Deep CNN Model Compression via Filter Pruning For Efficient ConvNets

A BTP Report

by

Nikhil Duppali

Mohammad Farazuddin

V Madhukar Reddy

Roll No. : S20170010045

Roll No. : S20170010097

Roll No. : S20170010183

INDIAN INSTITUTE OF INFORMATION

TECHNOLOGY SRICITY

26-12-2020

Deep CNN Model Compression via Filter Pruning For Efficient ConvNets

A BTP report

Submitted in partial fulfillment of the

requirements for the award of the degree

of

Bachelor of Technology

by

Nikhil Duppalli

Mohammad Farazuddin

Madhukar Reddy

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

DECEMBER 2020

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Deep CNN Model Compression via Filter Pruning for Efficient ConvNets**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2014 to January 2017 under the supervision of Prof. Viswanath Pulabaigari, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the student with date

(Nikhil Dupally)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date

(Prof. Viswanath Pulabaigari)

Nikhil Dupally has successfully completed his BTP Examination held on

Signature of BTP Advisor	BTP Coordinator
Date:	Date:

Signature of BTP Panel Member:1	Signature of BTP Panel Member:2	Signature of External Examiner
Date:	Date:	Date:

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Deep CNN Model Compression via Filter Pruning for Efficient ConvNets**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2014 to January 2017 under the supervision of Prof. Viswanath Pulabaigari, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the student with date

(Mohammad Farazuddin)

_ _

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date

(Prof. Viswanath Pulabaigari)

_ _

XYZ has successfully completed his BTP Examination held on

Signature of BTP Advisor	BTP Coordinator
Date:	Date:

Signature of BTP Panel Member:1	Signature of BTP Panel Member:2	Signature of External Examiner
Date:	Date:	Date:

_ _

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SRICITY

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the BTP entitled "**Deep CNN Model Compression via Filter Pruning for Efficient ConvNets**" in the partial fulfillment of the requirements for the award of the degree of B. Tech and submitted in the Indian Institute of Information Technology SriCity, is an authentic record of my own work carried out during the time period from January 2014 to January 2017 under the supervision of Prof. Viswanath Pulabaigari, Indian Institute of Information Technology SriCity, India.

The matter presented in this report has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the student with date

(V Madhukar Reddy)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of BTP Supervisor with date

(Prof. Viswanath Pulabaigari)

V Madhukar Reddy has successfully completed his BTP Examination held on

Signature of BTP Advisor

Date:

BTP Coordinator

Date:

Signature of BTP Panel Member:1

Date:

Signature of BTP Panel Member:2

Res Date:

Signature of External Examiner

Date:

Contents

0.1 Introduction Problem Statement

In recent years, Convolutional Neural Networks (CNN) have gained significant attention from researchers in the field of computer vision due to its impeccable performance in several tasks including classification and detection [1]. The wide usage of deep CNNs in numerous applications creates an increasing demand for memory (parameter storage) and computation. To address this key issue, various attempts have been made in the literature. One such attempt focuses on training the deep CNNs with limited data [2–4]. Another line of research has shown better performance in reducing the overhead of computational power and memory storage, which mainly focuses on model compression by pruning connections [5, 6] and filters [7–9].

0.2 Related Works

We illustrate the efforts found in the published literature for deep model compression, separately by two major categories of approaches. They are :-

- Connection Pruning
- Filter Pruning

0.2.1 Connection Pruning

Connection pruning methods induce sparsity into the neural network. A simple approach is to prune the connections with unimportant weights (parameters). However, this method requires quantifying the significance of the parameters. In this direction, Lecun [10] and Hassibil [11] have utilized second-order derivative information to quantify the importance of network connections (parameters). However, computing second-order derivatives of all the connections is expensive. Chen [12] used a low-cost hash function to group the weights into a single bucket such that weights in the same bucket have roughly the same parameter value. Hu [13] introduced a network trimming approach that iteratively prunes the zero-activation neurons. Wu [6] proposed a method called BlockDrop to dynamically learn which layers to execute during the inference to reduce the total computation time. Han [5] developed a pruning technique based on the absolute value of the parameter. In [5], the parameters with the absolute value below a certain threshold are fixed to zero. These pruning methods are suggested in the scenarios where the majority of the network parameters belong to Fully Connected (FC) layers. For deep models such as ResNet [14] and DenseNet [15], these kind of pruning methods might not be suitable. However, the use of modern deep learning models for different applications is the recent trend.

0.2.2 Filter Pruning

Compared to other network pruning methods, filter pruning methods are generic, which do not require the support of any special software/hardware. Due to this reason, filter pruning methods have gained popularity among researchers in recent years. In general, filter pruning methods [8, 16–18] compute the importance of filters so that unimportant filters can be pruned from the model. In filter-pruning methods, after each iteration, retraining is required to regain the classification performance, which is dropped due to pruning the filters. Abbasi [19] proposed a greedy-based compression scheme for filter pruning. Similarly, Li [9] employed a greedy approach to prune the filters with less filter norm. In [20], redundant channels are investigated based on the distribution of channel parameters. Ding [21] proposed an auto-balanced method to transfer the representation capacity of a convolutional layer to a fraction of filters belong to the same layer. Other methods such as Taylor expansion [22], low-rank approximation [7, 23, 24], group-wise sparsity [25–28], and many more are employed to prune the filters from deep neural networks. Very recently, Lin [16] proposed a filter pruning method based on the rank of feature maps in each layer such that the filters contributing to low-rank feature maps can be pruned.

0.3 Important Preliminaries

Here, we discuss the background details like computing the Floating Point Operations (FLOPs) involved in a CNN and the notations used in this paper.

0.3.1 Calculating Floating Point Operations

In order to compare the various CNN models, we primarily use the accuracy on the validation set as a metric to compare which model is the most accurate. However, when there are constraints on computational resources, we use the number of Floating Point Operations (FLOPs) as a metric to compare which model is more efficient. We use the terms "Heavy" and "Light" to represent the models with a higher and lower number of FLOPs, respectively. For a given input feature map, the number of FLOPs involved in a convolutional layer L_i, i.e., $(FLOP_{conv}(L_i))$, is computed as follows,

$$FLOP_{conv}(L_i) = F * F * C_{in} * H_{out} * W_{out} * C_{out} \tag{1}$$

Here, $F * F$ is the spatial dimension of the filter, C_{in} is the number of input channels of the input feature map, H_{out} and W_{out} are the height and width of the output feature map, and C_{out} is the number of channels in the output feature map. Similarly, for a given input feature map, the number of FLOPs for a Fully Connected (FC) or dense layer D_i, i.e., $(FLOP_{fc}(D_i))$, is given as,

$$FLOP_{fc}(D_i) = C_{in} * C_{out} \tag{2}$$

So for a model with say K convolution layers and N fully-connected layers the total number of FLOPs is calculated as,

$$FLOP_{total} = \sum_{i=1}^{K} FLOP_{conv}(L_i) + \sum_{j=1}^{N} FLOP_{fc}(D_j) \tag{3}$$

0.4 About our BTP

Most of the filter pruning methods discussed above prune the unimportant filters. However, these methods may not remove the redundant filters from the network. We propose a novel filter pruning technique that utilizes the training history to find the filters to be pruned. Moreover, in contrast to other class of pruning methods, the proposed filter pruning method does not require the support of any additional software/hardware. In brief, the contributions of this research are summarized as follows,

- We propose a novel method for pruning filters from convolutional layers based on the training history of a deep neural network. It facilitates the identification of stable, redundant filters throughout the training that can have a negligible effect on performance after pruning.
- We introduce an optimization step (custom regularizer) to reduce the information loss incurred due to filter pruning. It is achieved by increasing the redundancy level of selected filters for pruning.
- To establish the significance of the proposed pruning method, experiments are conducted on benchmark CNNs like LeNet-5 [29], VGG-16 [30], ResNet-56, and ResNet-110 [14]. The validation of the proposed pruning method is performed over two benchmark classification datasets, including MNIST and CIFAR-10.

0.4.1 History Based Filter Pruning

0.4.2 Notations

Consider a convolution layer L_i of a CNN, which has n filters, i.e., $\{f^1, f^2, f^3, ..., f^n\}$. Any two filters f^i, f^j belong to the k^{th} layer of a CNN are denoted as f_k^i, f_k^j, respectively. For instance, if the filter f_k^i is of dimension 3×3 then it consists of 9 parameters, i.e., $\{w_{k,1}^i, w_{k,2}^i, w_{k,3}^i, ..., w_{k,9}^i\}$. Here '$k$' represents the index of convolution layer and 'i' denotes the filter index.

Initially, our method computes the ℓ_1 norm of each filter within the same convolution layer using the formula given in Eq. 4. For example the ℓ_1 norm of filter f_k^i is computed as follows,

$$\ell_1(f_k^i) = \|f_k^i\|_1 = \sum_{p=1}^{9} w_{k,p}^i \tag{4}$$

where p is the number of parameters in the filter f_k^i. Here, we assume the filter f_k^i is of dimension 3×3.

Next, we compute the absolute difference between the ℓ_1 norms of each filter pair at every epoch as shown in Eq. 5. The absolute difference between $\ell_1(f_k^i)$ and $\ell_1(f_k^j)$ is computed as follows,

$$d_{f_k^i,f_k^j}(t) = \ell_1(f_k^i) - \ell_1(f_k^j). \tag{5}$$

Here, 't' indicates the epoch number. Then, this difference is summed over all the epochs and denoted by $D_{f_k^i,f_k^j}$. The sum of differences of filter pairs over the training epochs is considered as the metric for filter pruning and given as,

$$D_{f_k^i,f_k^j} = \sum_{t=1}^{N} d_{f_k^i,f_k^j}(t) \tag{6}$$

where 'N' indicates the maximum number of epochs used for training the networks.

Our method computes the sum of differences of pairs $D_{f_k^i,f_k^j}$ value for n_{C_2} filter pairs (assuming a convolution layer has n filters). The total difference $D_{f_k^i,f_k^j}$ which is summed over all the epochs is used for filter selection. More concretely, the top $M\%$ of the filter pairs (from n_{C_2} pairs in the same layer) with the least $D_{f_k^i,f_k^j}$ value are formed as redundant filter pairs which represent roughly the same information. The overview of the proposed HBFP method is presented in Fig. 1. The proposed History Based Filter Pruning (HBFP) method involves two key steps, i) Filter selection and ii) Optimization which are presented in the next section. Our pruning method aims at making a deep neural network computationally efficient. This is achieved by pruning the redundant filters, i.e., removing of the filters whose removal do not cause much hindrance to the classification performance. We identify the redundant filters by observing the similar patterns in the weights (parameters) of filters during the network training, which we refer to as the network's training history. We start with a pre-trained CNN model. During the network training, we observe and form pairs of filters whose weights follow a similar trend over the training epochs. In each iteration, we pick some ($M\%$) of the top filter pairs with high similarity (based on the D value computed using Eq. 6, low D value means high similarity). Instead of pruning the filters at this stage, we increase the similarity between the filters that belong to

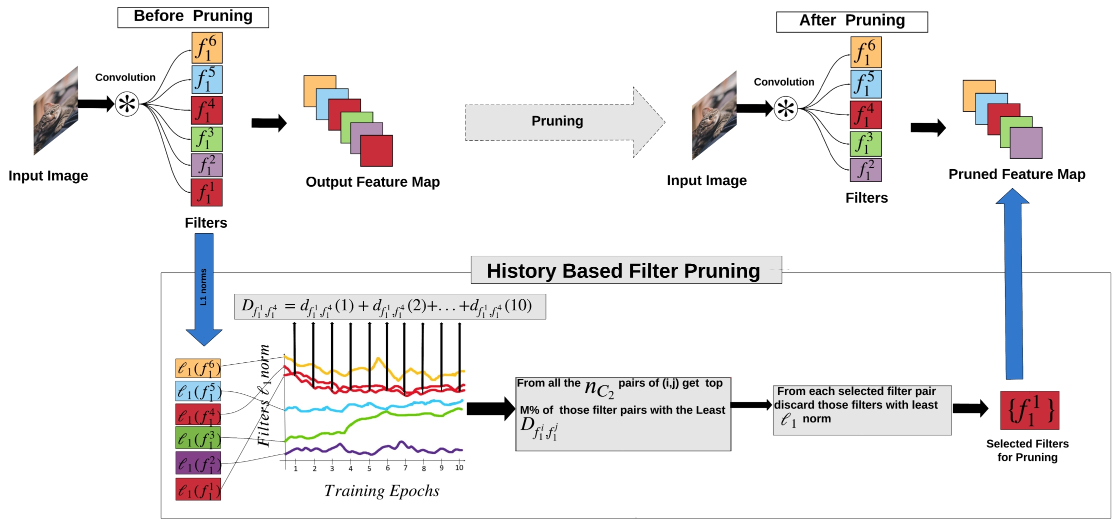
Before Pruning
Convolution
Input Image
Filters
Output Feature Map
Pruning
After Pruning
Convolution
Input Image
Filters
Pruned Feature Map
History Based Filter Pruning
L1 norms
$D_{f_1^1 f_1^4} = d_{f_1^1 f_1^4}(1) + d_{f_1^1 f_1^4}(2) + \ldots + d_{f_1^1 f_1^4}(10)$
Filters ℓ_1 norm
Training Epochs
From all the $^nC_2$ pairs of (i,j) get top M% of those filter pairs with the Least $D_{f_1^i f_1^j}$
From each selected filter pair discard those filters with least ℓ_1 norm
Selected Filters for Pruning

Figure 1: The overview of the proposed pruning method. At each iteration, we compute the sum of differences $D_{f_k^i, f_k^j}$ of all filter pairs. In filter selection stage, we choose $M\%$ of filter pairs for which $D_{f_k^i, f_k^j}$ is minimum. Here, filters f_1^1 and f_1^4 have the least $D_{f_k^i, f_k^j}$ (absolute point-to-point difference), so we select them as a pair and discard one from the pair with least magnitude. The feature maps corresponding to the pruned filters get dropped.

the selected filter pairs by introducing an optimization step. This optimization is achieved with a custom regularizer whose objective is to minimize the difference between the filter's norms (belong to a filter pair) at each epoch. After optimization, one filter from each pair is discarded (pruned). From a filter pair, we prune one filter based on the criteria employed in [9]. This whole process is repeated until the model's performance drops below a certain threshold. Our main contributions are made specifically in the filter selection and optimization steps.

0.4.3 Filter Selection

In the beginning, our method takes a heavy-weight CNN and selects the top $M\%$ of filter pairs from each convolution layer for which the difference D computed in Eq. 6 is minimum. More concretely, the filter pair having the least $D_{f_k^i, f_k^j}$ value is formed as the first redundant filter pair. Similarly, the next pair having the second least $D_{f_k^i, f_k^j}$ value is formed as another filter pair and so on. Likewise, in each iteration, $M\%$ of filter pairs from each convolution layer are selected as redundant which are further considered for optimization. Let us define two more terms that are used in the proposed pruning method. "Qualified-for-pruning (Q_i)" and "Already Pruned (P_i)". Here Q_i represents the set of filter pairs that are ready (selected) for pruning from a convolution layer L_i. Whereas, P_i indicates the filters that are pruned from the network (one filter from each pair of Q_i). Hence, if $M\%$ of filter pairs are chosen in Q_i, then $P = M\%$, i.e., from each convolution layer $M\%$ of filters are pruned in every iteration by the proposed method.

0.4.4 Optimization

Singh [31] reported that by introducing an optimization with a custom regularizer decreases the information loss incurred due to filter pruning. Motivated by this work, we add a new regularizer to the objective function to reduce the difference $d_{f_k^i, f_k^j}(t)$ between the filters belonging to Q_i at each epoch during network training, i.e., increasing the similarity between the filters belong to the same pair. Let $C(W)$ be the objective function (Cross-entropy loss function) of the deep convolutional neural network with W as the network parameters. To minimize the information loss and to maximize the regularization capability of the network, we employ a custom regularizer to the objective function, which is given as follows:

$$C_1 = \exp\left(\sum_{f_k^i, f_k^j \in Q_i} d_{f_k^i, f_k^j}(t) \right) \tag{7}$$

where t denotes the epoch number and $t \in 1, 2, 3, ..., N$ (assuming we train the model for N epochs). With this new regularizer, the final objective of the proposed HBFP method is given by,

$$W =_W \left(C(W) + \lambda * C_1 \right) \tag{8}$$

where λ is the regularizer term which is a hyperparameter. Optimizing the Eq. 8 decreases the difference $d_{f_k^i, f_k^j}$ between the filter pairs that belong to the set Q_i at every epoch without affecting the model's performance much.

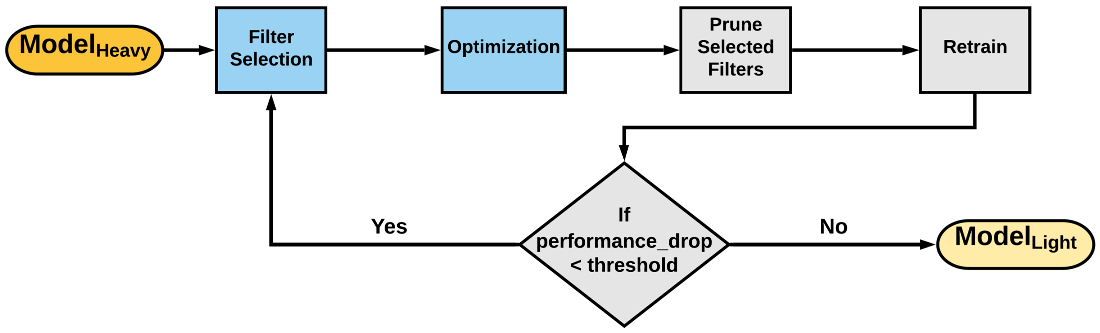

Figure 2: Initially, we start with a heavier model, after identifying the redundant filters, instead of pruning the filters naively at this stage, we optimize the model to minimize the complimentary information loss that will occur due to filter pruning. After optimizing the network with the custom regularizer, one filter from each filter pair is pruned. Later the network is re-trained to regain the classification performance. This process is repeated iteratively to obtain a light-weight CNN.

0.4.5 Pruning and Re-training

Using the process of minimizing the difference $d_{f_k^i, f_k^j}$ between the filters corresponding to a pair (which belongs to Q_i), we can increase the similarity between the filters that belong to the same filter pair. Thereby, one filter is pruned from each pair without affecting the model's performance much. The pruned model contains the reduced number of trainable parameters W,

$$W = W \backslash \{p_1, p_2, ..., p_k\} \tag{9}$$

where $p_1, p_2, ..p_k$ are the filters that are selected for pruning after optimization. Further, we re-train the network w.r.t. the reduced parameters W to regain the classification performance. As we prune the redundant filters from the network, the information loss is minimum. Therefore, re-training (fine-tuning) makes the network to recover from the loss incurred due to filter pruning.

The algorithm for the complete HBFP is demonstrated as a flow chart as shown in 2.

0.4.6 Results of LeNet-5 on MNIST

We utilize LeNet-5 architecture which has two convolution layers $Conv1$ and $Conv2$ with 20 and 50 filters, respectively, of spatial dimension 5×5. The first convolution layer $Conv1$ is followed by a max-pooling layer Max_pool1 with 2×2 filter which results in a feature map of dimension $12 \times 12 \times 20$. Similarly, the second convolution layer $Conv2$ is followed by another max-pooling layer Max_pool2 with 2×2 dimensional filter which results in a $4 \times 4 \times 50$ dimensional feature map. The feature map resulted by Max_pool2 layer is flattened into a 800×1 dimensional feature vector which is given as input to the first Fully Connected (FC) layer $FC1$. The $FC1$ and $FC2$ layers have 500 and 10 neurons, respectively. The LeNet architecture corresponds to 431080 trainable parameters and 4.4M FLOPs.

We conduct the pruning experiment on LeNet-5 over the MNIST dataset using the proposed HBFP method. Training the network from scratch results in 0.83% base error. The comparison among the benchmark pruning methods for LeNet-5 is shown in Table 1. Compared to the previous pruning methods, the proposed HBFP method achieves a higher reduction in the FLOPs, i.e., 97.98%, however, still results in a less error rate, i.e., 1.4%. Structured Sparsity Learning (SSL) method pruned 93.42% FLOPs with 1% error rate. From Table 1 (the rows corresponding to the proposed HBFP method), we can examine that the proposed method achieves better classification performance with a high percent of pruning compared to other methods. The previous work on Correlation Filter Pruning (CFP) by Singh [31] has reported a similar FLOPs reduction, i.e., 97.98%, however, their error rate is 1.77% which is quite high compared to our method. Singh [31] reported that employing a regularizer (in the form of an optimization) reduces the information loss occurs due to filter pruning. Motivated by this work, we also optimize the network to increase the similarity between the filters belong to a redundant filter pair. In this process, we reproduce the results of CFP [31] for which we obtain the top-1 error rate as 2.61% with the same percent of reduction in the FLOPs (row 10 of Table 1).

Table 1: The pruning results of LeNet-5 on MNIST dataset. The rows corresponding to HBFP-I, HBFP-II, HBFP-III, and HBFP-IV indicate the pruning results of the last four iterations of the proposed method. Here * indicates the reproduced results. The results are arranged in the increasing order of pruned FLOPs reduction.

Method	r1, r2	Top-1% Error	Pruned FLOPs
Baseline	20,50	0.83	4.4M (0.0%)
Sparse-VD [32]	-	0.75	2.0M (54.34%)
SBP [33]	-	0.86	0.41M (90.47%)
SSL-3 [26]	3,12	1.00	0.28M (93.42%)
HBFP-I (Ours)	4,5	0.98	0.19M (95.57%)
GAL [34]	2,15	1.01	0.1M (95.6%)
HBFP-II (Ours)	3,5	1.08	0.15M (96.41%)
Auto balanced	3,5	2.21	0.15M (96.41%)
CFP [31]	2,3	1.77	0.08 (97.98%)
CFP [31]*	2,3	2.61	0.08 (97.98%)
HBFP-III (Ours)	3,4	1.2	0.13M (96.84%)
HBFP-IV (Ours)	2,3	1.4	0.08M (97.98%)

0.4.7 Results of VGG-16 on Cifar-10 for HBFP

In 2014, Simonyan [30] proposed a VGG-16 CNN model that received much attention due to improved performance over ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The same architecture and settings are used in [30] with few modifications such as batch normalization layer [35] is added after every convolution layer. The VGG-16 consists of $14,982474$ trainable parameters and 313.73M FLOPs. Training the VGG-16 network from scratch enables the model to achieve 93.96% top-1 accuracy on the CIFAR-10 object recognition dataset. The comparison among the state-of-the-art filter pruning methods for VGG-16 on CIFAR-10 available in the literature is performed in Table 2. The proposed method prunes 83.42% of FLOPs from VGG-16 which results in 91.99% of top-1 accuracy. The Geometric Median method proposed in [36] reported 93.58% top-1 accuracy with a 35.9% reduction in the FLOPs. The recent works, such as HRank [16] and Correlation Filter Pruning (CFP) [31] are able to prune 76.5% and 81.93% FLOPs with the error rate of 8.77% and 7.02%, respectively, while the proposed

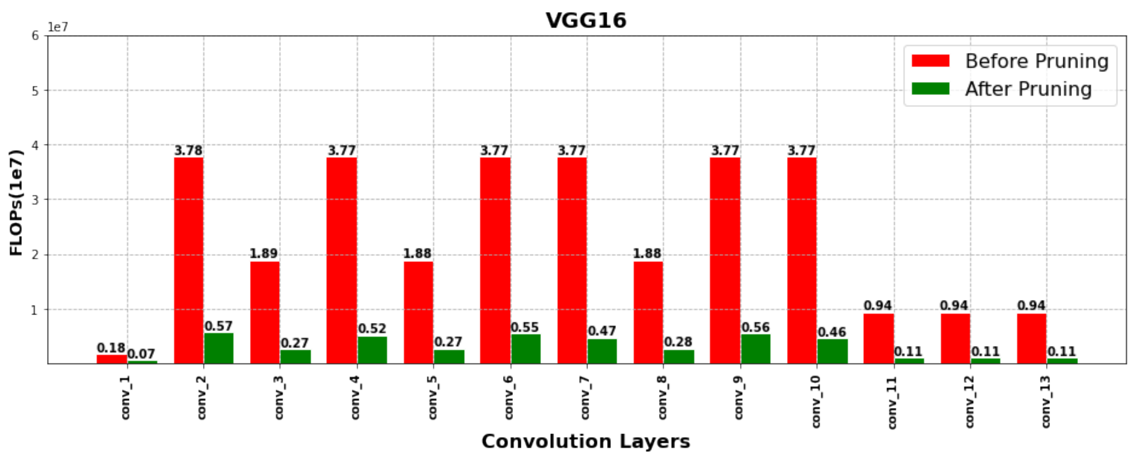

Figure 3: The total number of FLOPS involved in each convolution layer before and after pruning for VGG-16 trained on CIFAR-10 using the proposed history based filter pruning approach.

Table 2: The pruning results of VGG-16 on CIFAR-10 dataset. The reported results are taken from the respective research article, except * which is reproduced.

Model	Top-1%	Pruned FLOPs	Parameters Reduction
VGG-16 [30]	93.96	313.73M (0.0%)	14.98M (0.0%)
ℓ_1-norm [9]	93.40	206.0M (34.3%)	5.40M (64.0%)
GM [36]	93.58	201.1M (35.9%)	-
VFP [20]	93.18	190.0M (39.1%)	3.92M (73.3%)
SSS [37]	93.02	183.13M (41.6%)	3.93M (73.8%)
GAL [34]	90.73	171.89M (45.2%)	3.67M (82.2%)
HBFP-I (Ours)	93.04	90.23M (71.21%)	4.2M (71.8%)
HBFP-II (Ours)	92.54	75.05M (76.05%)	3.5M (76.56%)
HRank [16]	91.23	73.7M (76.5%)	1.78M (92.0%)
HBFP-III (Ours)	92.3	62.3M (80.09%)	2.9M (80.47%)
CFP [31]*	91.83	59.15M (81.14 %)	2.8M (81.1%)
CFP [31]	92.98	56.7M (81.93%)	-
HBFP-IV (Ours)	91.99	51.9M (83.42%)	2.4M (83.77%)

HBFP method is able to prune 83.42% FLOPs with an error rate of 8.01%. The detailed comparison of pruning results for VGG-16 on the CIFAR-10 dataset is demonstrated in Table 2. The FLOPs in each convolution layer before and after employing the proposed pruning method on VGG-16 over CIFAR-10 is illustrated in Fig. 3.

Table 3: The pruning results of ResNet-56/110 on CIFAR-10 dataset.

Model	Top-1%	Pruned FLOPs	Parameters Reduction
ResNet-56 [14]	93.26	125.49M (0.0%)	0.85M (0.0%)
VFP [20]	92.26	96.6M (20.3%)	0.67M (20.49%)
ℓ_1-norm [9]	93.06	90.9M (27.6%)	0.73M (14.1%)
NISP [38]	93.01	81.0M (35.5%)	0.49M (42.4%)
HBFP-I (Ours)	92.42	70.81M (43.68%)	0.48M (46.38%)
AMC [39]	91.9	62.74M (50 %)	-
CP [8]	91.8	62.74M (50 %)	-
GAL [34]	90.36	49.99M (60.2%)	0.29M (65.9%)
HBFP-II (Ours)	92.25	49.22M (60.85%)	0.33M (60.85%)
HRank [16]	90.72	32.52M (74.1%)	0.27M (68.1%)
HBFP-III (Ours)	91.79	31.54M (74.91%)	0.21M (74.9%)
CFP [31]*	91.37	31.54M (74.91%)	0.21M (74.9%)
CFP [31]	92.63	29.5M (76.59 %)	3.4M (77.14%)
HBFP-IV (Ours)	91.42	27.1M (78.43%)	0.19M (76.97%)
ResNet-110 [14]	93.5	252.89M (0.0%)	1.72M (0.0%)
VFP [20]	92.96	160.7M (36.44%)	1.01M (41.27%)
ℓ_1-norm [9]	93.3	155.0M (38.7%)	1.16M (32.6%)
GAL [34]	92.55	130.2M (48.5%)	0.95M (44.8%)
HBFP-I (Ours)	93.01	119.7M (52.69%)	0.81M (52.66%)
HBFP-II (Ours)	92.91	98.9M (60.89%)	0.67M (41.27%)
HBFP-III (Ours)	92.83	80.2M (68.31%)	0.54M (68.28%)
HRank [16]	92.65	79.3M (68.6%)	0.53M (68.7%)
HBFP-IV (Ours)	91.96	63.3M (74.95%)	0.43M (74.92%)

0.4.8 Results of ResNet-56 on Cifar-10 for HBFP

0.4.9 ResNet-56/110 on CIFAR-10

We also use the deeper and complex CNN models such as ResNet-56 and ResNet-110 [14] to conduct the pruning experiments over the CIFAR-10 dataset using the proposed HBFP method. ResNet-56/110 have three blocks of convolutional layers with 16, 32, and 64 filters. Training these residual models (i.e., ResNet-56 and ResNet-110) with the same parameters as in [14] produce 93.26% and 93.5% top-1 accuracies, respectively. The HBFP method prunes 2 filters from the first block which has 16 filters, 4 filters from the second block which has 32 filters, and 8 filters from the third block which has 64 filters

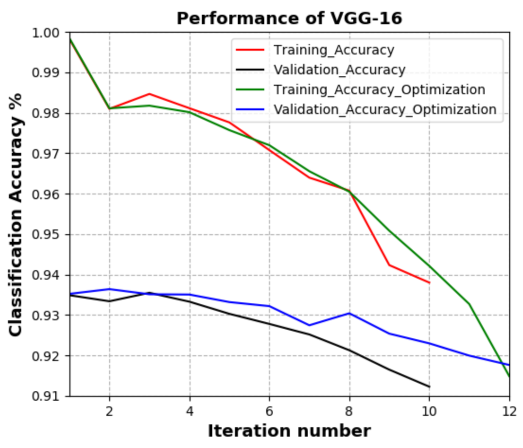

Figure 4: Illustrating the effect of pruning results obtained using the proposed pruning method with and without employing optimization. We can observe, introducing a custom regularizer to the objective increases the model's performance.

in each iteration of the proposed method. From Table 3, it is evident that the proposed pruning method produces state-of-the-art compression results for ResNet-56 and ResNet-110 on CIFAR-10 dataset.

ResNet-56: As depicted in Table 3, both AMC [39] and CP [8] methods have reduced 50.00% FLOPs while resulting in 8.1% and 8.2% error, respectively. The HRank [16] prunes 74.1% FLOPs with 9.28% error. From Table 3, it is clear that the proposed HBFP obtains top-1 accuracy 91.42% with high reduction of FLOPs (78.43%) compared to HRank [16]. However, the proposed method obtains the high FLOPs reduction with comparable performance 91.42% as compared to CFP [31]. Moreover, we achieve a better performance using the proposed HBFP as compared to the reproduced results using CFP [31].

ResNet-110: As per the results summarized in the lower part of Table 3, the filter's ℓ_1-norm based filter pruning method obtained 93.3% top-1 accuracy by pruning 38.7% of the FLOPs. The recent HRank [16] method achieved 92.65% top-1 performance with 68.6% FLOPs reduction. From Table 3, we can note that our method achieves a 74.95% FLOPs by removing 74.92% of the trainable parameters, with a minimum loss of 1.54% as compared to the baseline using ResNet-110 on CIFAR-10. Moreover, our HBFP-III performs better than HRank [16] in terms of the accuracy with comparable FLOPs reduction.

0.5 Conclusion

We propose a new filter pruning technique which uses the filters' information at every epoch during network training. The proposed History Based Filter Pruning (HBFP) method is able to prune a higher percent of convolution filters compared with state-of-the-art pruning methods. At the same time, the HBFP pruning can produce a less error rate. Eventually, it reduces the FLOPs available in LeNet-5 (97.98%), VGG-16 (83.42%), ResNet-56 (78.43%), and ResNet-110 (74.95%) models. The main finding of this paper is to prune the filters that exhibit similar behavior throughout the network training as the removal of one such filter from a filter pair does not affect the model's performance greatly. According to our study, employing the custom regularizer to the objective function also improves the classification results. We show the importance of the proposed pruning strategy through experiments, including LeNet-5 on MNIST dataset and VGG-16/ResNet-56/ResNet-110 on CIFAR-10 dataset. One possible direction of future research is pruning the filters further by considering the similarity among the filters from different layers.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 9516–9527.

[3] V. Kumar Verma, G. Arora, A. Mishra, and P. Rai, "Generalized zero-shot learning via synthesized examples," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4281–4289.

[4] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 7332–7342.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[6] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. S. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8817–8826.

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[8] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning filters for efficient convnets,” *arXiv preprint arXiv:1608.08710*, 2016.

[10] Y. LeCun, J. S. Denker, and S. A. Solla, “Optimal brain damage,” in *Advances in neural information processing systems*, 1990, pp. 598–605.

[11] B. Hassibi and D. G. Stork, “Second order derivatives for network pruning: Optimal brain surgeon,” in *Advances in neural information processing systems*, 1993, pp. 164–171.

[12] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, “Compressing neural networks with the hashing trick,” in *International conference on machine learning*, 2015, pp. 2285–2294.

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

[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[16] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, “Hrank: Filter pruning using high-rank feature map,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1529–1538.

[17] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, “Soft filter pruning for accelerating deep convolutional neural networks,” *arXiv preprint arXiv:1808.06866*, 2018.

[18] J.-H. Luo, J. Wu, and W. Lin, “Thinet: A filter level pruning method for deep neural network compression,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[19] R. Abbasi-Asl and B. Yu, “Structural compression of convolutional neural networks based on greedy filter pruning,” *arXiv preprint arXiv:1705.07356*, 2017.

[20] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, “Variational convolutional neural network pruning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2780–2789.

[21] X. Ding, G. Ding, J. Han, and S. Tang, “Auto-balanced filter pruning for efficient convolutional neural networks,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, “Pruning convolutional neural networks for resource efficient inference,” *arXiv preprint arXiv:1611.06440*, 2016.

[23] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up convolutional neural networks with low rank expansions,” *arXiv preprint arXiv:1405.3866*, 2014.

[24] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun, “Efficient and accurate approximations of nonlinear convolutional networks,” in *Proceedings of the IEEE Conference on Computer Vision and pattern Recognition*, 2015, pp. 1984–1992.

[25] V. Lebedev and V. Lempitsky, “Fast convnets using group-wise brain damage,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2554–2564.

[26] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, “Learning structured sparsity in deep neural networks,” in *Advances in neural information processing systems*, 2016, pp. 2074–2082.

[27] H. Zhou, J. M. Alvarez, and F. Porikli, “Less is more: Towards compact cnns,” in *European Conference on Computer Vision*. Springer, 2016, pp. 662–677.

[28] J. M. Alvarez and M. Salzmann, “Learning the number of neurons in deep networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 2270–2278.

[29] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[30] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[31] P. Singh, V. K. Verma, P. Rai, and V. Namboodiri, "Leveraging filter correlations for deep model compression," in *The IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 835–844.

[32] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2498–2507.

[33] K. Neklyudov, D. Molchanov, A. Ashukha, and D. P. Vetrov, "Structured bayesian pruning via log-normal multiplicative noise," in *Advances in Neural Information Processing Systems*, 2017, pp. 6775–6784.

[34] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2790–2799.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

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

[37] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 304–320.

[38] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203.

[39] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "Amc: Automl for model compression and acceleration on mobile devices," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 784–800.