Feature Statistics Guided Efficient Filter Pruning

Hang Li[1] , **Chen Ma**[1*] , **Wei Xu**[2] and **Xue Liu**[1]
[1]School of Computer Science, McGill University
[2]Institute for Interdisciplinary Information Sciences, Tsinghua University
{hang.li3, chen.ma2}@mail.mcgill.ca, weixu@tsinghua.edu.cn, xueliu@cs.mcgill.ca

Abstract

Building compact convolutional neural networks (CNNs) with reliable performance is a critical but challenging task, especially when deploying them in real-world applications. As a common approach to reduce the size of CNNs, pruning methods delete part of the CNN filters according to some metrics such as $l1$-norm. However, previous methods hardly leverage the information variance in a single feature map and the similarity characteristics among feature maps. In this paper, we propose a novel filter pruning method, which incorporates two kinds of feature map selections: diversity-aware selection (DFS) and similarity-aware selection (SFS). DFS aims to discover features with low information diversity while SFS removes features that have high similarities with others. We conduct extensive empirical experiments with various CNN architectures on publicly available datasets. The experimental results demonstrate that our model obtains up to 91.6% parameter decrease and 83.7% FLOPs reduction with almost no accuracy loss.

1 Introduction

Deep convolutional neural networks (CNNs) have evolved to the state-of-the-art technique on various tasks, including image classification [Krizhevsky *et al.*, 2012], object detection [Girshick *et al.*, 2014] and sentence classification [Kim, 2014]. Due to the parameter sharing and local connectivity schemes, CNNs own the powerful representation and approximation ability that can benefit downstream tasks. For example, the classification accuracy of CNNs in the ImageNet challenge has increased from 84.7% in 2012 (AlexNet [Krizhevsky *et al.*, 2012]) to 96.5% in 2015 (ResNet-152 [He *et al.*, 2016a]).

Although CNNs yield the state-of-the-art performance on various tasks, they still suffer from high storage and computation overheads. Specifically, CNNs cost a huge space to store millions or even billions of parameters; the floating-point operations (FLOPs) of CNNs are intensive since a large quantity of multiplication and addition operations are executed in convolutional layers. These two drawbacks impede deploying CNNs in real-world applications especially when the storage and computation resources are limited.

Filter pruning, as a promising solution to address the aforementioned issues, has drawn significant interests from both academia and industry. The reasons are two-fold. First, most filter pruning methods are conducted on predefined architectures without extra designs [Han *et al.*, 2015]. Second, filter pruning techniques do not introduce sparsity to architectures like weight pruning methods [Li *et al.*, 2016]. In particular, local pruning methods remove less important filters according to the pruning ratios in each layer, which leads to a fixed architecture with finely trained weights. For example, [Li *et al.*, 2016] prunes filters with a low $l1$-norm in each layer. However, [Liu *et al.*, 2018] shows that once the pruned architecture has been defined, the performance relies more on the architecture rather than learned weights, which makes local pruning less effective. Compared with local pruning, global pruning distinguishes the importance of filters across all layers, which can achieve better performance. The pruned architecture is automatically determined by the global pruning algorithm. As a representative approach of global pruning, [Liu *et al.*, 2017] imposes a sparsity-induced regularization on the scaling factors of batch normalization layers and prunes those channels with smaller factors.

Even though previous works have proposed effective methods to compress the CNN architecture, we argue that two factors of feature maps are rarely incorporated. On the one hand, the information variance of a feature map can be a good indicator for discriminating the effectiveness of feature maps. That is, if the values in a feature map do not vary a lot, the amount of information it contains may be limited. On the other hand, the relationships (e.g. similarity) between feature maps play a significant role in preserving effective feature maps. If two features have high similarities, then one of them can be considered as redundant. However, previous works mainly utilize the metrics within a feature map without considering the similarities between feature maps. For example, [Li *et al.*, 2016] only applies $l1$-norm to select feature maps. Solely depending on $l1$-norm may keep multiple similar feature maps with the same $l1$-norm value, which will lead to incomplete pruning.

To incorporate the aforementioned intuitions, we propose

*Corresponding Author

a framework containing two steps of feature map selections to prune less diverse feature maps and corresponding filters. The first step employs the diversity-aware feature selection (DFS) to remove feature maps with less information variance. In particular, we apply the mean standard deviation (M-std) of values in a feature map to measure the information variance degree. The feature map with the lowest M-std will be pruned. The second step is the similarity-aware feature selection (SFS) for deleting feature maps that have high similarities with each others. We compute the cosine similarity among all features and delete the feature whose similarity is larger than a threshold. Moreover, we observe that the M-std distribution varies a lot in different layers of ResNet. This motivates us to adopt a fine-grained pruning strategy, which contains two pruning processes working on different parts of a residual block. We extensively evaluate our method with many state-of-the-art approaches and different metrics on three publicly available datasets. The experimental results show the improvements of our model over other baselines.

Our contributions are summarized as follows:

- We introduce an effective filter pruning method to compress CNN models that have a large number of parameters and FLOPs. We employ a diversity-aware feature selection to distinguish informative feature maps and propose a similarity-aware feature selection to remain representative feature maps.
- Our method is a global filter pruning method that compares the redundancy degree of feature maps across all layers. More importantly, we propose a fine-grained pruning strategy for ResNet, which differs from many existing methods [He *et al.*, 2019; Li *et al.*, 2016; Luo *et al.*, 2017].
- We extensively evaluate our methods with multiple CNN architectures on three datasets. We show the significantly improved effectiveness of our proposed method, which can reduce parameters of MobileNet by up to 91.6% and FLOPs decrease up to 83.7% with limited accuracy drop.

2 Related Works

Convolutional neural networks have many parameters to provide enough model capacity, making them both computationally and memory intensive. Pruning methods aim to reduce the number of parameters in a CNN model. Based on the over-parameterization hypothesis [Ba and Caruana, 2014], a considerable amount of parameters can be pruned while CNNs maintain a promising accuracy. By reducing the number of weights or filters, we can save the storage space and also reduce the computational complexity and memory footprint of a model during testing.

Weight pruning is a strategy that deletes parameters with small "saliency". [LeCun *et al.*, 1990] introduces the Optimal Brain Damage(OBD) method which is one of the earliest attempts at pruning neural networks. OBD defines the "saliency" by calculating the second derivative of the objective function concerning the parameters. [Han *et al.*, 2015] present a generic iterative framework for neural network pruning, in which all weights below a threshold are removed from the network. [Guo *et al.*, 2016] propose a dynamic compression algorithm named Dynamic Network Surgery(DNS) to prune or rebuild the connections during the learning process. Weight pruning is fairly efficient in terms of reducing model size. However, networks compressed by weight pruning methods become sparse, requiring additional sparse libraries or even specialized hardware to run. The filter structure of CNNs allows us to perform filter pruning, which is a naturally structured method without introducing sparsity. And in this paper, we also focus on the filter pruning method.

Filter pruning methods prune the convolutional filters or channels of CNN models which make the deep networks thinner. Based on the assumption that CNNs usually have a significant redundancy among filters, researchers introduce various metrics [Li *et al.*, 2016; Luo *et al.*, 2017; Liu *et al.*, 2017; Wang *et al.*, 2019] to measure the importance of filters, which is the key issue of filter pruning. [Li *et al.*, 2016] measures the relative importance of a filter in each layer by calculating the sum of its absolute kernel weights. Beyond such a magnitude-based method, [Hu *et al.*, 2016] proposes that if the majority of the value in a filter is zero, this filter is likely to be redundant. They compute the Average Percentage of Zeros (APoZ) of each filter as its importance score.

Except for adopting the magnitude of a filter as an important metric, feature maps are also significant. Reconstruction-based methods seek to do filter pruning by minimizing the reconstruction error of feature maps between the pruned model and a pre-trained model. [Luo *et al.*, 2017] propose ThiNet which transforms the filter selection problem into the optimization of reconstruction error computed from the next layer's feature map. [He *et al.*, 2017] uses the LASSO regression to obtain a subset of filters that can reconstruct the corresponding output in each layer. Even though this reconstruction-based method considers the feature map information, it has a limitation that the reconstructed feature map might have high similarity which is redundant information.

Besides the aforementioned methods that prune filters after obtaining a trained neural network, there are some explorations [Liu *et al.*, 2019; He *et al.*, 2018] to get the importance of each filter during the training process. In [Liu *et al.*, 2017], a scaling parameter γ is introduced to each channel and is trained simultaneously with the rest of the weights by adding L1-norm of γ in the loss function. Channels with small factors are pruned and the network is fine-tuned after pruning. [Yamamoto and Maeno, 2018] inserts a self-attention module in the pre-trained convolutional layer or fully connected layer to learn the importance of each channel. [Luo and Wu, 2018] learn a 0-1 indicator which can multiply with feature maps as the input of the next layer in a joint training manner.

However, our proposed method is different from previous approaches. We apply a diversity-aware feature selection process to remove features with lower information variance. Besides, a similarity-aware feature selection process is utilized to discover those closely related features.

3 Methodology

3.1 Preliminaries

Given a convolutional neural network (CNN) with L convolution layers, we assume the dimension of the input feature $\mathbf{X}_i$ at the i_{th} layer is $\mathbb{R}^{N_i \times W_i \times H_i}$, where N_i, H_i and W_i de-

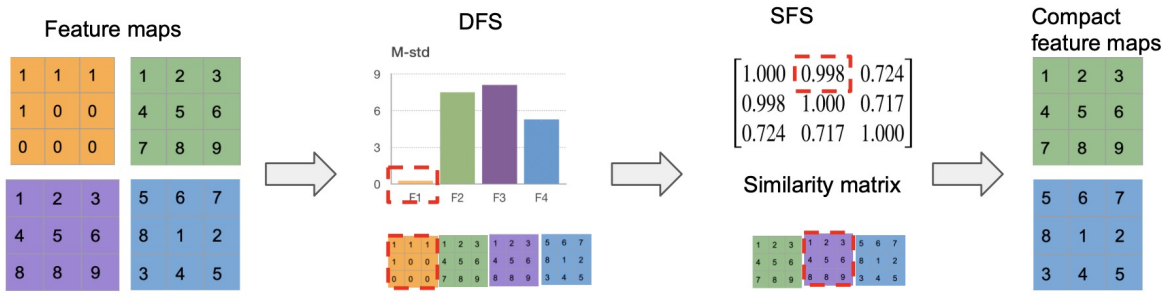


Figure 1: The framework of our proposed feature pruning approach. After obtaining feature maps of a trained CNN model, we first do the diversity-aware feature selection (DFS) by removing feature maps with the smaller M-std value. Then the similarity-aware feature selection (SFS) is further used to prune the redundant feature maps with higher cosine similarity.

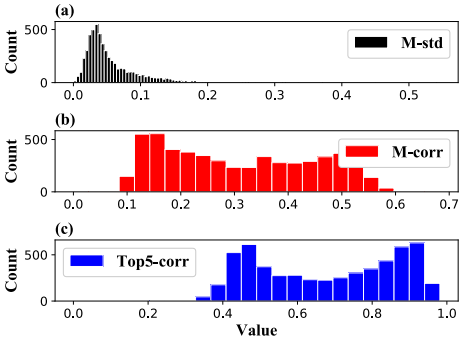


Figure 2: Distributions of M-std, M-corr and Top5-corr of features in VGGNet trained on CIFAR 10 dataset.

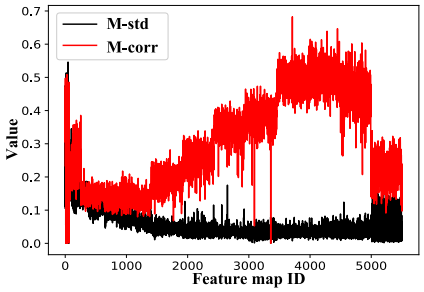


Figure 3: M-std and M-corr of all feature maps in VGGNet trained on CIFAR 10 dataset.

note the number of channels, rows and columns, respectively. The output dimension at this layer is $\mathbb{R}^{N_{i+1}\times W_{i+1}\times H_{i+1}}$. The corresponding filter set of the i_{th} layer is $\mathcal{F}_i = \{\mathbf{F}_{i,1}, \mathbf{F}_{i,2}, ..., \mathbf{F}_{i,N_{i+1}}\}$, where $\mathbf{F}_{i,j} \in \mathbb{R}^{N_i\times K\times K}$ and $K \times K$ is the kernel size. The convolutional operation of the i_{th} layer is denoted as:

$$\mathbf{X}_{i+1,j} = \mathbf{F}_{i,j} * \mathbf{X}_i \ , \quad 1 \leqslant j \leqslant N_{i+1}, \tag{1}$$

where $\mathbf{X}_{i,k} \in \mathbb{R}^{W_i\times H_i}$ represents the feature map of the k_{th} channel at the i_{th} layer.

Given a model $\mathbf{M}$ trained on dataset $\{(\hat{x}_i, \hat{y}_i)\}$, our task is to delete redundant feature maps and corresponding filters.

3.2 Diversity and Similarity in Feature Maps

To measure the diversity and similarity of feature maps, we adopt two metrics:

- Mean standard deviation (M-std). The mean standard deviation of each feature map is defined as:

$$\text{M-std}(\mathbf{X}_{i,j}) = \frac{1}{T}\sum_{m=1}^{T}\sqrt{\frac{\sum_{p=1}^{W_iH_i}(x_p^m - \overline{x}^m)^2}{W_iH_i - 1}}, \tag{2}$$

where x_p^m is the p_{th} element in $\mathbf{x}_{i,j}^m \in \mathbb{R}^{1\times W_iH_i}$ of a flat feature map $\mathbf{X}_{i,j}^m \in \mathbb{R}^{W_i\times H_i}$, m ($m \leqslant T$) represents the sample index, and $\overline{x}^m$ is the mean of $\{x_p^m\}$. Smaller M-std value means the corresponding feature map has less heterogeneity. This lower information diversity contributes more inadequate to further feature extraction.

- Mean cosine similarity (M-corr). We use the cosine similarity to measure the relevance between feature maps. Since the dimensions of feature maps vary in different layers, we compute the feature similarity within the same layer. The M-corr can be computed as:

$$\text{M-corr}(\mathbf{X}_{i,j}) = \frac{1}{T}\sum_{m=1}^{T}\frac{\sum_{p=1}^{N_i}\left|\cos(\mathbf{x}_{i,j}^m, \mathbf{x}_{i,p}^m)\right|}{N_i}. \tag{3}$$

The feature map with a larger M-corr value tends to have high similarity with each feature in its layer. Comparing with M-corr, mean Top-k cosine similarity (Topk-corr) is an another commonly used metric that can also reflect the similarity characteristic of feature maps. The Topk-corr of a feature map $\mathbf{X}_{i,j}$ is $\frac{1}{T}\sum_{m=1}^{T}\frac{\sum_{p=1}^{N_i}A_{i,p}\cdot\left|\cos(\mathbf{x}_{i,j}^m, \mathbf{x}_{i,p}^m)\right|}{k}$. $A_{i,p} = 1$ if $\mathbf{x}_{i,p}^m$ is among the Top-k cosine values of $\mathbf{x}_{i,j}^m$, $A_{i,p} = 0$ otherwise.

To highlight the characteristics of feature maps, we train a VGGNet model on the CIFAR10 dataset and obtain the statistical information of feature maps. Figure 2 gives an illustration of the overall distributions of M-std, M-corr, and Topk-corr. As shown in Figure 2a, there are nearly a half number of feature maps with M-std less than 0.05, which reveals that these feature maps may not contain much information. Figure 2b indicates there are around half of the feature maps that

Algorithm 1 Our proposed filter pruning scheme

Input: Sample data $\{\hat{x}_i\}_{i=1}^{T}$, model $\mathbf{M}$, threshold ν
Output: Selected filter subset $\hat{\mathcal{F}}$

Construct feature maps $\{\mathbf{X}_i^m\}_{i=1}^{L}$ for each data sample
Compute M-std $\{std\}$ from Equation 2 for each feature using $\{\mathbf{X}_i^m\}_{i=1}^{L}$
Let $\hat{\mathbf{X}} \leftarrow \varnothing$, $\beta \leftarrow mean(\{std\})$
for $i \in \{1, 2, ..., L\}$ **do**
    Find $\hat{\mathbf{X}}_i$ according to Equation 4
    Obtain $\widetilde{\mathbf{X}}_i$ using Algorithm 2, $\widetilde{\mathbf{X}}_i \leftarrow \text{SFS}(\hat{\mathbf{X}}_i, \nu)$
    Let $\hat{\mathbf{X}} \leftarrow \hat{\mathbf{X}} \cup \widetilde{\mathbf{X}}_i$
end for
Find $\hat{\mathcal{F}}$ according to $\hat{\mathbf{X}}$

have M-corr value over 0.3. Figure 2c shows that there are about a quarter of feature maps with Top-5 cosine similarity values exceed 0.8, even 0.9. Similar feature maps can be treated as redundant features, making less contribution to the network. These metrics demonstrate there exist redundant feature maps in CNNs.

The details of the statistic value of each feature map are shown in Figure 3. We can see that most of the M-std values become lower with the increase of layer depth and most of the M-corr values enhance as the number of channels gets larger. M-std and M-corr have a weak negative correlation between each other in this case. Their Pearson correlation [Galton, 1886] is -0.38. These two criteria can complement each other when selecting important features.

3.3 Filter Pruning

We aim to prune redundant filters of deep CNNs in a simple but effective scheme. The central idea of our method has two steps of feature map selections (see Figure 1): diversity-aware feature selection (DFS) and similarity-aware feature selection (SFS). After obtaining a pre-trained CNN model, we first compute the M-std values for all feature maps. Then we prune feature maps with the smallest values and save the unpruned feature maps. Finally, we calculate the cosine similarity among the unpruned feature maps and prune those features with high similarity for further compression. The overall filter pruning scheme is illustrated in Algorithm 1.

We adopt M-std as the diversity criteria. Specifically, in the i-th layer, the selected feature $\hat{\mathbf{X}}_i$ at DFS is:

$$\hat{\mathbf{X}}_i = \{\mathbf{X}_{i,j} \mid \text{M-std}(\mathbf{X}_{i,j}) \geqslant \beta\},\ j = 1, 2, ..., N_i\ , \quad (4)$$

where β is the hyper-parameter which is chosen according to percentiles of M-std values of all feature maps, making our method a global pruning across all layers.

To select a subset of features with lower correlations, we employ a direct way to delete redundant features with high similarity. In particular, we compute the cosine similarity among all the features of $\hat{\mathbf{X}}_i$, which can form a correlation set $\mathbf{s}_i = \{s_{i_{j,p}}\}$,

$$s_{i_{j,p}} = \frac{1}{T}\sum_{m=1}^{T}\frac{|\cos(\mathbf{x}_{i,j}^m, \mathbf{x}_{i,p}^m)\ |}{T}, \mathbf{x}_{i,*} \in \hat{\mathbf{X}}_i. \quad (5)$$

Algorithm 2 Similarity-aware feature map selection (SFS)

Input: Feature map set $\mathbf{X}_i$, threshold ν
Output: Selected feature subset $\mathcal{B}_i$

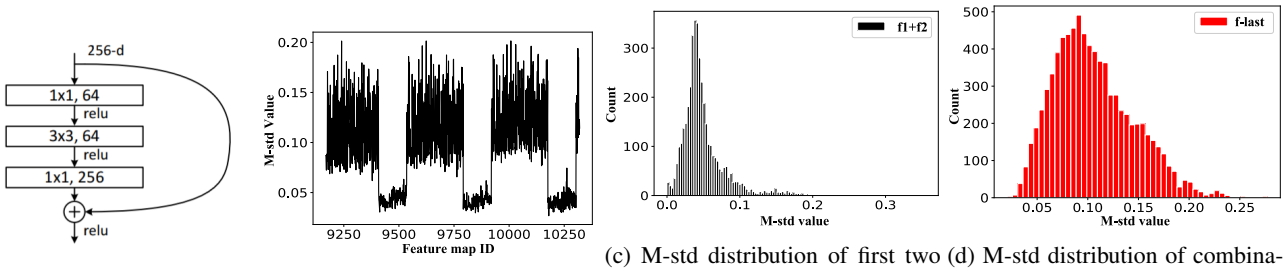
Initialize $\mathcal{B}_i \leftarrow \varnothing$
Form the correlation set $\mathbf{s}_i$ according to Equation 5
Find the max value max in $\mathbf{s}_i$
while $max > \nu$ **do**
    Find $\mathbf{X}_{i,r}$ and $\mathbf{X}_{i,c}$ have the max similarity value
    Let $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup \mathbf{X}_{i,r}$
    for $\mathbf{X}_{i,j} \in \mathbf{X}_i$ **do**
        if $s_{i_{r,j}} > \nu$ **then**
            Remove $s_{i_{r,j}}$ from $\mathbf{s}_i$ and remove $\mathbf{X}_{i,j}$ from $\mathbf{X}_i$
        end if
    end for
    Find max in $\mathbf{s}_i$
end while
Let $\mathcal{B}_i \leftarrow \mathcal{B}_i \cup \mathbf{X}_i$
return $\mathcal{B}_i$

We find the largest value in $\mathbf{s}_i$ and its corresponding feature pair, then we save one of the pair as a reference feature $\mathbf{x}_{i,r}$. As a result, we can safely delete features whose similarity with $\mathbf{x}_{i,r}$ is bigger than a pre-defined threshold ν because these features could be replaced by $\mathbf{x}_{i,r}$. SFS is summarized in Algorithm 2.

3.4 Pruning for Multiple Branch Networks

The multiple branch networks illustrate a kind of CNNs that the output of one layer may be the input of multiple subsequent layers, which are more complicated to prune than single branch networks. For instance, ResNet [He *et al.*, 2016a] is a representative example of multiple branch networks, which has a sequential branch and a shortcut branch. The outputs of these two branches will conduct an element-wise addition operation. Since the outputs require equal channel dimensions, this makes pruning ResNet more difficult.

We use two separate feature map selection processes for the sequential branch and the shortcut branch, respectively. The features except for the last layer within all sequential branches compose one group, the results after branches combination form another group. Filter pruning is operated on each group individually. These two separate filter pruning strategies are inspired by the statistic information of feature maps in PreResNet [He *et al.*, 2016b]. Figure 4a gives an example of the bottleneck architecture, which is one of the multiple-branch building blocks of ResNet. This bottleneck includes three layers with 1×1, 3×3, and 1×1 convolutional filters. The element-wise addition is performed channel by channel on two output feature maps of sequential and shortcut branches. We train a PreResNet-164 on the CIFAR10 dataset and compare the statistical information of feature maps between those in the first two layers of sequential branches (f1+f2) and those after the additional operation (f-last). Figure 4b shows the M-std values of part of the feature maps. The overall pattern of M-std for all feature maps is similar to Figure 4b. We can clearly see that all M-std values form two groups, the upper part corresponds to f1+f2, and the lower one represents f-last. Besides, the distribution of M-std



(a) bottleneck (b) M-std of part feature maps. (c) M-std distribution of first two layers within sequential branches. (d) M-std distribution of combination layers.

Figure 4: *Bottleneck* and M-std values of ResNet-164 on CIFAR-10.

Dataset	Method	Acc.(%)	FLOPs ↓(%)	Para.↓(%)
C10	VGGNet	93.74	0.0	0.0
	L1-Prune	93.12	-	88.5
	N-Slim∗	93.80	51.1	88.5
	PFGM∗	94.0	35.9	-
	Ours	**94.05**	**56.3**	**90.7**
C100	VGGNet	73.41	0.0	0.0
	L1-Prune	71.64	-	76.0
	N-Slim∗	73.48	37.1	75.1
	Ours	**73.69**	**45.0**	**75.9**

Table 1: Results of pruned VGGNet on CIFAR dataset. C10 and C100 mean the CIFAR 10 and CIFAR 100, respectively. Acc. is the classification accuracy, and Para. is short for parameters. The ↓ is the drop percent between the pruned model and the original model, the smaller, the better. Results with ∗ are got from original papers. − denotes the results are not reported.

Dataset	Method	Acc.(%)	FLOPs ↓(%)	Para.↓(%)
C10	PreResNet	94.86	0.0	0.0
	N-Slim∗	94.73	44.9	35.2
	Ours	**94.93**	**56.1**	**40.5**
C100	PreResNet	76.88	0.0	0.0
	N-Slim∗	76.09	50.6	29.7
	Ours	**76.18**	**53.4**	**35.9**

Table 2: Comparisons of pruning PreResNet on CIFAR dataset.

of f1+f2 is mainly from 0 to 0.1, which is shown in Figure 4c. From Figure 4d, we could conclude that the M-std of f-last is about 0.05 to 0.2. The percentile of f1+f2 is higher while the one of f-last is lower, which motivates us to use two different pruning thresholds. Thus, we utilize two feature map selection processes for f1+f2 and f-last layer, respectively.

4 Experiments

4.1 Experimental Setting

Dataset. We perform experiments on publicly available datasets. CIFAR10 and CIFAR100 [Krizhevsky *et al.*, 2009] are two widely used datasets with 32×32 colour natural images. They both contain $50,000$ training images and $10,000$ test images with 10 and 100 classes respectively. The data is normalized using channel means and standard deviations. And the data augmentation approach we used is consistent with [Liu *et al.*, 2017]. ILSVRC-2012 is a large-scale dataset with 1.2 million training images and $50,000$ validation images of 1000 classes. Following the common training procedure in [Liu *et al.*, 2017; He *et al.*, 2019], we adopt the same data augmentation approach and report the single-center-crop validation error of the final model.

Network models. We test the performance of our pruning method on several famous CNN models. VGGNet is a remarkable single branch network which is widely used for computer vision task. ResNet [He *et al.*, 2016a] and PreResNet [He *et al.*, 2016b] are two popular multiple branch network. MobileNet [Howard *et al.*, 2017] is a compact network designing for effective use on mobile devices.

Configuration. We train or fine-tune all the networks using SGD. For CIFAR, we set the mini-batch size to 64, epochs to 160 with a weight decay of 0.0015 and Nesterov momentum [Sutskever *et al.*, 2013] of 0.9. For ILSVRC-2012, we use the pre-trained ResNet-50 released by Pytorch. We train MobileNet for 60 epochs with a weight decay of 0.0015. The pruning ratio is determined by two factors, one is a percentile among M-std and the other is the threshold for SFS, i.e. 40% for DFS, 0.85 for SFS.

4.2 Compared Algorithms

- **L1-Prune**[1] [Li *et al.*, 2016] uses the $l1$-norm of filters as the important measurement.
- **ThiNet** [Luo *et al.*, 2017] is a feature-map based method that selects the filter subset reconstructing the next layer.
- **N-Slim** [Liu *et al.*, 2017] gets the importance of each filter during the training process according to the batch-normalization scaling factors.
- **PFGM** [He *et al.*, 2019] prunes redundant filters utilizing geometric correlation among filters in the same layer.

4.3 Experimental Results

Single branch network. We first prune the trained VGGNet on CIFAR10 and CIFAR100. We compare the performance of our method with the state-of-the-art methods. Table 1 lists the results of the classification accuracy, the reduction of parameters, and the decrease in FLOPs, respectively. Although all approaches can reduce the model size with limited accuracy drop, our method has the highest compression

[1]The result of L1-Prune is obtained from [Liu *et al.*, 2017].

Method	Acc.(%)	FLOPs ↓(%)	Para.↓(%)
ResNet	76.15	0.0	0.0
ThiNet	72.04	40.5	33.7
PFGM ∗	75.03	42.2	39.6
Ours	71.05	40.4	47.8

Table 3: Comparisons of pruning ResNet on ILSVRC-2012.

Dataset	Method	Acc.(%)	FLOPs ↓(%)	Para.↓(%)
C10	MobileNet	93.71	0.0	0.0
	R1	93.91	47.1	66.7
	R2	93.86	65.8	80.7
	R3	93.17	83.7	91.6
C100	MobileNet	74.19	0.0	0.0
	R1	75.40	29.3	43.8
	R2	74.77	47.8	58.3
	R3	72.73	62.6	68.3
ILSVRC	MobileNet	68.43	0.0	0.0
	R1	67.54	37.06	40.18
	R2	61.20	61.49	67.13

Table 4: Performance of pruned MobileNet on CIFAR and ILSVRC-2012. Rk denotes different compression step with β is 25% quantile of M-std and ν=0.85. R1 is the pruning based on the pre-trained model. R2 prunes the result of R1. R3 is the final pruning based on result of R2.

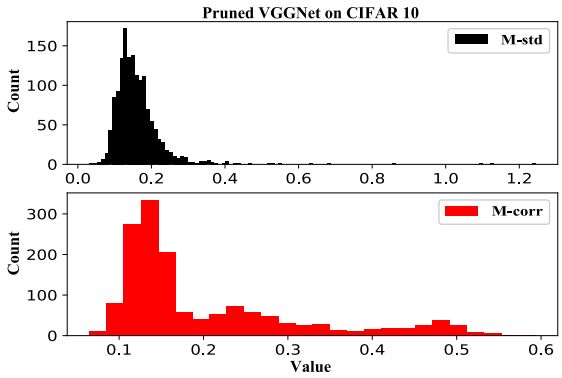


Figure 5: Distributions of M-std and M-corr of all feature maps in pruned VGGNet trained on CIFAR 10 dataset.

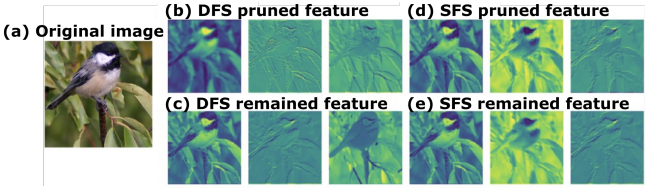


Figure 6: A demonstration of pruned and remained feature maps.

ratio. The scalar factors used in N-Slim are not powerful enough for compression since it does not consider relationships among features. Although PFGM can achieve satisfactory accuracy, it has low pruning ratio, since it neglects the feature diversity. On the other hand, our method considers both the diversity and similarity of features maps, which benefits the performance improvements of our method over other methods. When pruning 90.7% of parameters and 56.3% of FLOPs of VGGNet trained on CIFAR 10, our method surprisingly increases the accuracy by 0.31%. One possible reason is that our method reduces the unnecessary parameters that cause the overfitting of the original model.

Multiple branch network. We prune PreResNet-164 on CIFAR and ResNet-50 on ILSVRC-2012, respectively. The results are reported in Table 2 and Table 3. From the results, we can observe that even with multiple branch networks, our method can still compress the model to a satisfactory extent. After SFS and DFS, our method reduces up to 40% of parameters and 56.1% of FLOPs for PreResNet on CIFAR 10 while maintaining the accuracy as high as 94.93%.

Compact designed network. To further illustrate the generalization of our method, we prune MobileNet on both CIFAR and ILSVRC-2012 datasets. The performance of different prune ratios is given in Table 4. With the increase of the compression ratio, the accuracy of the pruned model drops gradually. Although the pruned model reduces 91.6% of parameters and 83.7% FLOPs, its accuracy as high as 93.17% on CIFAR 10.

The efficiency of feature map selection. The purpose of our two feature selections is to extract more diverse and less similar feature maps. We prune a VGGNet trained on CIFAR 10 and plot the statistic information of feature maps in Figure 5. It can be observed that most M-std values of feature maps are concentrated between 0.1 and 0.2, which is higher than the original model (i.e. 0.05 in Figure 2a). In addition, the M-corr values are almost smaller than 0.3, which is much lower compared with the original model (i.e. Figure 2b). These indicate the remained features have higher diversities and fewer similarities. Although there still exist feature maps with M-std values smaller than 0.05 and M-corr values bigger than 0.5, their percentage is quite small. As a result, these feature maps can keep the generalization ability of the model.

Visualization. We further visualize the pruned and remained feature maps to show the effectiveness of our approach. Figure 6 shows part of the feature maps of the first convolutional layer in ResNet-50 trained on LSVRC-2012. Each feature map in Figure 6b expresses limited information variance (e.g. ambiguous or no texture) compared with Figure 6c. The feature maps in Figure 6d are nearly the same as Figure 6e. Therefore, SFS prunes one of them to keep fewer similar features.

5 Conclusion

In this study, we investigate the statistical information of feature maps in CNN for analyzing the diversity and similarity. We propose two feature map selections, namely DFS and SFS, for removing redundant filters. The pruning method we proposed can significantly compress the size of CNNs and decrease the computational cost with almost no accuracy loss. To further validate the effectiveness of our proposed method, different pruning ratio strategies can be evaluated in the future. We will also explore more tasks, such as object detection and text classification.

References

[Ba and Caruana, 2014] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

[Galton, 1886] Francis Galton. Regression towards mediocrity in hereditary stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[Girshick *et al.*, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[Guo *et al.*, 2016] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016.

[Han *et al.*, 2015] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

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

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[He *et al.*, 2017] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[He *et al.*, 2018] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[He *et al.*, 2019] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[Howard *et al.*, 2017] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[Hu *et al.*, 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LeCun *et al.*, 1990] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[Li *et al.*, 2016] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

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

[Liu *et al.*, 2018] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[Liu *et al.*, 2019] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3296–3305, 2019.

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

[Luo *et al.*, 2017] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[Sutskever *et al.*, 2013] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[Wang *et al.*, 2019] Wenxiao Wang, Cong Fu, Jishun Guo, Deng Cai, and Xiaofei He. Cop: Customized deep model compression via regularized correlation-based filter-level pruning. In *International Joint Conference on Artificial Intelligence*, volume 2019, 2019.

[Yamamoto and Maeno, 2018] Kohei Yamamoto and Kurato Maeno. Pcas: Pruning channels with attention statistics for deep network compression. *arXiv preprint arXiv:1806.05382*, 2018.