LOSSLESS FILTER PRUNING VIA ADAPTIVE CLUSTERING FOR CONVOLUTIONAL NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

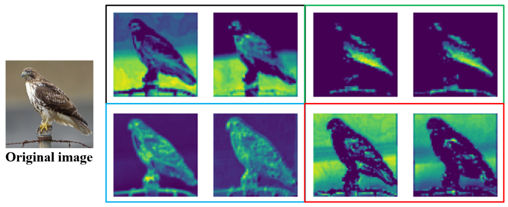
The filter pruning method introduces structural sparsity by removing selected filters and is thus particularly effective for reducing complexity. However, previous works face two common limitations. 1) The pruned filters are prevented from contributing to the final outputs, resulting in performance degradation, especially when it comes to a large pruning rate. 2) To recover accuracy, the time-consuming fine-tuning step is required. The cost in time and the need for training data make it difficult to deploy in real-world scenarios. To address the aforementioned limitations, we propose a novel filter pruning method called Cluster Pruning (CP). Our CP reconstructs the redundant filters from the perspective of similarity and removes them equivalently using the proposed channel addition operation in a lossless manner. Pruning in such a way allows CP to preserve as many learned features as possible while getting rid of the need for fine-tuning. Specifically, each filter is first distinguished by clustering and then reconstructed as the centroid to which it belongs. Filters are then updated to eliminate the effect caused by mistakenly selected. After convergence, CP can equivalently remove identical filters through the proposed channel addition operation. The strategies for adjusting the pruning rate and the adaptive coefficient for clustering make our CP even smoother and more efficient. Extensive experiments on CIFAR-10 and ImageNet datasets show that our method achieves the best trade-off between performance and complexity compared with other state-of-the-art algorithms.

1 INTRODUCTION

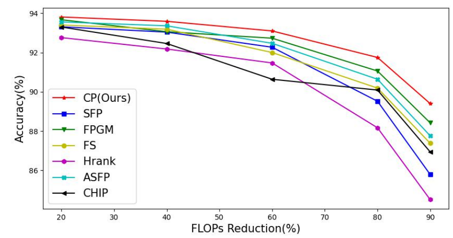
The state-of-the-art performance of deep convolutional neural networks (CNNs) (Ren et al., 2015) (Chen, 2015) (Lin et al., 2014) is based on deeper and wider architecture, which hinders the deployment in resource-limited devices due to its expensive computation cost and memory footprint. As an effective compression technique, network pruning has attracted numerous attention. Unlike fine-grained weight pruning (Guo et al., 2016) (Han et al., 2015) (LeCun et al., 1989) which causes unstructured sparsity, filter pruning directly deletes the whole filter and is efficient in saving memory usage and computational cost on general platforms.

Numerous studies have been conducted to study how to define unimportant filters. ”More-similar-less-important” is a promising viewpoint for filter pruning. The observation of these semantically similar feature map pairs, as shown in Fig. 1(a), which has also been pointed out by GhostNet (Han et al., 2020), indicates the prevalent redundancy on the feature level. For further investigation, we visualize filters after PCA in Fig. 1(c), which demonstrates the phenomenon of cluster formation. Given such natural clustering property, we intuitively seek to prune redundancy based on similarity. In comparison to the commonly used ℓ_p-norm criteria (Yu et al., 2021) (Lin et al., 2020c) (Li et al., 2020b), similarity-based criteria avoid the sensitivity to the filter distribution (He et al., 2019) which can lead to the erroneous removal of those relatively unimportant filters.

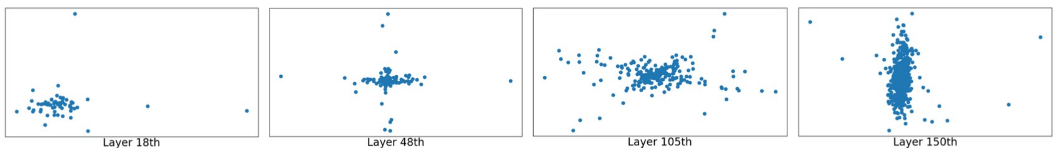
However, no matter which criteria the network is pruned under, previous works either directly remove filters or set them to zero (or masking). Although these filters are considered trivial under certain criteria, they do not make zero contribution and still contain semantic information helpful for further processing. Pruning in such manner will undoubtedly degrade performance, especially at



(a) Similar feature map pairs



(b) Performance of CP compared with other works.



(c) Filter distribution of ResNet-50.

Figure 1: (a) Similar feature map pairs of the second residual block in ResNet-50. Different pairs are framed in different colors. (b) Comparison with other state-of-the-art filter pruning methods of various FLOPs reduction when pruning ResNet-56 on CIFAR-10. We can see the advantage of CP under same configuration, especially at large pruning rate. (c) Visualization of filter distribution of ResNet-50 after decomposition.

large pruning rates. Furthermore, even if the accuracy can be regained through fine-tuning, the high computational cost and the requirement for training data make it inefficient to carry out in realistic scenarios.

In this paper, we propose Cluster Pruning (CP), a novel filter pruning method based on clustering. The core of our CP is that instead of removing or zeroing out unimportant filters, we intend to use the equivalence of the convolutional layer to remove filters without impact. To accomplish this, we gradually replace filters with the value of cluster centroids while training, which still preserves their ability to extract features. After convergence, we can obtain a network whose filters are divided into several clusters and exactly identical within each cluster. The proposed channel addition operation can discard all but one filter from each cluster by leveraging the linear and combinational properties of the convolutional layer. Our CP no longer requires the time-consuming fine-tuning process, since it is equivalent before and after pruning. As shown in Fig. 1(b), our method can maintain comparable performance even at a large pruning rate, demonstrating our advantage in the trade-off between complexity and performance.

To summarize, our main contributions are three-fold as follows:

- We investigate the similarity between filters and propose our Cluster Pruning (CP), which prunes the network based on such similarity. We propose an adaptive coefficient for clustering which well preserves the extracted features. Various strategies for the pruning rate are designed to smooth the pruning process.
- The channel addition operation is proposed to equivalently remove generated identical filters. It enables CP to omit the time-consuming fine-tuning step and avoids steep accuracy degradation especially at large pruning rates.
- Extensive experiments on CIFAR-10 and ImageNet datasets demonstrate that the proposed CP can achieve the best trade-off between complexity and performance compared with other state-of-the-art methods.

2 Related Work

In this section, we will give a comprehensive overview of different pruning methods, from the aspects of pruning structure, pruning criteria and pruning manner.

Pruning Structure. Early works of network pruning concentrate on fine-grained granularities: weight-level (LeCun et al., 1989), vector-level (Wen et al., 2016) and kernel level (Li et al., 2017).

For architecture consistency, this type of non-structured method can only zeroize the unnecessary parameters rather than prune. The need of saving special coordinates for every weight makes it difficult to satisfy nowadays trillion-level models. During inference, although the model sizes and the number of multiply-accumulate operations are dramatically decreased, the irregular structure of dense matrices requires additional computations and special hardware designs for acceleration.

Pruning Criteria. Selecting a proper criterion to identify filters which need to be pruned is a major point of network pruning. (Li et al., 2017) prunes filters with small ℓ_1-norm in each layer, while (Han et al., 2016) (Lin et al., 2018) are based on ℓ_2-norm. (Molchanov et al., 2017) uses the Taylor series to estimate the loss change after each filter's removal and prune the filters that cause minimal training loss change. Network slimming (Liu et al., 2017) applies LASSO on the scaling factors of BN, by setting the BN scaling factor to zero, channel-wise pruning is enabled. FPGM (He et al., 2019) prunes filters nearest to the geometric median of each layer using Euclidian distance, refraining from the limits of norm-based criterion. (Hu et al., 2016) indicates activation may also be an indicator, and it introduces Average Percentage Of Zeros (APoZ) to judge if one output activation map is contributing to the result. Regardless of the criteria used to identify redundancy, previous works either remove or zero out unimportant parameters, which will inevitably result in information loss. Our method, however, solves these drawbacks by an equivalent approach, which will be discussed in Section 3.2

Pruning Manner. The traditional filter pruning pipeline will reduce the model capacity of the original models, thus facing the problem of unrecoverable performance loss after incorrect pruning. Besides, the overreliance on pre-trained models and the huge time cost of fine-tuning make it unsuitable to deploy in real-world scenarios. To overcome that, SFP (He et al., 2018) zeroizes pruned filters with a binary mask and updates filters in a soft manner to maintain the capacity of the network. Based on SFP, ASFP (He et al., 2020b) gradually increases the pruning rate to alleviate the accuracy drop caused by pruning and stabilize the whole pruning process. STP (Rong et al., 2020) uses the first-order Taylor series to measure the importance of filters while SRFP (Cai et al., 2021a) removes filters smoothly by gradually decaying to zero so that it can better preserve the trained information. (Cai et al., 2021b) analysis the characteristic of hard manner and soft manner, and propose GHFP to smoothly switch from soft to hard to achieve a balance between performance and convergence speed. Similarly, our work can be also classified as a SFP-basd method, which well guarantees the model capacity after pruning.

3 Methodology

3.1 Formulation

In this section, we formally introduce the symbol and notation. The deep CNN network can be parameterized by $W_i \in \mathbb{R}^{N_{out}^i * N_{in}^i * h_i * w_i}$, where $1 \leqslant i \leqslant L$ and L is the total number of convolutional layers in a network, h_i and w_i represent the height and width of a kernel. N_{out}^i and N_{in}^i denote the number of output channels and input channels for the i-th convolution layer, respectively. The output feature map $O_i \in \mathbb{R}^{N_{out}^i * h_{i+1} * w_{i+1}}$ is calculated by the convolution operation of input feature map $I_i \in \mathbb{R}^{N_{in}^i * h_i * w_i}$ and W_i, shown as below:

$$O_i = W_i * I_i, \tag{1}$$

where $O_{i,j}$ and $W_{i,j}$ denote the j-th output channel of output feature maps and the j-th filter of the i-th layer, respectively. Assume that the pruning rate of i-th layer is P_i, then the number of filters after pruning will reduce from N_{out}^i to $N_{out}^i * (1 - P_i)$. Accordingly, the size of the pruned output tensor would be $N_{out}^i * (1 - P_i) * h_{i+1} * w_{i+1}$. Given a dataset $D = \{(x_i, y_i)\}_{i=1}^n$ and a desired sparsity level k (i.e., the number of remaining filters), filter pruning can be formulated as:

$$\begin{aligned} &\min_W \ell\left(W^{'}; D\right) = \min_W \frac{1}{n} \sum_{i=1}^{n} \ell\left(W^{'}; (x_i, y_i)\right), \\ &\text{s.t.} \quad \left\|W^{'}\right\|_0 \leqslant k, \end{aligned} \tag{2}$$

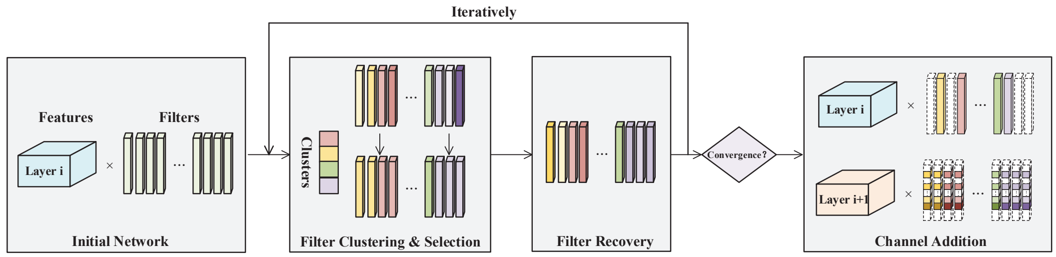


Figure 2: Overview of our CP. Filters are clustered and manually reconstructed in the filter clustering and selection step, and are retrained in the filter recovery step. This pipeline is executed iteratively till convergence before channel operation operation is conducted to obtain the compact model. Filters in similar colors are considered close to each other in Euclidean space. (Best viewed in color.)

where $\ell(.)$ is the loss function (e.g., cross-entropy loss), $W^{'}$ is the filter set of the pruned network. Typically, hard filter pruning prunes the model layer by layer and fine-tune iteratively to complement the degradation of the performance, while soft filter pruning (SFP) and its variants prune filters by simply zeroizing them with a Boolean matrix, indicates whether the filter is pruned or not.

3.2 Cluster Pruning

Based on the observation that filters are naturally clustered in each layer, our CP intends to manually introduce redundancy by generating identical filters. As depicted in Fig. 2, our pipeline is composed of four steps, 1) Clustering Step 2) Selection Step 3) Recovery Step and 4) Channel Addition. The first three steps are carried out iteratively till convergence, right before those redundant filters are pruned with the proposed channel addition operation.

Clustering Step: We first reshape 4D tensor W_i into a 2D matrix of size $(N_{in}^i * h_i * w_i) * N_{out}^i$. Therefore, each column of this 2D matrix stands for the filter of the weight tensor. Then we can regard them as coordinates in high dimensional space and conduct clustering as:

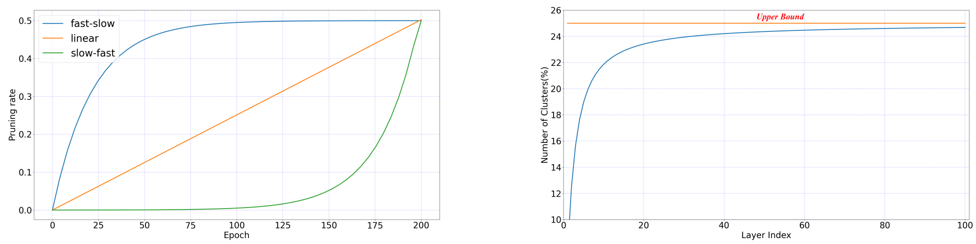
$$\min \frac{1}{N_{out}^i} \sum_{j=1}^{N_{out}^i} \left\| W_{ij} - \mu_{c_i^{(j)}} \right\|_2, \tag{3}$$

where c_i is the amount of clusters of i-th layer, $\mu_{c_i^{(j)}}$ is the cluster centroid which W_{ij} is assigned to. The optimization can be solved by $kmeans$ (MacQueen, 1967) or other clustering algorithms. After clustering, filters in the i-th layer can be divided into c_i clusters, where their norm are relatively close and may have a certain linear relationship (He et al., 2019). Filters within each cluster can generate similar feature maps of the next layer, therefore, the clusters with more filters can be identified as more redundant.

Selection Step: We adopt the Euclidean distance to evaluate the similarity of filters as Eq. 4. In general, those filters close to assigned cluster centroids are considered much more redundant. Based on this understanding, we can find the "most similar" filters and replace them with the centroids to which they belong. Since the distance between these filters and corresponding clustering centroids is small, such reconstruction does not have too much negative impact with a small pruning rate.

$$\begin{aligned} W_{ij^*} &\in \underset{j\in\left[1,N_{out}^i\right]}{arg\min} \left\| W_{ij} - \mu_{c_i^{(j)}} \right\|_2, \\ W_{ij^*} &= \mu_{c_i^{(j^*)}},\ \text{for } 1 \leqslant j^* \leqslant N_{out}^i * P_i. \end{aligned} \tag{4}$$

To further avoid the severe impact of such reconstruction when 1) the pruning rate is large 2) pruning the network from scratch, we limit P_i to gradually increase from the initial value towards the goal pruning rate P_{goal}. The definition of P_i is listed as follows:



(a) Different pruning strategy

(b) Adaptive number of clusters

Figure 3: (a) Illustration of different strategies to control the speed of pruning ResNet-56 on CIFAR-10 dataset, where the goal pruning rate is 0.5. (b) The adaptive coefficient for clustering, where number of clusters (vertical axis) represents the percentage of total number of filters of each layer.

$$P_i = f\left(P_i^{init}, P_i^{goal}, epoch\right), \tag{5}$$

where P_i^{init} represents the initial pruning rate for the i-th layer. To stabilize the pruning process, we consider two kinds of strategies, which are linear increase and exponential increase respectively.

The linear increase strategy can be written as:

$$P_i = \frac{P_{goal}}{epoch_{\max}} \times epoch, \tag{6}$$

where $epoch_{max}$ is the number of total training epochs. Starting from zero, the pruning rate will linearly increase until it reach the goal pruning rate, as illustrated in Fig. 3(a). Similarly, the exponential increase strategy can be given by:

$$P_i = k_1 \times \left(e^{k_2 \times epoch} - 1\right), \tag{7}$$

where k_1 and k_2 are two hyper-parameters to control the growth speed of the pruning rate. For different setting of hyper-parameters, the trend of pruning rate can be fast to slow, or vice versa, as shown in Fig. 3. With the adaptive strategies of pruning rate, the whole pruning process will become smoother, which is intuitively reflected on the final performance.

Recovery Step: Filters can be wrongly reconstructed since the cluster centroids are changing along the whole iterative pruning process. Thus after the selection step, we retrain the network for some epochs so that it can recover from the wrong assignment. As the pruning step is integrated into the normal training schema, the model can be trained and pruned synchronously.

Channel Addition: Our CP iterates over the clustering, selection and recovery steps till the model converges. Afterwards, each layer contains lots of identical filters within different clusters, where we can utilize the proposed channel addition operation to obtain a actual compact model. As shown in Fig. 2, suppose filters $W_{i,m}$ and $W_{i,n}$ are identical, we can safely prune $W_{i,n}$ by adding the n-th channel to the m-th of the filters in the next layer, which can be represented by:

$$\begin{aligned} & W_{i+1} * I_{i+1} = W_{i+1}^{'} * I_{i+1}^{'}, \\ & \text{s.t.} \quad \begin{cases} W_{i,m} = W_{i,n}, \\ W_{i+1,:,m}^{'} = W_{i+1,:,m} + W_{i+1,:,n}, \\ I_{i+1}^{'} = W_{i,n}^{C} * I_i, \end{cases} \end{aligned} \tag{8}$$

where $I_{i+1}^{'}$ denotes the input feature map of i+1-th layer after pruning $W_{i,n}$. $W_{i+1,:,m}$ and $W_{i+1,:,n}$ represent the m-th and n-th channel of W_{i+1}, respectively. $W_{i,n}^{C}$ is the complementary set of $W_{i,n}$. It is worth noting that the mainstream CNN structures consist of the common Conv-BN cell. Thus, in order to omit the time-consuming fine-tuning step after channel addition, inspired by ConvNeXt

(Liu et al., 2022) which only preserve part of normalization layers, we remove those BN layers after the target convolutional layers. The details of our CP are illustrated in Algorithm 1.

3.3 Adaptive Coefficient for Clustering

Setting the hyper-parameter c_i properly is quite critical since it determines both the performance and complexity of the pruned network. One possible approach is to set c_i to a constant across layers, which is simple and intuitive. However, as deeper layers suppose to have more channels to extract various features, setting it to a constant will do harm to the diversity of features. Another possible way is to use metrics such as Silhouette Coefficient, which allows for fine-grained settings for each layer. Although it can provide better accuracy, the time-consuming process for traversing each layer makes it less efficient. Besides, it can be hard to control the overall FLOPs reduction, as it is automatically determined. For the best trade-off between accuracy and FLOPs reduction, we propose our adaptive coefficient for clustering as follows.

Considering that the network logically should be assigned more clusters as it goes deeper, we scale the pruning rate to the worse case and restrict it to satisfy the realistic pruning rate P_r as in Eq. 9, which factually treats P_r as the lower pruning bound.

$$
\begin{gathered}
P_r \leqslant \frac{\sum_{i=1}^{L}\left(P_i - \max\left(\frac{c_i}{N_{out}^i}\right)\right) N_{out}^i}{\sum_{i=1}^{L} N_{out}^i} \leqslant \frac{\sum_{i=1}^{L}\left(P_i - \frac{c_i}{N_{out}^i}\right) N_{out}^i}{\sum_{i=1}^{L} N_{out}^i}, \\
then \quad \max\left(\frac{c_i}{N_{out}^i}\right) \leqslant \frac{\sum_{i=1}^{L}\left(P_i - P_r\right) N_{out}^i}{\sum_{i=1}^{L} N_{out}^i}
\end{gathered}
\tag{9}
$$

In our experiment, since the pruning rate is same across layers, we can simply scale the arctangent function to modify its upper bound, as shown in Fig. 3(b). The ablation studies validate the advantages of the proposed adaptive coefficient with other methods.

Algorithm 1 Cluster Pruning Algorithm

Input: training set D, pruning rate P_i, number of clusters c_i, model with parameters $W = \{W_i, 0 \leqslant i \leqslant L\}$, total epochs t_{epoch}

1: **for** $epoch = 1; epoch \leqslant t_{epoch}; epoch++$ **do**
2: Update the model parameter W based on D;
3: **for** $i = 1; i \leqslant L, i++$ **do**
4: Filter Clustering based on c_i;
5: Filter Selection base on Equation 4;
6: Set each selected filter W_{ij^*} to centroid $\mu_{c_i^{(j^*)}}$ it belongs to;
7: **end for**
8: **end for**
9: Obtain the pruned model by Channel Addition;

Output: The compact model with parameter $W' = W^{t_{epoch}}$

4 Experiment

4.1 Experimental Settings

Datasets. We conduct experiments on two publicly available datasets CIFAR-10 (Krizhevsky, 2009) and ImageNet (Russakovsky et al., 2015) to show the efficiency of our method. CIFAR-10 dataset contains 60000 32*32 images with 10 classes, 50000 images for training while the remaining for testing. ImageNet is a large-scale dataset which contains 1.28 million 224*224 training images and 50k validation images drawn from 1000 categories. For both datasets, we first preprocess the data by subtracting the mean and dividing the standard-deviation, then adopt the same data augmentation scheme as (He et al., 2018) (He et al., 2019).

Table 1: Pruning results of ResNet-56 and VGG16 on CIFAR-10. "PR" denotes the actual pruning rate. CP obtains the maximum FLOPs reduction with close accuracy drop on ResNet-56. On VGG16, CP outperforms other methods with both performance and complexity under different pruning rates.

Arch	Method	Baseline(%)	Pruned Accu.(%)	↓ Accu. (%)	↓ FLOPs(%)
ResNet-56	FPGM (He et al., 2019)	93.59	93.49	0.10	52.6
	HRank (Lin et al., 2020a)	93.26	93.17	0.09	50.0
	ABCPruner (Lin et al., 2020b)	93.26	93.23	0.03	54.1
	HFP (Enderich et al., 2021)	93.30	93.30	0.00	56.0
	GDP (Guo et al., 2021)	93.90	93.55	0.35	34.3
	FS (Lin et al., 2021)	93.26	93.19	0.07	41.5
	SCOP (Tang et al., 2020)	93.70	93.64	0.06	56.0
	CP (ours, PR=50%)	93.39	93.34	0.05	**58.6**
VGG-16	AutoPrune (Xiao et al., 2019)	92.40	91.50	0.90	23.0
	VCNNP (Zhao et al., 2019)	93.25	93.18	0.07	39.1
	CP (ours, PR=25%)	93.87	93.92	↑**0.05**	37.1
	GAL (Lin et al., 2019)	93.96	90.73	3.23	45.2
	GS (Li et al., 2020a)	94.02	93.59	0.43	60.9
	CP (ours, PR=40%)	93.87	93.51	**0.36**	**61.9**
	HRank (Lin et al., 2020a)	93.96	92.34	1.62	65.3
	CP (ours, PR=50%)	93.87	92.72	**1.15**	**72.2**

Network Architecture. We study the performance on various mainstream CNN models, including VGGNet (Simonyan & Zisserman, 2015) with a plain structure and ResNet series (He et al., 2016) with residual blocks. For CIFAR-10 dataset, we test our CP on VGGNet and ResNet-56, while on ImageNet, we test it on commonly used ResNet-50. Since ResNet is less redundant than VGGNet, we will focus more on it.

Configurations. The models are trained from scratch using SGD with a weight decay of 0.0005 and Nesterov momentum of 0.9 (Sutskever et al., 2013). For CIFAR-10, we train the model with a batchsize of 128 for 200 epochs and use an initial learning rate as 0.1, while on ImageNet, the number of epochs, batchsize and the initial learning rate are set to 100, 256 and 0.1, respectively. The learning rate is divided by 5 at epoch 60, 120, 160 on CIFAR-10, and is divided by 10 every 30 epochs on ImageNet. We prune all the convolutional layers of VGGNet for it has a plain structure. While for ResNet, due to the existence of shortcut, we prune all the convolutional layers except for the last one of every residual block for simplification. We compare our results with other state-of-the-art filter pruning methods and all the results are obtained from the original papers. Each experiment is conducted 3 times and we report the mean value for comparison.

4.2 Pruning Results on CIFAR-10

ResNet-56. We summarize the results of ResNet-56 on CIFAR-10 in Table 1. Our CP achieves comparable performance on CIFAR-10 compared with other filter pruning methods. For example, GDP accelerates ResNet-56 by 34.3% with 0.35% accuracy drop while our CP can prune 58.6% of total FLOPs with only 0.05% accuracy drop. Furthermore, CP outperforms FS on accuracy loss (0.05% v.s. 0.07%) and pruned FLOPs (58.6% v.s. 41.5%) on ResNet-56.

VGG-16. Table 1 shows the performance of various pruning methods on VGG-16. Those numbers in brackets denote the overall pruning rate. Our CP provides lower accuracy loss compared to AutoPrune and VCNNP (-0.05% v.s. 0.07%, 0.9%) with similar FLOPs reduction. Compared with GAL and Hrank, CP outperforms each of them in all aspects (45.2% v.s. 61.9% and 65.3% v.s. 72.2% in FLOPs reduction, 0.36% v.s. 3.23% and 1.15% v.s. 1.62% in accuracy loss), which proves its ability to compress and accelerate a neural network with a plain structure.

Table 2: Pruning results of ResNet on ImageNet, where “BL” and ”PR” denote baseline and pruned, respectively. Our method achieves only 1.06% Top-1 drop with the most 61.8% FLOPs pruned.

Arch	Method	Top-1 Accu. BL/PR(%)	Top-5 Accu. BL/PR(%)	Top-1 Accu.↓(%)	Top-5 Accu.↓(%)	FLOPs↓(%)
ResNet-50	SFP (He et al., 2018)	76.15/74.61	92.87/92.06	1.54	0.81	41.8
	FPGM (He et al., 2019)	76.15/74.83	92.87/92.32	1.32	0.55	53.5
	LFPC (He et al., 2020a)	76.15/74.46	92.87/92.04	1.69	0.83	60.8
	MetaPruning (Liu et al., 2019)	76.60/75.40	-	1.20	-	51.1
	HRank (Lin et al., 2020a)	76.15/74.98	92.87/92.33	1.17	0.54	43.8
	ABCPruner (Lin et al., 2020b)	76.01/73.52	92.96/91.51	2.49	1.45	56.6
	FS (Lin et al., 2021)	76.13/74.68	92.86/92.17	1.45	0.69	45.5
	SCOP (Tang et al., 2020)	76.15/75.26	92.87/92.53	0.89	0.34	54.6
	CP (ours)	75.56/74.50	92.70/92.02	1.06	0.68	**61.8**

Table 3: Comparison of test accuracies between different guidelines for clustering when pruning ResNet-56 on CIFAR-10.

	Accuracy(%)	FLOPs ↓(%)
Constant ($c_i = 5$)	92.98	54.6
Silhouette Coefficient	93.39	41.2
Adaptive Coefficient	93.34	58.6

Table 4: Comparison of test accuracies between different clustering methods when pruning ResNet-56 on CIFAR-10.

	Accuracy(%)
Kmeans	93.34
Spectral	93.19
Agglomerative	93.23
Meanshift	93.31

4.3 PRUNING RESULTS ON IMAGENET

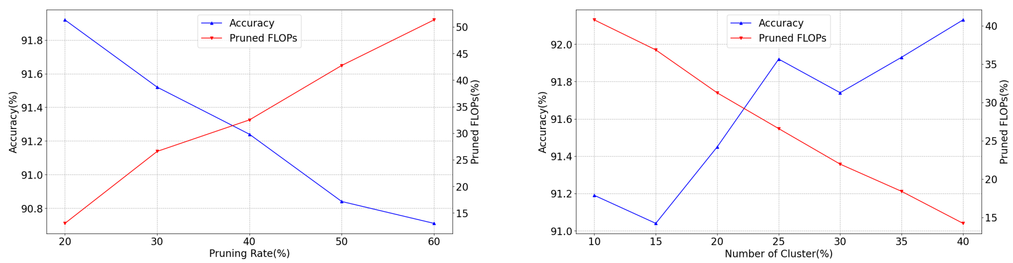
For the ImageNet dataset, we evaluate our CP on ResNet-50 and compare our results with other state-of-the-art methods. Table 2 shows the performance of our CP compared with the previously mentioned methods. As a result, our method can achieve better FLOPs reduction and lower accuracy drop. For instance, our CP can prune 61.8% FLOPs of ResNet-50 with only 1.06% top-1 and 0.68% top-5 accuracy drop, while ABCPruner can only prune 56.6% of total FLOPs with 2.49% top-1 accuracy drop. Compared with HRank, CP achieves 0.11% less top-1 accuracy drop and yields 18.0% more FLOPs reduction. This proves our CP can obtain a better trade-off between performance and complexity compared with other state-of-the-art pruning methods, which is highly attributed to the preservation of feature extraction when CP reconstructs filters.

4.4 ABLATION STUDY

Varying pruning rates. To better shed light on the performance of our CP, we present the test accuracy under different pruning rates for ResNet-20 in Fig. 4(a). The network is trained from scratch with the number of clusters c_i is fixed at 25%. As shown in Fig. 4(a), with the decrease in pruning rate, the test accuracy increases linearly, and the pruned FLOPs decreases gradually as well. When the pruning rate is too large, it is difficult to recover even after fine-tuning. Thus, for a better trade-off between model complexity and performance, the pruning rate should be chosen carefully.

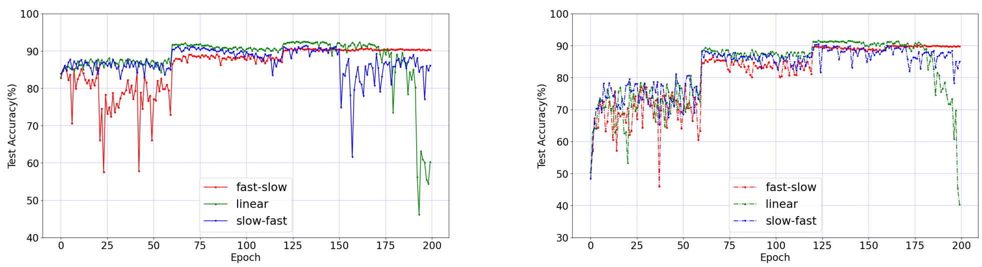
Adaptive coefficient for clustering. We demonstrate the effectiveness of the proposed adaptive coefficient for clustering in Table. 3. Compare with the constant setting, the adaptive coefficient can achieve better performance and more FLOPs reduction. This is mainly because it allows deeper layers to preserve more various features. We also compare our method with using Silhouette Coefficient, whose accuracy is slightly higher than ours. However, as it automatically determines c_i for each layer, the total FLOPs reduction is uncontrollable. It tends to not prune the network as much as possible. Besides, the time cost of traversing all possible choices is intolerant. In summary, our adaptive coefficient can well balance performance and complexity.

Influence of the number of clusters. In order to explore the influence of c_i, we compare the test accuracy of pruned ResNet-20 with different c_i on CIFAR-10 dataset. As shown in Fig. 4(b), the overall trend of accuracy is that the larger c_i is, the higher test accuracy becomes. However, the



(a) Different pruning rate on ResNet-20 (b) Different number of clusters on ResNet-20

Figure 4: Comparison of test accuracies of different pruning rates and number of clusters for ResNet-20 on CIFAR-10 dataset with the fast-slow stratgy.



(a) ResNet-20 from pre-train (b) ResNet-20 from scratch

Figure 5: The training process of ResNet-20 on CIFAR-10 with different strategies of pruning rate while the goal pruning rate is set to 50%. Top-1 accuracy is reported for comparison.

pruned FLOPs is also downward and there exist some knee points of accuracy, which suggests that c_i should not be too large and it needs to be well designed to balance the general performance.

Different pruning strategies. We compare the results of three decay strategies when pruning ResNet-20 on CIFAR-10 with the training epochs increasing. As shown in Fig. 5, no matter if it is trained from scratch or not, the **fast-slow** strategy outperforms others although it causes a relatively larger accuracy loss at the initial stage of the training process. The **linear** strategy creates identical filters in a much smoother manner which leads to a disastrous non-convergence issue.

Choice of clustering methods. We compare the experimental results under different clustering methods, as shown in Table. 4. Clustering methods including Kmeans, Spectral, Birch, and Agglomerative are evaluated, respectively. No significant performance differences are observed, which further validates the robustness of our method. A logical explanation of such a phenomenon can be that our reconstruction of filters mainly concentrates on locations where density is high and these clustering methods all well identify such locations.

5 Conclusion

To conclude, in this paper, we point out the limitations of previous works and propose a novel filter pruning method based on clustering, named Cluster Pruning, to accelerate deep CNNs. Specifically, CP considers the similarity between filters as redundancy and removes them in a softer way without information loss. Thanks to these, CP allows filters to be pruned more stable as the training procedures run and achieves state-of-the-art performance in several benchmarks. In the future, we plan to work on how to combine CP with other norm-based criteria and more importantly, other acceleration algorithms, e.g., knowledge distillation and matrix decomposition, to push the performance to a higher stage.

REFERENCES

Linhang Cai, Zhulin An, Chuanguang Yang, and Yongjun Xu. Softer pruning, incremental regularization. *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 224–230, 2021a.

Linhang Cai, Zhulin An, Chuanguang Yang, and Yongjun Xu. Soft and hard filter pruning via dimension reduction. *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2021b.

Yahui Chen. Convolutional neural network for sentence classification. 2015.

Lukas Enderich, Fabian Timm, and Wolfram Burgard. Holistic filter pruning for efficient deep neural networks. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 2595–2604, 2021.

Yi Guo, Huan Yuan, Jianchao Tan, Zhangyang Wang, Sen Yang, and Ji Liu. Gdp: Stabilized neural network pruning via gates with differentiable polarization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5219–5230, 2021.

Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *ArXiv*, abs/1608.04493, 2016.

Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1577–1586, 2020.

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural network. *ArXiv*, abs/1506.02626, 2015.

Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *arXiv: Computer Vision and Pattern Recognition*, 2016.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *ArXiv*, abs/1808.06866, 2018.

Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4335–4344, 2019.

Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2006–2015, 2020a.

Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, Chenggang Clarence Yan, and Yi Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Transactions on Cybernetics*, 50:3594–3604, 2020b.

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

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In *NIPS*, 1989.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *ArXiv*, abs/1608.08710, 2017.

Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8015–8024, 2020a.

Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8015–8024, 2020b.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1526–1535, 2020a.

Mingbao Lin, Rongrong Ji, Yu xin Zhang, Baochang Zhang, Yongjian Wu, and Yonghong Tian. Channel pruning via automatic structure search. *ArXiv*, abs/2001.08565, 2020b.

Mingbao Lin, Rongrong Ji, Shaojie Li, Qixiang Ye, Yonghong Tian, Jianzhuang Liu, and Qi Tian. Filter sketch for network pruning. *IEEE transactions on neural networks and learning systems*, PP, 2021.

Shaohui Lin, R. Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, 2018.

Shaohui Lin, R. Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David S. Doermann. Towards optimal structured cnn pruning via generative adversarial learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2785–2794, 2019.

Shaohui Lin, R. Ji, Yuchao Li, Cheng Deng, and Xuelong Li. Toward compact convnets via structure-sparsity regularized filter pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 31:574–588, 2020c.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, K. Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3295–3304, 2019.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2755–2763, 2017.

Zhuang Liu, Hanzi Mao, Chaozheng Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. 2022.

J. MacQueen. Some methods for classification and analysis of multivariate observations. 1967.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv: Learning*, 2017.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

Jintao Rong, Xiyi Yu, Mingyang Zhang, and Linlin Ou. Soft taylor pruning for accelerating deep convolutional neural networks. *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pp. 5343–5349, 2020.

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

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2015.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. On the importance of initialization and momentum in deep learning. In *ICML*, 2013.

Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing Xu, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. *ArXiv*, abs/2010.10732, 2020.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Helen Li. Learning structured sparsity in deep neural networks. *ArXiv*, abs/1608.03665, 2016.

Xia Xiao, Zigeng Wang, and Sanguthevar Rajasekaran. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *NeurIPS*, 2019.

Fang Yu, Chuanqi Han, Pengcheng Wang, Ruoran Huang, Xi Huang, and Li Cui. Hfp: Hardware-aware filter pruning for deep convolutional neural networks acceleration. *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 255–262, 2021.

Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2775–2784, 2019.