Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration

Yang He[1] Ping Liu[1,2] Ziwei Wang[3] Zhilan Hu[4] Yi Yang[1,5]*
[1]CAI, University of Technology Sydney [2]JD.com
[3]Information Science Academy, CETC [4]Huawei [5]Baidu Research
{yang.he-1}@student.uts.edu.au {pino.pingliu,wangziwei26,yee.i.yang}@gmail.com

Abstract

Previous works utilized "smaller-norm-less-important" criterion to prune filters with smaller norm values in a convolutional neural network. In this paper, we analyze this norm-based criterion and point out that its effectiveness depends on two requirements that are not always met: (1) the norm deviation of the filters should be large; (2) the minimum norm of the filters should be small. To solve this problem, we propose a novel filter pruning method, namely Filter Pruning via Geometric Median (FPGM), to compress the model regardless of those two requirements. Unlike previous methods, FPGM compresses CNN models by pruning filters with redundancy, rather than those with "relatively less" importance. When applied to two image classification benchmarks, our method validates its usefulness and strengths. Notably, on CIFAR-10, FPGM reduces more than 52% FLOPs on ResNet-110 with even 2.69% relative accuracy improvement. Moreover, on ILSVRC-2012, FPGM reduces more than 42% FLOPs on ResNet-101 without top-5 accuracy drop, which has advanced the state-of-the-art. Code is publicly available on GitHub: https://github.com/he-y/filter-pruning-geometric-median

1. Introduction

The deeper and wider architectures of deep CNNs bring about the superior performance of computer vision tasks [6, 26, 45]. However, they also cause the prohibitively expensive computational cost and make the model deployment on mobile devices hard if not impossible. Even the latest architecture with high efficiencies, such as residual connection [12] or inception module [34], has millions of parameters requiring billions of float point operations (FLOPs) [15]. Therefore, it is necessary to attain the deep CNN models which have relatively low computational cost

*Corrsponding Author. Part of this work was done when Yi Yang was visiting Baidu Research during his Professional Experience Program.

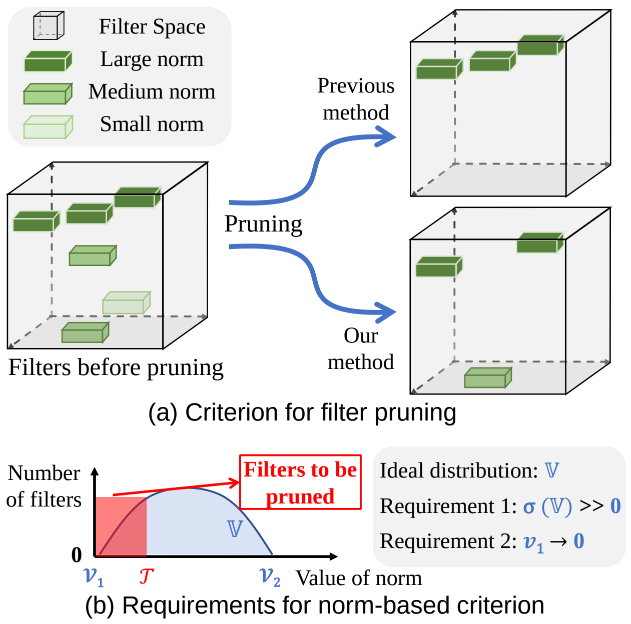


Figure 1. An illustration of (a) the pruning criterion for norm-based approach and the proposed method; (b) requirements for norm-based filter pruning criterion. In (a), the green boxes denote the filters of the network, where deeper color denotes larger norm of the filter. For the norm-based criterion, only the filters with the largest norm are kept based on the assumption that smaller-norm filters are less important. In contrast, the proposed method prunes the filters with redundant information in the network. In this way, filters with different norms indicated by different intensities of green may be retained. In (b), the blue curve represents the ideal norm distribution of the network, and the v_1 and v_2 is the minimal and maximum value of norm distribution, respectively. To choose the appropriate threshold $\mathcal{T}$ (the red shadow), two requirements should be achieved, that is, the norm deviation should be large, and the minimum of the norm should be arbitrarily small.

but high accuracy.

Recent developments on pruning can be divided into two categories, *i.e.*, weight pruning [11, 1] and filter pruning [21, 39]. Weight pruning directly deletes weight values in a filter which may cause unstructured sparsities. This irregular structure makes it difficult to leverage the high-efficiency Basic Linear Algebra Subprograms (BLAS) libraries [25]. In contrast, filter pruning directly discards the whole selected filters and leaves a model with regular structures. Therefore, filter pruning is more preferred for accelerating the networks and decreasing the model size.

Current practice [21, 38, 15] performs filter pruning by following the "smaller-norm-less-important" criterion, which believes that filters with smaller norms can be pruned safely due to their less importance. As shown in the top right of Figure 1(a), after calculating norms of filters in a model, a pre-specified threshold $\mathcal{T}$ is utilized to select filters whose norms are smaller than it.

However, as illustrated in Figure 1(b), there are two prerequisites to utilize this "smaller-norm-less-important" criterion. First, the deviation of filter norms should be significant. This requirement makes the searching space for threshold $\mathcal{T}$ wide enough so that separating those filters needed to be pruned would be an easy task. Second, the norms of those filters which can be pruned should be arbitrarily small, *i.e.*, close to zero; in other words, the filters with smaller norms are expected to make absolutely small contributions, rather than relatively less but positively large contributions, to the network. An ideal norm distribution when satisfactorily meeting those two requirements is illustrated as the blue curve in Figure 1. Unfortunately, based on our analysis and experimental observations, this is not always true.

To address the problems mentioned above, we propose a novel filter pruning approach, named Filter Pruning via Geometric Median (FPGM). Different from the previous methods which prune filters with ***relatively less contribution***, FPGM chooses the filters with ***the most replaceable contribution***. Specifically, we calculate the Geometric Median (GM) [8] of the filters within the same layer. According to the characteristics of GM, the filter(s) $\mathcal{F}$ near it can be represented by the remaining ones. Therefore, pruning those filters will not have substantial negative influences on model performance. Note that FPGM does not utilize norm based criterion to select filters to prune, which means its performance will not deteriorate even when failing to meet requirements for norm-based criterion.

Contributions. We have three contributions:

(1) We analyze the norm-based criterion utilized in previous works, which prunes the relatively less important filters. We elaborate on its two underlying requirements which lead to its limitations;

(2) We propose FPGM to prune the most replaceable filters containing redundant information, which can still achieve good performances when norm-based criterion fails;

(3) The extensive experiment on two benchmarks demonstrates the effectiveness and efficiency of FPGM.

2. Related Works

Most previous works on accelerating CNNs can be roughly divided into four categories, namely, *matrix decomposition* [42, 35], *low-precision weights* [44, 43, 32], knowledge distilling [17, 19] and *pruning*. *Pruning*-based approaches aim to remove the unnecessary connections of the neural network [11, 21, 24]. Essentially, ***weight pruning*** always results in unstructured models, which makes it hard to deploy the efficient BLAS library, while ***filter pruning*** not only reduces the storage usage on devices but also decreases computation cost to accelerate the inference. We could roughly divide the filter pruning methods into two categories by whether the training data is utilized to determine the pruned filters, that is, *data dependent* and *data independent* filter pruning. Data independent method is more efficient than data dependent method as the utilizing of training data is computation consuming.

Weight Pruning. Many recent works [11, 10, 9, 36, 1, 15, 41, 4] focus on pruning fine-grained weight of filters. For example, [11] proposes an iterative method to discard the small weights whose values are below the predefined threshold. [1] formulates pruning as an optimization problem of finding the weights that minimize the loss while satisfying a pruning cost condition.

Data Dependent Filter Pruning. Some filter pruning approaches [23, 25, 16, 27, 7, 33, 39, 37, 46, 14, 18, 22] need to utilize training data to determine the pruned filters. [25] adopts the statistics information from the next layer to guide the filter selections. [7] aims to obtain a decomposition by minimizing the reconstruction error of training set sample activation. [33] proposes an inherently data-driven method which use Principal Component Analysis (PCA) to specify the proportion of the energy that should be preserved. [37] applies subspace clustering to feature maps to eliminate the redundancy in convolutional filters.

Data Independent Filter Pruning. Concurrently with our work, some data independent filter pruning strategies [21, 15, 38, 47] have been explored. [21] utilizes an ℓ_1-norm criterion to prune unimportant filters. [15] proposes to select filters with a ℓ_2-norm criterion and prune those selected filters in a soft manner. [38] proposes to prune models by enforcing sparsity on the scaling parameter of batch normalization layers. [47] uses spectral clustering on filters to select unimportant ones.

Discussion. To the best of our knowledge, only one previous work reconsiders the smaller-norm-less-important criterion [38]. We would like to highlight our advantages compared to this approach as below: (1) [38] pays more attention to enforcing sparsity on the scaling parameter in the batch normalization operator, which is not friendly to the structure without batch normalization. On the contrary, our approach is not limited by this constraint. (2) After pruning channels selected, [38] need fine-tuning to reduce performance degradation. However, our method combines the pruning operation with normal training procedure. Thus extra fine-tuning is not necessary. (3) Calculation of the gradient of scaling factor is needed for [38]; therefore lots of computation cost are inevitable, whereas our approach could accelerate the neural network without calculating the gradient of scaling factor.

3. Methodology

3.1. Preliminaries

We formally introduce symbols and notations in this subsection. We assume that a neural network has L layers. We use N_i and N_{i+1}, to represent the number of input channels and the output channels for the i_{th} convolution layer, respectively. $\mathcal{F}_{i,j}$ represents the j_{th} filter of the i_{th} layer, then the dimension of filter $\mathcal{F}_{i,j}$ is $\mathbb{R}^{N_i \times K \times K}$, where K is the kernel size of the network[1]. The i_{th} layer of the network $\mathbf{W}^{(i)}$ could be represented by $\{\mathcal{F}_{i,j}, 1 \leq j \leq N_{i+1}\}$. The tensor of connection of the deep CNN network could be parameterized by $\{\mathbf{W}^{(i)} \in \mathbb{R}^{N_{i+1} \times N_i \times K \times K}, 1 \leq i \leq L\}$.

3.2. Analysis of Norm-based Criterion

Figure 1 gives an illustration for the two requirements for successful utilization of the norm-based criterion. However, these requirements may not always hold, and it might lead to unexpected results. The details are illustrated in Figure 2, in which the blue dashed curve and the green solid curve indicates the norm distribution in ideal and real cases, respectively.

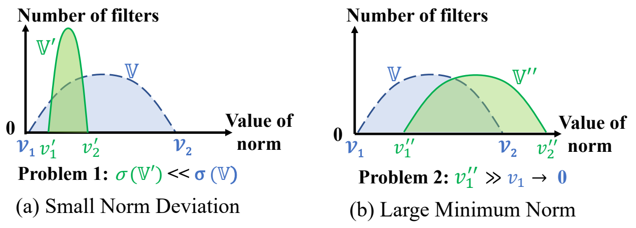


Figure 2. Ideal and Reality of the norm-based criterion: (a) Small Norm Deviation and (b) Large Minimum Norm. The blue dashed curve indicates the ideal norm distribution, and the green solid curve denotes the norm distribution might occur in real cases.

(1) *Small Norm Deviation*. The deviation of filter norm distributions might be too small, which means the norm values are concentrated to a small interval, as shown in Figure 2(a). A small norm deviation leads to a small search space, which makes it difficult to find an appropriate threshold to select filters to prune.

(2) *Large Minimum Norm*. The filters with the minimum norm may not be arbitrarily small, as shown in the Figure 2(b), $v_1'' >> v_1 \to 0$. Under this condition, those filters considered as the least important still contribute significantly to the network, which means every filter is highly informative. Therefore, pruning those filters with minimum norm values will cast a negative effect on the network.

[1]Fully-connected layers equal to convolutional layers with $k = 1$

3.3. Norm Statistics in Real Scenarios

In Figure 3, statistical information collected from pre-trained ResNet-110 on CIFAR-10 and pre-trained ResNet-18 on ILSVRC-2012 demonstrates previous analysis. The small green vertical lines show each observation in this norm distribution, and the blue curves denote the Kernel Distribution Estimate (KDE) [30], which is a non-parametric way to estimate the probability density function of a random variable. The norm distribution of first layer and last layer in both structures are drawn. In addition, to clearly illustrate the relation between norm points, two different x-scale, *i.e.*, linear x-scale and log x-scale, are presented.

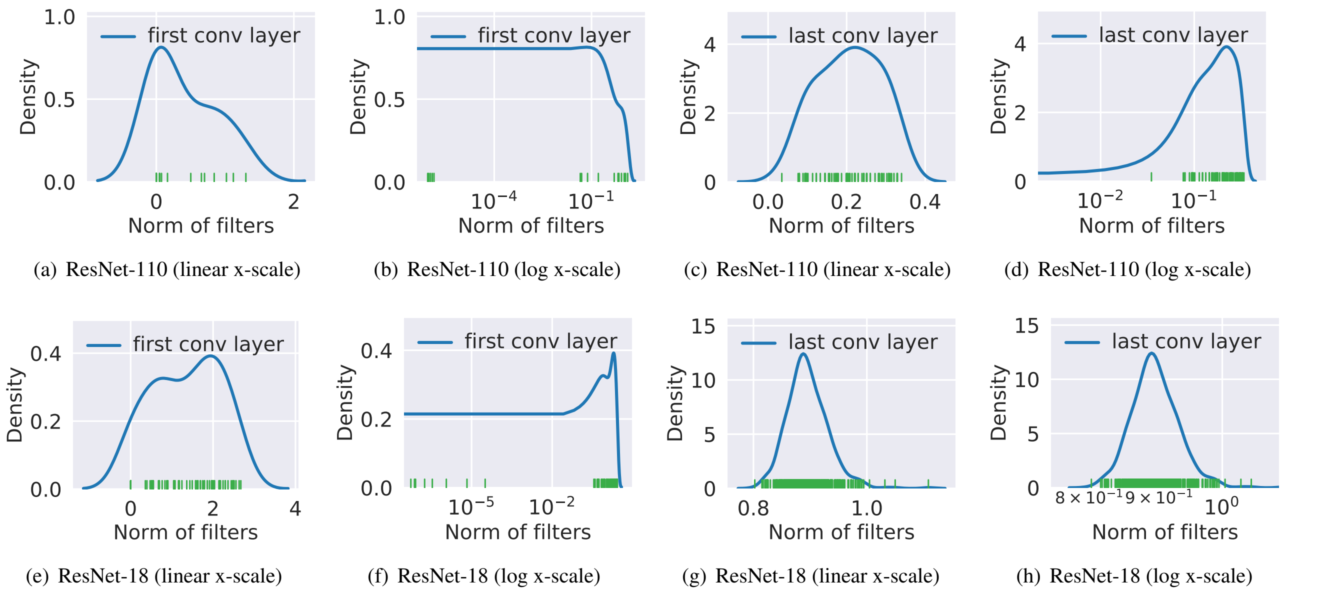
(1) *Small Norm Deviation in Network*. For the first convolutional layer of ResNet-110, as shown in Figure 3(b), there is a large quantity of filters whose norms are **concentrated** around the magnitude of 10^{-6}. For the last convolutional layer of ResNet-110, as shown in Figure 3(c), the interval span of the value of norm is roughly **0.3**, which is much smaller than the interval span of the norm of the first layer (**1.7**). For the last convolutional layer of ResNet-18, as shown in Figure 3(g), most filter norms are between the interval $[0.8, 1.0]$. In all these cases, filters are distributed too densely, which makes it difficult to select a proper threshold to distinguish the important filters from the others.

(2) *Large Minimum Norm in Network*. For the last convolutional layer of ResNet-18, as shown in Figure 3(g), the minimum norm of these filters is around **0.8**, which is **large** comparing to filters in the first convolutional layer (Figure 3(e)). For the last convolutional layer of ResNet-110, as shown in Figure 3(c), only one filter is arbitrarily small, while the others are not. Under those circumstances, the filters with minimum norms, although they are relatively less important according to the norm-based criterion, still make significant contributions in the network.

3.4. Filter Pruning via Geometric Median

To get rid of the constraints in the norm-based criterion, we propose a new filter pruning method inspired from geometric median. The central idea of geometric median [8] is as follows: given a set of n points $a^{(1)}, \dots, a^{(n)}$ with each $a^{(i)} \in \mathbb{R}^d$, find a point $x^* \in \mathbb{R}^d$ that minimizes the sum of Euclidean distances to them:

$$x^* \in \underset{x \in \mathbb{R}^d}{\arg\min} f(x) \quad \text{where} \quad f(x) \overset{\text{def}}{=} \sum_{i \in [1,n]} \|x - a^{(i)}\|_2 \quad (1)$$



(a) ResNet-110 (linear x-scale) (b) ResNet-110 (log x-scale) (c) ResNet-110 (linear x-scale) (d) ResNet-110 (log x-scale)

(e) ResNet-18 (linear x-scale) (f) ResNet-18 (log x-scale) (g) ResNet-18 (linear x-scale) (h) ResNet-18 (log x-scale)

Figure 3. Norm distribution of filters from different layers of ResNet-110 on CIFAR-10 and ResNet-18 on ILSVRC-2012. The small green vertical lines and blue curves denote each norm and Kernel Distribution Estimate (KDE) of the norm distribution, respectively.

As the geometric median is a classic robust estimator of centrality for data in Euclidean spaces [8], we use the geometric median $\mathcal{F}_i^{GM}$ to get the common information of all the filters within the single i_{th} layer:

$$\mathcal{F}_i^{GM} \in \underset{x \in \mathbb{R}^{N_i \times K \times K}}{\arg\min} \; g(x), \tag{2}$$

where

$$g(x) \overset{\text{def}}{=} \sum_{j' \in [1, N_{i+1}]} \|x - \mathcal{F}_{i,j'}\|_2. \tag{3}$$

In the i_{th} layer, if some filters have the same, or similar values as the geometric median in that layer, which is:

$$\mathcal{F}_{i,j^*} \in \underset{j' \in [1, N_{i+1}]}{\arg\min} \; \|\mathcal{F}_{i,j'} - \mathcal{F}_i^{GM}\|_2, \tag{4}$$

then those filters, $\mathcal{F}_{i,j^*}$, can be represented by the other filters in the same layer, and therefore, pruning them has little negative impacts on the network performance.

As geometric median is a non-trivial problem in computational geometry, the previous fastest running times for computing a $(1+\epsilon)$-approximate geometric median were $\widetilde{O}(dn^{4/3} \cdot \epsilon^{-8/3})$ by [2], $O(nd \log^3(n/\epsilon))$ by [3]. In our case, as the final result $\mathcal{F}_{i,j^*}$ are a list of know points, that is, the candidate filters in the layer, we could relax the above problem.

We assume that

$$\|\mathcal{F}_{i,j^*} - \mathcal{F}_i^{GM}\|_2 = 0, \tag{5}$$

so the Equation.4 is achieved. Then the above Equation.2 becomes to

$$\begin{aligned} \mathcal{F}_{i,j^*} \in & \underset{j^* \in [1, N_{i+1}]}{\arg\min} \sum_{j' \in [1, N_{i+1}]} \|x - \mathcal{F}_{i,j'}\|_2 \\ = & \underset{j^* \in [1, N_{i+1}]}{\arg\min} \; g(x) \end{aligned} \tag{6}$$

Note that even if the *filter need to be pruned*, $\mathcal{F}_{i,j^*}$, is not included in the calculation of the geometric median in Equation.6[2], we could also achieve the same result. In this setting, we want to find the filter

$$\mathcal{F}_{i,j^{*\prime}} \in \underset{j^* \in [1, N_{i+1}]}{\arg\min} \; g'(x), \tag{7}$$

where

$$g'(x) = \sum_{j' \in [1, N_{i+1}], j' \neq j^*} \|x - \mathcal{F}_{i,j'}\|_2. \tag{8}$$

With the above Equation.6 and Equation.8, we could get that:

$$\begin{aligned} g'(x) &= g(x) - \sum_{j' = j^*} \|x - \mathcal{F}_{i,j'}\|_2 \\ &= g(x) - \|x - \mathcal{F}_{i,j^*}\|_2. \end{aligned} \tag{9}$$

[2]To select multiple filters, we choose several j that makes $g(x)$ to the smallest extent.

Algorithm 1 Algorithm Description of FPGM

```
Input: training data: $\mathbf{X}$.
1: Given: pruning rate $P_i$
2: Initialize: model parameter $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \le i \le L\}$
3: for $epoch = 1; epoch \le epoch_{max}; epoch++$ do
4:     Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
5:     for $i = 1; i \le L; i++$ do
6:         Find $N_{i+1}P_i$ filters that satisfy Equation 6
7:         Zeroize selected filters
8:     end for
9: end for
10: Obtain the compact model $\mathbf{W}^*$ from $\mathbf{W}$
Output: The compact model and its parameters $\mathbf{W}^*$
```

then we could get

$$\begin{aligned} \min g'(x) &= \min\{g(x) - \|x - \mathcal{F}_{i,j^*}\|_2\} \\ &= \min g(x) - \min \|x - \mathcal{F}_{i,j^*}\|_2 \\ &= g(\mathcal{F}_{i,j^*}) - \min \|x - \mathcal{F}_{i,j^*}\|_2. \end{aligned} \tag{10}$$

For the second component of the right side for Equation.10, when $x = \mathcal{F}_{i,j^*}$, we can get:

$$\mathcal{F}_{i,j^{*\prime}} = \mathcal{F}_{i,j^*} \tag{11}$$

since $\|x - \mathcal{F}_{i,j'}\|_2 = 0$

Since the geometric median is a classic robust estimator of centrality for data in Euclidean spaces [8], the selected filter(s), $\mathcal{F}_{i,j^*}$, and left ones share the most common information. This indicates the information of the filter(s) $\mathcal{F}_{i,j^*}$ could be replaced by others. After fine-tuning, the network could easily recover its original performance since the information of pruned filters can be represented by the remaining ones. Therefore, the filter(s) $\mathcal{F}_{i,j^*}$ could be pruned with negligible effect on the final result of the neural network. The FPGM is summarized in Algorithm 1.

3.5. Theoretical and Realistic Acceleration

3.5.1 Theoretical Acceleration

Suppose the shapes of input tensor $\mathbf{I} \in N_i \times H_i \times W_i$ and output tensor $\mathbf{O} \in N_{i+1} \times H_{i+1} \times W_{i+1}$. Set the filter pruning rate of the i_{th} layer to P_i, then $N_{i+1} \times P_i$ filters should be pruned. After filter pruning, the dimension of input and output feature map of the i_{th} layer change to $\mathbf{I}' \in [N_i \times (1 - P_i)] \times H_i \times W_i$ and $\mathbf{O}' \in [N_{i+1} \times (1 - P_i)] \times H_{i+1} \times W_{i+1}$, respectively.

If setting pruning rate for the $(i+1)_{th}$ layer to P_{i+1}, then only $(1 - P_{i+1}) \times (1 - P_i)$ of the original computation is needed. Finally, a compact model $\{\mathbf{W}^{*(i)} \in \mathbb{R}^{N_{i+1}(1-P_i) \times N_i(1-P_{i-1}) \times K \times K}\}$ is obtained.

3.5.2 Realistic Acceleration

In the above analysis, only the FLOPs of convolution operations for computation complexity comparison is considered, which is common in previous works [21, 15]. This is because other operations such as batch normalization (BN) and pooling are insignificant comparing to convolution operations.

However, non-tensor layers (e.g., BN and pooling layers) also need the inference time on GPU [25], and influence the realistic acceleration. Besides, the wide gap between the theoretical and realistic acceleration could also be restricted by the IO delay, buffer switch, and efficiency of BLAS libraries. We compare the theoretical and practical acceleration in Table 5.

4. Experiments

We evaluate FPGM for single-branch network (VGGNet [31]), and multiple-branch network (ResNet) on two benchmarks: CIFAR-10 [20] and ILSVRC-2012 [29][3]. The CIFAR-10 [20] dataset contains $60,000$ 32×32 color images in 10 different classes, in which $50,000$ training images and $10,000$ testing images are included. ILSVRC-2012 [29] is a large-scale dataset containing 1.28 million training images and 50k validation images of 1,000 classes.

4.1. Experimental Settings

Training setting. On CIFAR-10, the parameter setting is the same as [13] and the training schedule is the same as [40]. In the ILSVRC-2012 experiments, we use the default parameter settings which is same as [12, 13]. Data argumentation strategies for ILSVRC-2012 is the same as PyTorch [28] official examples. We analyze the difference between starting from scratch and the pre-trained model. For pruning the model from scratch, We use the normal training schedule without additional fine-tuning process. For pruning the pre-trained model, we reduce the learning rate to one-tenth of the original learning rate. To conduct a fair comparison of pruning scratch and pre-trained models, we use the same training epochs to train/fine-tune the network. The previous work [21] might use fewer epochs to finetune the pruned model, but it converges too early, and its accuracy can not improve even with more epochs, which can be shown in section 4.2.

Pruning setting. In the filter pruning step, we simply prune *all* the weighted layers with the *same* pruning rate at the same time, which is the same as [15]. Therefore, only one hyper-parameter $P_i = P$ is needed to balance the acceleration and accuracy. The pruning operation is conducted at

[3]As stated in [21], "comparing with AlexNet or VGG (on ILSVRC-2012), both VGG (on CIFAR-10) and Residual networks have fewer parameters in the fully connected layers", which makes pruning filters in those networks challenging.

Depth	Method	Fine-tune?	Baseline acc. (%)	Accelerated acc. (%)	Acc. ↓ (%)	FLOPs	FLOPs ↓(%)
20	SFP [15]	✗	**92.20** (±0.18)	90.83 (±0.31)	1.37	2.43E7	42.2
	Ours (FPGM-only 30%)	✗	**92.20** (±0.18)	91.09 (±0.10)	1.11	2.43E7	42.2
	Ours (FPGM-only 40%)	✗	**92.20** (±0.18)	90.44 (±0.20)	1.76	**1.87E7**	**54.0**
	Ours (FPGM-mix 40%)	✗	**92.20** (±0.18)	**91.99** (±0.15)	**0.21**	**1.87E7**	**54.0**
32	MIL [5]	✗	92.33	90.74	1.59	4.70E7	31.2
	SFP [15]	✗	**92.63** (±0.70)	92.08 (±0.08)	0.55	4.03E7	41.5
	Ours (FPGM-only 30%)	✗	**92.63** (±0.70)	92.31 (±0.30)	0.32	4.03E7	41.5
	Ours (FPGM-only 40%)	✗	**92.63** (±0.70)	91.93 (±0.03)	0.70	**3.23E7**	**53.2**
	Ours (FPGM-mix 40%)	✗	**92.63** (±0.70)	**92.82** (±0.03)	**-0.19**	**3.23E7**	**53.2**
56	PFEC [21]	✗	93.04	91.31	1.75	9.09E7	27.6
	CP [16]	✗	92.80	90.90	1.90	–	50.0
	SFP [15]	✗	**93.59** (±0.58)	92.26 (±0.31)	1.33	**5.94E7**	**52.6**
	Ours (FPGM-only 40%)	✗	**93.59** (±0.58)	**92.93** (±0.49)	**0.66**	**5.94E7**	**52.6**
	Ours (FPGM-mix 40%)	✗	**93.59** (±0.58)	92.89 (±0.32)	0.70	**5.94E7**	**52.6**
	PFEC [21]	✓	93.04	93.06	-0.02	9.09E7	27.6
	CP [16]	✓	92.80	91.80	1.00	–	50.0
	Ours (FPGM-only 40%)	✓	**93.59** (±0.58)	**93.49** (±0.13)	**0.10**	**5.94E7**	**52.6**
	Ours (FPGM-mix 40%)	✓	**93.59** (±0.58)	93.26 (±0.03)	0.33	**5.94E7**	**52.6**
110	MIL [5]	✗	93.63	93.44	0.19	-	34.2
	PFEC [21]	✗	93.53	92.94	0.61	1.55E8	38.6
	SFP [15]	✗	**93.68** (±0.32)	93.38 (±0.30)	0.30	1.50E8	40.8
	Ours (FPGM-only 40%)	✗	**93.68** (±0.32)	93.73 (±0.23)	-0.05	**1.21E8**	**52.3**
	Ours (FPGM-mix 40%)	✗	**93.68** (±0.32)	**93.85** (±0.11)	**-0.17**	**1.21E8**	**52.3**
	PFEC [21]	✓	93.53	93.30	0.20	1.55E8	38.6
	NISP [39]	✓	–	–	0.18	–	43.8
	Ours (FPGM-only 40%)	✓	**93.68** (±0.32)	**93.74** (±0.10)	**-0.16**	**1.21E8**	**52.3**

Table 1. Comparison of pruned ResNet on CIFAR-10. In "Fine-tune?" column, "✓" and "✗" indicates whether to use the pre-trained model as initialization or not, respectively. The "Acc. ↓" is the accuracy drop between pruned model and the baseline model, the smaller, the better.

the end of every training epoch. Unlike previous work [21], sensitivity analysis is not essential in FPGM to achieve good performances, which will be demonstrated in later sections.

Apart from FPGM only criterion, we also use a mixture of FPGM and previous norm-based method [15] to show that FPGM could serve as a supplement to previous methods. FPGM only criterion is denoted as "FPGM-only", the criterion combining the FPGM and norm-based criterion is indicated as "FPGM-mix". "FPGM-only 40%" means 40% filters of the layer are selected with FPGM only, while "FPGM-mix 40%" means 30% filters of the layer are selected with FPGM, and the remaining 10% filters are selected with norm-based criterion [15]. We compare FPGM with previous acceleration algorithms, e.g., MIL [5], PFEC [21], CP [16], ThiNet [25], SFP [15], NISP [39], Rethinking [38]. Not surprisingly, our FPGM method achieves the state-of-the-art result.

4.2. Single-Branch Network Pruning

VGGNet on CIFAR-10. As the training setup is not publicly available for [21], we re-implement the pruning procedure and achieve similar results to the original paper. The result of pruning pre-trained and scratch model is shown in Table 3 and Table 4, respectively. Not surprisingly, FPGM achieves better performance than [21] in both settings.

4.3. Multiple-Branch Network Pruning

ResNet on CIFAR-10. For the CIFAR-10 dataset, we test our FPGM on ResNet-20, 32, 56 and 110 with two different pruning rates: 30% and 40%.

As shown in Table 1, our FPGM achieves the state-of-the-art performance. For example, MIL [5] without fine-tuning accelerates ResNet-32 by 31.2% speedup ratio with 1.59% accuracy drop, but our FPGM without fine-tuning achieves 53.2% speedup ratio with even 0.19% accuracy improvement. Comparing to SFP [15], when pruning 52.6% FLOPs of ResNet-56, our FPGM has only 0.66% accuracy drop, which is much less than SFP [15] (1.33%). For pruning the pre-trained ResNet-110, our method achieves a much higher (52.3% v.s. 38.6%) acceleration ratio with 0.16% performance increase, while PFEC [21] harms the performance with lower acceleration ratio. These results demonstrate that FPGM can produce a more compressed model with comparable or even better performances.

ResNet on ILSVRC-2012. For the ILSVRC-2012

Depth	Method	Fine-tune?	Baseline top-1 acc.(%)	Accelerated top-1 acc.(%)	Baseline top-5 acc.(%)	Accelerated top-5 acc.(%)	Top-1 acc. ↓(%)	Top-5 acc. ↓(%)	FLOPs↓(%)
18	MIL [5]	✗	69.98	66.33	89.24	86.94	3.65	2.30	34.6
	SFP [15]	✗	**70.28**	67.10	**89.63**	87.78	3.18	1.85	**41.8**
	Ours (FPGM-only 30%)	✗	**70.28**	67.78	**89.63**	88.01	2.50	1.62	**41.8**
	Ours (FPGM-mix 30%)	✗	**70.28**	**67.81**	**89.63**	**88.11**	**2.47**	**1.52**	**41.8**
	Ours (FPGM-only 30%)	✓	**70.28**	68.34	**89.63**	**88.53**	1.94	**1.10**	**41.8**
	Ours (FPGM-mix 30%)	✓	**70.28**	**68.41**	**89.63**	88.48	**1.87**	1.15	**41.8**
34	SFP [15]	✗	**73.92**	71.83	**91.62**	90.33	2.09	1.29	**41.1**
	Ours (FPGM-only 30%)	✗	**73.92**	71.79	**91.62**	**90.70**	2.13	**0.92**	**41.1**
	Ours (FPGM-mix 30%)	✗	**73.92**	**72.11**	**91.62**	90.69	**1.81**	0.93	**41.1**
	PFEC [21]	✓	73.23	72.17	–	–	**1.06**	–	24.2
	Ours (FPGM-only 30%)	✓	**73.92**	72.54	**91.62**	**91.13**	1.38	**0.49**	**41.1**
	Ours (FPGM-mix 30%)	✓	**73.92**	**72.63**	**91.62**	91.08	1.29	0.54	**41.1**
50	SFP [15]	✗	**76.15**	74.61	**92.87**	92.06	1.54	0.81	41.8
	Ours (FPGM-only 30%)	✗	**76.15**	**75.03**	**92.87**	**92.40**	**1.12**	**0.47**	42.2
	Ours (FPGM-mix 30%)	✗	**76.15**	74.94	**92.87**	92.39	1.21	0.48	42.2
	Ours (FPGM-only 40%)	✗	**76.15**	74.13	**92.87**	91.94	2.02	0.93	**53.5**
	ThiNet [25]	✓	72.88	72.04	91.14	90.67	0.84	0.47	36.7
	SFP [15]	✓	**76.15**	62.14	**92.87**	84.60	14.01	8.27	41.8
	NISP [39]	✓	–	–	–	–	0.89	–	44.0
	CP [16]	✓	–	–	92.20	90.80	–	1.40	50.0
	Ours (FPGM-only 30%)	✓	**76.15**	**75.59**	**92.87**	**92.63**	**0.56**	0.24	42.2
	Ours (FPGM-mix 30%)	✓	**76.15**	75.50	**92.87**	92.63	0.65	**0.21**	42.2
	Ours (FPGM-only 40%)	✓	**76.15**	74.83	**92.87**	92.32	1.32	0.55	**53.5**
101	Rethinking [38]	✓	**77.37**	75.27	–	–	2.10	–	**47.0**
	Ours (FPGM-only 30%)	✓	**77.37**	**77.32**	**93.56**	**93.56**	**0.05**	**0.00**	42.2

Table 2. Comparison of pruned ResNet on ILSVRC-2012. "Fine-tune?" and "acc. ↓" have the same meaning with Table 1.

Model \ Acc (%)	Baseline	Pruned w.o. FT	FT 40 epochs	FT 160 epochs
PFEC [21]	93.58 (±0.03)	77.45 (±0.03)	93.22 (±0.03)	93.28 (±0.07)
Ours	93.58 (±0.03)	**80.38** (±0.03)	**93.24** (±0.01)	**94.00** (±0.13)

Table 3. Pruning pre-trained VGGNet on CIFAR-10. "w.o." means "without" and "FT" means "fine-tuning" the pruned model.

Model	SA	Baseline	Pruned From Scratch	FLOPs↓(%)
PFEC [21]	Y	93.58 (±0.03)	93.31 (±0.02)	34.2
Ours	Y	93.58 (±0.03)	**93.54** (±0.08)	34.2
Ours	N	93.58 (±0.03)	93.23 (±0.13)	**35.9**

Table 4. Pruning scratch VGGNet on CIFAR-10. "SA" means "sensitivity analysis". Without sensitivity analysis, FPGM can still achieve comparable performances comparing to [21]; after introducing sensitivity analysis, FPGM can surpass [21].

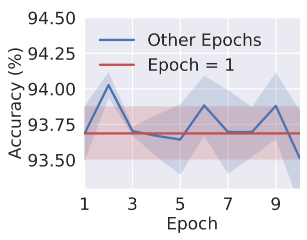
dataset, we test our FPGM on ResNet-18, 34, 50 and 101 with pruning rates 30% and 40%. Same with [15], we do not prune the projection shortcuts for simplification.

Table 2 shows that FPGM outperforms previous methods on ILSVRC-2012 dataset, again. For ResNet-18, pure

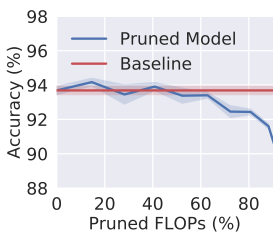
Model	Baseline time (ms)	Pruned time (ms)	Realistic Acc.(%)	Theoretical Acc.(%)
ResNet-18	37.05	26.77	27.7	41.8
ResNet-34	63.89	45.24	29.2	41.1
ResNet-50	134.57	83.22	38.2	53.5
ResNet-101	219.70	147.45	32.9	42.2

Table 5. Comparison on the theoretical and realistic acceleration. Only the time consumption of the forward procedure is considered.

FPGM without fine-tuning achieves the same inference speedup with [15], but its accuracy exceeds by 0.68%. FPGM-only with fine-tuning could even gain 0.60% improvement over FPGM-only without fine-tuning, thus exceeds [15] by 1.28%. For ResNet-50, FPGM with fine-tuning achieves more inference speedup than CP [16], but our pruned model exceeds their model by 0.85% on the accuracy. Moreover, for pruning a pre-trained ResNet-101, FPGM reduces more than 40% FLOPs of the model without top-5 accuracy loss and only negligible (0.05%) top-1 accuracy loss. In contrast, the performance degradation is 2.10% for Rethinking [38]. Compared to the norm-based criterion, Geometric Median (GM) explicitly utilizes the relationship between filters, which is the main cause of its superior per-



(a) Different pruning intervals



(b) Different pruned FLOPs

Figure 4. Accuracy of ResNet-110 on CIFAR-10 regarding different hyper-parameters. Solid line and shadow denotes the mean values and standard deviation of three experiments, respectively.

formance.

To compare the theoretical and realistic acceleration, we measure the forward time of the pruned models on one GTX1080 GPU with a batch size of 64. The results [4] are shown in Table 5. As discussed in the above section, the gap between the theoretical and realistic model may come from the limitation of IO delay, buffer switch, and efficiency of BLAS libraries.

4.4. Ablation Study

Influence of Pruning Interval In our experiment setting, the interval of pruning equals to one, *i.e.*, we conduct our pruning operation at the end of every training epoch. To explore the influence of pruning interval, we change the pruning interval from one epoch to ten epochs. We use the ResNet-110 under pruning rate 40% as the baseline, as shown in Fig. 4(a). The accuracy fluctuation along with the different pruning intervals is less than 0.3%, which means the performance of pruning is not sensitive to this parameter. Note that fine-tuning this parameter could even achieve better performance.

Varying Pruned FLOPs We change the ratio of Pruned FLOPs for ResNet-110 to comprehensively understand FPGM, as shown in Fig. 4(b). When the pruned FLOPs is 18% and 40%, the performance of the pruned model even exceeds the baseline model without pruning, which shows FPGM may have a regularization effect on the neural network.

Influence of Distance Type We use ℓ_1-norm and cosine distance to replace the distance function in Equation 3. We use the ResNet-110 under pruning rate 40% as the baseline, the accuracy of the pruned model is 93.73 ± 0.23 %. The accuracy based on ℓ_1-norm and cosine distance is 93.87 ± 0.22 % and 93.56 ± 0.13, respectively. Using ℓ_1-norm as the distance of filter would bring a slightly better result, but cosine distance as distance would slightly harm the performance of the network.

[4]Optimization of the addition of ResNet shortcuts and convolutional outputs would also affect the results.

Combining FPGM with Norm-based Criterion We analyze the effect of combining FPGM and previous norm-based criterion. For ResNet-110 on CIFAR-10, FPGM-mix is slightly better than FPGM-only. For ResNet-18 on ILSVRC-2012, the performances of FPGM-only and FPGM-mix are almost the same. It seems that the norm-based criterion and FPGM together can boost the performance on CIFAR-10, but not on ILSVRC-2012. We believe that this is because the two requirements for the norm-based criterion are met on some layers of CIFAR-10 pre-trained network, but not on that of ILSVRC-2012 pre-trained network, which is shown in Figure 3.

4.5. Feature Map Visualization

We visualize the feature maps of the first layer of the first block of ResNet-50. The feature maps with red titles (7,23,27,46,56,58) correspond to the selected filter activation when setting the pruning rate to 10%. These selected feature maps contain outlines of the bamboo and the panda's head and body, which can be replaced by remaining feature maps: (5,12,16,18,22, *et al.*) containing outlines of the bamboo, and (0,4,33,34,47, *et al.*) containing the outline of panda.

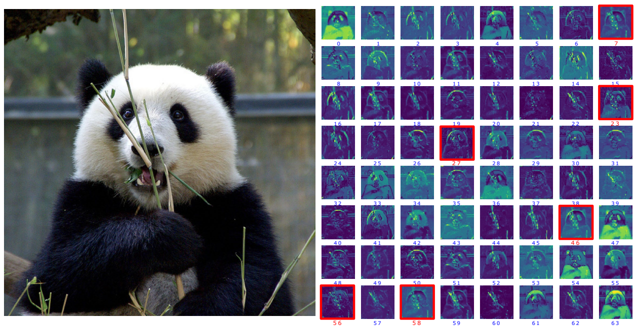


Figure 5. Input image (left) and visualization of feature maps (right) of ResNet-50-conv1. Feature maps with red bounding boxes are the channels to be pruned.

5. Conclusion and Future Work

In this paper, we elaborate on the underlying requirements for norm-based filter pruning criterion and point out their limitations. To solve this, we propose a new filter pruning strategy based on the geometric median, named FPGM, to accelerate the deep CNNs. Unlike the previous norm-based criterion, FPGM explicitly considers the mutual relations between filters. Thanks to this, FPGM achieves the state-of-the-art performance in several benchmarks. In the future, we plan to work on how to combine FPGM with other acceleration algorithms, e.g., matrix decomposition and low-precision weights, to push the performance to a higher stage.

References

[1] M. A. Carreira-Perpinán and Y. Idelbayev. learning-compression algorithms for neural net pruning. In *CVPR*, 2018. 1, 2

[2] H. H. Chin, A. Madry, G. L. Miller, and R. Peng. Runtime guarantees for regression problems. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 269–282. ACM, 2013. 4

[3] M. B. Cohen, Y. T. Lee, G. Miller, J. Pachocki, and A. Sidford. Geometric median in nearly linear time. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 9–21. ACM, 2016. 4

[4] X. Dong, S. Chen, and S. Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 4857–4867, 2017. 2

[5] X. Dong, J. Huang, Y. Yang, and S. Yan. More is less: A more complicated network with less inference complexity. In *CVPR*, 2017. 6, 7

[6] X. Dong and Y. Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[7] A. Dubey, M. Chatterjee, and N. Ahuja. Coreset-based neural network compression. In *ECCV*, 2018. 2

[8] P. T. Fletcher, S. Venkatasubramanian, and S. Joshi. Robust statistics on riemannian manifolds via the geometric median. In *CVPR*, 2008. 2, 3, 5

[9] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient DNNs. In *NIPS*, 2016. 2

[10] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2015. 2

[11] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015. 1, 2

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 5

[13] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 5

[14] Y. He and S. Han. ADC: Automated deep compression and acceleration with reinforcement learning. *arXiv preprint arXiv:1802.03494*, 2018. 2

[15] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, 2018. 1, 2, 5, 6, 7

[16] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 2, 6, 7

[17] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS*, 2015. 2

[18] Q. Huang, K. Zhou, S. You, and U. Neumann. Learning to prune filters in convolutional neural networks. In *WACV*, 2018. 2

[19] J. Kim, S. Park, and N. Kwak. Paraphrasing complex network: Network compression via factor transfer. In *NIPS*, 2018. 2

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. 5

[21] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient ConvNets. In *ICLR*, 2017. 1, 2, 5, 6, 7

[22] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *CVPR*, 2019. 2

[23] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 2

[24] Z. Liu, J. Xu, X. Peng, and R. Xiong. Frequency-domain dynamic pruning for convolutional neural networks. In *NIPS*, 2018. 2

[25] J.-H. Luo, J. Wu, and W. Lin. ThiNet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017. 2, 5, 6, 7

[26] Y. Luo, L. Zheng, T. Guan, J. Yu, and Y. Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. In *CVPR*, 2019. 1

[27] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *ICLR*, 2017. 2

[28] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 5

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 5

[30] B. W. Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018. 3

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 5

[32] S. Son, S. Nah, and K. Mu Lee. Clustering convolutional kernels to compress deep neural networks. In *The European Conference on Computer Vision (ECCV)*, 2018. 2

[33] X. Suau, L. Zappella, V. Palakkode, and N. Apostoloff. Principal filter analysis for guided network compression. *arXiv preprint arXiv:1807.10585*, 2018. 2

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[35] C. Tai, T. Xiao, Y. Zhang, X. Wang, et al. Convolutional neural networks with low-rank regularization. In *ICLR*, 2016. 2

[36] F. Tung and G. Mori. Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *CVPR*, 2018. 2

[37] D. Wang, L. Zhou, X. Zhang, X. Bai, and J. Zhou. Exploring linear relationship in feature map subspace for convnets compression. *arXiv preprint arXiv:1803.05729*, 2018. 2

[38] J. Ye, X. Lu, Z. Lin, and J. Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*, 2018. 2, 3, 6, 7

[39] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. NISP: Pruning networks

using neuron importance score propagation. In *CVPR*, 2018. 1, 2, 6, 7

[40] S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016. 5

[41] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. *arXiv preprint arXiv:1804.03294*, 2018. 2

[42] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE T-PAMI*, 2016. 2

[43] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*, 2017. 2

[44] C. Zhu, S. Han, H. Mao, and W. J. Dally. Trained ternary quantization. In *ICLR*, 2017. 2

[45] F. Zhu, L. Zhu, and Y. Yang. Sim-real joint reinforcement transfer for 3d indoor navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[46] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu. Discrimination-aware channel pruning for deep neural networks. In *NIPS*, 2018. 2

[47] H. Zhuo, X. Qian, Y. Fu, H. Yang, and X. Xue. Scsp: Spectral clustering filter pruning with soft self-adaption manners. *arXiv preprint arXiv:1806.05320*, 2018. 2