolutional neural network pruning with structural redundancy reduction. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 14908–14917 (2021)
47. Wang, Z., Li, C., Wang, X.: Convolutional neural network pruning with structural redundancy reduction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14913–14922 (2021)
48. Wang, Z., jun Liu, X., Huang, L., Chen, Y., Zhang, Y., Lin, Z., Wang, R.: Model pruning based on quantified similarity of feature maps. ArXiv **abs/2105.06052** (2021)
49. You, Z., Yan, K., Ye, J., Ma, M., Wang, P.: Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. ArXiv **abs/1909.08174** (2019)
50. Zhang, Y., Gao, S., Huang, H.: Exploration and estimation for model compression. 2021 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 477–486 (2021)
51. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.H.: Discrimination-aware channel pruning for deep neural networks. In: NeurIPS (2018)