MaskSparsity: Pruning the Convolutional Neural Networks via Fine-Grained Sparse Regularization

Anonymous CVPR 2021 submission

Paper ID 2672

Abstract

Structural neural network pruning aims to remove the redundant channels in the deep convolutional neural networks (CNNs) by pruning the filters of less importance to the final output accuracy. To reduce the degradation of performance after pruning, many methods utilize sparse regularization loss to produce structured sparsity. In this paper, we analyze these sparsity-training-based methods and find out two problems of these methods. First, the regularization of unpruned channels is unnecessary, which also restricts the network's capacity. Second, the distribution of important filters and unimportant filters are not well distinguishable after sparsity-training. This challenges the selecting of pruning threshold. The consequence is that a part of pruned parameters are still not small enough to reduce the effect of pruning on the network's output. To solve this problem, we propose a novel pruning method with fine-grained sparse regularization, named MaskSparsity. MaskSparsity imposes fine-grained sparse regularization on the specific filters selected by a pruning mask, rather than all the filters of the model. Before the fine-grained sparse regularization of MaskSparity, we run the global sparse regularization to get the pruning mask. MaskSparsity achieves 63.03%-FLOPs reduction on ResNet-110 by removing 60.34% of the parameters, with no top-1 accuracy on CIFAR-10. Moreover, on ILSVRC-2012, MaskSparsity reduces more than 53.76% FLOPs on ResNet-50 by removing 54.06% of the parameters, with only a loss of 0.93% in the top-1 accuracy.

1. Introduction

Convolutional Neural Networks (CNNs) have demonstrated a great success for a variety of computer vision tasks, like image classification [34], detection [29], and semantic segmentation [4]. However, the increasing depth and width of the CNNs also leads to the higher computing resources demands and excessive memory footprint requirements. Typically, the widely used ResNet models [9] have millions of parameters, requiring billions of float point operations (FLOPs), which makes it a great challenge to deploy most state-of-the-art CNNs on edge devices. As recent studies have reveal existence of the significant redundancy in deep CNNs [5], various methods were proposed to compress and accelerate CNNs, including network pruning [25], parameter quantization [19], knowledge distillation [16] and low-rank decomposition [6]. Network pruning is attracting much attention from the researchers. It removes the parameters in the deep CNNs and reduces the required FLOPs and memory footprint.

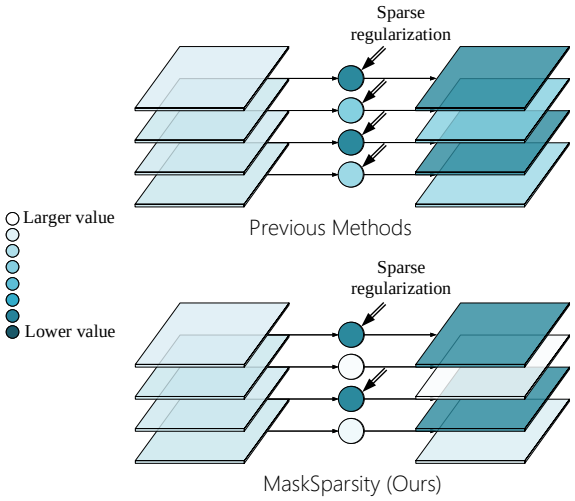

Figure 1. Visual comparison between the previous sparsity-training-based methods and the proposed MaskSparsity method.

Existing network pruning methods can be divided into non-structural and structural pruning methods, according to the difference in the pruning granularity. Non-structural pruning methods delete the unimportant connections in the filter. They would result in irregular network structures, which is not supported on most deep learning libraries and devices. This limits the acceleration and applications of pruned network on general-purpose hardware. Structural

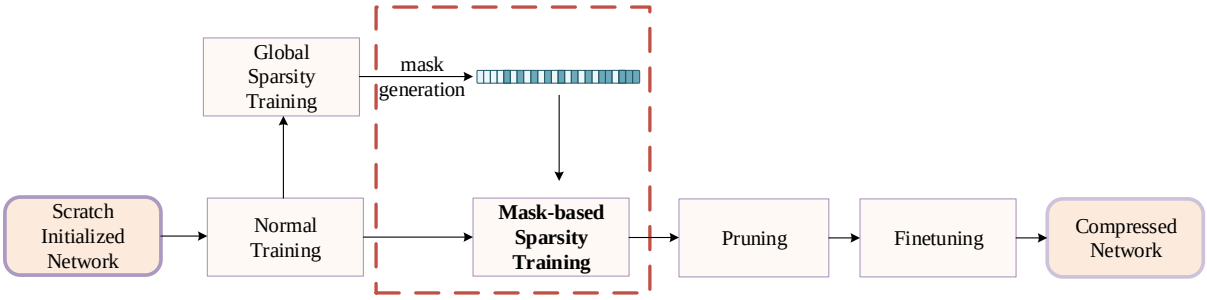

Figure 2. The pipeline of the proposed MaskSparsity method.

pruning removes the whole filters of less importance according to a certain criterion. The pruned structure still consists of regular convolution layers. Therefore, this category of methods are more implementation-friendly, and they have more practical values.

A typical scheme of network pruning consists of three stages: 1) training an over-parameterized model normally; 2) pruning the model according to a certain criterion; and 3) fine-tune the pruned model to reduce the degradation caused by pruning. Some of the existing network pruning methods apply a sparsity training stage after the Step 1). These methods apply sparse regularization on the filter weights of the convolution layers [1, 37] or scaling factors [17, 30] of the batch normalization layers. After the sparsity training, the corresponding filter weights or scaling factors of unimportant channels are considered to be near 0. Then these channels could be safely pruned without affecting the output values of the corresponding layers too much. We call these methods as sparsity-training-based methods.

However, we find that existing sparsity-training-based methods have the following two problems. First, the important channels are preserved after the pruning. However, the weights of these channels are still regularized in the sparsity training stage. We think this prevents the better coverage of the network in the sparsity training stage, which affects the final performance of the fine-tuned pruned network. Second, we find in the existing sparsity-training-based methods, the bad distribution of parameters after sparsity-training challenges the selecting of pruning threshold. The consequence is that a part of pruned parameters are still not small enough to be pruned without affecting the output of the network. In other words, these methods are not able to make the filter weights of these unimportant channels coverage to 0. In other words, these methods are not able to make the filter weights of these unimportant channels coverage to 0. We find this is cause by the full regularization on all channels of each layer, which leads to a lager gradients from the loss function for these unimportant layers. The gradients prevents the filter weight from decaying to 0 by the sparse regularization.

To address the problems mentioned above, we propose a novel sparsity-training-based channel pruning approach, named MaskSparsity. Different from the previous sparsity-training-based methods which with impose regularization on all channels of each layer of the network, MaskSparsity only imposes regularization on the specific channels selected by mask indicating the unimportant channels. In other words, we will only impose regularization on the channels to be pruned, and only prune the channels where regularization is applied. The perfect match between the sparse channels and the pruning channels allows us to minimizes the impact of sparse regularization and maximize the accuracy of the pruned networks, which solves the first problem above. With the the MaskSparsity method, the sparse training is able to push the filter weights of unimportant channels to a nearer-to-zero value, which makes the pruning threshold better to be set.

To summarize, our main contributions are three-fold:

(1) We analyze the previous sparsity-training-based methods in the previous work, which simply impose L1 regularization on all channels of model. We elaborate on many of its problems.

(2) We propose MaskSparsity to solve these problems by more fine-grained sparsity-training.

(3) The extensive experiments on two benchmarks demonstrates the effectiveness and efficiency of MaskSparsity.

2. Related Work

We mainly focus on the structural pruning methods in this paper. In this section, we first review the closely related works, *i.e.* the sparsity-training-based structural pruning methods. After that, we list other structural pruning methods.

2.1. Sparsity-training-based Pruning Methods

To make the network adaptively converge to a sparse structure and alleviate the damage of the pruning process to the network's output, some sparsity-training-based pruning methods are prosed. There are mainly two categories of these methods according to the place where sparse regularization is applied.

The first category of methods are the group sparsity based methods which apply the spare regularizations on the filter weights. Alvarez and Salzmann [1] proposed to use group sparsity regularizer to determine the number of channels of each layer. Wen *et al*. [37] proposed a Structured Sparsity Learning (SSL) method to regularize the structure to obtain a hardware-friendly pruned structure. Alvarez and Salzmann [2] added a low-rank regularizer to improve the pruning performance. Zhang *et al*. proposed a group ordered weighted L1 regularizer. Li and Qi [23] pointed out the correlations between two consecutive layers and proposed to combine one out-channel in current layer and the corresponding inchannel in next layer as a regularization group. Li and Gu proposed the Hinge [24] by combining the filter pruning and low-rank decomposition into the group sparsity training framework.

The second category of methods are the indirect group-sparse methods, which apply the sparse regularization on the scaling factors of each layer. The representative method is NetSlim [30] method, which sparsely regularizes the scaling factors of BN layers to get the sparse structure and remove less important channels. Huang and Wang [17] proposed to add a new scaling factor vector to each layer to apply the sparse regularization. Srinivas and Subramanya [36] proposed to impose sparse constraint over each weight with additional gate variables, and achieve high compression rates by pruning connections with zero gate values. Ye and You [39] proposed to to prune channels with layer dependent thresholds according to the different weight distribution of each layer. [42] develop the norm-based importance estimation by taking the dependency between the adjacent layers into consideration.

These methods all apply the global sparse regularization on the network channels, which over-regularize the important channels. Our MaskSparsity method solves this problem and is able to improve the performance of sparsity-training-based methods to the new state-of-the-art.

2.2. Non-sparsity-training-based Pruning Methods

Recently, many non-sparsity-training-based pruning methods also show good performance. These methods usually evaluate the importance of each channels with a handcraft criterion first. After that, they directly prune the unimportant channels and finetune the network. For instance, Li and Kadav [21] propose to prune filters with smaller L1 norm values in a network. Base on the theory of Geometric Median (GM) [8], He and Liu proposed FPGM[14] to prune the filters with the most replaceable contribution. Inspired by the discovery that the average rank of multiple feature maps generated by a single filter is always the same, Lin et al [26] propose to prune filters by exploring the High Rank of feature maps (HRank). In this paper, we compare the performance of the proposed MaskSparsity with these methods, and shows the good pruning performance.

3. Methodology

3.1. Notations and background

We assume that a convolutional neural network consists of multiple convolutional layers and each convolution layer is followed by a batch normalization (BN) [18] layer. For the i-th convolutional layer, we use $\mathcal{C}^i$ and $\mathcal{D}^i$ to represent the number of its input channels and output channels, and $k^i \times k^i$ represent the kernel size. We use $\mathcal{W}_i = \{\mathbf{w}_1^i, \mathbf{w}_2^i, ..., \mathbf{w}_{\mathcal{D}_i^i}^i\} \in \mathbb{R}^{\mathcal{C}^i \times \mathcal{D}^i \times k^i \times k^i}$ represent the filters of the i-th convolutional layer, and the input feature maps and the output feature maps of filters and be denoted as $\mathcal{I}^i = \{\mathbf{i}_1^i, \mathbf{i}_2^i, ..., \mathbf{i}_{\mathcal{C}_i}^i\} \in \mathbb{R}^{\mathcal{C}_i \times B \times h_i \times w_i}$ and $\mathcal{O}^i = \{\mathbf{o}_1^i, \mathbf{o}_2^i, ..., \mathbf{o}_{\mathcal{D}_i}^i\} \in \mathbb{R}^{\mathcal{D}_i \times B \times h_i{}' \times w_i{}'}$. h_i, w_i, $h_i{}'$ and $w_i{}'$ are the heights and widths of the input and output feature maps respectively. B is the batch size of the input images. The j-th feature map $\mathbf{o}_j^i \in \mathbb{R}^{g \times h_i{}' \times w_i{}'}$ is generated by $\mathbf{w}_j^i$ and $\mathbf{i}_j^i \in \mathbb{R}^{g \times h_i \times w_i}$. For the i-th BN layer with mean μ_i, standard deviation σ_i, learned scaling factor γ_i and bias $\beta_i \in \mathbb{R}^{\mathcal{D}^i}$ following the i-th convolutional layer, regardless of bias of convolutional layer, we have

$$\mathcal{O}^i = ((\mathcal{I}^i \times \mathcal{W}_i) - \mu_i)\frac{\gamma_i}{\sigma_i} + \beta_i \,. \tag{1}$$

Existing sparsity-training-based methods utilizes sparsity regularization loss to produce structured sparsity. In the methods which apply sparse regularization on scaling factors, the training objective function is shown in Equation 2:

$$L_{\text{Sparsity}}(\mathcal{I}^0, y, \mathcal{W}) = L(f(\mathcal{I}^0, \mathcal{W}), y) + \lambda L_{\text{Reg}}(\gamma) \,, \tag{2}$$

where $(\mathcal{I}^0\ ,\ y)$ denote the training samples and the labels, $\mathcal{W}$ denotes the trainable weights, the $L(f(\mathcal{I}^0, \mathcal{W}), y)$ is the objective function of normal training, $L_{\text{Reg}}(\gamma)$ is a sparsity regularization penalty on the scaling factors γ, and λ is the factor of controlling the strength of sparsity. L_{Reg} is usually set as L1 regularization.

Usually, the two kind of gradients of the scaling factors received from the normal training loss L and the sparsity regularization L_{Reg} is against to each other during training. The former aims at improving the model performance, while the latter aims at increasing the structure sparsity. The L_{Reg} loss tend to retain the capacity of the network by pushing the learnable parameters to 0. The unimportant channels' gradients from the normal training loss L is usually small, making them being more likely to be pushed to near 0 by L_{Reg}. For the important channels, the gradients from the L are large enough to prevent the scaling factors being degrades to 0 by L_{Reg}.

3.2. Analysis of Previous Sparsity-training-based Methods

Figure 3 shows the statistical results of two sets of scaling factors (absolute value) collected from the normally-trained and sparsely-trained ResNet-50 [9] on ILSVRC-2012. When we sparsely train the ResNet-50 network, we use $\lambda = 5e-4$ for the regularization loss (L_{Reg} in Equation 2). In Figure 3, the purple histogram shows the distribution of the scaling factors of the normally-trained network, while the green histogram represents that of the sparse-trained network. Figure 3 shows that the scaling factors of the normally-trained network form one peak and those of the sparsely-trained network form two peaks, which is consistent with the bimodal-distribution observation of OT [40]. Obviously, the left scaling-factors peak of the sparsely-trained network represents the unimportant channels and the right peak represents the important channels. With Figure 3, we demonstrate the two problems of existing sparsity-training-based methods, which are mentioned in the Section 1.

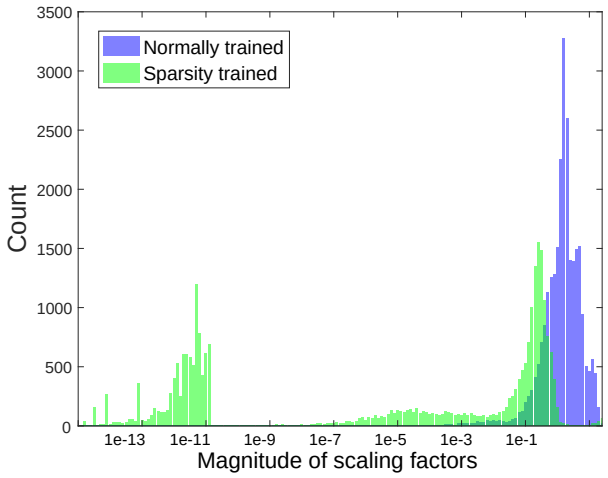

Figure 3. Distribution of scaling factors of ResNet50 before and after global sparsity training.

First, from Figure 3, it can be seen that the right peak of the sparsely-trained network moves to 0 in some order of magnitude, compared with their location in the histogram of normally-trained network. Since the network channels corresponding to the scaling factors of this peak are preserved after the network pruning, this phenomenon indicates that the unpruned important channels are over-regularized. We think this over-regularization prevents the better coverage of the network in the sparsity training stage, which affects the final performance of the fine-tuned pruned network.

Second, there are many in-between bars between the main bodies of the two peaks. This bad distribution of scaling factors challenges the threshold selecting. The channels of the in-between ares is difficult to be classified as important or not. This challenges the selecting of a optimal pruning threshold. With a sub-optimal threshold, some pruned channels may still be important for the network.

In this paper, we think the two problems are caused by the uniformly applied sparse regularization on all channels, regardless of the importance of each individual channels. Therefore, we propose a fine-grained sparsity training method which only apply the sparse regularization on the unimportant channels to keep a maximum representation ability of the important channels.

Algorithm 1 Algorithm Description of MaskSparsity

Input: training data: $\{\mathcal{X}, y\}$, pruning threshold θ.

Initialize: pretrained model parameter $\mathcal{W} = \{\mathcal{W}_i, 0 \leq i \leq L\}$;
for $epoch = 1$; $epoch \leq epoch_{max}$; $epoch++$ **do**
    Update the model parameter $\mathcal{W}$ based on $\{\mathcal{X}, y\}$ and, using the global sparse regularization as in Equation 2;
end for
Obtain the pruning mask $\mathcal{M}$ by thresholding $\mathcal{W}$ with θ;
Reinitialize: pretrained model parameter $\mathcal{W} = \{\mathcal{W}_i, 0 \leq i \leq L\}$;
for $epoch = 1$; $epoch \leq epoch_{max}$; $epoch++$ **do**
    Update the model parameter $\mathcal{W}$ based on $\{\mathcal{X}, y\}$, the mask-guided sparse regularization as shown in Equation 4 with the mask $\mathcal{M}$;
end for
Obtain the compact model $\mathcal{W}^*$ from $\mathcal{W}$;
Finetune the compact model $\mathcal{W}^*$;

Output: The compact model and its parameters $\mathcal{W}^*$.

3.3. MaskSparsity for Fine-grained Sparse Regularization

In this paper, we find the task of sparsity training consists of two sub-tasks implicitly. The first sub-task is identifying the unimportant channels. The second sub-task is pushing the filter weights or scaling factors of unimportant channels to 0 by the sparse regularization loss. Existing sparsity-training based methods accomplish the two sub-tasks simultaneously in the sparsity training stage. We propose to decouple the two sub-tasks. By doing this, we are able to apply the fine-grained spare regularization, which only sparse out the unimportant channels.

Figure 2 shows the training pipeline of the proposed MaskSparsity. We transform the sparsity training stage of the existing methods into two stages. The first stage is the sparsity training stage with global sparse regularization, which is aimed to get the indexes of the unimportant channels. The indexes are transformed into a binary pruning mask in previous methods. In our method, we use the mask

to identify which channels to apply the sparse regularization in the second stage. To get the pruning mask, we directly threshold the scaling factors of the normally trained network. The details is shown in Equation 3:

$$\mathcal{M} = \{\mathbb{1}(\gamma < \theta) | \gamma \in \Gamma\}, \tag{3}$$

where $\mathbb{1}$ is the indicator function, Γ is all the scaling factors of the network, and θ is the predefined pruning threshold of the pruning method. Actually, the pruning mask $\mathcal{M}$ consists of the unimportant-channel mask of each layers, *i.e.* $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, ..., \mathcal{M}_L\}$, where L is the layer count of the network. According to Equation 3, the pruning mask $\mathcal{M}_i$ is a binary vector consisting of 0 and 1.

As discussed above, the over-regularization on the important channels damages the network capacity. Therefore, in this paper, we design a fine-grained sparse training strategy to alleviate the damage of the sparse regularization loss on important channels. Specifically, we propose to apply the sparse regularization only on the unimportant channels. Based on Equation 2, we can describe our MaskSparsity method as the Equation 4:

$$L_{\text{Reg}}(\mathcal{I}^0, y, \mathcal{W}) = L(f(\mathcal{I}^0, \mathcal{W}), y) + \lambda \cdot \mathcal{M} \cdot Reg(\gamma)\,, \tag{4}$$

where $\mathcal{M}$ denote the binary mask indicating the unimportant channels of the whole network. For the important channels, the values in the channel mask are 0. Therefore, these channels are not affected by the sparse regularization and are trained as normal.

Pruning Strategy for Residual Connections. Many modern networks have the structure of residual-connections [9]. The residual-connections based module usually adopts the element-wise add operation. This operation requires the summed feature maps have the same channel count. Moreover, when pruning this structure, all the corresponding channels of the inputs of the add operation should be pushed-to-zero together before they can be pruned. In this scene, the pushed-to-zero channels that have non-sparse corresponding channels are also over-regularized by global sparse regularization to some extend. On the contrary, MaskSparsity can better use these channels to maximum the network's accuracy, since it does not apply the sparse regularization on the above mentioned not-to-prune channels. An illustration is shown in Figure 4.

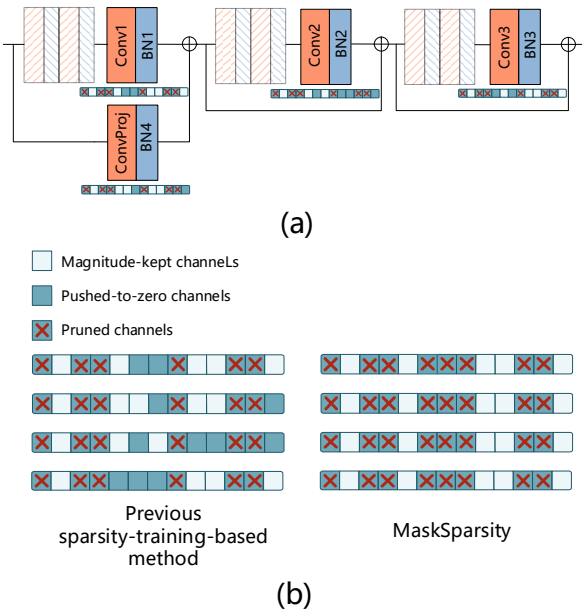

Figure 4. An illustration of pruning with Residual-Connections. (a) shows the structure of one typical network stage. Only the last convolution layers of each ResBlock is highlighted. (b) shows the pruning strategy of previous sparsity-training-based method and MaskSparsity method. The shallow blue boxes means the important channels. The deep blue boxes means the unimportant channels. The red cross means that the channel can be pruned.

4. Experiments

4.1. Experimental Settings

Datasets and Baselines. To demonstrate the effectiveness of Mask Sparsity in reducing model complexity, we evaluate MaskSparsity on both small and large datasets, *i.e.*, CIFAR-10 [20] ,and ILSVRC-2012 [34]. The CIFAR-10 dataset consists of natural images of 10 classes with resolution 32×32 and the train and test sets contain 50,000 and 10,000 images respectively. The ImageNet dataset consists of natural images of 1000 classes with resolution 224×224 and the train and test sets contain 1.2 million and 50,000 images respectively. We experiment with ResNet-50 [10] on ILSVRC-2012 , and experiment with ResNet-56 [9] and ResNet-110 [9] on CIFAR-10.

Evaluation Protocols. For fair comparison of the model complexity, we use the number of parameters and the FLOPS [9]. To evaluate the accuracy, to be consistent with other methods, we use top-1 and top-5 score of full-size models and pruned models on ILSVRC-2012 and top-1 score only on CIFAR-10.

Training and Pruning setting.

On CIFAR-10, we train models for 200 epochs with a batch size of 128, a weight decay of 0.0005, a Nesterov momentum of 0.9 without dampening in every stage, and an initial learning rate of 0.1 which is divided by 5 at epochs 60, 120 and 160 on four NVIDIA GTX 1080Ti GPUs; On ILSVRC-2012, we train models for 100 epochs with a batch size of 256, a weight decay of 0.0001, a Nesterov momentum of 0.9 with dampening in every stage, and an initial

Table 1. Evaluation results using ResNet-50 on ILSVRC-2012.

Method	Base Top-1	Base Top-5	Pruned Top-1	Pruned Top-5	Top-1 ↓	Top-5 ↓	FLOPs ↓%
NS [30]	75.04	-	69.60	-	5.44	-	50.51
OT [39]	75.04	-	70.40	-	4.64	-	52.88
SFP [12]	76.15	92.87	74.61	92.06	1.54	0.81	41.8
GAL-0.5 [28]	76.15	92.87	71.95	90.94	4.20	1.93	43.03
HRank [27]	76.15	92.87	74.98	92.33	1.17	0.54	43.76
Hinge [24]	-	-	74.7	-	-		46.55
HP [38]	76.01	92.93	74.87	92.43	1.14	0.50	50
MetaPruning [31]	76.6	-	75.4	-	1.2	-	51.10
Autopr [33]	76.15	92.87	74.76	92.15	1.39	0.72	51.21
FPGM [14]	76.15	92.87	74.83	92.32	1.32	0.55	53.5
DCP [43]	76.01	92.93	74.95	92.32	1.06	0.61	55.76
ThiNet [32]	75.30	92.20	72.03	90.99	3.27	1.21	55.83
MaskSparsity (ours)	**76.44**	**93.22**	**75.50**	**92.72**	**0.93**	**0.49**	53.761
HRank [27]	76.15	92.87	71.98	91.01	4.17	1.86	62.10

learning rate of 0.1 which is divided by 5 at epochs 30, 60 and 90 on eight NVIDIA GTX 1080Ti GPUs.

In the mask acquisition step, we first perform sparsity training on resnet56 with penalty factors $\lambda = 2e^{-4}$, resnet110 with penalty factors $\lambda = 5e^{-5}$ on CIFAR-10, and resnet50 with penalty factors $\lambda = 2e^{-4}$ on ILSVRC-2012. Then we prune the sparse model base on the "smaller-norm-less-important" criterion and obtain the sparse mask according to the target pruning rate.

In the mask sparsity training step, we reinitialize the network with a pretrained model and perform the mask sparsity training for the model with pruning the mask and a higher λ. On CIFAR-10, we choose $\lambda = 1e^{-3}$ for ResNet-56 and ResNet-110, and we choose $\lambda = 5e^{-4}$ for ResNet-50 On ILSVRC-2012.

In the formal pruning stage, we prune the model with a threshold of $1e^{-2}$. The pruning mask and sparse mask almost completely coincide, which means that we only perform a sparse penalty on the pruned channel.

After pruning, we fine-tune the pruned model with an initial learning rate of 0.001, and other parameter settings are the same as the previous step.

4.2. Results and Analysis

4.2.1 Results on ILSVRC-2012

As shown in Table 1, our proposed MaskSparsity achieves the new state-of-the-art. NS [30] is the baseline method of MaskSparsity, which adopt the global sparse regularization on the scaling factors of BN layers. OT [39] is the improved NS method, which set an optimal threshold for each layer. It can be seen that the MaskSparsity outperforms them significantly in the respect of accuracy drop (Top-1 ↓ and Top-5 ↓ in Table 1) under roughly the same level of FLOPS drop. This shows that with the fine-grained sparse regularization, we avoid the bad effect of the sparse regularization on the unpruned channels. Moreover, as shown in Section 4.3, with MaskSparsity, the pruning threshold on the scaling factors is easier to set.

In Table 1, it can be seen that we also outperform the non-sparsity-training-based methods under the same FLOPS decrease rate, *e.g.*, FPGM [14] (53.5% FLOPS reduced), and DCP [43] (55.76% FLOPS reduced), MetaPruning [31] (51.10% FLOPS reduced). While the FLOPS reduction of MaskSparsity is less than HRank (53.76% vs 62.1%), the accuracy of the pruned model is much higher than that of HRank (0.93 vs 4.17 in Top-1 accuracy drop). This shows that GroupSparsity's superiority over the previous state-of-the-art methods.

4.2.2 Results on CIFAR-10

Table 2 shows the experimental results of ResNet-56 on CIFAR-10. On this small dataset, MaskSparsity also achieves the state-of-the-art performance. Under similar FLOPs reduction with FPGM [14] and Hinge [24], MaskSparsity achieves 0.31% Top-1 accuracy drop with ResNet-56, which is slightly better than FPGM [14] (0.31% vs 0.33%) and Hinge [24] (0.31% vs 0.74%).

Table 3 shows the experimental results of another network, ResNet-110 on CIFAR-10. With this deeper network, MaskSparity achieves a better performance. As shown in Table 3, our MaskSparsity outperforms the other state-of-the-art methods, like HRank [27] (at 58.2% FLOPS reduction), under the roughly the same ratio of FLOPS reduction. MaskSparsity has roughly the same accuracy increase with FPGM [14] and C-SGD [7] (0.02 vs 0.06 and 0.03), but MaskSparsity has a larger FLOPS reduction than these two methods(63.03% vs 52.3% and 60.89%).

Table 4 shows the experimental results of VGG-16, which is a straight network structure that is different from ResNet. We compare MaskSparsity with NetSlim (NS)

Table 2. Evaluation results using ResNet-56 on CIFAR-10.

Method	Base top-1	Pruned top-1	Top-1 ↓%	FLOPs↓%
NISP [41]	-	-	0.03	42.6
Hinge [24]	93.69	92.95	0.74	50
Channel Pruning [15]	92.8	91.8	1.0	50
AMC [13]	92.8	91.9	0.9	50
He *et al.* [15]	93.26	90.80	2.46	50.6
LeGR [3]	93.9	93.7	0.2	52
FPGM [14]	93.59	93.26	0.33	52.6
LFPC [11]	93.59	93.24	0.35	52.9
MaskSparsity (ours)	**94.50**	**94.19**	**0.31**	54.88
GAL-0.8 [28]	93.26	90.36	2.9	60.2
HRank [27]	93.26	90.72	2.54	74.1

Table 3. Evaluation results using ResNet-110 on CIFAR-10.

Method	Base top-1	Pruned top-1	Top-1 ↓%	FLOPs↓%
Li et al. [22]	93.53	93.30	0.23	38.60
SFP [12]	93.68	93.86	-0.18	40.8
NISP-110 [41]	-	-	0.18	43.78
GAL-0.5 [28]	93.50	92.74	0.76	48.5
FPGM [14]	93.68	93.74	-0.06	52.3
HRank [27]	93.50	93.36	0.14	58.2
LFPC [11]	93.68	93.07	0.61	60.3
MaskSparsity (ours)	**94.70**	**94.72**	**-0.02**	**63.03**
HRank [27]	93.50	92.65	0.85	68.6
C-SGD [7]	94.38	94.41	-0.03	60.89
HRank	93.50	92.65	0.85	68.6
SASL [35]	93.83	93.80	0.03	70.2

[30], FPGM [14], and PFEC [22]. It shows that we outperforms NS [30] and PFEC [22] on both accuracy and FLOPS reduction. Compared with FPGM [14], we are 0.04 % less than FPGM on the increase of the accuracy, which is very minor. However, we decrease 18.01% more FLOPS than FPGM. Therefore, we also outperforms FPGM in general pruning performance. Moreover, we want to

Table 4. Evaluation results using VGG-16 on CIFAR-10.

Method	Base top-1	Pruned before FT	FT	Top-1 ↓%	FLOPs↓%
PFEC [22] (from [14])	93.58	77.45	93.28	0.3	34.2
FPGM [14]	93.58	80.38	94.00	**-0.42**	34.2
NS [30]	93.66	-	93.80	-0.14	51
MaskSparsity(ours)	**93.86**	**94.16**	**94.24**	-0.38	**52.21**

4.3. Ablation Study

Visualization of the distribution of the scaling factors after using MaskSparsity. In Section 3.2 and Figure 3, we show that the distribution of scaling factors meets two problems that might harm the pruning performance. In Figure 5 we draw the same set of scaling factors after the mask-guided sparse regularization, and compare it with that of the pre-trained networks. It can be seen that the two problems are alleviated significantly. First, the right peak of the sparsity trained network does not move towards the 0 like in Figure 3. Second, the two peaks are well distinguishable, with almost no in-between bars in the middle area. This demonstrates the effectiveness of the fine-grained sparse regularization of MaskSparsity, which would benefit the pruning performance.

The convergence analysis by visualizing the gradients. To validate the statement that the sparse regularization on unpruned channels restricts the network's capacity, we visualize the gradients of important and unimportant channels after the sparsity training using global and fine-grained sparse regularization, respectively. The result is shown in Figure 6. The gradients are collected by continue training for 1,000 iterations from the end of the sparsity training stage of ResNet-50 on ILSVRC-2012. In Figure 6 (a), it can be seen that the gradient norm of the important channels is still large for the important channels with the global sparse regularization, while Figure 6 (b) shows the important channel has a small gradient norm with our fine-grained MaskSparsity sparse regularization. Figure 6 (b) and (d) shows that both the gradients of unimportant channels with the two kind of sparse regularization methods are almost the same. According to these figures, the global sparse regularization leads to a larger gradient on the important channels though the network's accuracy and sparsity have converged. We infer the large gradient prevent the network converge to a better local-minimum.

The effectiveness of fine-grained sparse regularization on residual modules. In Section 3.3 and Figure 4, we claim that the MaskSparsity avoids to apply sparse regularization on the unpruned unimportant channels, which can make the network make full use of all kept channels to maximum its accuracy. In Figure 7, we plot the binary unimportant-channel masks of each layer of the third stage of ResNet-56, which are generated by thresholding the scaling factors. It can be seen that the binary masks of MaskSparsity is much consistent among the channels of the same channel index. This shows the benefit of the fine-grained sparsity from MaskSparsity.

The robustness to various pruning thresholds.

Comparing fine-tuning and train-from-scratch. In Table 5, we list the network model's accuracy and computational complexity at different pruning stages. Moreover, we also list the result that train the pruned model from scratch without exploiting its weights. It can be seen that the train-from-scratch result is lower than the fine-tuning result. We think this demonstrates the effectiveness of the fine-grained sparsely-trained pre-trained weights.

In order to make a fair comparison and remove the influence of insufficient convergence, we use the same parameter configuration as normal train, except for the base learning

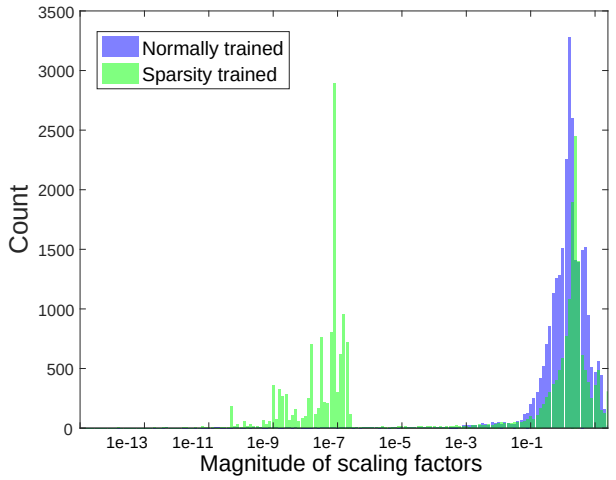

Figure 5. Distribution of scaling factors of ResNet50 on ILSVRC-2012 before and after the MaskSparsity's sparsity training.

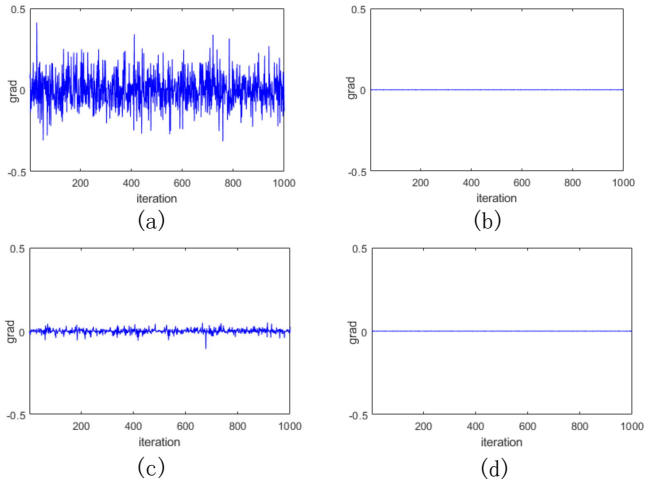

Figure 6. The gradients of the scaling factors of a certain channel over the additional 1, 000 sparsity-training iterations after the ending of the sparsity training stage. The spare regularization keep the same as that used in the original sparsity training. We choose an important channel and an unimportant channel to conduct this experiment. (a) The important channel, global sparse regularization. (b) The unimportant channel, global sparse regularization. (c) The important channel, fine-grained sparse regularization of MaskSparsity. (d) The unimportant channel, fine-grained sparse regularization of MaskSparsity.

rate, and enough epochs for scratch training and fine-tune. As show in Tab 5. The results show that fine-tune result is nearly 1% top-1 accuracy higher than training from scratch.

5. Conclusion

In this paper, to solve the problem that existing sparsity-training-based methods over-regularize the important channels, we design a network pruning method with the fine-grained sparse regularization, named MaskSparsity. MaskSparsity apply the sparse regularization only on the unimportant channels which are to be pruned. Therefore, MaskSparsity can minimize the negative impact of the sparse regularization on the important channels. The method is effective and efficient, without too many hyper-parameter settings. The experimental results show that it outperforms the other sparsity-training-based pruning methods and achieves the state-of-the-art on the bench-

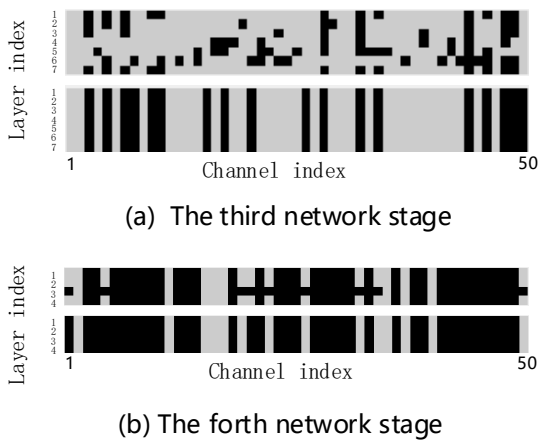

Figure 7. The unimportant-channel masks in every last layers of ResBlocks in the third and forth network stage. The data were obtained from ResNet50 trained with traditional sparsity-training-based methods and MaskSparsity on ILSVRC-2012.

Table 5. The stage-wise performance in the case of pruning ResNet-56 on CIFAR-10.

Model state	Top-1%	FLOPs	Parameters
Normally trained	94.50	126M	853K
MaskSparsity trained	94.02	126M	853K
Pruned	92.67	57M	419K
Finetune	94.19	57M	419K
Train from Scratch	93.60	57M	419K

Table 6. Pruning ResNet56 on CIFAR-10.

Pruning threshold	Top-1%	FLOPs	Parameters
Before pruning	94.02	126M	853K
$1e^{-1}$	85.63	54M	408K
$1e^{-2}$	92.67	57M	419K
$1e^{-3}$	92.56	57M	419K
$1e^{-4}$	92.59	57M	421K
$1e^{-5}$	93.01	58M	422K
$1e^{-6}$	94.02	90M	620K

marks. In the future, we plan to work on how to obtain better pruning masks.

References

[1] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, pages 2270–2278, 2016. 2, 3

[2] Jose M Alvarez and Mathieu Salzmann. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, pages 856–867, 2017. 3

[3] Ting-Wu Chin, Ruizhou Ding, C. Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1515–1525, 2020. 7

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[5] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013. 1

[6] Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014. 1

[7] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal SGD for pruning very deep convolutional networks with complicated structure. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4943–4953. Computer Vision Foundation / IEEE, 2019. 6, 7

[8] P. T. Fletcher, Suresh Venkatasubramanian, and S. Joshi. Robust statistics on riemannian manifolds via the geometric median. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 4, 5

[10] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *ArXiv*, abs/1603.05027, 2016. 5

[11] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7

[12] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 2234–2240, 2018. 6, 7

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, pages 815–832. Springer, 2018. 7

[14] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4340–4349. Computer Vision Foundation / IEEE, 2019. 3, 6, 7

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *International Conference on Computer Vision (ICCV)*, volume 2, page 6, 2017. 7

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[17] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018. 2, 3

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[19] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. 1

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, 2009. 5

[21] Hao Li, Asim Kadav, Igor Durdanovic, H. Samet, and H. Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2017. 3

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *5th International Conference on Learning Representations*, 2017. 7

[23] Jiashi Li, Q. Qi, J. Wang, Ce Ge, Yujian Li, Zhangzhang Yue, and Haifeng Sun. Oicsr: Out-in-channel sparsity regularization for compact deep neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7039–7048, 2019. 3

[24] Yawei Li, Shuhang Gu, C. Mayer, L. Gool, and R. Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8015–8024, 2020. 3, 6, 7

[25] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8018–8027, 2020. 1

[26] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020. 3

[27] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *CoRR*, abs/2002.10179, 2020. 6, 7

[28] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David S. Doermann. Towards optimal structured CNN pruning via generative adversarial learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 2790–2799. Computer Vision Foundation / IEEE, 2019. 6, 7

[29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755, 2014. 1

[30] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 2, 3, 6, 7

[31] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3295–3304. IEEE, 2019. 6

[32] Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. Thinet: Pruning CNN filters for a thinner net. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(10):2525–2538, 2019. 6

[33] Jian-Hao Luo and Jianxin Wu. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *arXiv preprint arXiv:1805.08941*, 2018. 6

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1, 5

[35] Jun Shi, Jianfeng Xu, Kazuyuki Tasaka, and Zhibo Chen. SASL: saliency-adaptive sparsity learning for neural network acceleration. *CoRR*, abs/2003.05891, 2020. 7

[36] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. Training sparse neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 455–462, 2017. 3

[37] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016. 2, 3

[38] Xiaofan Xu, Mi Sun Park, and Cormac Brick. Hybrid pruning: Thinner sparse networks for fast inference on edge devices. *arXiv preprint arXiv:1811.00482*, 2018. 6

[39] Yun Ye, Ganmei You, J. Fwu, Xia Zhu, Q. Yang, and Y. Zhu. Channel pruning via optimal thresholding. *ArXiv*, abs/2003.04566, 2020. 3, 6

[40] Yun Ye, Ganmei You, Jong-Kae Fwu, Xia Zhu, Qing Yang, and Yuan Zhu. Channel pruning via optimal thresholding. *arXiv preprint arXiv:2003.04566*, 2020. 4

[41] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018. 7

[42] Kai Zhao, Xinyu Zhang, Q. Han, and Ming-Ming Cheng. Dependency aware filter pruning. *ArXiv*, abs/2005.02634, 2020. 3

[43] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 883–894, 2018. 6