Computer Vision and Image Understanding
Authorship Confirmation

As corresponding author I, _____, hereby confirm on behalf of all authors that:

1. This manuscript, or a large part of it, has not been published, was not, and is not being submitted to any other journal.
2. If presented at or submitted to or published at a conference(s), the conference(s) is (are) identified and substantial justification for re-publication is presented below. A copy of conference paper(s) is(are) uploaded with the manuscript.
3. If the manuscript appears as a preprint anywhere on the web, e.g. arXiv, etc., it is identified below. The preprint should include a statement that the paper is under consideration at Computer Vision and Image Understanding.
4. All text and graphics, except for those marked with sources, are original works of the authors, and all necessary permissions for publication were secured prior to submission of the manuscript.
5. All authors each made a significant contribution to the research reported and have read and approved the submitted manuscript.

Signature_____ Date_____

List any pre-prints: Not Applicable

Relevant Conference publication(s) (submitted, accepted, or published): Not Applicable

Justification for re-publication: Not Applicable

Research Highlights

- A novel adaptive pruning method for CNN to produce compact architecture
- Determining filter importance using class-specific feature map activation
- Filter importances are globally comparable across the convolutional layers
- Trimming insignificant filters in a non-uniform manner from each convolutional layer

Adaptive CNN filter pruning using global importance metric

Milton Mondal[a,*], Bishshoy Das[a,*], Sumantra Dutta Roy[a], Pushpendra Singh[b], Brejesh Lall[a], Shiv Dutt Joshi[a]

[a]*Department of Electrical Engineering, Indian Institute of Technology Delhi, Hauz Khas, New Delhi - 110016, India*
[b]*Department of Electronics and Communication Engineering, National Institute of Technology Hamirpur, Hamirpur (HP) - 177005, India*

Abstract

The depth and width of CNNs have increased over the years so as to learn a better representation of the input-output mapping of a dataset. However, a significant amount of redundancy exists among different convolutional kernels. Several methods on pruning suggest that trimming redundant parameters can produce compact structures with minor degradation in classification performance. Existing pruning methods reduce the number of filters at a uniform rate (i.e. pruning same percentage of filters from each layer) in every convolutional layer, which is suboptimal. In this paper, we conduct experiments to observe the sensitivity of each and every filter towards the final performance of the neural network. The essence of comparing filter importance on a global scale and subsequently pruning the neural network adaptively, is highlighted for the first time in this paper. Based on our observations, we propose a novel method named 'Global Filter Importance based Adaptive Pruning (GFI-AP)' that assigns importance scores to all filters based on how the network learns the input-output mapping of a dataset, which can then be compared across all the other convolutional filters. Our results show that non-uniform pruning achieves better compression as compared to uniform pruning. We demonstrate that GFI-AP significantly decreases the number of FLOPs (floating point operations) of VGG and ResNet networks in ImageNet and CIFAR datasets, without substantial drop in classification accuracy. GFI-AP reduces more number of FLOPs compared to existing pruning methods, for example, the ResNet50 variant of GFI-AP provides an additional 11% reduction in FLOPs over Taylor-FO-BN-72% while achieving higher accuracy.

Keywords: adaptive pruning, convolutional neural networks, filter pruning, model compression

1. Introduction

Convolutional neural networks (CNN) have become the primary tool for solving most of the computer vision applications which include multi-class classification, semantic segmentation, captioning of an image, and many more challenging tasks [1–3]. The depth and width of the deep neural networks have increased over the years so that the neural networks can achieve a better representations. With more complicated tasks a greater number of parameters are required to capture the patterns within data. The depth of the neural network is often increased to achieve higher accuracy but at the cost of significant increase in computations [4].

The number of parameters and floating point operations (FLOPs) increase as the number of hidden layers grow for a CNN [5]. The number of parameters and the network structure determines the model size. Simultaneously, the size of the feature map and the number of floating-point computations at the time of inference determine the run-time resource demand. While dealing with deep architectures, storing millions of parameters and performing billions of floating point operations during inference have

*Equal contribution

Email addresses: milton.mondal@ee.iitd.ac.in (Milton Mondal), bishshoy.das@ee.iitd.ac.in (Bishshoy Das), sumantra@ee.iitd.ac.in (Sumantra Dutta Roy), spushp@nith.ac.in (Pushpendra Singh), brejesh@ee.iitd.ac.in (Brejesh Lall), sdjoshi@ee.iitd.ac.in (Shiv Dutt Joshi)

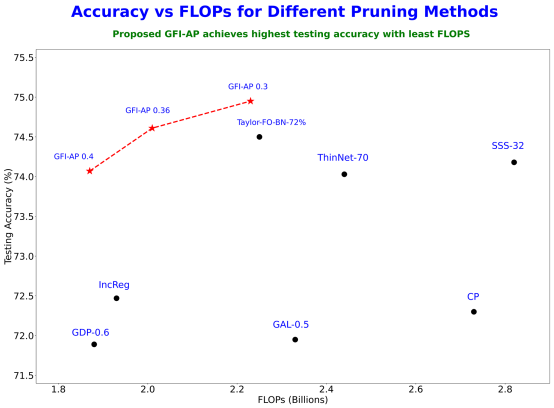

Figure 1: Performance of the pruned network after pruning filters from ResNet50, ImageNet classification task. GFI-AP provides the highest classification accuracy with the least number of FLOPs compared to competing pruning methods.

become common on workstations and server grade systems [3]. The requirement of memory for the storage of the network parameters in handheld devices, and computational resources for performing convolutions thereof cannot be fulfilled due to resource limitations in mobile devices or any other embedded systems when the depth and width of CNN are too high [6]. It is observed that a significant amount of redundancy exists among different convolution kernels and even in a single kernel for deep neural networks [7]. Pruning redundant parameters produce compact structures without significant drop in accuracy [8]. Network trimming while training [5] and network pruning after completion of training [9] are two significant ways to compress deep architectures. Introducing the sparsity objective in the original loss function during training forces neural network structures to become more compact [10]. Trimming is often performed after analyzing the redundancies of the weight matrices [9] or feature maps [11] of the trained network . Deletion of a layer [12], removal of a filter [2], and elimination of insignificant neurons [13] are three possible ways to prune a neural network. The memory and computation expenses reduce drastically due to severe pruning in case of layer pruning [12]. However, it is most favorable when the model depth is sufficiently large for the dataset. Discarding insignificant weights reduces the classification accuracy by the least amount, however, it creates unstructured sparsity in the network. So, to deal with such irregular structure produced by neuron pruning custom software and hardware support is required to perform classification [4]. In this paper, we focus particularly on filter pruning instead of layer or neuron pruning as these methods suffer from structural irregularity issues [3]. Filter pruning reduces computational constraint as fewer convolutions need to be performed in the pruned network.

In most of the methods used for filter pruning, firstly for all filters, an importance score (or a significance metric) is computed . Afterwards, redundant filters are trimmed if the importance score of a filter is less than some threshold value [3, 9, 11, 14]. Generally, in filter pruning, there can be two possibilities. If one wants to prune $p\%$ of the filters from a neural network, then one can either prune each layer uniformly by $p\%$, or one can non-uniformly prune each layer such that the total reduction in filters is $p\%$. Most of the existing pruning methods reduce the number of filters uniformly. Uniform pruning methods are mainly restricted in two ways: (1) these methods determine only the local importance of filters, i.e., the significance of filters within a layer. The local importance score cannot be used to prune filters adaptively throughout the entire network as the significance metric is not comparable across filters belonging to different layers. So, these methods either prune filters from each layer with an equal factor (uniform pruning) [15] or (2) determines the pruning fraction per layer based on hyperparameters [4, 7]. Few methods assign a significance metric to each filter in such a manner that filter relevance across the layers is comparable. However, most of these methods choose to prune filters uniformly or adaptively as a design choice [3, 13, 16] without understanding the necessity of global pruning. In this paper, we highlight the significance of global comparison of filter importance and adaptive pruning of CNN which has not been highlighted in a detailed manner so far in the literature to the best of our knowledge. The key contributions of the paper are as follows:

1. We design few experiments to study the effect of pruning in each layer. We observe that pruning the same percentage of filters from each layer degrades the performance of the classifier in an inconsistent way. These observations suggest that different convolution layers have different sensitivity towards pruning. So, pruning filters by the same factor for

all the layers will result in a sub-optimally pruned network.

2. We propose a novel method of pruning that assigns global importance score to all filters in an convolutional neural network based on the characteristics of how the network learns the input-output mapping of a dataset. More precisely, we develop a significance metric for filters which can be compared across all convolutional layers. We call our method, global filter importance based adaptive pruning (GFI-AP). The proposed method is computationally inexpensive as it does not use any optimization technique to find the filter importance. Instead, the filter importance scores are assigned based on the properties of the feature maps generated by every filter.
3. We highlight the essence of adaptive pruning by comparing the performance of the pruned network after adaptive and uniform pruning. We observe that GFI based non-uniform (or adaptive) trimming provides better classification accuracy than pruning all the layers by a constant factor (uniform pruning).

As we observe from Fig. 1, our method outperforms other competing methods at various FLOPs targets. At 2.23 billion FLOPs, our method reduces the total FLOPs count by 42.7%, at the expense of only 1% top-1 test accuracy.

2. Related works: A brief overview of filter pruning methods

Filter pruning methods in the literature follow specific patterns. There are mainly six types of filter pruning methods depending on the pruning criteria, which are as follows- filter pruning using (i) **filter norm**, (ii) **feature map properties**, (iii) **feature map reconstruction error**, (iv) **derivative of the loss function**, (v) **additional pruning agent, tool or module**, (vi) **modified loss to promote sparsity**.

(i) **Filter norm:** pruning based on filter norm is highlighted in [2–4, 9, 17, 18]. Filters with low norm produce low activations in the corresponding feature maps of CNN. This data free pruning strategy reduces 30% FLOPs in VGG16 while performing CIFAR10 classification [9]. In most of the cases, either the minimum filter norm is large or the deviation of all filter norms is small. To solve this problem, Y. He et al. [3] proposes to prune filters whose norms are close to the geometric median as all the other filters can learn this information over iterations.

(ii) **Feature map properties:** Like filter norms, there are methods based on feature norm [1, 19]. In [1], global average pooling is applied on each feature map to find activation entropy per feature using all input images. Filters corresponding to low entropy feature maps are pruned. In [19], rank selection based inter-kernel and interspatial redundancies are eliminated. Low feature map variance is another widely used criterion for filter pruning [6, 7, 11, 20]. Insignificant feature maps are removed by imposing L1 regularization on scaling factors of all batch normalization layers [6]. Trimming the neurons that have a larger average percentage of zeros (APoZ) activations than one standard deviation from the mean APoZ, do not degrade the classification performance significantly [11]. Both filters and filter channels are pruned whenever low variance in corresponding feature map activations is observed [7]. J. Ye et al. [20] proposes a stochastic method of training that forces the least important channels to be constant and then eliminates these constant channels by adjusting the biases of the resulting network.

(iii) **Feature map reconstruction error:** Another popular category of filter pruning method is minimizing feature map reconstruction error while trimming [13, 16, 21]. Y. He et al. [21] applies least absolute shrinkage and selection operator (LASSO) regression to detect most representative channels. These informative channels are retained, and linear least square-based reconstruction method is used to reconstruct the original output. Likewise, R. Yu et al. [13] backpropagates the importance from the fully connected layer to all the initial layers and trims the network while minimizing the weighted average reconstruction error (WARE). Luo et al. [16] proposes ThinNet, which uses next layer statistics to determine the redundant filters of the current layer. Least square error minimization while reconstructing output with the pruned network is performed in [16].

(iv) **Derivative of the loss:** Methods such as optimal network selection and optimal brain surgeon methods are based on Taylor expansion of loss gradient with feature map activation [22–25]. In [22], filters with flat gradient are pruned for acceleration in deep neural networks. Whereas in [24], Molchanov et al. follows the first-order approximation on Taylor expansion of squared change in

cost due to a neuron estimates the contribution of neurons in classification. Dong et al. [23] proposes a layer-wise optimal brain surgeon and determines the least important parameters after computing the second-order derivatives of the cost. Here light retraining helps to regain the original performance for the pruned network as the total error is bounded by a linear combination of layer-wise reconstruction error. Similarly, neural network synthesis tool uses gradient of the loss to produce compact network [26].

(v) **Additional pruning agent, tool or module:** Use of pruning agent, network, or module, is another popular way of measuring neuron or filter importance [2, 5, 27–30]. Huang et al. [2] uses an additional neural network as a pruning agent that makes binary decisions on retaining or pruning the filters based on the filter weights. These filter weights act as an input to the pruning agent. Likewise, a multi-task network whose last layer contains binary-valued outputs (one per layer) accelerates deep CNN by pruning redundant filters in [28]. Dong et al. [30] introduces a low-cost collaborative block along with a convolutional block in a few layers to produce discriminate features and thereby reduces the inference time. A min-max framework with the adaptive filter pruning (AFP) module and pruning rate controller (PRC) module is designed in [5] to prune maximally with minimal redundancy. Determining sensitive filters with the introduction of auxiliary loss function [27] and introducing the differentiable pruning criterion sampler [29] are two unique methods that incorporate additional blocks for trimming.

(vi) **Modified loss to promote sparsity:** One of the most widely used criterion for trimming the redundant parameters is sparsity promoting optimization [10, 31–34]. Flexible, dynamic, and progressive pruning methods are proposed in [2, 35]. Alternating updates with lagrange multiplier (AULM) scheme [10] alternates between promoting structure sparsity and minimizing the recognition loss of CNNs. Zhang et al. formulates weight pruning as a constrained nonconvex optimization problem and solves this problem using the Alternating Direction Method of Multipliers (ADMM) [36]. In [37] kernel sparsity and entropy (KSE) is used as an indicator to determine filter importance whereas in [33], sparse connections are trimmed with sparsity promoting stochastic gradient descent strategy. Advanced probabilistic methods for filter pruning include, runtime dynamic pruning with the use of reinforcement learning to explore sparse networks [8, 38, 39]. Other filter pruning methods which produce sparse networks include channel saliency-based variational CNN pruning [40], trimming CNN after training with discrimination-aware loss [41] and dynamic pruning of filters with low Discrete Cosine Transform (DCT) coefficients [42].

Existing filter pruning techniques have two primary flaws. (a) Determining filter significance scores based on feature map statistics: this is computed over all training samples without taking class labels into account [6, 7, 11]. (b) In other methods such as [3, 4, 7, 13, 16], a fixed compression ratio is used per layer without observing the significance of layer sensitivity. In this paper, we propose a novel pruning method that determines filter importance using class-specific feature map activation. Our method is an adaptive filter pruning technique based on norm of the corresponding feature map (category **(ii)**). Our observations suggest that non-uniform or adaptive pruning is essential to achieve high compression with minimal performance degradation. We prune filters in such a manner that any filter significance can be compared with all other filters globally after incorporating layer sensitivity. We explain the details of our method in subsequent sections.

3. Methodology

3.1. Analysis of layer sensitivity towards pruning

We design one experiment with three different cases of single layer, two layer and three layer pruning to investigate whether the pruned model can provide the same classification accuracy as the base model or not. In this experiment, we prune randomly instead of using any pruning method available in the literature as this is a simple experiment to observe three key points which are as follows:

(a) Does reducing parameters in the architecture reduces the performance of the classifier?

(b) Does pruning the same percentage of filters from different layers provide the same accuracy drop?

(c) In case the accuracy drops after pruning, does the CNN regain the performance with retraining?

As we are interested in observing the sensitivity of layers towards pruning, ideally we should prune a layer from the network completely and observe the performance of the classifier. However, the model depth reduces in that case and the model structure needs to be redefined. So we

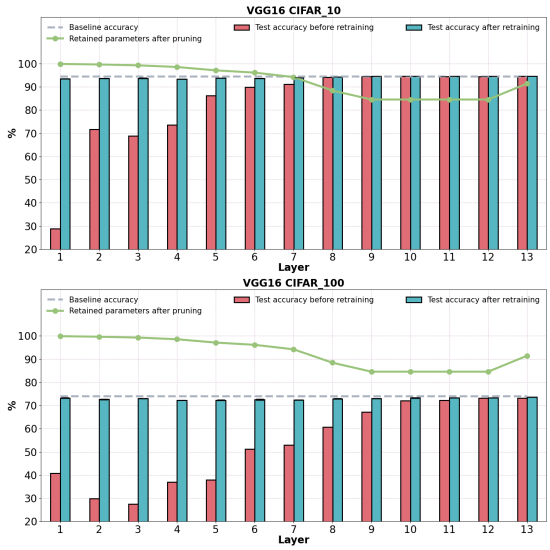

Figure 2: Performance of the pruned network after pruning 99% of the total number of filters from a single layer of VGG16, trained for **(top)** CIFAR10, **(bottom)** CIFAR100. The x-label (layer index) indicates the layer from which the filters are trimmed. The baseline test accuracy reduces from 94.48% to 28.87% for CIFAR10, and it drops from 73.97% to 40.69% for CIFAR100, while trimming the 1^{st} layer. We verified that even after retraining for forty epochs, the baseline accuracy cannot be recovered. On the contrary, we can recover almost the same classification accuracy as the unpruned network within twelve epochs of retraining if the same percentage of filters are pruned from the 13^{th} layer.

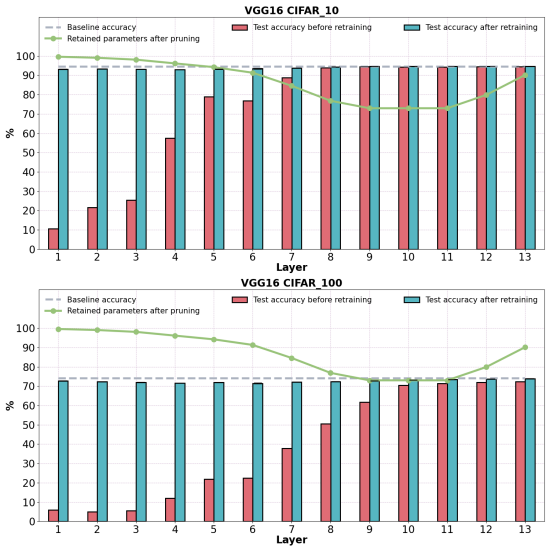

Figure 3: Performance of the pruned network after pruning 99% of filters from each of the two consecutive layers of VGG16, trained on **(top)** CIFAR10, **(bottom)** CIFAR100. The x-label (layer index) indicates that the starting index of two consecutive layers from which filters are deleted. Trimming filters from the first two layers reduces the test accuracy to 10.5% and after retraining it reaches 93.13% in the CIFAR10 classification task. The test accuracy remains at 94.36% after pruning filters from 10^{th} and 11^{th} layers. Interestingly, the accuracy reaches to 94.6% within twelve epochs of retraining which is better than the unpruned network's performance. The pruned network becomes less complex for CIFAR10 classification and thus provides lower generalization error than the unpruned network.

choose 99% filter pruning from a layer in our experiments. In the first case, we prune 99% filters randomly from a single layer. For instance, we prune 99% filters from the first layer only and observe the accuracy drop in the pruned model with respect to the original model. Thereafter, starting with the original model again, we prune 99% filters from the second layer only and observe the accuracy drop in the residual model and so on. Likewise, we repeat this procedure with all the thirteen convolutional layers of VGG16 [43], a widely used CNN architecture. We observe that the pruned model provides lower classification accuracy than the base model when filters are pruned from the initial layers. Whereas, there is a small or no degradation in the performance of the classifier if 99% filters are trimmed from the deep layers. In fact, pruning filters from deep layers sometimes improve classification accuracy by small amount while using VGG16 architecture for CIFAR10 and CIFAR100 classification.The observation details are shown in Fig. 2 (refer supplementary material for more details).

We find that a pruned network can match the performance of the original network, even if a severe accuracy drop occurs while pruning. This is possible by retraining the pruned CNN with few epochs like twelve epochs in our case. The observations from this experiment suggest that the pruned model with a smaller number of parameters can still maintain the same testing (or, evaluation) accuracy as the base model. Additionally, we observe that the degradation in the performance of the classifier after pruning is different for different layers. Generally in filter pruning, filters from more than one layer are pruned. So, we further perform two more experiments for multi-layer pruning in which 99% filters are pruned randomly from each layer. We trim filters from two or three consecutive

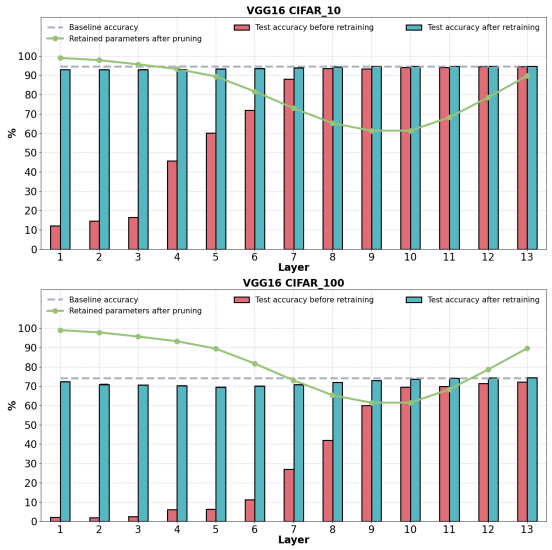

Figure 4: Performance of the pruned network after pruning 99% filters from each of three consecutive layers of VGG16, trained on **(top)** CIFAR10, **(bottom)** CIFAR100. The x-label (layer index) indicates the starting index of the three consecutive layers from which filters are deleted. In the CIFAR100 classification task, 98.97% and 89.54% parameters are retained while trimming from the first and the last three layers respectively as shown by the 'green' lines. When the deeper layers of the network are pruned, we get a smaller accuracy drop with retraining or even without retraining, than the case in which the initial layers of the network are pruned.

layers instead of a single layer as done in the previous experiment. These two experiments are designed to check the behaviour of performance degradation of the pruned model across the layers in multi-layer pruning. Here, once again we notice that the same percentage of pruning does not provide the same classification accuracy as observed in a single-layer pruning experiment. It depends on from which two or three successive layers filters are pruned. The classification accuracy is highly dependent on the extent of redundant parameters present in a layer of the trained model as shown in Fig. 3 and Fig. 4.

In Fig. 2-4, the reason for the reduction in the number of retained parameters near layers five, six, and seven is following: If n_l is the number of filters present in layer l, d_l is the depth of a filter belonging to layer l, and $k \times k$ is the spatial size of the filter, then trimming a single filter from the l^{th} layer removes $k \times k \times d_l$ parameters from the current layer and $n_{l+1} \times k \times k$ parameters of the next layer. So, a pruning a single filter leads to a reduction of $k^2(d_l+n_{l+1})$ parameters. Therefore, the number of parameters in pruned network is largely reduced near 5^{th}, 6^{th}, and 7^{th} layers as d and n both have high values in these layers compared to the initial layers. For the two and three layer pruning cases (Fig. 3, Fig. 4), the reduction in parameters is even greater in these layers relative to single layer pruning case (Fig. 2). Layer thirteen is attached to a densely connected layer and so the amount of parameter reduction happens differently than other convolutional layers.

3.2. Requirement of class-specific importance score

The test accuracy of VGG16 reduces more for the two-layer (Fig. 3) and three-layer pruning cases (Fig. 4) as compared to the single-layer case (Fig. 2). This is because a higher number of parameters are removed in the former cases. Additionally, we observe in all three cases, the test accuracy drops more if initial layers are pruned. On the other hand, if the deeper layers are pruned, the drop in accuracy is low. Trimming filters from initial layers reduces the number of low level features which are common to all classes, and therefore the accuracy takes a hit. This also means that pruning low level filters by large amount makes the network deviate farther from its optimal state. The gap in accuracy between the initial and final layers of the pruned network reduces with retraining. It is clearly visible from Fig. 2–4 **(top)** that trimming filters from deep layers does not require retraining in CIFAR10 classification. Whereas Fig. 2–4 **(bottom)** show that in CIFAR100 classification, retraining is required for all the convolutional layers as it is a more challenging dataset. All the above experiments indicate that different layers have different sensitivity towards pruning.

A characteristic of convolutional neural networks is that in the initial layers most of the feature maps are activated for almost all examples of all classes. However, with increasing network depth, feature activations become more and more class specific. So, features of different layers are responsible for classification in a different manner (refer supplementary material for more details). Existing methods [4, 7, 15] either uniformly prune $p\%$ of the filters in each layer or they determine the pruning fraction of each layer by hyperparameters. Feature norm or feature variance based filter importance scores that are computed over all training examples are suboptimal, because these methods do not determine the significance of a filter for a

particular class. Whereas class specific importance based pruning methods (GFI-AP) determine the responsibility of a feature map towards classification. Also the spatial size of the feature map reduces due to pooling operations with increasing layer depth. So, filter importance scores have to be normalized in such a manner that the scores across layers are comparable. In our method, we have included an appropriate normalization scheme as shown in Eq. 6 so that importance scores are comparable across different layers. We have also observed that different layers of the CNN have different sensitivity towards pruning (Fig. 2–4). We define a global importance metric with Eq. 7 after taking all of these effects into account.

3.3. Global filter importance based adaptive pruning (GFI-AP)

We propose 'GFI-AP', a method to find global filter importance (GFI). It is a score to measure the importance of filters based on the properties of the feature maps it produces. Every training example, given as an input, produces different feature map activations for every filter in the network. Existing filter pruning methods assign filter importance scores based on the statistics of the feature maps computed over all the training examples irrespective of their classes. However, feature maps that have very small values over several training examples can exhibit high value for certain examples belonging to specific classes. The corresponding filter responsible for producing that feature map can still play an essential role in image classification. Therefore, the filter responsible for detecting dominant patterns for a single class must have high importance and should not be pruned. Unlike conventional methods, here we first compute the absolute spatial mean of the feature map over training examples belonging to individual classes. Thereafter, we assign the maximum mean of the feature map among all the classes as global filter importance of that filter. As layer depth increases the spatial dimension of the feature map reduces due to max-pooling operations and the depth of the feature map increases due to an increase in the number of feature detectors. So, we normalize the mean of the feature map by its size to compute the importance score in our method.

The estimation problem of filter pruning can be described as follows. If,

$\mathbf{X} \leftarrow$ Input image

$\mathbf{Y} \leftarrow$ Ground truth labels

$f \leftarrow$ The neural network's input-output mapping

$\mathcal{L} \leftarrow$ Loss function

$\mathbf{K} \leftarrow$ Set of filters of the unpruned model

$\mathbf{K}_1 \leftarrow$ Trained filter weights of the unpruned model

$\widehat{\boldsymbol{K}} \leftarrow$ Set of filters of the pruned model

$\mathbf{K}_2 \leftarrow$ Trained filter weights of the pruned model after retraining

$n_{\mathbf{K}_1} \leftarrow$ Number of filters present in the base model

$n_{\mathbf{K}_2} \leftarrow$ Number of filters present in the pruned model

$p \leftarrow$ Filter pruning fraction

$I(\mathbf{K}) \leftarrow$ List of importance scores of all filters of the base model

$th \leftarrow$ Threshold to prune $p\%$ least important filters

$\odot \leftarrow$ Symbol representing element-wise multiplication

$\mathbb{1} \leftarrow$ Indicator function

The trained state of the unpruned model provides the state of $\mathbf{K}$ for which the training error is minimum, and the optimal parameters are obtained as

$$\mathbf{K}_1 = \underset{\mathbf{K}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{Y}, f_{\mathbf{K}}(\mathbf{X})). \tag{1}$$

In global filter pruning, we retain those filters whose importance score is greater than some threshold. The set of the filters present in the pruned model are represented as

$$\widehat{\boldsymbol{K}} = \mathbf{K}_1 \odot \mathbb{1}(I(\mathbf{K}) > th). \tag{2}$$

Our objective is to find a state for $\widehat{\boldsymbol{K}}$ for which $\mathcal{L}(\mathbf{Y}, f_{\widehat{\boldsymbol{K}}}(\mathbf{X}))$ is as close to $\mathcal{L}(\mathbf{Y}, f_{\mathbf{K}}(\mathbf{X}))$ as possible. To achieve this, we retrain the pruned model for a few epochs until convergence occurs. The final state of the pruned model that we obtain after retraining as

$$\mathbf{K}_2 = \underset{\widehat{\boldsymbol{K}}}{\operatorname{argmin}}\ \mathcal{L}(\mathbf{Y}, f_{\widehat{\boldsymbol{K}}}(\mathbf{X})) \tag{3}$$

where

$$n_{\mathbf{K}_2} = n_{\mathbf{K}_1}(1 - p). \tag{4}$$

In GFI based pruning, pruning of filters depend on the intensity of the corresponding feature map where the intensity is computed using maximally activated class examples. The filter importance can be compared across the layers (globally) with the proposed GFI based pruning method. Specifically, if we want to prune $p\%$ of filters

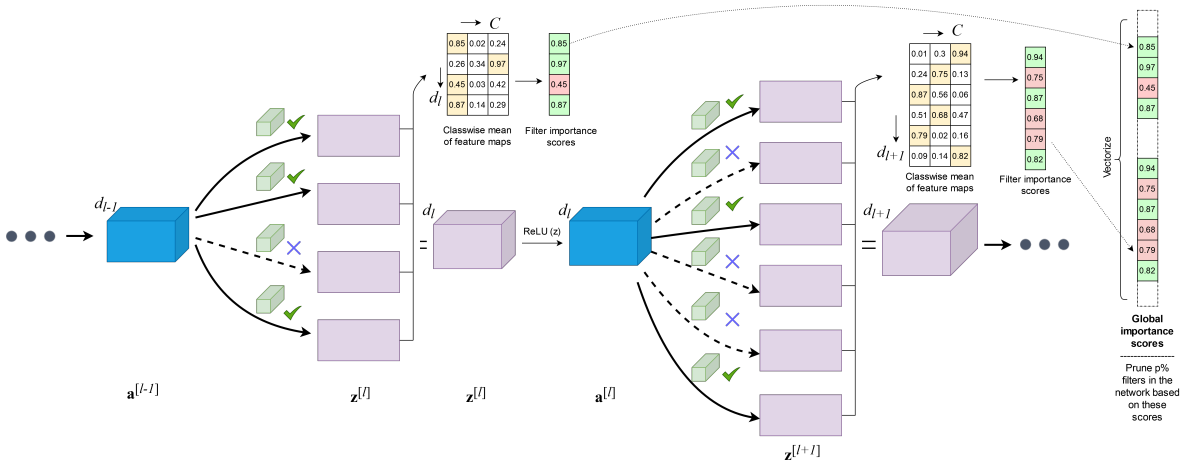

Figure 5: Illustration of GFI-AP's pruning pipeline. Here, the $(l-1)^{th}$ layer's output ($a^{[l-1]}$) having (d_{l-1}) channels is convolved with four filters to produce $Z^{[l]}$. The four filters' class specific normalized mean is computed in $d_l \times C$ matrix (top-left) and the filter importance is determined by the normalized mean of the maximally activated class. Filter scores for $(l+1)^{th}$ layer is computed in a similar fashion. Then to prune p% filters from the entire network, the importance scores of all filters are compared globally and the least p% are pruned. Importance score of retained and pruned filters are highlighted in green and red respectively in the Global Importance scores vector (top-right). Ticks and crosses indicate filters that are retained and pruned respectively. Blue blocks, purple blocks and purple rectangles indicate feature tensors. Green cubes indicate filters of a particular layer.

from the CNN then the proposed method performs adaptive pruning (AP) of filters from all the layers. This global comparison of filter importance values is possible as the method incorporates the characteristics of CNN feature maps to determine the importance score. Additionally, 'GFI-AP' works well for both moderate and high pruning cases.

3.4. Proposed algorithm

We first describe the notations used in Algorithm 1, which are as follows:

$C \leftarrow$ Total number of classes
$\sigma \leftarrow$ ReLU activation function
$\mathbf{a}^{[0]} \leftarrow$ Input image
$\mathbf{z}_j^{[l]} \leftarrow j^{th}$ input feature map in l^{th} layer
$\mathbf{a}_j^{[l]} \leftarrow j^{th}$ output feature map in l^{th} layer
$\mathbf{a}^{[L]} \leftarrow$ Predicted probabilities
$\mathbf{y} \leftarrow$ Ground truth labels
$\mathbf{k}_j^{[l]} \leftarrow j^{th}$ filter in l^{th} layer
$W_j^{[l]} \leftarrow$ Width of j^{th} feature map in l^{th} layer
$H_j^{[l]} \leftarrow$ Height of j^{th} feature map in l^{th} layer
$I(k_j^{[l]}) \leftarrow$ Importance score of j^{th} filter in l^{th} layer
$th \leftarrow$ Threshold to prune $p\%$ least important filters

Here,
$\mathbf{z}_j^{[l]} = \text{CONV}(\mathbf{a}^{[l-1]}, \mathbf{k}_j^{[l]})$
$\mathbf{a}_j^{[l]} = \sigma(\mathbf{z}_j^{[l]})$

GFI computes the importance of a filter by taking into account the characteristics of feature maps. We provide a detailed schematic of the pruning pipeline of GFI-AP in Fig. 5.

To prune the redundant filters of CNN following steps are employed:

1. First, the feature map is vectorized and the ℓ_1 norm of the vectorized feature map corresponding to j^{th} filter is computed for every example belonging to the c^{th} class. Here we add the ℓ_1 norms of j^{th} feature map corresponding to all examples belonging to the c^{th} class and normalize it by a factor of the number of class examples and feature map spatial size. This normalized mean value is denoted by $\mu_j^{c[l]}$ for j^{th} feature of the l^{th} layer from the c^{th} class as

$$\mu_j^{c[l]} = \frac{\sum_{i=1}^{n_c} \|\mathbf{z}_j^{c[l](i)}\|_1}{D(n_c, W_j^{[l]}, H_j^{[l]})}. \tag{5}$$

We calculate the normalization factor as

$$D(n_c, W_j^{[l]}, H_j^{[l]}) = n_c * W_j^{[l]} * H_j^{[l]} \quad (6)$$

where, n_c is the number of training examples belonging to the c^{th} class and $W_j^{[l]} * H_j^{[l]}$ is the spatial size (width and height respectively) of the j^{th} feature map of l^{th} layer.

2. We compute the importance score $I(k_j^{[l]})$ of j^{th} filter in l^{th} layer by taking the maximum of the normalized mean for that feature map as

$$I(k_j^{[l]}) = \max_{c \in \{1,2,\ldots,C\}} \mu_j^{c[l]}. \quad (7)$$

3. Once $I(k_j^{[l]})$ is computed, we compare the important scores of all filters present in the entire network globally. To prune $p\%$ of the filters from the CNN, we first sort the filters' importance scores in ascending order.
4. The first $p\%$ of the filters are to be pruned, and hence the threshold value is set to the importance score of the filter whose index is $p \times |K|/100$, where $|K|$ is the total number of filters in the network. Thereafter, filters having an importance score lower than the global threshold are pruned from the network.
5. We retrain the pruned model after pruning filters from each layer and finetune the pruned model after pruning filters from the entire network.

In residual networks, we do not trim the last convolutional layer within a residual block because there can be a size mismatch in the number of channels between the output tensor of the residual path and the tensor coming from the skip path. However, in case of feedforward networks such as VGG, no such restrictions exist, and so we trim all convolutional layers.

4. Results and discussion

In this section, we perform pruning experiments on different networks using different datasets and illustrate the performance of our proposed method (GFI-AP).

Algorithm 1: GFI-AP

Input: training data X_{train}, Y_{train}; pretrained CNN ($\mathbf{K}_1$)
Given: desired pruning percentage p
Output: Pruned CNN ($\mathbf{K}_2$)

```
for layer l ← 1 to L do
    for filter j ← 1 to J_l do
        for class c ← 1 to C do
            Compute normalization factor D with Eq.6
            Calculate normalized mean of each feature map with Eq. 5
        end
        Determine importance score of each filter with Eq. 7
    end
end
Find global threshold (th) such that p% filters are pruned
for layer l ← 1 to L do
    for filter j ← 1 to J_l do
        prune all filters for which I(k_j^[l]) <= th
    end
    Retrain model after pruning filters from 'l'th layer
end
Finetune the pruned model and obtain compact K_2
```

4.1. Adaptive Pruning vs Uniform Pruning

First, we observe the performance of the pruned CNN after applying GFI based adaptive pruning to the network. To do this, we compare the performance of the proposed GFI based adaptively pruned network (APN) with uniformly pruned network (UPN), with the same number of *parameters* in both the networks. The importance scores for all the filters in each network (APN and UPN) are evaluated using the proposed scheme, as described in section 3.4, for both networks.

Fig. 6**(a)**, 6**(b)**, and 6**(c)** show the impact of the pruning in three different cases viz. VGG16_CIFAR10, VGG16_CIFAR100, and ResNet32_CIFAR10 [44] respectively. These figures highlight the training accuracy and test accuracy for both UPN and APN. Our objective here is to compare the performance of the pruned model

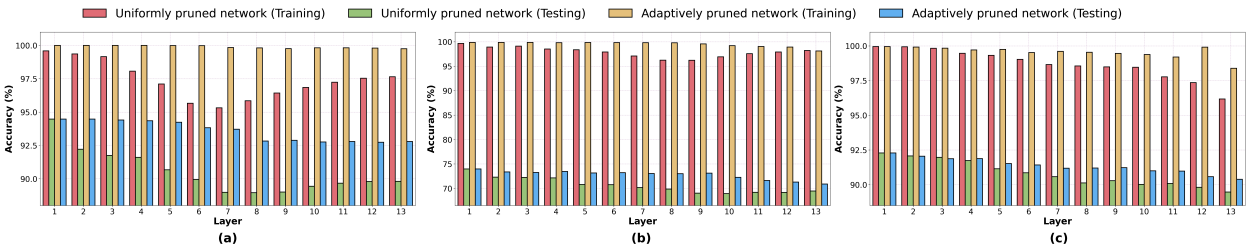

Figure 6: Performance comparison of uniformly pruned network (UPN) vs GFI-based adaptively pruned network (APN) with total pruned parameters **(a)** 78% in VGG16_CIFAR10, **(b)** 53% in VGG16_CIFAR100, **(c)** 41% in ResNet32_CIFAR10. The x-label (layer index) indicates the last layer up to which the filters are trimmed. **(a)** There is no substantial variation in the performance of UPN and APN when only the first layer is trimmed in VGG16_CIFAR10. However, APN provides (99.75 – 97.65) = 2.1% and (92.8 – 89.81) = 2.99% accuracy improvement over the corresponding UPN for the training and test set respectively after removing filters from all thirteen convolutional layers. **(b)** For VGG16_CIFAR100, APN provides (70.9 – 69.46) = 1.44% test accuracy improvement over the UPN after pruning filters from all convolutional layers. **(c)** Thirteen prunable convolutional layers are present in ResNet32. APN provides (96.96 – 94.08) = 2.88% and (89.66 – 87.94) = 1.72% training and test accuracy improvement respectively over the corresponding UPN method.

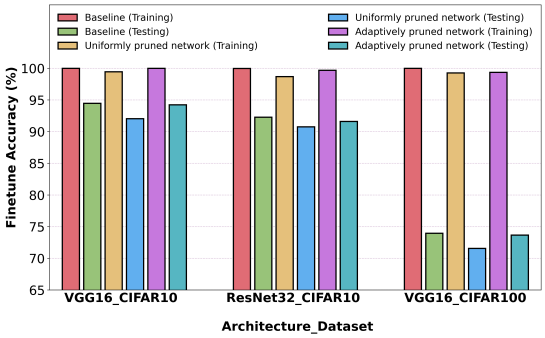

Figure 7: Accuracy comparisons among the base network, uniformly pruned network (UPN) and adaptively pruned network (APN) after fine tuning. APN provides (94.23 – 92.05) = 2.18% test accuracy improvement over UPN for the VGG16_CIFAR10 classification task. GFI-AP performs better than uniform pruning by (73.68 – 71.57) = 2.11% for the VGG16_CIFAR100 classification task. APN performs better in general than UPN across different architectures and datasets.

obtained by uniform pruning versus adaptive pruning. In both cases, we choose a pruning fraction in such a manner that both models have the same percentage of parameters after being pruned. In uniform pruning, one can prune filters randomly from each layer without taking into account the filters' importance. The pruned network would deviate much farther from the optimal state in that case. However, if the filter importance scores are taken into account to determine the filter index for pruning, then the deviation from the optimal state would be smaller. We use the proposed method to compute the filter importance scores and thereafter eliminate the least important filters by intra-layer comparison in case of uniform filter pruning. This ensures that the pruned network reaches its optimal state with few epochs of retraining. In case of UPN, we define local pruning fraction as the percentage of filters deleted from each convolutional layer. Whereas in APN, insignificant filters are determined after comparing the importance of all filters present in the entire network. In APN, we define global pruning fraction (GPF) as the percentage of filters that is trimmed from the full network. Given a value of the GPF, we calculate the global pruning threshold after comparing importance scores of all the filters. However, we prune only restricted pruning fraction (RPF) of the total number of filters present in a convolutional layer to restrict the maximum allowable number of filters pruning when a very large number of filters of a convolutional layer (> RPF) have a lower importance score than the global pruning threshold. We define RPF = GPF + (1 − GPF)/2 and mark the trimming type of a layer as restricted pruning in those cases. In experiments with VGG16, we observe that for a few layers the importance scores are very small. For these layers, we prune RPF amount filters so as to not eliminate the entire layer or have just a few filters in a layer. We observe that the tenth, eleventh and twelfth layer of VGG16_CIFAR10 hits the RPF threshold. This is also the case for the twelfth and the thirteenth convolutional layer in VGG16_CIFAR100, when the parameter pruning percentage is greater than 40% (such cases are termed as 'extreme pruning'). In case of ResNet architectures, all convolutional layers are pruned to GPF only.

In a case of extreme pruning, we reduce 78% parameters of VGG16 and then we examine the performance of the retained network while performing CIFAR10 classification. Here, we start pruning the filters from the first layer and then retrain the network by twelve epochs if the training accuracy drops by 0.4% of the unpruned network's training accuracy. We then proceed to prune followed by retraining the next layers up until the last convolutional layer. Trimming filters from each convolutional layer by a uniform pruning rate of 0.55 reduces 78% of the network's parameters. In APN, fixing the GPF as 0.6 and the RPF as 0.8 also contributes to 78% reduction in parameters. The results are shown in Fig. 7.

Table 1: FLOPs comparison across different pruning methods. Arch_Data denotes the architecture and the dataset used in the experiments. GFI-AP provides higher reduction of FLOPs while achieving better test accuracies than the existing methods.

Arch_Dataset	Methods		Accuracy	FLOPs
VGG16_CIFAR10	Feature-Cluster[45]	Baseline	93.8%	313M
		Pruned	93.67%	186M
	GFI-AP	Baseline	94.48%	315M
		Pruned	94.23%	**177M**
VGG16_CIFAR100	Slimming[6]	Baseline	73.26%	797M
		Pruned	73.48%	501M
	GFI-AP	Baseline	73.97%	315M
		Pruned	73.68%	**247M**

It is evident from our analysis that uniform filter pruning is not an appropriate design choice (Fig. 7). Instead, retaining significant filters based on their global importances maintain compact representation of features in APN. Fig. 7 illustrates that GFI-AP outperforms uniform pruning for every dataset and architecture we've tested.

4.2. Comparison with other methods

Next we compare the performance of the proposed GFI based adaptive pruning (GFI-AP) method with other pruning methods. Existing pruning methods focus on the reduction of inference time. Floating point operations (FLOPs) is a popular metric that is commonly used to determine the inference speed of the pruned network. Our results show that GFI-AP, provides better classification accuracy with fewer FLOPs than existing pruning methods as listed in Table 1. The retained FLOPs in each layer after pruning ResNet50 is shown in Fig. 8. Our base model for ResNet50 has the same configuration as described in [44] i.e., we use stride 2 for the first $(1 * 1)$ convolutional layer of each residual block where downsampling happens. We choose a GPF value of 0.6 for VGG16_CIFAR10, and we get a 43.8% reduction in FLOPs with 0.25% drop in accuracy. For VGG16_CIFAR100, we choose a GPF value of 0.4 and we achieve a net FLOPs reduction of 21.6% without substantial drop in accuracy (0.29%). We compare our results with existing methods for CIFAR10 classification using ResNet32 architecture in Table 2. GFI-AP with GPF value 0.41% reduces 42.5% FLOPs with 0.49% drop in accuracy for ResNet32_CIFAR10 configuration.

Table 2: Comparison of accuracy and FLOPs across different pruning methods for CIFAR10 classification with ResNet32. Global pruning fractions for GFI-AP are indicated by value of p. All experiments have been conducted three times to get the mean and standard deviation. Best results are highlighted in bold. Acc.↓ and FLOPs↓ indicate the drop in accuracy and the reduction in FLOPs respectively for the pruned model compared to unpruned model

Pruning Method	Baseline Acc. (%)	Pruned Acc. (%)	FLOPs (M)	Acc. ↓(%)	FLOPs ↓(%)
MIL [30]	92.33	90.74	47	1.59	31.2
SFP [4]	92.63 (± 0.7)	92.08 (± 0.08)	40.3	0.55	41.5
LFPC [29]	92.63 (± 0.7)	**92.12** (± 0.3)	32.7	0.51	52.6
GFI-AP (p=0.41)	92.54	92.09 (± 0.15)	40.2	0.45	42.5
GFI-AP (p=0.52)	92.54	91.59 (± 0.11)	32.2	0.95	53.96

Table 3: Comparison of accuracy and FLOPs across different pruning methods for ImageNet classification with ResNet50. Global pruning fractions for GFI-AP are indicated by the value of p. Best results are highlighted in bold. Acc.↓ and FLOPs↓ indicate the drop in accuracy and the reduction in FLOPs respectively for the pruned model compared to unpruned model

Methods	Baseline		Pruned			
	Acc. (%)	FLOPS (B)	Acc. (%)	FLOPS (B)	Acc. ↓(%)	FLOPs ↓(%)
GDP 0.6 [46]	75.13	3.86	71.89	1.88	3.24	51.3
IncReg [35]	75.60	3.86	72.47	1.93	3.13	50
ThinNet70 [16]	75.30	3.86	74.03	2.44	1.27	36.79
GAL-0.5 [47]	76.15	4.09	71.95	2.33	4.20	43.03
CP [21]	75.30	4.09	72.30	2.73	3.00	33.25
SSS-32 [31]	76.12	4.09	74.18	2.82	1.94	31.05
SFP [4]	76.15	4.09	74.61	2.38	1.54	41.8
Taylor-FO-BN-72% [24]	76.18	4.09	74.50	2.25	1.68	44.99
LFPC [29]	76.15	4.09	74.46	**1.6**	1.69	**60.8**
GFI-AP (p=0.3)	75.95	3.89	**74.95**	2.23	**1.00**	42.67
GFI-AP (p=0.36)	75.95	3.89	74.61	2.01	1.34	48.33
GFI-AP (p=0.4)	75.95	3.89	74.07	1.87	1.88	51.93

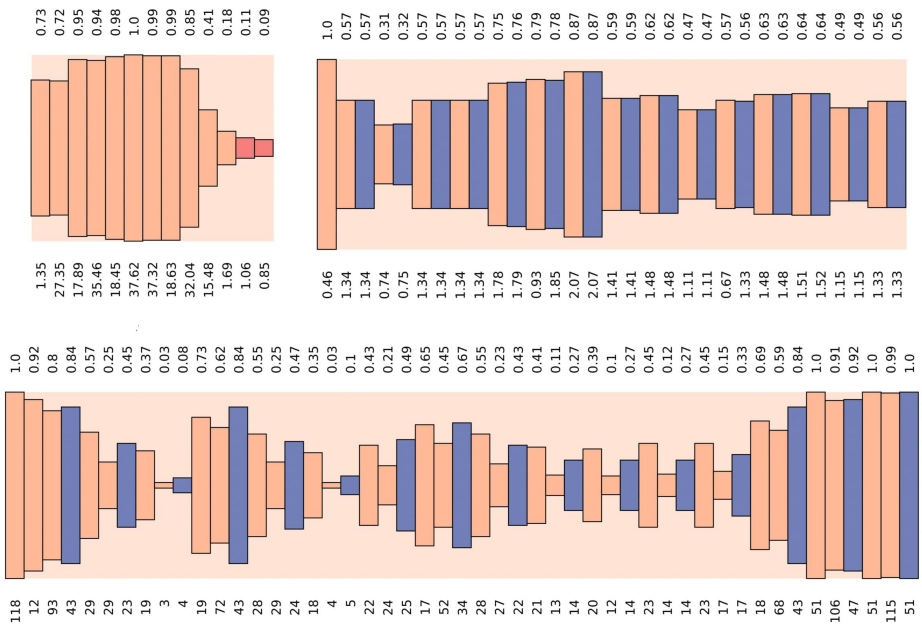

Figure 8: Retained FLOPs in each layer after applying GFI-AP. Total reduction of FLOPs is **(top left)** 22% in VGG16_CIFAR100, **(top right)** 40% in ResNet32_CIFAR10, and **(bottom)** 48% in ResNet50_ImageNet. Numeric values underneath the diagrams represent the number of FLOPs (in millions) present in each layer in the pruned network and the numeric value above the diagrams represent the fraction of FLOPs retained per layer. In residual networks, we prune all convolutional layers except the last convolutional layer of a residual block (highlighted in 'dark blue'). **(top left)** shows that GFI-AP prunes more filters from deeper layers than initial layers in VGG16. The last two convolutional layers are pruned using a restricted pruning fraction (RPF), highlighted in 'dark pink'. It matches our previous observation that deeper layers are less sensitive towards pruning for the VGG16 architecture. However, **(top right)** shows that different amounts of filters are trimmed from different convolutional layers irrespective of the layer position when filters are trimmed adaptively in ResNet32. **(bottom)** shows that the filter importance metric is comparable across all prunable convolutional layers.

GFI-AP is also applied to 'ResNet50_ImageNet' configuration. GFI-AP provides 43% (1.66B) reduction in FLOPs for an accuracy drop of just 1%. As shown in Table 3, GFI-AP outperforms other competing methods when minimal accuracy drop is the main objective. In cases where the target reduction in FLOPs is same for GFI-AP and other methods, we see a higher final test accuracy for GFI-AP, while in cases where the final accuracy target is held constant, we see a net higher reduction in FLOPs for GFI-AP. For example, GFI-AP provides 0.45% improvement in test accuracy with 20M fewer FLOPs than competing pruning methods such as Taylor-FO-BN-72% [24]. This empirically demonstrates that GFI-AP is more efficient as a pruning strategy than other methods.

4.3. More explorations

In this section, we first perform experiments where we observe the drop in the test accuracy while pruning filters by varying the global pruning fraction from 0.05 to 0.95. We find that there is no drop in test accuracy for CIFAR10 classification using ResNet32 architecture even at 30% reduction in FLOPs. The details are shown in Fig. 9. We also notice more than half FLOPs reduction with only 1% drop in test accuracy for ResNet32_CIFAR10 configuration. The classification accuracy drops more rapidly when the global pruning fraction (GPF) is set larger than 0.55.

In the next experiment, we observe the sensitivity of GFI-AP for different input image shapes. To perform this experiment, we resize the CIFAR images to 24×24 (downscaled) and 48×48 (upscaled) from the original size of 32×32. We have found out that the accuracy drop is

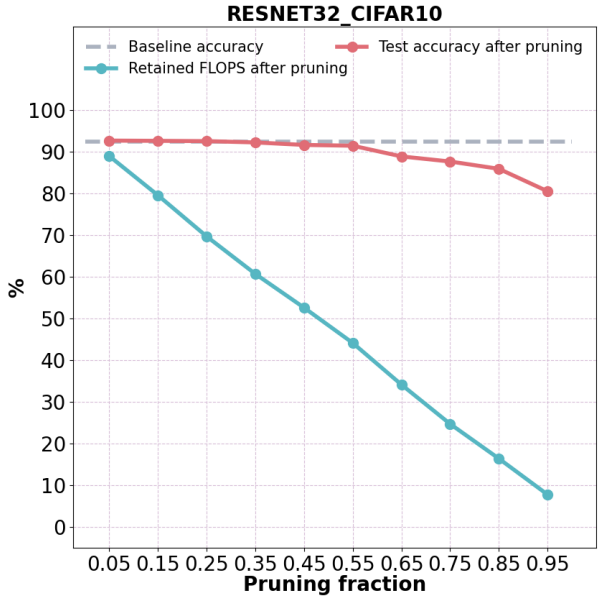

Figure 9: Performance of GFI-AP on CIFAR10 classification using ResNet32 for various values of p (pruning fraction). At p=0.25, GFI-AP reduces 30% FLOPs without any drop in accuracy. At p=0.55, the total FLOPs reduces by more than half, with just 1% drop in test accuracy. The red line shows the accuracy of the pruned model at different pruning fractions. The blue line indicates the reduction in FLOPs.

Table 4: Sensitivity of GFI-AP while modifying the spatial resolution of the input image for ResNet32_CIFAR10 configuration. Size of the input image becomes (24, 24, 3) and (48, 48, 3) via downscaling and upscaling operation respectively. The performance of GFI-AP is not affected due to change in the size of input image as we observe similar accuracy drop for same percentage of FLOPs reduction in all the cases

Input Image Size	Base Acc. (%)	Pruned Acc. (%)	FLOPs (M)	Acc ↓ (%)	FLOPs ↓ (%)
(24, 24, 3)	90.87	90.27 (± 0.17)	40.2	0.6	42.6
(32, 32, 3)	92.54	92.09 (± 0.15)	40.2	0.45	42.5
(48, 48, 3)	93.12	92.68 (± 0.11)	40.6	0.44	42

Table 5: Performance of GFI-AP while freezing some of the initial layers. Freeze initial layer 'n' indicates that we have not pruned any filters from the first 'n' convolutional layers, while maintaining close to the same target reduction in FLOPs. These experiments are conducted on a system with 1x Nvidia RTX 2080 Ti.

Freeze initial layers	Target FLOPs (M)	Pruned Acc. (%)	Acc. Drop(%)	Finetune Time (s)
None	40.2	92.23	0.31	3517
1	40.6	91.73	0.81	3458
2	40.6	91.21	1.33	3729
3	40.2	91.23	1.31	3245

less than 0.6% in all cases at a FLOPs reduction of more than 42% for the ResNet32_CIFAR10 configuration. In Table 4, we observe that GFI-AP is robust to varying input shapes, as we do not find any significant change in accuracy drop rates for any input sizes at similar FLOPs reduction percentages.

We further explore GFI-AP with the following experiments. We freeze some of the initial convolutional layers and then apply GFI-AP to determine whether the classification performance of the pruned model improves or not, compared to the case when we apply GFI-AP to the entire network. We do three experiments by freezing the first, second and third convolutional layers of ResNet32 and compare the results with GFI-AP in Table 5 for CIFAR10 dataset. The first column in Table 5 (Freeze initial layers), 'n' denotes the first 'n' layers are abstained from pruning. We observe that if we restrict GFI-AP and freeze initial layers, then the performance of pruned model is worse than applying GFI-AP to the whole network. It indicates that GFI-AP detects redundant filters from all layers in a better manner than manually freezing a layer or a few layers.

In the next experiment, we modify GFI-AP and propose another variant of it named GFI-AP-NC. GFI-AP-NC is the version of GFI-AP that is '*Not Class*' specific. GFI-AP-NC prunes filters from the network after computing importance score by taking the spatial mean over all feature maps of all training examples irrespective of their class. Hence, there is not class specific information in the GFI-AP-NC variant. We find that GFI-AP outperforms GFI-AP-NC for approximately the same target FLOPs as shown in Table 6. GFI-AP improves test accuracy by 0.31% and 0.3% over GFI-AP-NC for VGG16_CIFAR10 and ResNet32_CIFAR10 when the target FLOPs is less than 59%.

We conduct all our experiments on a system with 4x Nvidia RTX 2080 Ti. Code for our experiments are available at `https://github.com/miltonmondal/GFI-AP`.

Table 6: Comparison between the performance of GFI-AP and GFI-AP-NC. GFI-AP-NC is the version of GFI-AP that is Not Class specific. GFI-AP-NC determines the filter importance by taking the normalized mean using all training examples irrespective of their class.

Arch_Dataset	Method	Target FLOPs (%)	Target FLOPs (M)	Baseline acc (%)	Pruned acc (%)
VGG16_CIFAR10	GFI-AP-NC	58.72	178.7	94.48	93.92
	GFI-AP	56.2	177	94.48	**94.23**
VGG16_CIFAR100	GFI-AP-NC	78.41	247	73.98	73.41
	GFI-AP	78.41	247	73.98	**73.68**
ResNet32_CIFAR10	GFI-AP-NC	57.86	40.5	92.54	91.93
	GFI-AP	57.5	40.2	92.54	**92.23**

5. Conclusion

We designed several experiments to study the effects of pruning on classification performance of a convolutional neural network. We observe that pruning filters in the initial layers of a neural network drops the classification performance by the highest amount. A similar pruning fraction applied at deeper layers drops the classification accuracy by fewer points and often with no loss of accuracy, as compared to pruning initial layers. It indicates that the classification accuracy of a neural network is highly dependent on the amount of redundant parameters that are present in the network. The fraction of redundancy varies across layers. We compute filter importance scores in such a manner that the filter significance metric can be compared with all other filters across different layers. We propose a method to find the importance score of filters based on the class specific statistics of learned feature maps. Our observations suggest that layer-wise adaptive pruning is essential in achieving high compression ratio with minimal degradation in classification performance. This in comparison to uniform pruning yields superior performance and pruning efficiency. In particular, the proposed Global Filter Importance based Adaptive Pruning (GFI-AP) reduces the number of parameters in VGG16 by more than 50% (in both CIFAR10 and CIFAR100 classification tasks), with less than 0.3% reduction in accuracy. GFI-AP also eliminates 41% of the net parameters of ResNet32 and saves 40% floating point operations while trading off less than 0.45% accuracy in CIFAR10 classification task. The ResNet50 variant of GFI-AP provides 74.95% Top-1 test accuracy for ImageNet classification with a reduction of 1.66B (42.67%) FLOPs.

References

[1] J.-H. Luo, J. Wu, An entropy-based pruning method for cnn compression, arXiv preprint arXiv:1706.05791 (2017).

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

[3] Y. He, P. Liu, Z. Wang, Z. Hu, 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.

[4] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, arXiv preprint arXiv:1808.06866 (2018).

[5] P. Singh, V. K. Verma, P. Rai, V. P. Namboodiri, Acceleration of deep convolutional neural networks using adaptive filter pruning, IEEE Journal of Selected Topics in Signal Processing (2020).

[6] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2736–2744.

[7] A. Polyak, L. Wolf, Channel-level acceleration of deep face representations, IEEE Access (3) (2015) 2163–2175.

[8] H. Mostafa, X. Wang, Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization, arXiv preprint arXiv:1902.05967 (2019).

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

[10] S. Lin, R. Ji, Y. Li, C. Deng, X. Li, Toward compact convnets via structure-sparsity regularized filter pruning, IEEE transactions on neural networks and learning systems 31 (2) (2019) 574–588.

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

[12] S. Chen, Q. Zhao, Shallowing deep networks: Layer-wise pruning based on feature representations, IEEE transactions on pattern analysis and machine intelligence 41 (12) (2018) 3048–3056.

[13] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, 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.

[14] G. Roffo, S. Melzi, M. Cristani, Infinite feature selection, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4202–4210.

[15] A. Alqahtani, X. Xie, M. W. Jones, E. Essa, Pruning cnn filters via quantifying the importance of deep visual representations, Computer Vision and Image Understanding 208 (2021) 103220.

[16] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, W. Lin, Thinet: pruning cnn filters for a thinner net, IEEE transactions on pattern analysis and machine intelligence 41 (10) (2018) 2525–2538.

[17] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, Y. Yang, Asymptotic soft filter pruning for deep convolutional neural networks, IEEE transactions on cybernetics (2019).

[18] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, arXiv preprint arXiv:1810.05270 (2018).

[19] L. Zeng, X. Tian, Accelerating convolutional neural networks by removing interspatial and interkernel redundancies, IEEE transactions on cybernetics 50 (2) (2018) 452–464.

[20] J. Ye, X. Lu, Z. Lin, J. Z. Wang, Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers, arXiv preprint arXiv:1802.00124 (2018).

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

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

[23] X. Dong, S. Chen, S. Pan, Learning to prune deep neural networks via layer-wise optimal brain surgeon, Advances in Neural Information Processing Systems 30 (2017) 4857–4867.

[24] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, J. Kautz, Importance estimation for neural network pruning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 11264–11272.

[25] V. Lebedev, 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] X. Dai, H. Yin, N. K. Jha, Nest: A neural network synthesis tool based on a grow-and-prune paradigm, IEEE Transactions on Computers 68 (10) (2019) 1487–1497.

[27] P. Singh, V. S. R. Kadi, N. Verma, V. P. Namboodiri, Stability based filter pruning for accelerating deep cnns, in: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 1166–1174.

[28] V. K. Verma, P. Singh, V. P. Namboodiri, P. Rai, A" network pruning network" approach to deep model compression, in: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2020, pp. 2998–3007.

[29] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, Y. Yang, Learning filter pruning criteria for deep convolutional neural networks acceleration, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 2009–2018.

[30] X. Dong, J. Huang, Y. Yang, S. Yan, More is less: A more complicated network with less inference complexity, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5840–5848.

[31] Z. Huang, 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.

[32] M. A. Carreira-Perpinán, Y. Idelbayev, "learning-compression" algorithms for neural net pruning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8532–8541.

[33] C.-H. Lee, I. Fedorov, B. D. Rao, H. Garudadri, Ssgd: Sparsity-promoting stochastic gradient descent algorithm for unbiased dnn pruning, in: ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2020, pp. 5410–5414.

[34] J. Guo, W. Zhang, W. Ouyang, D. Xu, Model compression using progressive channel pruning, IEEE Transactions on Circuits and Systems for Video Technology (2020).

[35] H. Wang, X. Hu, Q. Zhang, Y. Wang, L. Yu, H. Hu, Structured pruning for efficient convolutional neural networks via incremental regularization, IEEE Journal of Selected Topics in Signal Processing (2019).

[36] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, Y. Wang, A systematic dnn weight pruning framework using alternating direction method of multipliers, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 184–199.

[37] Y. Li, S. Lin, B. Zhang, J. Liu, D. Doermann, Y. Wu, F. Huang, R. Ji, Exploiting kernel sparsity and entropy for interpretable cnn compression, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2800–2809.

[38] J. Lin, Y. Rao, J. Lu, J. Zhou, Runtime neural pruning, in: Advances in neural information processing systems, 2017, pp. 2181–2191.

[39] H. Liu, F. Du, X. Tang, H. Liu, Z. Yu, Network architecture reasoning via deep deterministic policy gradient, in: 2020 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2020, pp. 1–6.

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

[41] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, J. Zhu, Discrimination-aware channel pruning for deep neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 875–886.

[42] Z. Liu, J. Xu, X. Peng, R. Xiong, Frequency-domain dynamic pruning for convolutional neural networks, in: Advances in Neural Information Processing Systems, 2018, pp. 1043–1053.

[43] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

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

[45] B. O. Ayinde, T. Inanc, J. M. Zurada, Redundant feature pruning for accelerated inference in deep neural networks, Neural Networks 118 (2019) 148–158.

[46] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, B. Zhang, Accelerating convolutional networks via global & dynamic filter pruning., in: IJCAI, 2018, pp. 2425–2432.

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