Pruning the Unimportant or Redundant Filters? Synergy Makes Better

Yucheng Cai[1], Zhuowen Yin[1], Kailing Guo[1,2*], Xiangmin Xu[1,2,3]
[1]*School of Electronic and Information Engineering, South China University of Technology, Guangzhou, China*
[2]*UBTECH-SCUT Union Laboratory, Guangzhou, China*
[3] *Institute of Modern Industrial Technology of SCUT in Zhongshan, Zhongshan, China*
{nanocyc, yinzwjohn}@gmail.com, {guokl, xmxu}@scut.edu.cn

***Abstract*—Filter pruning is a hot topic in convolutional neural network compression due to its friendliness to hardware implementation. Most pruning methods prune filters according to their importance, *i.e.*, removing the filters that have little effect on the final performance of the network. While from another perspective, some recent works propose to prune upon the redundancy of filters. Filters pruned in this way usually have non-negligible effects on the final performance, whereas those effects could be compensated by the remaining filters. Since importance and redundancy pruning respectively captures the local and global information of a convolutional layer, which are mutually complementary to some extent, we propose a new pruning criterion that synergies both of those previous pruning criteria to make full use of the filter information. Comprehensive experiments on benchmark image classification datasets show the effectiveness of our proposed pruning criterion.**

***Index Terms*—convolutional network, network pruning, norm-based criterion, distance-based criterion**

I. Introduction

Many deep convolutional neural networks (CNNs) (*e.g.*, VGGNet [1], ResNet [2]) have been developed in recent years and achieved great success in various computer vision tasks. However, most of these networks have millions of floating-point parameters, which hinders their implementation on terminal devices with limited computing power and memory space. For example, ResNet-152 has approximately 60 million float parameters with 231MB storage space. This hinders developing the integration of CNN with the Internet of Things and many traditional industries. Numerous network compression methods have been proposed to reduce the computing power and memory space required for CNNs, such as pruning [3]–[5], quantization [6], [7] and distillation [8], [9]. In this paper, we focus on pruning methods, which cut out unimportant parameters or filters. Pruning individual parameters makes the filters unstructured. It would be hard to deploy such strategies on existing hardware and benefit from the commonly used hardware acceleration library Basic Linear Algebra Subprogram (BLAS). Filter pruning methods are attracting more attention because they are more friendly to implement on hardware.

Previous works on filter pruning could be roughly divided into importance-based pruning and redundancy-based pruning. For importance-based pruning methods, there are different ways to define filters' importance. Intuitively, the norms of the filters, *e.g.* ℓ_1-norm [10] and ℓ_2-norm [3], are employed as their importance. Other than that, instead of directly measuring filter importance by norms, some works resort to the closely related modules of filters, like introducing sparsity on the scaling parameters of batch normalization (BN) layers [11], [12] or using the average percentage of zeros of activation to represent filter importance [13]. These methods focus on individual filter information yet have neglected the relations among filters. Based on filters' mutual relations, an important filter can be pruned if it is replaceable. Therefore, from this perspective, Filter Pruning via Geometric Median (FPGM) [14] utilizes geometric median in Euclidean space as a metric to prune redundant filters that other filters can replace. Compared to importance-based pruning methods, this redundancy-based method captures global information of filters and outperforms importance-based methods.

According to FPGM [14], it seems the answer to the question "Pruning the unimportant or redundant filters?" is the latter. However, considering that the importance and redundancy criteria are complementary to some extent, we synergize them to make a new criterion and put forward our answer, "Synergy makes better". In [14], a combination strategy called "FPGM-mix" is also studied. Nonetheless, they solely prune the filters in successive steps with the two criteria respectively [1], which is hard to obtain a good trade-off between the two criteria. It is also verified in the experiment results in [14] since they show that "FPGM-mix" can not consistently outperform "FPGM-only" (only using the

This work was supported in part by the National Natural Science Foundation of China under Grant 61802131, Grant U1801262, and Grant 61972163, in part by the Natural Science Foundation of Guangdong Province, China, under Grant 2018A030313474, Grant 2020A1515010781, and Grant 2019B010154003, in part by Key-Area Research and Development Program of Guangdong Province, China, under grant 2019B010154003, in part by the Foundation for Characteristic Innovation in Higher Education of Guangdong Province, China, under grant 2019KTSCX112, in part by the Guangzhou Key Laboratory of Body Data Science, under Grant 201605030011, in part by Science and Technology Project of Zhongshan, under Grant 2019AG024, and in part by the Fundamental Research Funds for the Central Universities, SCUT, under Grant 2019PY21, and Grant 2019MS028.

*Corresponding author

[1]The combination strategy is not stated in detail in the original paper. We get it from the official released code: https://github.com/he-y/filter-pruning-geometric-median

FPGM criterion). In this paper, we propose a new criterion that combines importance and redundancy holistically. We adopt ℓ_2-norm to measure filter importance and distance to the geometric median to judge filter redundancy. Since the two criteria are usually not in the same magnitude order, we normalize them by dividing them by their maximums before their combination. Experiment results show that this normalization strategy is critical for the performance of the proposed pruning criterion.

Take the norm-based criterion as an absolute importance factor of filters. The distance-based criterion is a relative importance factor of filters. We name our method Pruning with Absolute and Relative Importance (PARI). Fig. 1 illustrates the pruning criteria' difference using a toy model (a point on a two-dimensional position represents a filter). Filters with darker colors have larger pruning criterion values. Fig. 1 indicates that our method prunes regarding both norm-based and distance-based criteria and is thus doing a trade-off between them upon each filter.

We conduct experiments on benchmark image classification datasets CIFAR-10 [15], CIFAR-100 [15], and ILSVRC-2012 [16] with different network architecture. Experiment results demonstrate that PARI is an effective filter pruning method. We also show that PARI is insensitive to the combination trade-off parameter.

II. Related Works

A. Weight Pruning and Filter Pruning

Most previous works on CNNs pruning could be divided into two categories as weight pruning and filter pruning.

Weight pruning works eliminate specific weight parameters. Considering the p-norm magnitude as a criterion of the importance of weight parameters, Han et al. [17] prune the parameters with ℓ_1 or ℓ_2-norm magnitude less than a threshold. Ding et al. [18] proposed a systematic pruning scheme by introducing global sparsity. Zhang et al. [19] presented a systematic weight pruning framework using alternating direction method of multipliers. However, all these methods would result in unstructured models. Thus, they need sophisticated hardware of software implementations to achieve actual acceleration.

Filter pruning, which prunes an entire filter, is friendly to hardware implementation and becomes the dominant network compression method. Focusing on the filter itself, "smaller-norm-less-important" is intuitively adopted as a pruning criterion, e.g., Soft Filter Pruning (SFP) [3]. Recent works prune filters utilizing related modules or extra information. Channel pruning (CP) [20] and ThiNet [21] select the pruned channels based on minimizing reconstruction error. Modern CNNs usually have BN layers, which apply an affine transformation to the convolution layer's output. Since a smaller scaling parameter indicates that the corresponding convolution filter's effect is more negligible, the works [11] and [12] enforce the scaling parameter of BN to be sparse and prune filters with a small scaling parameter. Noticing that the activation layer's output influences the following convolutional layer, network trimming [13] proposes to prune zero activation neurons. Extra discriminative information [22], [23] is utilized to guide the pruning process. Importance score propagation [24] and Taylor expansions [25] prune filters according to their contribution to the final loss. Saliency-Adaptive Sparsity Learning (SASL) combines consumed computational resources with Taylor expansion to measure the saliency of a filter. Straight-through estimator [26] and differentiable Markov channel pruning [27] directly search the optimal sub-structure. Without resorting to other modules or information, FPGM goes back to the filters themselves by using a geometric median-induced criterion. Lee et al. [28], on the other hand, measures the importance of channels based on gradients of mutual information. In this paper, we try to make the best use of filter information and go far from FPGM by combining norm-based and geometric median-induced criteria. Note that our method can be further combined with methods that use extra modules or information.

B. Other DNN Accelerating Method

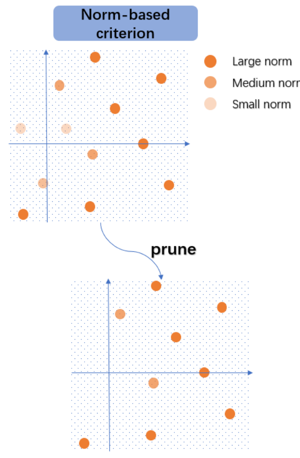
Apart from pruning, there are several other kinds of methods to compress and accelerate deep neural networks. Architecture Design manually devises lightweight networks [29], [30] and achieves excellent success these years. Low-Cost Collaborative Network (LCCN) [31] proposes a more complicated architecture but with less inference complexity by predicting the zero elements and skipping some calculation of high-cost convolution. Low-rank decomposition [32] removes redundancies according to the rank of the weight matrix. Distillation [9] distills knowledge from a sophisticated teacher network to a small student network. Network quantization [33] compress the model from full-precision to low-bit with certain quantization strategy. Neural Architecture Search (NAS), which intends to obtain an effective network by searching architecture in a pre-defined space, has been used to search efficient compact networks, e.g., EfficientNets [34] and MobileNetV3 [35]. NAS is also used to search the hyper-parameters of pruning for better performance, e.g., Learning Filter Pruning Criteria (LFPC) [36] and Differentiable Sparsity Allocation (DSA) [37]. To make a better balance between training efficiency and architecture discrimination in the NAS-based pruning method, propose a new strategy of parameter sharing, Affine Parameter Sharing (APS) [38] is proposed.

Besides, it is worth noting that our approach is parallel to some of these methods, which means that our method can be combined with some of them to achieve better performance.

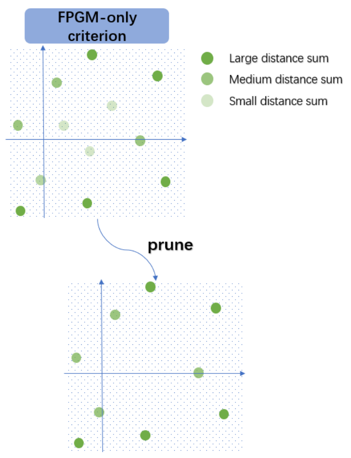
III. Methodology

A. Preliminaries and Definitions

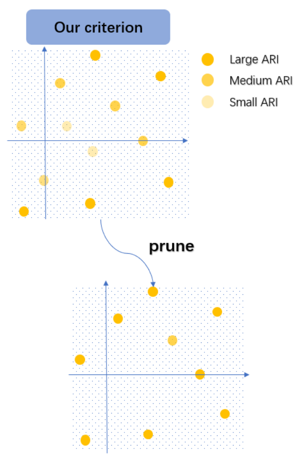
A neural network with L layers is composed of the weights $\mathcal{W} = \{\mathcal{W}^{(i)} \in \mathbb{R}^{N_{i+1}\times N_i \times K_i \times K_i}, 0 \leq i \leq L\}$, where N_i and N_{i+1} respectively represent the input and output channels of the i_{th} layer and K_i is the kernel size. The j_{th} filter of the



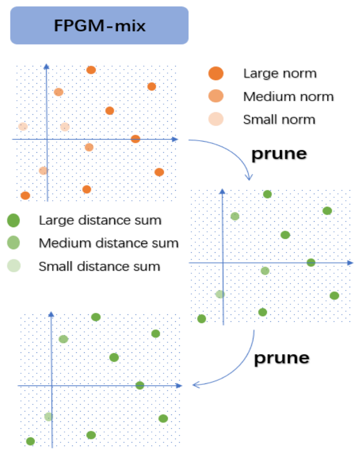
(a) Norm-based pruning



(b) FPGM-only pruning



(c) PARI



(d) FPGM-mix pruning

Fig. 1: An illustration of four different criteria. Filters in a single layer are represented by dots in these diagrams, in which color accounts for the type of the criterion used and transparency means the magnitude defined in the criterion.

i_{th} layer is denoted by $\mathcal{F}_{i,j} \in \mathbb{R}^{N_i \times K_i \times K_i}$, and the weights of the i_{th} layer $\mathcal{W}^{(i)}$ is represented by $\{\mathcal{F}_{i,j}, 1 \leq j \leq N_i\}$.

B. Absolute Importance and Relative Importance

"Smaller-norm-less-important" is an intuitive criterion for pruning filters. To be compatible with geometric median-based criterion defined based on ℓ_2-norm, we utilize ℓ_2-norm to evaluate the absolute importance of each filter as follows:

$$I_a(i,j) = \|\mathcal{F}_{i,j}\|_2 = \sqrt{\sum_{n=1}^{N_i}\sum_{k_1=1}^{K_i}\sum_{k_2=1}^{K_i} |\mathcal{F}_{i,j}(n,k_1,k_2)|^2}. \quad (1)$$

The geometric median is a robust estimator of centrality for Euclidean space data points, which get the common information of all the data points [14]. Considering a filter $F_{i,j}$ as a point in Euclidean space, the definition of geometric median for the i_{th} layer is given by

$$x^{GM} = \underset{x \in \mathbb{R}^{N_i \times K_i \times K_i}}{\arg\min} \sum_{j' \in [1, N_{i+1}]} ||x - \mathcal{F}_{i,j'}||_2. \quad (2)$$

A filter near the geometric median shares similar common information with the geometric median and can be thus considered relatively less important. However, since calculating the geometric median is a time-consuming task, yet what we need is to find the nearest point to geometric median, we can instead compute the sum of the distance from one filter to others to find the relative least important filter. This geometric median-induced criterion is given by

$$I_r(i,j) = \sum_{j' \in [1, N_{i+1}]} ||\mathcal{F}_{i,j} - \mathcal{F}_{i,j'}||_2. \quad (3)$$

C. The Proposed Pruning Criterion

Without misunderstanding, we use I_a and I_r for short to denote $I_a(i,j)$ and $I_r(i,j)$, respectively. Fig. 2 illustrates the Kernel Distribution Estimation (KDE) [39] of I_a and I_r of filters in several chosen layers of ResNet-56 on CIFAR-10. We can see that the magnitude orders of I_a and I_r are different in the same layer and vary significantly in different layers, resulting in one of them dominating the filter selection. In order to eliminate the influence of magnitude, we normalize both I_a and I_r by dividing them by their maximum values, respectively, which maps their value between 0 and 1, given that they are both positive numbers. The process is shown in Eq. (4) and Eq. (5), where $I_{a,max}$ stands for the maximum value of I_a in the current layer, and accordingly, $I_{r,max}$ stands for the maximum value of I_r.

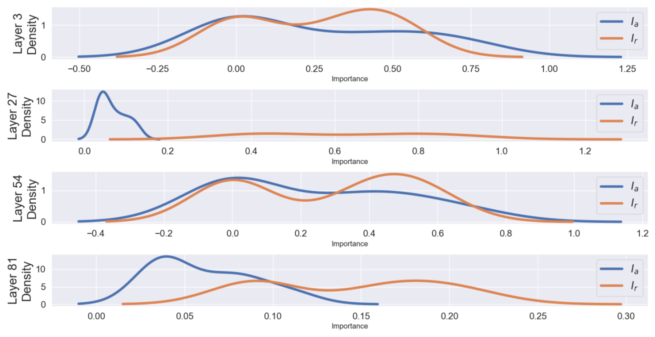


Fig. 2: KDE of the distribution of I_r and I_a of filters in layers 3, 27, 54, and 81 of ResNet-56 on CIFAR-10.

$$\hat{I}_a = I_a / I_{a,max}. \quad (4)$$

$$\hat{I}_r = I_r / I_{r,max}. \quad (5)$$

Then we introduce a trade-off parameter $w \in [0,1]$ that indicates the dominance of I_r, where higher w leads to more consideration of I_r. we therefore obtain a uniform metric represented as below.

$$I_{ar} = (1-w) \times \hat{I}_a + w \times \hat{I}_r. \quad (6)$$

The detailed setting of w is given in Section III-D

D. Training Scheme

In this subsection, we devise a training scheme to work with the proposed pruning criterion. Following [14], we use a uniform pruning rate R for all the layers so that we only need one hyper-parameter to balance the acceleration and accuracy. Specifically, for the i_{th} layer with N_{i+1} filters, it means $N_{i+1}R$ filters are pruned. We first prune filters with the lowest I_{ar} values in each layer after initialization. Then, we train the model with stochastic gradient descent (SGD) by setting the gradients of the pruned filters to be zeros. At the end of each training epoch, we re-compute the I_{ar} values and prune filters according to the new I_{ar}. The details of the training scheme is described in Algorithm 1.

Algorithm 1 Algorithm Description of PARI

```
Input: Training data $X$, pruning rate $R$, and trade-off parameter $w$.
 1: Initialize: Model parameters $\mathcal{W} = \{\mathcal{W}^{(i)}, 0 \le i \le L\}$.
 2: for $i = 1, \cdots, L$ do
 3:     Select $N_{i+1}R$ filters with lowest values of $I_{ar}$.
 4:     Zeroize the selected filters and generate a binary selection mask $M_i$.
 5: end for
 6: for $epoch = 1, \cdots, epoch_{max}$ do
 7:     For every iteration, compute the gradients of each layer and zeroize the gradients of the selected filters according to the masks $M_i's$, and then update the parameter $\mathcal{W}$ with SGD.
 8:     for $i = 1, \cdots, L$ do
 9:         Compute the value $I_{ar}$ of each filter by Eq. (6).
10:         Select $N_{i+1}R$ filters with lowest values of $I_{ar}$.
11:         Zeroize the selected filters and generate a binary selection mask $M_i$.
12:     end for
13: end for
14: Obtain pruned model $\mathcal{W}^*$.
Output: Parameters $\mathcal{W}^*$ of the compressed model.
```

E. Computational Complexity and Theoretical Acceleration

The complexity of computing the ℓ_2 norm of a filter (Eq. (1)) is $O(N_iK_i^2)$ and the complexity of computing I_r (Eq. (3)) is $O(N_{i+1}N_iK_i^2)$. Thus, the total complexity of computing I_{ar} is $O((N_{i+1}+1)N_iK_i^2)$, which is the same order as FPGM since N_{i+1} is usually much larger than 1. Considering that we just compute I_ar at the end epoch, the extra computation burden brought by our method is negligible compared to the total training procedure.

Since convolution occupies most amount of computation in CNNs, we approximately analyze the acceleration solely with convolution. Suppose the input and output feature maps of the i_{th} layer are of the size $N_i \times H_i \times W_i$ and $N_{i+1} \times H_{i+1} \times W_{i+1}$, respectively. Because the output of the current layer is the input of the next layer, both input and output channels are reduced after pruning. Then, the FLOPs of the i_{th} layer is changed from $N_iK_i^2N_{i+1}H_iW_i$ into $N_{i(1-R)}K_i^2N_{i+1}(1-R)H_iW_i$. Therefore, the theoretical acceleration during inference is $1/(1-R)^2$.

IV. Experiments

A. Datasets and Experimental Settings

Datasets. In our experiments, three datasets (CIFAR-10 [15], CIFAR-100 [15], and ILSVRC-2012 [16]) are leveraged to verify the effectiveness of the proposed method. CIFAR-10 is a small image dataset for CNN training and testing, which contains 50,000 training and 10,000 testing color images of 32×32 in 10 categories. CIFAR-100 is similar to CIFAR-10 except that it contains 100 categories. ILSVRC-2012, as a large-scale classification dataset, contains 1.28 million training images and 50k validation images in 1,000 categories.

Experiment settings. For the CIFAR datasets, we use the same hyper-parameter setup as [40] and the training schedule of [41]. We train our models for 200 epochs and the learning rate decays to one tenth with the schedule of epoch 60, 120 and 160. For ILSVRC-2012, we follow the default parameter settings of [2]. The data augmentation strategies for ILSVRC-2012 are the same as PyTorch official examples. For ResNets, we do not prune the projection shortcuts since the shortcuts have negligible influence on the overall cost. We conduct each following experiment three times and report the mean and standard deviation of the classification accuracy. Our proposed pruning strategy is plug-in to existing training methods, so that we can prune neural networks by training from scratch without fine-tuning. Note that the performance of the pruned network can be further improved with fine-tuning but we do not adopt in this paper since we focus more on the intrinsic effect of the pruning strategy.

B. Results on Small Scale Datasets CIFAR-10 and CIFAR-100

For CIFAR-10, we test PARI with ResNet-32, 56, and 110, and VGG-16. For fair comparison with other methods, we set the pruning rate to 40% for ResNet, and 20% for VGG-16. We also test PARI on CIFAR-100 to show its generalization performance with ResNet-20 and 56, and the pruning rate is set to 30%.

TABLE I shows the results of ResNet on CIFAR-10. "Acc.$\downarrow$" indicates the accuracy drop between the baseline model and the pruned model. "FLOPS$\downarrow$" indicates the reduced FLOPs. For PARI, the number after 40% is the value of w. For example, "PARI(40%-0.3)" indicates that $w = 0.3$. PARI achieves better performance than other state-of-the-art pruning methods. Our method accelerates ResNet-32 by reducing 53.2% FLOPs, which is larger than LFPC and SFP, and is the same as FPGM. The accuracy of ResNet-32 pruned by PARI with $w = 0.3/0.7$ is better than the other methods. For ResNet-56, the two settings of PARI achieve the second and third best performance. Note that LFPC is a criterion searching method and needs re-training after pruning, which is more complicated than our method. PARI is simple but achieves considerable results, and it outperforms LFPC at the other

TABLE I. Comparison of pruned ResNet on CIFAR-10

Model	Method	Baseline Acc. (%)	Pruned Acc. (%)	Acc.↓(%)	FLOPs	FLOPs↓ (%)
ResNet-32	LFPC [36]	92.63 ± 0.70	92.12 ± 0.32	0.51	3.27E7	52.6
	SFP [3]	92.63 ± 0.70	92.08 ± 0.08	0.55	4.03E7	41.5
	FPGM-only 40% [14]	92.63 ± 0.70	91.93 ± 0.03	0.7	3.23E7	53.2
	FPGM-mix 40% [14]	92.63 ± 0.70	91.91 ± 0.21	0.72	3.23E7	53.2
	PARI(40%-0.3)	92.63 ± 0.70	92.21 ± 0.18	0.42	3.23E7	53.2
	PARI(40%-0.7)	92.63 ± 0.70	$\mathbf{92.32} \pm 0.3$	**0.31**	3.23E7	53.2
ResNet-56	LFPC [36]	93.59 ± 0.58	$\mathbf{93.34} \pm 0.08$	**0.25**	5.91E7	52.9
	CP [20]	92.80	90.90	1.90	-	34.2
	SFP [3]	93.59 ± 0.58	92.26 ± 0.31	1.33	5.94E7	52.6
	DSA [37]	93.59 ± 0.58	92.91	0.68	-	47.8
	FPGM-only 40% [14]	93.59 ± 0.58	92.93 ± 0.49	0.66	5.94E7	52.6
	FPGM-mix 40% [14]	93.59 ± 0.58	92.89 ± 0.32	0.70	5.94E7	52.6
	PARI(40%-0.3)	93.59 ± 0.58	93.05 ± 0.25	0.54	5.94E7	52.6
	PARI(40%-0.7)	93.59 ± 0.58	92.98 ± 0.20	0.61	5.94E7	52.6
ResNet-110	LCCN [31]	93.63	93.44	0.19	-	34.2
	LFPC [36]	93.68 ± 0.32	93.79 ± 0.38	-0.11	1.01E8	60.3
	SFP [3]	93.68 ± 0.32	92.26 ± 0.31	1.33	1.50E8	40.8
	FPGM-only 40% [14]	93.68 ± 0.32	93.73 ± 0.23	0.66	1.21E8	52.3
	FPGM-mix 40% [14]	93.68 ± 0.32	93.85 ± 0.11	-0.17	1.21E8	52.3
	PARI(40%-0.3)	93.68 ± 0.32	93.70 ± 0.34	-0.02	1.21E8	52.3
	PARI(40%-0.7)	93.68 ± 0.32	$\mathbf{94.03} \pm 0.07$	**-0.35**	1.21E8	52.3

TABLE II. Comparison of pruned ResNet on CIFAR-100

Model	Method	Baseline Acc. (%)	Pruned Acc. (%)	Acc.↓ (%)	FLOPs	FLOPs↓ (%)
ResNet-20	SFP [3]	68.18 ± 0.38	65.43 ± 0.08	2.75	2.43E7	42.2
	FPGM-only 30% [14]	68.18 ± 0.38	65.56 ± 0.12	2.62	2.43E7	42.2
	FPGM-mix 30% [14]	68.18 ± 0.38	65.36 ± 0.70	2.82	2.43E7	42.2
	PARI(30%-0.3)	68.18 ± 0.38	$\mathbf{66.03} \pm 0.14$	**2.15**	2.43E7	42.2
	PARI(30%-0.7)	68.18 ± 0.38	65.62 ± 0.38	2.56	2.43E7	42.2
ResNet-56	SFP [3]	70.43 ± 0.34	69.26 ± 0.58	1.17	7.40E7	41.1
	FPGM-only 30% [14]	70.43 ± 0.34	69.44 ± 0.32	0.99	7.40E7	41.1
	FPGM-mix 30% [14]	70.43 ± 0.34	69.53 ± 0.30	0.90	7.40E7	41.1
	PARI(30%-0.3)	70.43 ± 0.34	$\mathbf{69.73} \pm 0.28$	**0.70**	7.40E7	41.1
	PARI(30%-0.7)	70.43 ± 0.34	69.54 ± 0.08	0.89	7.40E7	41.1

depths. For ResNet-110, our accuracy could reach 94.03% when $w = 0.7$, which outperforms other methods of the same depth level. SFP and FPGM-only respectively prune the filters according to importance and redundancy. PARI outperforms both of them, which suggests that both criteria are useful for pruning. Although FPGM-mix also combines both importance and redundancy, the combination is in separated steps. PARI outperforms FPGM-mix, which shows that combining them in a unified way is better than separated combination. It's worth noting that when the architecture of the network gets deeper, the accuracy drop of our method gets lower, which indicates that our method could be more efficient when implemented on deeper neural networks.

The results on CIFAR-100 are reported in TABLE II. We can see that, PARI still achieves the best performance on CIFAR-100. Note that we do not tune the trade-off parameter w but just set it the same as CIFAR-10.

Furthermore, to test the performance of our proposed method on single branch networks, we train and prune from scratch with VGG-16 on CIFAR-10. We prune 20% of the parameters in VGG-16 following [14]. The result is shown in TABLE III. We can see that PARI also outperforms FPGM in single branch networks.

TABLE III. Comparison of pruned VGG-16 on CIFAR-10

Method	Baseline Acc. (%)	Pruned Acc. (%)	FLOPs↓ (%)
FPGM-20% [14]	93.58 ± 0.03	93.23 ± 0.08	35.9
PARI(20%-0.7)	93.58 ± 0.03	$\mathbf{93.35} \pm 0.11$	35.9

C. Results on Large Scale Dataset ILSVRC-2012

For the ILSVRC-2012 dataset, we use our method to accelerate ResNet-18 and ResNet-50 with the pruning rate set to 30% for all models.

The experimental results are shown in TABLE IV. For ResNet-18, all the compared methods do not need fine-tuning

TABLE IV. Comparison of pruned ResNet on ILSVRC-2012

Model	Method	Top-1 Acc. (%) Baseline	Top-5 Acc. (%) Baseline	Top-1 Acc. (%) Pruned	Top-5 Acc. (%) Pruned	Top-1 Acc.↓ (%)	Top-5 Acc.↓ (%)	FLOPs↓ (%)
ResNet-18	LCCN [31]	69.98	89.24	66.33	86.94	3.65	2.3	34.6
	SFP [3]	70.28	89.63	67.1	87.78	3.18	1.85	41.8
	FPGM-only 30% [14]	70.28	89.63	67.78	88.01	2.5	1.62	41.8
	FPGM-mix 30% [14]	70.28	89.63	67.81	88.11	2.47	1.52	41.8
	PARI(30%-0.3)	70.28	89.63	**67.92**	**88.20**	**2.36**	**1.43**	41.8
	PARI(30%-0.7)	70.28	89.63	67.83	88.07	2.45	1.56	41.8
ResNet-50	ThiNet [21]	72.88	91.14	72.04	90.67	**0.84**	0.47	36.7
	SFP [3]	76.15	92.87	74.61	92.06	1.54	0.81	41.8
	LFPC [36]	76.15	92.87	74.18	91.92	1.97	0.95	60.8
	FPGM-only 30% [14]	76.15	92.87	75.03	92.40	1.12	0.47	42.2
	FPGM-mix 30% [14]	76,15	92.87	74.94	92.39	1.21	0.48	42.2
	PARI(30%-0.3)	76.15	92.87	**75.18**	**92.49**	0.97	**0.38**	42.2
	PARI(30%-0.7)	76.15	92.87	75.08	**92.49**	1.07	**0.38**	42.2

and PARI is the best when $w = 0.3$. Compared to LCCN, it achieves 1.59% accuracy gain while reducing 7.2% of the FLOPs. For ResNet-50, PARI achieves the highest top-1 and top-5 accuracy among all the methods. Although the accuracy drop of ThiNet is a little less than PARI, ThiNet needs extra reconstruction information, fine-tuning, and more FLOPs computation. PARI consistently outperforms SFP and FPGM, which again verifies the effectiveness of the proposed pruning criterion.

D. Parameter Analysis

TABLE V. Classification accuracy drop (%) compared to baseline when pruning different networks on CIFAR-10 with different w

w	ResNet-20	ResNet-32	ResNet-56	ResNet-110
0.1	1.49	0.95	0.62	0.33
0.3	1.91	0.57	0.54	-0.03
0.5	2.02	1.09	0.93	0.34
0.7	1.47	0.31	0.61	-0.35
0.9	1.67	0.78	1.04	1.00

To investigate the effect of w on pruning CNNs of different depths, we conducted a series of experiments on CIFAR-10 and posted the results in TABLE V and Fig. 3. We set the pruning rate of our method to 40%. It can be seen from the results that PARI is insensitive to the parameter w. When the values of w are 0.3 and 0.7, the network can be compressed and accelerated with less accuracy drop. Because we have the same compression ratio as FPGM [14], we take not only the original model but also FPGM [14] for comparison in Fig. 3. As the depth of the layers goes deeper, the accuracy of our method even outperforms the original one, which may be because very deep CNNs tend to overfit small-scale datasets and have more unimportant and redundant filters.

E. Ablation Study

Normalization is applied to make I_r and I_a compatible in PARI. Here we show that normalization is critical for the final performance by comparing with a combination strategy

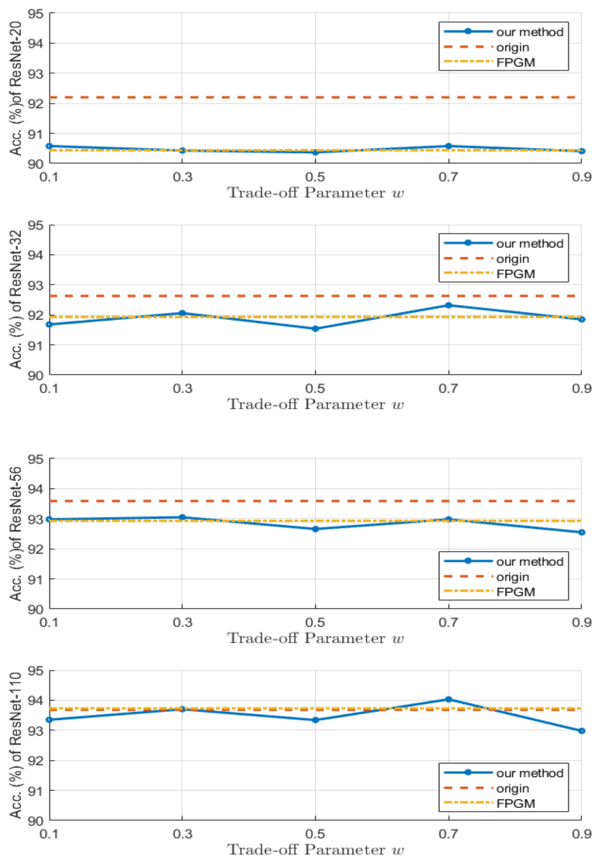


Fig. 3: The effect of different trade-off parameters w

without normalization. We conduct experiments on CIFAR-10 with ResNet-56 under the pruning rate of 40%. For the strategy with normalization, the trade-off parameter w is set to 0.3. The pruning criterion is computed as $I'_{ar} = I_a + w' \times I_r$ if without normalization. We tune w' in a wide range for the strategy without normalization. As shown in TABLE VI, without normalization, there is a decline in classification accuracy no matter how we tune the parameter w'. The best performance of

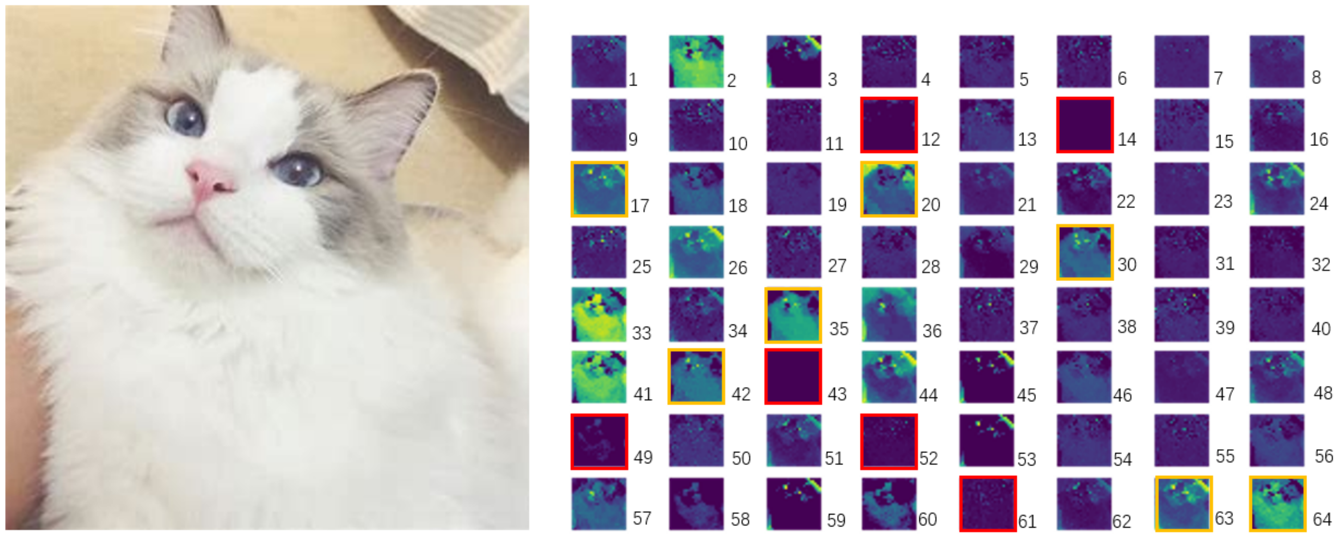


Fig. 4: Feature map visualization of a cat image. The left one is input image and the right ones are the feature maps of ResNet-50-conv1. Feature maps in red bounding boxes and yellow bounding boxes are the filters to be pruned.

the strategy without normalization is even worse than FPGM-only and FPGM-mix, which suggests that normalization is critical for PARI.

TABLE VI. Pruning with and without normalization. "w/o n." means "without normalization" and the following number indicates the value of w'.

Method	Accuracy (%)	Method	Accuracy (%)
w/o n.-0.0001	91.54	FPGM-only	92.93
w/o n.-0.001	92.35	FPGM-mix	92.89
w/o n.-0.01	92.44	PARI	**93.05**
w/o n.-0.1	91.74		
w/o n.-1	92.71		
w/o n.-10	92.74		
w/o n.-100	92.67		
w/o n.-1000	91.73		

F. Realistic Acceleration

To compare the theoretical and realistic acceleration efficiency of different methods, we take the inference time as an indicator and give out experimental results shown in TABLE VII. We evaluate our models on one GTX1080 GPU with the test dataset of CIFAR-10 and a batch size of 128. Not surprisingly, the acceleration ratios of the results are lower than our theoretical expected values. The gap between theoretical and realistic inference time may be caused by buffer switch, IO delay or BLAS. However, it is worthing noting that the deeper the network is, the closer the acceleration ratio is to the theoretical value. It is because the IO delay is nearly the same as the parameters of the deep network increase.

G. Feature Visualization

To show the effectiveness of the proposed method more clearly, we visualize the feature maps of the first convolutional layer of ResNet-50 in Fig. 4. The images with red and yellow boxes are channels to be pruned. Based on our theory, we use red bounding boxes to indicate filters mainly pruned by importance-based criterion and yellow bounding boxes to indicate filters mainly pruned by redundancy-based criterion. The filters with red boxes are almost filled with one single dark color, which has nearly no information within. The filters with yellow boxes carry the information (both outline and details) that can be represented by other images. For example, picture 30 can be roughly represented by the combination of picture 46 and 24, for one of them has the profile of the cat and the other captures the illumination information.

TABLE VII. Comparison between theoretical speedup ratio and realistic speedup ratio

Model	Original time (ms)	Accelerated time (ms)	Realistic speedup ratio	Theoretical speedup ratio
ResNet-20	488	335	31.4%	54.0%
ResNet-32	599	389	35.0%	53.2%
ResNet-56	1131	679	39.9%	52.6%
ResNet-110	1777	1061	40.3%	52.3%

V. Conclusion

In this paper, we explore the potential of pruning CNNs through filter information itself. We propose a synergized criterion for filter pruning that combines the importance indicator ℓ_2 norm and the redundancy indicator distance sum together. The effectiveness of the proposed method is verified by comprehensive experiments with popular network architectures on benchmark datasets. We show that importance and redundancy are both useful for pruning, and their combination makes better.

References

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[3] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018, pp. 2234–2240.

[4] Y. Wang, X. Zhang, L. Xie, J. Zhou, H. Su, B. Zhang, and X. Hu, "Pruning from scratch," in *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 07, 2020, pp. 12 273–12 280.

[5] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2016, pp. 2074–2082.

[6] M. Nagel, M. van Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1325–1334.

[7] P. Yin, J. Lyu, S. Zhang, S. J. Osher, Y. Qi, and J. Xin, "Understanding straight-through estimator in training activation quantized neural nets," in *Proceedings of the International Conference on Learning Representations*, 2019.

[8] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *in arXiv preprint arXiv:1503.02531*, 2015.

[9] S. Hegde, R. Prasad, R. Hebbalaguppe, and V. Kumar, "Variational student: Learning compact and sparser networks in knowledge distillation framework," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 3247–3251.

[10] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *Proceedings of the International Conference on Learning Representations*, 2017.

[11] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *Proceedings of the International Conference on Learning Representations*, 2018.

[12] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2755–2763.

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

[14] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4340–4349.

[15] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," *Univ. Toronto, Toronto, ON, Canada*, vol. 1, no. 4, 2009.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, and e. a. Zhiheng Huang, "Imagenet large scale visual recognition challenge," *in International Journal Computer Vision*, no. 3, pp. 211–252, 2015.

[17] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural network," in *Proceedings of the Advances in Neural Information Processing Systems*, 2015.

[18] X. Ding, G. Ding, X. Zhou, Y. Guo, J. Han, and J. Liu, "Global sparse momentum SGD for pruning very deep neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2019, pp. 6379–6391.

[19] 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," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 184–199.

[20] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1398–1406.

[21] J. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5068–5076.

[22] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel for deep neural networks," in *Proceedings of the Advances in Neural Information Processing Systems*, 2018, pp. 883–894.

[23] N. Gkalelis and V. Mezaris, "Fractional step discriminant pruning: A filter pruning framework for deep convolutional neural networks," in *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops*, 2020, pp. 1–6.

[24] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203.

[25] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 264–11 272.

[26] S. Gao, F. Huang, J. Pei, and H. Huang, "Discrete model compression with resource constraint for deep neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1896–1905.

[27] S. Guo, Y. Wang, Q. Li, and J. Yan, "DMCP: differentiable markov channel pruning for neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1536–1544.

[28] M. K. Lee, S. Lee, S. H. Lee, and B. C. Song, "Channel pruning via gradient of mutual information for light-weight convolutional neural networks," in *Proceedings of the IEEE International Conference on Image Processing*, 2020, pp. 1751–1755.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[30] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 6848–6856.

[31] X. Dong, J. Huang, Y. Yang, and S. Yan, "More is less: A more complicated network with less inference complexity," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1895–1903.

[32] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7370–7379.

[33] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. D. Reid, "Towards effective low-bitwidth convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7920–7928.

[34] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, 2019, pp. 6105–6114.

[35] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[36] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, and Y. Yang, "Learning filter pruning criteria for deep convolutional neural networks acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2006–2015.

[37] X. Ning, T. Zhao, W. Li, and P. Lei, "DSA: More efficient budgeted pruning via differentiable sparsity allocation," in *Proceedings of the European Conference on Computer Vision*, 2020, pp. 592–607.

[38] J. Wang, H. Bai, J. Wu, X. Shi, J. Huang, I. King, M. Lyu, and J. Cheng, "Revisiting parameter sharing for automatic neural channel number search," in *Proceedings of the Advances in Neural Information Processing Systems*, 2020.

[39] S. J. Sheather, "Density estimation," *in Statistical science*, pp. 588–597, 2004.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the European conference on computer vision*, 2016, pp. 630–645.

[41] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*, 2016, pp. 1–12.