Meta Filter Pruning to Accelerate Deep Convolutional Neural Networks

Yang He[1] Ping Liu[1,2] Linchao Zhu[1] Yi Yang[1]
[1]CAI, University of Technology Sydney [2]JD.com
{yang.he-1}@student.uts.edu.au {pino.pingliu,zhulinchao7,yee.i.yang}@gmail.com

Abstract

Existing methods usually utilize pre-defined criterions, such as ℓ_p-norm, to prune unimportant filters. There are two major limitations in these methods. First, the relations of the filters are largely ignored. The filters usually work jointly to make an accurate prediction in a collaborative way. Similar filters will have equivalent effects on the network prediction, and the redundant filters can be further pruned. Second, the pruning criterion remains unchanged during training. As the network updated at each iteration, the filter distribution also changes continuously. The pruning criterions should also be adaptively switched.

In this paper, we propose Meta Filter Pruning (MFP) to solve the above problems. First, as a complement to the existing ℓ_p-norm criterion, we introduce a new pruning criterion considering the filter relation via filter distance. Additionally, we build a meta pruning framework for filter pruning, so that our method could adaptively select the most appropriate pruning criterion as the filter distribution changes. Experiments validate our approach on two image classification benchmarks. Notably, on ILSVRC-2012, our MFP reduces more than 50% FLOPs on ResNet-50 with only 0.44% top-5 accuracy loss.

1. Introduction

The computational cost of deep convolutional neural networks (CNNs) keeps increasing due to the complex architectures of modern CNNs, *e.g.*, VGGNet [26], ResNet [10], *etc*. To deploy those complicate models on low resource devices, network pruning becomes necessary since it reduces the storage demand and computational cost (floating point operations, a.k.a, FLOPs) of neural networks. Filter pruning [18, 33] is preferred comparing to weight pruning [9] for the fact that filter pruning could discard the whole filters and make the model with structured sparsity.

Basically, the filter pruning criterion utilized in previous works [18, 32, 13] can be categorized into two branches. The first category is the "smaller-norm-less-important" criterion. This criterion believes that filters with smaller norms

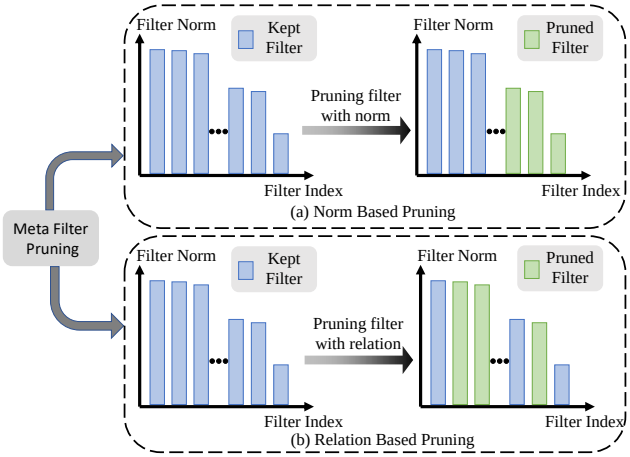


Figure 1. Meta filter pruning with the norm-based criterion (a) and the relation-based criterion (b). The x-axis and y-axis denote the filter index and filter norm, respectively. The blue filters and green filters represent the kept ones and pruned ones, respectively. For the norm-based criterion, filters with small norm are pruned. For the relation-based criterion, the filters with similar norm may have a similar contribution to the network, so they are redundant and should be pruned. We utilize the meta filter pruning module to select suitable criteria based on the current filter distribution.

are less critical, and therefore, pruning those filters with smaller L1-norm [18] or L2-norm [13] will not bring dramatic performance drop. One disadvantage of the ℓ_p-norm based criterion is that, as pointed out by [14], it focuses on the magnitude information of individual filters but ignores the correlations between them, and thus might lead to inappropriate pruning results. Here is an example, suppose we have three filters to prune, each of which is a three-dimension vector: $A = (1, 1, 1), B = (1.1, 1, 1), C = (0.5, 0.3, 0.2)$. Norm-based criterion would prune C since it has the smallest norm. However, if we look closer at A and B, we will find that they are statistically similar and therefore make a very similar, if not the same, contribution to the network, which makes pruning anyone of A or B more reasonable.

The second pruning branch "Relational Criterion" utilizes the relation between filters for filter pruning, which is

a complement for the first branch [14]. Specifically, [14] proposes to use Geometric Median to select the most replaceable filters in the network. They believe that filters near Geometric Median have similar contributions with the remaining filters, and thus pruning those redundant ones would have little negative influence on the network.

Now, here arises two questions: (1) considering the promising results of Relational Criterion, is there any other measurement, except Geometric Median utilized in [14], can be employed for Relational Criterion; (2) as the filter distribution keeps changing during the training (pruning) process, is it appropriate to pre-specify one criterion and keep it **fixed** during the whole process? More specifically, is there any adaptive approach to decide which one criterion to deploy in each training (pruning) step?

For the first question, we introduce new measurements to model correlations between filters. Specifically, we propose to employ Minkowski distance and Cosine distance, to measure the similarities between the filters. When utilizing those two distance functions for two filters, the smaller of the function value, the more similar of the two filters. And therefore, we can prune one of the two filters since they make a similar contribution to the network.

To solve the second question, we build a Meta Filter Pruning (MFP) framework to adaptively select the most suitable criteria based on the current state of filter distribution. Our proposed *meta-pruning* framework borrows the idea from *meta-learning*. In a traditional meta-learning framework [1], given a query dataset Q, we select a similar dataset from previously processed training datasets ($S \in T$). Then the performance of a candidate algorithm of the query Q and the chosen S should be similar. Similarly, in our meta-pruning framework, we view the original unpruned model as S: the dataset that should be used as a benchmark and compared. Besides, the pruned model are regarded as Q: a query dataset that should have similar attributes with S. Following [1], we call the measure of the similarity between S and Q as *meta-attributes*. During the decision process, we minimize the meta-attributes of the pruned model and the original model, by doing which we make the pruned models perform as well as the original models on the pre-defined task.

Contributions. We have three contributions:

(1) We introduce two measurements to model the relation between the filters, which expands the options in Relational Criterion.

(2) This is the first work to adaptively select suitable pruning criteria according to the filter distribution. We conduct pruning in a *meta-learning* way by minimizing the meta-attributes of the pruned and original models.

(3) The experiment on two benchmarks demonstrates the effectiveness of our MFP. Notably, it accelerates ResNet-110 by two times with even 0.16% relative accuracy improvement on CIFAR-10. Additionally, we reduce more than 50% FLOPs on ResNet-50 with only 0.44% top-5 accuracy loss.

2. Related Works

Previous work on pruning can be categorized into weight pruning and filter pruning. [9, 8, 7, 30, 2, 36, 4] focus on pruning fine-grained weight of filters, which leads to unstructured sparsity in models. Filter pruning could achieve the structured sparsity, so the pruned model could take full advantages of high-efficiency Basic Linear Algebra Subprograms (BLAS) libraries to achieve better acceleration.

If we consider whether to utilize the training data to determine the pruned filters, we could further divide the filter pruning methods into two categories:

Training Data Dependent Filter Pruning. [19, 20, 15, 21, 6, 29, 33, 31, 37, 16, 12] utilize the training data to determine the pruned filters. [29] adopts the Principal Component Analysis (PCA) method to specify which part of the network should be preserved. [20] proposes to use the information from the next layer to guide the filter selection. [6] minimizes the reconstruction error of training set sample activations and applies Singular Value Decomposition (SVD) to obtain a decomposition of filters. [31] explores the linear relationship identified in different feature map to eliminate the redundancy in convolutional filters.

Training Data Independent Filter Pruning. Some training data independent filter pruning methods [18, 13, 32, 38] have been proposed. [18] prunes the filters with small ℓ_1-norm. [13] utilizes ℓ_2-norm criterion to select filters and prune those selected filters softly. [32] introduces sparsity on the scaling parameters of batch normalization (BN) layers to prune the network. [38] clusters the filters in the spectral domain to select the unimportant ones.

The second category is more practical than the first one as the training data may not be available during pruning operations. Note that the first category could be viewed as a special kind of "Relational Criterion", mentioned in the Section 1, since the training data and all filters work together to determine the pruned filters. But in this paper, the "Relational Criterion" just utilize the filter value to guide pruning, and it belongs to the training data independent category just the same as the norm-based criterion.

Some previous work [25, 4, 16, 3] has the "meta" word or some similar name such as "learning to prune", but the core idea is different. [25] focus on modeling the complex software systems at high-levels of abstraction, not the neural network. The pruning in this situation refers to the removal of unnecessary classes and properties of software systems, not filters of the neural network. [4] conducts pruning based on second order derivatives of a layer-wise error function. The "learning" word means second order derivatives. [16] utilizes reinforcement learning framework to determine the

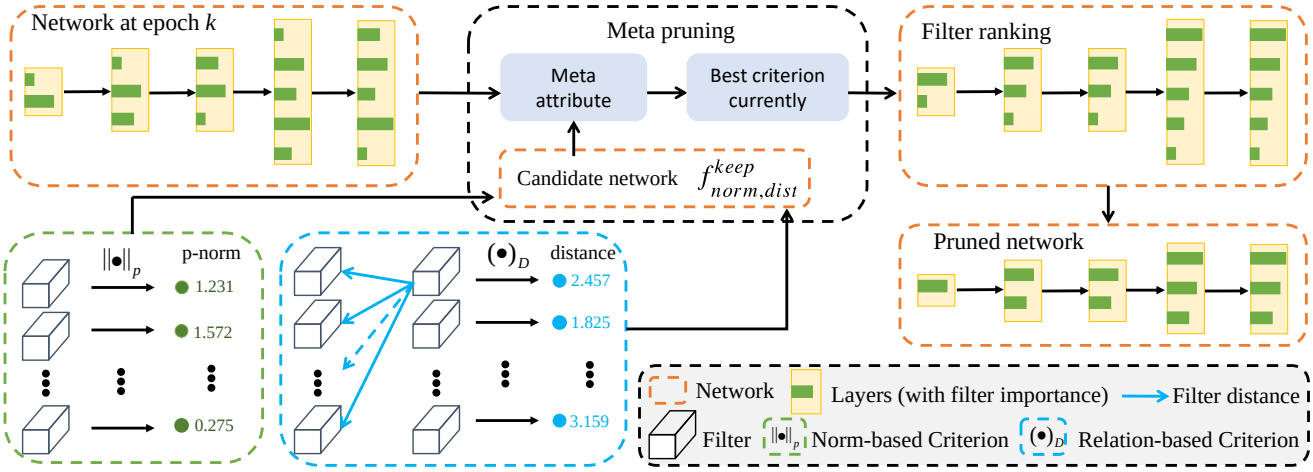


Figure 2. The process of meta filter pruning. The orange dashed box, the yellow box and the white cube indicate the network, the layer of the network, and the filter of the network, respectively. The green and blue dashed box denote the norm-based and relation-based criteria. The black dashed box concludes the process of meta-pruning.

pruning filters. To the best of our knowledge, none of these previous works mentions the meta-learning framework and the meta-attributes for similarity measuring.

3. Meta Filter Pruning

3.1. Preliminaries and Definitions

First, we assume that a neural network has L layers and the number of input and output channels in i_{th} [1] convolution layer is N_i and N_{i+1}, respectively. Suppose K is the kernel size of the network[2], we use $\mathcal{F}_{i,j}$ to represent the j_{th} filter in the i_{th} layer, and $\mathcal{F}_{i,j} \in \mathbb{R}^{N_i \times K \times K}$. For the i_{th} layer, it is consisted of a set of filters denoted by $\{\mathcal{F}_{i,j}, 1 \le j \le N_{i+1}\}$ and parameterized by $\{\mathbf{W}^{(i)} \in \mathbb{R}^{N_{i+1} \times N_i \times K \times K}, 1 \le i \le L\}$. The convolutional operation of the i_{th} layer can be written as:

$$\mathbf{O}_{i,j} = \mathcal{F}_{i,j} * \mathbf{I} \text{ for } 1 \le j \le N_{i+1}, \tag{1}$$

where $\mathbf{I}$ is the input tensor with a shape of $N_i \times H_i \times W_i$, $\mathbf{O}$ is the output tensor with a shape of $N_{i+1} \times H_{i+1} \times W_{i+1}$.

For the convenience of discussion, we assume $\mathcal{F}_{i,j}$ consists of two subsets: the pruned filter set $\mathcal{F}^{pruned}$, and remaining filter set $\mathcal{F}^{keep}$, then we have:

$$\begin{aligned} \mathcal{F}^{keep} \cup \mathcal{F}^{pruned} &= \mathcal{F} \\ \mathcal{F}^{keep} \cap \mathcal{F}^{pruned} &= \emptyset \end{aligned} \tag{2}$$

Now our target becomes clear: given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, and a desired sparsity level κ (*i.e.*, the number of remaining non-zero filters), we are solving a constrained optimization problem defined as follows:

$$\min_{\mathcal{F}^{keep}} \ell(\mathcal{F}^{keep}; \mathcal{D}) = \min_{\mathcal{F}^{keep}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathcal{F}^{keep}; (\mathbf{x}_i, \mathbf{y}_i)), \tag{3}$$
$$\text{s.t.} \quad \mathsf{N}(\mathcal{F}^{keep}) \le \kappa, \quad \mathcal{F} \in \mathbb{R}^{N \times K \times K}.$$

where $\ell(\cdot)$ is a standard loss function (*e.g.*, cross-entropy loss), $\mathcal{F}^{keep}$ is the set of remaining filters of the neural network and N is the cardinality of the filter set. Obviously, after achieving the optimal $\mathcal{F}^{keep}$ in Equation 3, a corresponding $\mathcal{F}^{pruned}$ can be found by solving Equation 2.

Existing works [18, 13] prune the filters based on a predefined criterion, without considering the change of filter distribution due to network architecture updating. In this section, we introduce two new measurements for relational criterion as well as the framework of meta filter pruning (MFP), by which we could choose a suitable criteria based on the state at a specific training or re-training stage.

3.2. Relational Criterion Based on Distance

As the ℓ_p-norm only models the magnitude information of the filters, we introduce the distance of filters to reflect the relation between them.

Minkowski Distance of Filters. First, we utilize the Minkowski distance [27] as one of our choice. To avoid curse of dimensionality [22], we reshape or extend the 3-dimension filter $\mathcal{F}_{i,j}$ to 1-dimension vector. Then i_{th} convolution layer could be written as $\mathcal{Z} \in \mathbb{R}^{N_{i+1} \times G_i}$, which means totally N_{i+1} vectors and the length of each vector is $G_i = N_i \times K \times K$. If we choose two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^{1 \times G_i}$. Then the Minkowski distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is:

$$D_{Mink}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\sum_{i=1}^{G_i} |x_i - y_i|^p}, \tag{4}$$

[1] $1 \le i \le L$

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

Cosine Distance of Filters. We utilize the cosine distance as another effective measurement for filter correlations, which is defined as:

$$D_{cos}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||}. \quad (5)$$

We calculate the distance between every two filters in the network and get the distance matrix of filters. To comprehensively consider the relation of a specific filter $\boldsymbol{x}$ with all the other filters, we define an *average distance* for this filter, denoted as $AveD(\boldsymbol{x})$:

$$AveD(\boldsymbol{x}) = \frac{\sum_{j=1, \mathcal{Z}_j \neq \boldsymbol{x}}^{N_{i+1}} D(\boldsymbol{x}, \mathcal{Z}_j)}{N_{i+1}} = \frac{\sum_{j=1}^{N_{i+1}} D(\boldsymbol{x}, \mathcal{Z}_j)}{N_{i+1}}. \quad (6)$$

As the distance between the filter and itself euqals to zero, we have $D(\boldsymbol{x}, \mathcal{Z}_j) = 0$ when $\mathcal{Z}_j = \boldsymbol{x}$.

If a filter has a large $AveD$, it has little or weak correlations with other filters in the network. In other words, it is independent with other filters and plays a special role in the network, and thus its contribution is hard to be replaced. It is difficult to prune a filter with a larger $AveD$ without hurting the final performance. On the contrary, if a filter has a small $AveD$, it has a similar role to some other filters. In this situation, it is safe to prune this filter since it has a similar or even the same contribution with some others. Based on this understanding, we choose filters with a small average distance to prune:

$$\mathcal{F}_{i,j^*}^{Mink,Cos} \in \underset{j \in [1, N_{i+1}]}{\arg\min} \frac{\sum_{j=1}^{N_{i+1}} D_{Mink,Cos}(\boldsymbol{x}, \boldsymbol{y}_j)}{N_{i+1}}. \quad (7)$$

3.3. Meta Pruning Framework

In this section, we illustrate our proposed meta-pruning framework, which can adaptively choose an appropriate criterion based on the current state of the network.

Traditional Meta-learning Given a query dataset, a standard meta-learning method selects a similar dataset from previously processed datasets (training datasets). Then the performance of a candidate algorithm of the query dataset and the chosen training dataset should be similar [1]. Therefore, measuring the similarity between datasets is a crucial part of traditional meta-learning. The general, statistical and information theoretic *measures* to characterize the datasets are called *meta-attributes* of the the datasets [1].

Proposed Meta-pruning In the filter pruning scenario, it is the network that should be measured. Therefore, we make comparisons between the original network ($\mathcal{F}$) and the pruned network ($\mathcal{F}^{keep}$). If the meta-attributes of these two networks are similar, then the performance from them should be similar or even the same. In this way, $\mathcal{F}^{keep}$ could achieve a performance as good as $\mathcal{F}$ do. Note that there exists a difference between traditional meta-learning and our proposed meta-pruning. The former setting has multiple candidates, while our setting only has one for comparison, *i.e.*, the original model $\mathcal{F}$.

Our target is to minimize the meta-attributes of the two networks before and after pruning. Following [15, 13], we combine the pruning process and training process. Therefore, we propose to model the meta-process of filter pruning as a sequential decision process in a greedy way [3]. The whole process is illustrated in the Figure 2, and we elaborate it below.

- S is a set of states. In our case, $s_t \in S$ at each time step t represents the state available to the meta-process: the remaining filter set $\mathcal{F}_t^{keep}$. We check the state at the end of each training/retraining epoch.

- At the t-th step, given the state s_t, the meta-process takes an action $a_t \in \mathcal{A}$ to choose the proper pruning criterion. If we totally have $\mathcal{H}$ candidate pruning criteria, then we have $\mathcal{A} \in \mathbb{R}^{\mathcal{H} \times 1}$. For every dimension of $\mathcal{A}$, we use the binary value 0,1 to indicate whether a specific criterion is selected (1 means yes, and 0 means no). In our setting, only one criterion are chosen for current step, that is,

$$\sum_{i=1}^{\mathcal{H}} \mathcal{A}_i = 1, \quad \mathcal{A}_i \in \{0, 1\}. \quad (8)$$

  In our experiment, we utilize norm based criterion and relational criterion discussed in Section 3.2, so we have $\mathcal{F}^{keep} = u(\mathcal{F}, \|\mathcal{F}_{i,j}\|_p, AveD(\boldsymbol{x}))$. Here $u()$ denotes a pruning algorithm in Section 3.2. Note that our framework is compatible with any new pruning criterion.

- $\phi : S \to \mathcal{A}$ is the policy employed by the meta-process to generate its action: $\phi(s_t) = a_t$. For filter pruning, the policy should aim at reducing the meta-attributes of the pruned models and the original models:

$$\min \mid \mathbf{M}(\mathcal{F}^{keep}) - \mathbf{M}(\mathcal{F}) \mid, \quad (9)$$

  where $\mathbf{M}$ represents the meta-attributes of the network. Several measures could be utilized as meta-attributes, such as sparsity level κ, the mean value of weights, top-5 loss, top-1 loss and so on. From the empirical analysis, we find the top-5 loss is the best meta-attributes, as shown in Section 4.5.

- After we train the model $\mathcal{F}_t^{keep}$ for several epochs, we enter time step $t+1$. Then we take action a_{t+1} based on the state s_{t+1} of the trained model to get $\mathcal{F}_{t+1}^{keep}$.

[3] Our method is different from reinforcement learning, which is based on delayed rewards setting.

Depth	Method	Pre-train?	Baseline acc. (%)	Accelerated acc. (%)	Acc. ↓ (%)	FLOPs	FLOPs ↓(%)
32	MIL [5]	✗	92.33	90.74	1.59	4.70E7	31.2
	Ours (40%)	✗	**92.63** (±0.70)	**91.85** (±0.09)	**0.78**	**3.23E7**	**53.2**
56	PFEC [18]	✗	93.04	91.31	1.75	9.09E7	27.6
	CP [15]	✗	92.80	90.90	1.90	–	50.0
	SFP [13]	✗	**93.59** (±0.58)	92.26 (±0.31)	1.33	**5.94E7**	**52.6**
	Ours (40%)	✗	**93.59** (±0.58)	**92.76** (±0.03)	**0.83**	**5.94E7**	**52.6**
	PFEC [18]	✓	93.04	93.06	-0.02	9.09E7	27.6
	CP [15]	✓	92.80	91.80	1.00	–	50.0
	FPGM [14]	✓	**93.59** (±0.58)	93.26 (±0.03)	0.33	**5.94E7**	**52.6**
	Ours (40%)	✓	**93.59** (±0.58)	**93.56** (±0.16)	**0.03**	**5.94E7**	**52.6**
110	PFEC [18]	✗	93.53	92.94	0.61	1.55E8	38.6
	MIL [5]	✗	93.63	93.44	0.19	-	34.2
	SFP [13]	✗	**93.68** (±0.32)	93.38 (±0.30)	0.30	1.50E8	40.8
	Ours (40%)	✗	**93.68** (±0.32)	**93.69** (±0.31)	**-0.01**	**1.21E8**	52.3
	Ours (50%)	✗	**93.68** (±0.32)	**93.38** (±0.16)	**0.30**	**9.40E7**	**62.8**
	PFEC [18]	✓	93.53	93.30	0.20	1.55E8	38.6
	NISP [33]	✓	–	–	0.18	–	43.8
	Ours (40%)	✓	**93.68** (±0.32)	**93.31** (±0.08)	**0.37**	1.21E8	**52.3**

Table 1. Comparison of the pruned ResNet on CIFAR-10. In "Pre-train?" column, "✓" and "✗" indicate 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. A negative value in "Acc. ↓" indicates an improved model accuracy.

3.4. Acceleration Analysis

In the above analysis, the ratio of pruned FLOPs is $1-(1-P_{i+1})\times(1-P_i)$ theoretically. As other operations such as batch normalization (BN) and pooling are insignificant comparing to convolution operations, it is common to utilize the FLOPs of convolution operations as the FLOPs of the network [18, 13].

However, in the real scenario, non-tensor layers (*e.g.*, pooling and BN layers) also need the computation time on GPU [20] and the realistic acceleration may be influenced. Besides, other factors such as buffer switch, IO delay and the efficiency of BLAS libraries also lead to the gap between the realistic and theoretical acceleration. We compare the different acceleration ratio in Table 4.

4. Experiments

4.1. Experimental Setting

Datasets. In this section, we validate the effectiveness of our acceleration method on two benchmark datasets, CIFAR-10 [17] and ILSVRC-2012 [24]. The CIFAR-10 dataset contains $50,000$ training images and $10,000$ testing images, in total $60,000$ 32×32 color images in 10 different classes. ILSVRC-2012 [24] contains 1.28 million training images and 50k validation images of $1,000$ classes.

Architecture Setting. As the ResNet has the shortcut structure, existing works [5, 20, 15] claim that ResNet has less redundancy than VGGNet [26] and accelerating ResNet is more difficult than accelerating VGGNet. Therefore, we focus on pruning the challenging ResNet model. Moreover, to validate our method on the single branch network, we follow [18] to conduct a test on the VGGNet [34].

Training Setting. For VGGNet on CIFAR-10, we follow the setting in [18]. As the training setup is not publicly available, we re-implement the pruning procedure and achieve similar results to the original paper. For ResNet on CIFAR-10, we utilize the same training schedule as [35]. For CIFAR-10 experiments, we run each setting three times and report the "mean ± std". In the ILSVRC-2012 experiments, we use the default parameter settings which is same as [10, 11] and the same data argumentation strategies as the official PyTorch [23] examples.

Pruning Setting. For VGGNet on CIFAR-10, we use the same pruning rate as [18]. For experiments on ResNet, we prune *all* the weighted layers with the *same* pruning rate at the same time, which is the same as [13]. Therefore, only one hyper-parameter, the pruning rate of $P_i = P$ is used to balance the acceleration and accuracy. Note that choosing different rates for different layers could improve the performance [18], but it also introduces extra hyper-parameters. Our pruning operation is conducted at the end of every two training epochs, which gives us a balance of accuracy and consumption of pruning operation.

We choose ℓ_p-norm criterion with $p = 1, 2$ as the norm-based criterion in our meta-pruning framework. For relational criterion in our meta-pruning, we use Minkowski distance with $p = 1, 2$ and cosine distance. We analyze the difference between pruning a scratch model and the pre-trained model. For pruning the scratch model, we utilize the regular training schedule without additional fine-tuning. For pruning the pre-trained model, we reduce the learning

Depth	Method	Pre-train?	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 [13]	✗	**70.28**	67.10	**89.63**	87.78	3.18	1.85	**41.8**
	Ours (30%)	✗	**70.28**	**67.66**	**89.63**	**87.90**	**2.62**	**1.73**	**41.8**
	Ours (30%)	✓	**70.28**	**68.31**	**89.63**	**88.28**	**1.97**	**1.35**	41.8
	Ours (40%)	✓	**70.28**	67.11	**89.63**	87.49	3.17	2.14	**51.8**
50	Ours (40%)	✗	**76.15**	74.13	**92.87**	91.94	2.02	0.93	**53.5**
	ThiNet [20]	✓	72.88	72.04	91.14	90.67	0.84	0.47	36.7
	SFP [13]	✓	**76.15**	62.14	**92.87**	84.60	14.01	8.27	41.8
	NISP [33]	✓	–	–	–	–	0.89	–	44.0
	CP [15]	✓	–	–	92.20	90.80	–	1.40	50.0
	LFC [28]	✓	75.30	73.40	92.20	91.40	1.90	0.80	50.0
	ELR [31]	✓	–	–	92.20	91.20	–	1.00	50.0
	Ours (30%)	✓	**76.15**	**75.67**	**92.87**	**92.81**	**0.48**	**0.06**	42.2
	Ours (40%)	✓	**76.15**	74.86	**92.87**	92.43	1.29	0.44	**53.5**

Table 2. Comparison of the pruned ResNet on ImageNet. "Pre-train?" and "acc. ↓" have the same meaning with Table 1.

rate to one-tenth of the original learning rate. To conduct a fair comparison, we use the same training epochs for scratch and pre-trained models, which is the same as [13].

"Ours (40%)" in Table 1 and Table 2 means we prune 40% filters of the layers. We compare our method with existing state-of-the-art acceleration algorithms, *e.g.*, MIL [5], PFEC [18], CP [15], ThiNet [20], SFP [13], NISP [33], FPGM [14], LFC [28], ELR [31]. Experiments show that our MFP achieves the comparable performance with the state-of-the-art results.

4.2. VGGNet on CIFAR-10

The result of pruning scratch and pre-trained VGGNet is shown in Table 3. Not surprisingly, MFP achieves better performance than [18] in both settings. With the pruning criterion selected by our method, we could achieve better accuracy than [18] when pruning the random initialized VGGNet (93.54% *v.s.*93.31%). In addition, The pruned model without fine-tuning has better performance than [18] (84.80% *v.s.*77.45%). After fine-tuning 40 epochs, our model achieves similar accuracy with [18]. Notably, if more fine-tuning epochs (160) are utilized, [18] achieve similar result with fine-tuning 40 epochs (93.28% *v.s.*93.22%), while our method could get much better performance (93.76% *v.s.*93.26%).

Setting \ Acc (%)	PFEC [18]	Ours
Baseline	93.58 (±0.03)	93.58 (±0.03)
Prune from scratch	93.31 (±0.03)	**93.54** (±0.03)
Prune pretrain w.o. FT	77.45 (±0.03)	**84.80** (±0.03)
FT 40 epochs	93.22 (±0.03)	**93.26** (±0.03)
FT 160 epochs	93.28 (±0.03)	**93.76** (±0.08)

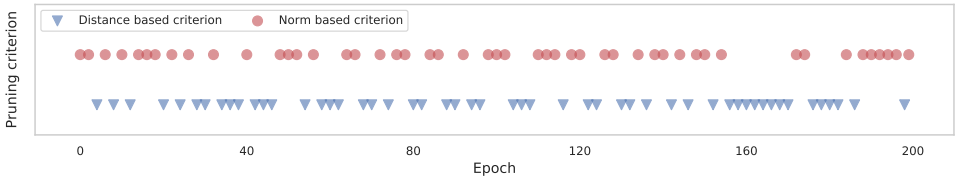
Table 3. Pruning scratch and pre-trained VGGNet on CIFAR-10. "w.o." means "without" and "FT" means "fine-tuning".

4.3. ResNet on CIFAR-10

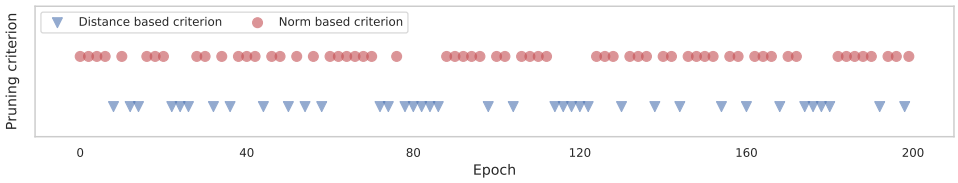
For the CIFAR-10 dataset, we test our MFP on ResNet (depth 32, 56 and 110). We use two different pruning rates 40% and 50%. As shown in Table 1, the experiment results validate the effectiveness of our method.

Result Explanation. For example, MIL [5] accelerates the random initialized ResNet-32 by 31.2% speedup ratio with 1.59% accuracy drop, but our MFP achieves 53.2% speedup ratio with only 0.78% accuracy drop. Comparing to SFP [13], when we prune 52.6% FLOPs of the random initialized ResNet-56, our MFP has 0.50% accuracy improvement over SFP [13] (0.83% *v.s.*1.33%). For pruning the pre-trained ResNet-56, our method achieves a higher (52.3% *v.s.*50.0%) acceleration ratio than CP [15] with 0.97% accuracy increase over CP [15]. Comparing to PFEC [18], our method accelerates the random initialized ResNet-110 by 52.3% speedup ratio with even 0.01% accuracy improvement, while PFEC [18] achieves only 38.6% acceleration (13.7% less than our method) with 0.61% accuracy drop. For NISP [33], we achieve higher speedup ratio (52.3% *v.s.*43.8%) with similar accuracy drop. The first reason for our superior result is that our criterion explicitly models the correlation between filters by taking advantages of three corresponding measurements. Besides, we adaptively select the suitable criteria to match the current filter distribution, which might keep changing during the pruning process. Previous works [5, 13, 15, 18, 33] pre-specify a criterion before pruning and keep it fixed during the whole pruning process, failing to consider that the selected criterion might be not suitable any more after epochs of training.

Different Criteria During Training. The pruning criterion during the training process of two different initialization are shown in Figure 3. The norm-based criterion includes 1-norm and 2-norm. The relational criterion in-



(a) Initialization 1



(b) Initialization 2

Figure 3. Learned pruning criterion during the training process of ResNet-110 on CIFAR-10 under different initialization. The pruning rate is 40%. The red and blue marker denotes the norm based criterion and the distance based criterion, respectively.

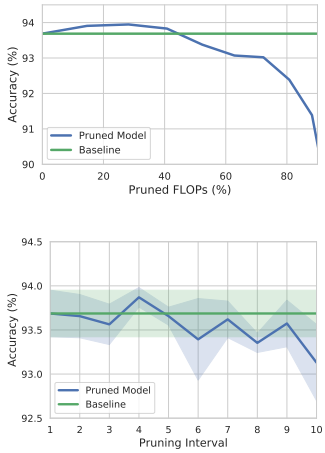


Figure 4. Accuracy of ResNet-110 on CIFAR-10 regarding different FLOPs and pruning interval.

cludes Cosine distance, Manhattan distance, and Euclidean distance.[4] Comparing those two figures, we conclude that our MFP could adaptively select proper criterion during the training process with different initializations. For the selected pruning criteria, we find that during the early training process, the distance-based criteria are adopted less than norm-based criteria. The above phenomena might be caused by the training knowledge from the training set. During the early training stage, the filters have not learned enough training set knowledge, and the filters have a small correlation with other filters, so the norm-based criteria are preferred. After more training epochs, the filters obtain the information of training set and become correlated with others; then the distance-based criteria start taking power.

4.4. ResNet on ILSVRC-2012

For the ILSVRC-2012 dataset, we test our method on ResNet-18, ResNet-50; and we use pruning rate 30% and 40% for these models. Same as [13], we do not prune the projection shortcuts. Table 2 shows that our MFP outperforms existing methods on the ILSVRC-2012 dataset.

Result Explanation. For the random initialized ResNet-18, MIL [5] accelerates the network by 34.6% speedup ratio with 3.65% accuracy drop, but our MFP achieves 41.8% speedup ratio (7.20% better) with only 2.62% accuracy drop (1.03% better). Comparing to SFP [13], when we prune the same ratio (41.8%) of FLOPs of the ResNet-18, our MFP has 0.56% accuracy improvement over SFP [13].

For pruning the pre-trained ResNet-50, our MFP reduces 41.8% FLOPs of the network with only 0.06% top-5 accuracy drop. In contrast, ThiNet [20] reduces 36.7% FLOPs (5.1% worse than ours) with 0.47% top-5 accuracy drop (0.41% worse than ours). In addition, SFP achieves the same acceleration ratio with the 8.27% top-5 accuracy drop (8.21% worse than ours). Comparing to NISP [33], we achieve a similar acceleration ratio with smaller accuracy drop (0.48% vs 0.89%). When we prune 53.5% FLOPs of the pre-trained ResNet-50, our MFP has 0.44% top-5 accuracy drop, while CP [15] reduces 50.0% FLOPs of the network with 1.40% top-5 accuracy (0.96% worse than ours). The superior performance may come from that our method consider the magnitude information and the correlation information of the filters.

Realistic Acceleration 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 result is shown in Table 4. As discussed in the above section, the gap between the theoretical and the realistic acceleration may come from the limitation of IO delay, buffer switch and efficiency of BLAS libraries.

Model	Baseline time (ms)	Pruned time (ms)	Realistic Speedup(%)	Theoretical Speedup(%)
ResNet-18	37.50	26.17	30.2	41.8
ResNet-50	136.24	84.33	38.1	53.5

Table 4. Comparison on the theoretical and realistic acceleration. Only the time consumption of the forward procedure is considered.

4.5. Ablation Study

Varying Pruned FLOPs. We change the ratio of pruned FLOPs for ResNet-110 on CIFAR-10 to comprehensively understand our MFP, as shown in Figure 4(a). We could prune more than 40% of the filters of the network without hurting the performance. When the ratio of pruned FLOPs

[4]Cosine distance is not selected in this training process. Manhattan distance and Euclidean distance are special cases when the exponentiation parameter in Minkowski distance equals one and two, respectively.

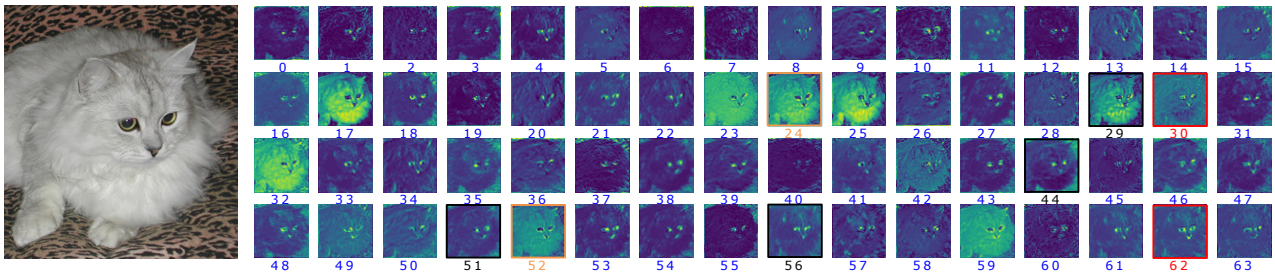


Figure 5. Input image (left) and visualization of feature maps (right, the number from 0 to 63). Selected channels to be prund with norm-based criterion and relation-based criterion. The feature maps are extracted from the first convolutional layer of the first block of ResNet-18. The pruning rate is 10%. Feature maps with black title and box (channel 29, 44, 51, 56) denote the common channels selected by both criteria. Feature maps with red title and box (30, 62) denote the channels only selected by norm-based criterion, while feature maps with orange title and box (24, 52) denote the channels only selected by distance-based criterion.

is less than 40%, the performance of the pruned model even exceeds the baseline model without pruning. This means our MFP could choose the proper criterion and prune the suitable filters. In addition, our MFP may have a regularization effect on the neural network.

Varying Pruning Interval. The pruning interval means how many training epochs between two pruning operations. We change the pruning interval from one epoch to ten epochs, as shown in Fig. 4(b). It is shown that the model accuracy has no large fluctuation along with the different pruning intervals. This means the performance of our framework is not sensitive to the pruning interval.

Varying Meta-attributes. We compare several meta-attributes to comprehensively understand the MFP. The meta-attributes includes top5 loss, top1 loss, the mean value of the network, sparsity level κ, and so on. The sparsity level κ meta-attributes is directly related to the acceleration ratio of the network. If we pre-define the expected acceleration ratio, the sparsity level κ of pruned model would be the same. Hence, we should consider other meta-attributes to distinguish the pruned models. We find that top5 loss is a better meta-attribute comparing to top1 loss as it reflects more information and thus is more general. The improvement of top5 meta-attributes over random meta-attributes validates the effectiveness of meta pruning process. From a statistical perspective, we also use the mean value of the network as a meta-attribute. The poor performance of mean value meta-attributes might come from the fact that too much information about the network is lost in the mean calculation. Finding a better meta-attributes still need to be explored later.

Meta attrbute	Top5	Top1	Mean	Random
Acc.	$93.52_{\pm 0.56}$	$93.39_{\pm 0.40}$	$91.52_{\pm 0.34}$	$91.82_{\pm 0.41}$

Table 5. Accuracy of ResNet-110 on CIFAR-10 regarding different meta-attributes. The pruning rate is 40%.

4.6. Feature Map Visualization and Explanation

We visualize the feature maps of the first layer of the first block of ResNet-18, as shown in Figure 4.5. We rank the 64 channel[5] in this layer with number 0 to 63 and set the pruning rate to 10% to choose six filters to be pruned. We select channel (44, 29, 62, 56, 30, 51) via $L2$-norm criterion and select channel (44, 56, 29, 51, 52, 24) via Euclidean distance criterion. Both criteria select channel (29, 44, 51, 56), but order of the channels is different.

We focus on the different channels selected by these criteria to illustrate their difference. In addition to the common part, the norm-based criterion select (30, 62), while the distance-based criterion select (24, 52). Channel (30, 62) have rather small activation values and might be meaningless to the network, so they are selected by the norm-based criterion. For channel (24, 52), the rough shape of the cat in these channels are similar to other channels such as (17, 23, 25, 32, 59), so the distance-based criterion (relational criterion) prefers to prune these channels. These results validate our points that norm based criterion and distance based criterion consider different aspects of the network. In this way, during the updating of the network and the filter distribution, adaptively selecting those criteria is necessary.

5. Conclusion and Future Work

In this paper, we propose a new meta filter pruning (MFP) strategy for deep CNNs acceleration. Different from the existing norm-based criterion, MFP explicitly considers the relation between filters. More than that, as a meta-framework, MFP selects the suitable criteria adaptively during training, to fit the current filter distribution. MFP achieves comparable performance with state-of-the-art methods in several benchmarks.

In the future, we could consider utilizing different criterion for different layers of the network. Even within a network layer, we could combine different criteria for filter

[5]The channels correspond the filters in the network.

pruning. Besides, whether a better meta-attribute exists still needs to be explored. Moreover, other parallel acceleration algorithms, *e.g.*, matrix decomposition, and low-precision weights, could be used as a complementary method to improve the performance further.

References

[1] P. B. Brazdil, C. Soares, and J. P. Da Costa. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, 2003. 2, 4

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

[3] T.-W. Chin, C. Zhang, and D. Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *arXiv preprint arXiv:1810.00518*, 2018. 2

[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. 5, 6, 7

[6] A. Dubey, M. Chatterjee, and N. Ahuja. Coreset-based neural network compression. *arXiv preprint arXiv:1807.09810*, 2018. 2

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

[8] 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

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

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

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

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

[13] 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, 3, 4, 5, 6, 7

[14] Y. He, P. Liu, Z. Wang, and Y. Yang. Pruning filter via geometric median for deep convolutional neural networks acceleration. In *CVPR*, 2019. 1, 2, 5, 6

[15] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017. 2, 4, 5, 6, 7

[16] Q. Huang, K. Zhou, S. You, and U. Neumann. Learning to prune filters in convolutional neural networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 709–718. IEEE, 2018. 2

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

[18] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient ConvNets. In *ICLR*, 2017. 1, 2, 3, 5, 6

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

[20] 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

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

[22] E. Novak and K. Ritter. The curse of dimension and a universal method for numerical integration. In *Multivariate approximation and splines*, pages 177–187. Springer, 1997. 3

[23] 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

[24] 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

[25] S. Sen, N. Moha, B. Baudry, and J.-M. Jézéquel. Meta-model pruning. In *International Conference on Model Driven Engineering Languages and Systems*, pages 32–46. Springer, 2009. 2

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 5

[27] A. Singh, A. Yadav, and A. Rana. K-means with three different distance metrics. *International Journal of Computer Applications*, 67(10), 2013. 3

[28] P. Singh, V. K. Verma, P. Rai, and V. P. Namboodiri. Leveraging filter correlations for deep model compression. *arXiv preprint arXiv:1811.10559*, 2018. 6

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

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

[31] 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, 6

[32] 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. 1, 2

[33] 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, 5, 6, 7

[34] S. Zagoruyko. 92.45% on cifar-10 in torch. http://torch.ch/blog/2015/07/30/cifar.html, 2015. 5

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

[36] 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. *ECCV*, 2018. 2

[37] 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

[38] 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