Accurate and Efficient Channel pruning via Orthogonal Matching Pursuit

Kiran Purohit
Department of Computer Science
Indian Institute of Technology, Kharagpur, India
kiran.purohit@kgpian.iitkgp.ac.in

Anurag Parvathgari
Department of Computer Science
Indian Institute of Technology, Kharagpur, India
anuragparvathgari7@gmail.com

Soumi Das
Department of Computer Science
Indian Institute of Technology, Kharagpur, India
soumi_das@iitkgp.ac.in

Sourangshu Bhattacharya
Department of Computer Science
Indian Institute of Technology, Kharagpur, India
sourangshu@cse.iitkgp.ac.in

ABSTRACT

The deeper and wider architectures of recent convolutional neural networks (CNN) are responsible for superior performance in computer vision tasks. However, they also come with an enormous model size and heavy computational cost. Filter pruning (**FP**) is one of the methods applied to CNNs for compression and acceleration. Various techniques have been recently proposed for filter pruning. We address the limitation of the existing state-of-the-art method and motivate our setup. We develop a novel method for filter selection using sparse approximation of filter weights. We propose an orthogonal matching pursuit (OMP) based algorithm for filter pruning (called **FP-OMP**). We also propose **FP-OMP Search**, which address the problem of removal of uniform number of filters from all the layers of a network. FP-OMP Search performs a search over all the layers with a given batch size of filter removal. We evaluate both FP-OMP and FP-OMP Search on benchmark datasets using standard ResNet architectures. Experimental results indicate that FP-OMP Search consistently outperforms the baseline method (LRF) by nearly $0.5 - 3\%$. We demonstrate both empirically and visually, that FP-OMP Search prunes different number of filters from different layers. Further, timing profile experiments show that FP-OMP improves over the running time of LRF.

KEYWORDS

Filter Pruning, OMP, Multiple channels, Weight compensation

1 INTRODUCTION

Computer vision has been revolutionized with the advent of convolutional neural networks (CNNs) such as AlexNet [21], VGG [29], and ResNet [10]. In order to obtain higher performance, additional heavier networks [17, 31] were later developed. The massive computing that these networks require has been made possible by technological advancements. However, the computing resources are constrained in some contexts, such as in mobile devices. Also, in certain contexts higher efficiency at similar performance levels can lead to newer applications being more practical given a resource budget. Hence, methods that can prune unnecessary weights are in demand. Moreover, it has been argued that pruning individual weights [9] leads to irregular sparsity [27]. Hence, it is not effective for reduction of computation in case of convolutional neural networks [19]. This has lead to development of many channel pruning methods, e.g. geometric medians [13], learning filter pruning criteria [11], transformable architecture search [6], etc.

Recently, linearly replacable filters (LRF) [19] has outperformed the existing methods by removing filters which can be expressed as linear combinations of other filters in the same layer. However, there are two shortcomings with the method. Firstly, the filters are removed one at a time and the weights are compensated through fine tuning after each removal, causing this method to be time consuming. Secondly, same fraction of filters are removed from all the layers. However, certain layers may have a higher fraction of redundant filters compared to other layers. In this paper, we seek to alleviate these limitations.

In order to alleviate the first shortcoming, we propose to remove multiple filters at a time. We develop the formulation based on sparse approximation of filter weights and propose an orthogonal matching pursuit (OMP) [30] based algorithm (called **FP-OMP**) for the solving the problem. FP-OMP also performs the initial weight compensation using 1x1 convolutions for the existing filters. This decouples the pruning of multiple layers while incurring minimal overhead in an optimized final network. Note that the original weight compensation developed in [19] only compensates weights for a single removed filter.

We also propose **FP-OMP Search**, which aims to alleviate the second limitation, by performing a search over all the layers with a given batch size of filter removal. We evaluate both FP-OMP and FP-OMP Search on benchmark datasets using standard ResNet architectures. Experimental results indicate that FP-OMP Search consistently outperforms the baseline method (LRF) by $\sim 0.5 - 3\%$. Also, we show that FP-OMP Search prunes very different number of filters from different layers. Further, anecdotal evidence suggests that the layers from which lower number of filters are pruned encode more diverse feature maps compared to layers with higher number of pruned filters. We also demonstrate that FP-OMP improves over the running time of LRF, albeit FP-OMP Search is more computationally expensive.

2 RELATED WORK

Network Pruning: A neural network that has already been trained can have its superfluous weights removed while still performing well. Early pruning [9] started with the elimination of small-norm weights or nodes from the Deep Neural Networks(DNN); however,

as CNN developed, the practical effectiveness of removing a single weight or node decreased. Channel pruning or filter pruning thus became widely used in this industry. At the early stage of channel pruning research, a number of heuristic techniques that make use of the average proportion of zero activations [16] and Lasso regression [14, 27] were developed. Since then, a number of methods have been suggested, including regularising the scaling factor of BatchNorm (BN) [18] to prune the appropriate channel [24], to prune filters that have a small effect on the output accuracy [22], assessing the relevance score by backprop [34], and identifying essential channels by adding a new loss to the intermediate layer [35]. More novel techniques have recently surfaced [13, 23, 33], such as investigating the value of pruning [26] or using meta-learning [25].

A straightforward method of filter selection is to remove the filters with lower value of norm as they are expected to contribute relatively less to the output feature maps. Filter Pruning via Geometric Median (FPGM) [13] notes the two limitations with this criterion: (1) The deviation of filter norms should be significant else, the lower norm filters that are removed would be almost as important as the remaining filters. (2) The norms of the pruned filters should be close to zero. However these prerequisites are not always true. FPGM addresses these limitations and proposes a better criterion. According to the geometric median's [8] property, the remaining filters can be used to represent filters that are close to it in a layer of filters. Thus they can be pruned without affecting the network and since this isn't norm based, it doesn't suffer from the above limitations.

The earlier techniques employ the same pruning criteria for various network layers, according to Learning Filter Pruning Criteria LFPC [11]. Setting a single pruning criterion for each layer may not be appropriate because various layers have distinct filter distributions. A Differentiable Criteria Sampler (DCS), the main element of LFPC, learns the appropriate criteria from a set of preset criteria to utilise for various layers. LFPC simultaneously takes into account all layers and pruning criteria.

Transformable Architecture Search (TAS)[6] is a differentiable searching algorithm that learns the optimal architecture size of the network. Once each layer's ideal number of filters is established, a network with this architecture is trained through knowledge distillation [15] using the unpruned original network in the final step. This approach is in sharp contrast with other papers as it allows the number of channels per layers to be explicitly optimized to get the smaller model. One problem with this approach is that the first step of searching the optimal network architecture is time expensive. This is due to the increase in the candidate search space.

Linearly Replaceable Filter (LRF) [19] frames each filter as a synthesis of other filters in a specific layer. If the approximation for a certain filter as a linear combination of remaining filters is accurate enough, we can consider this filter to be replaceable such that the pruned network is unaffected. Our work is essentially built on this idea of filter representation.

A large body of dataset selection works exist such as those of data valuation [5, 28, 32] that constructs a value function based on the validation set, using which a set of high value data samples get selected; facility location based approaches [4, 7] that select data in a streaming setting using convex approximation; uncertainty based method [3] that believes including samples with low confidence in the subsets have higher impact on optimisation; submodular optimisation based coreset approaches [1, 2] that finds a set of representative samples fulfilling a defined objective function; gradient based methods [20] that use Orthogonal Matching Pursuit (OMP) algorithms to match the gradients on full set with that of the subset. Owing to OMP being a sparse approximation algorithm, that is also used in the area of data selection for speed enhancement, we resort to adapting this method in the area of filter selection.

3 FILTER PRUNING FOR CNNS

In this section, we describe the setup for filter pruning/selection for the architecture of Convolutional Neural Networks. We lay the background of the problem in Section 3.1 where we describe the limitations of the existing state-of-the-art methodology. This serves as a motivation for our proposed pruning framework in Section 3.2. Inspired from Joo et al.[19], we derive the weight compensation module for our setup in Section 3.3. Additionally, we finally motivate and describe a variant of our proposed framework in Section 3.4.

3.1 Background and Problem Setup

With the rapid progress in the field of deep learning, the size of the models are getting larger, leading to several computational bottlenecks like decreasing speed, increasing energy requirements and carbon footprints. Pruning models is one of the ways to recover from these bottlenecks. We focus our attention on filter channel pruning for the CNN architectures. This serves as an effective way of reducing computation along with the size of the model.

Inspired from Joo et al.[19], we setup the foundation of the pruning framework. Given a trained model with any $K \times K$ convolutional layer c having its filter weights as $\mathcal{F}_c \in \mathcal{R}^{KxKxmxn}$, where m and n are the number of input and output channels, any filter $f_{:,j} \in \mathcal{R}^{KxKxm}\ \forall j = \{1, 2, .., n\}$ can be represented as a linear combination of other $n-1$ filters using Equation 1.

$$f_{:,j} = \sum_{l \neq j} \lambda_{j,l} f_{:,l} + \epsilon_j \tag{1}$$

Here ϵ is the approximation error and $\lambda_{j,l}$ is the weight coefficient of the respective filters $l \in \{1, 2, .., n\} \setminus \{j\}$ in representing filter j. The objective is to find a set of filters that best approximate Equation 1. The intuition behind this is that the filter with the lowest approximation error can be well represented by the other filters and hence can be pruned.

Limitations of Joo et al.[19]: The proposed technique *Linearly Replaceable Filters(LRF)* by Joo et al.[19] uses $1x1$ convolution at the top of the convolutional layers to be pruned, preventing any effect of pruning on the computation of the network model. The algorithm proceeds by solving for $\lambda_{j,l}$ for each $j \in \{1, 2, .., n\}$ using Equation 2, followed by computing ϵ_j from Equation 1.

$$\min ||f_{:,j} - \sum_{l \neq j} \lambda_{j,l} f_{:,l}||^2 \tag{2}$$

The filter with minimum approximation error ($argmin_j ||\epsilon_j||$) gets pruned at the end of each turn. However, for pruning filters by a fraction of β using their algorithm, one needs to prune the network β of N_c times where N_c is the total number of filters in layer c. It

is followed by fine-tuning the network each time after removing a filter. Hence, this approach is both slow and sub-optimal. In order to speed up the pruning method, we develop an algorithm that prunes the filters of a layer for a given fraction together, followed by fine tuning, thus reducing the pruning time by a reasonable margin. Next, we describe our proposed algorithm.

3.2 Identifying Multiple Channels for Pruning

In this section, we describe our algorithm for pruning multiple channels together. We develop an Orthogonal Matching Pursuit (OMP) based algorithm for filter pruning, *Filter Pruning-OMP (FP-OMP)* that addresses the limitations of the existing state-of-the-art method [19]. Starting from Equation 2, we arrive at our formulation for sparse approximation as:

$$S^*, \lambda^* = \min_{|S| \leq (N_c - \beta * N_c), \lambda} \sum_{j \in \{1,2,..,N_c\}} ||f_{:,j} - \sum_{l \in S} \lambda_{j,l} f_{:,l}||^2 \ , \forall j \in \{1, 2, ..., N_c\} \tag{3}$$

where S is the set of the selected/retained filters of layer l, N_c is the total number of filter in layer c, and β is the pruning fraction of filters in layer c.

Therefore, we can rewrite Equation 1 as

$$f_{:,j} = \sum_{l \in S} \lambda_{j,l} f_{:,l} + \epsilon_j \quad , \forall j \in \{1, 2, .., N_c\} \tag{4}$$

Algorithm 1 describes our approach of selecting filters into S from layer l, that are to be retained. We can hence obtain the filters that are to be pruned from $\{1, 2, ..., n\} \setminus S$. The key idea behind the algorithm is to (a) develop a sparse approximation approach that is faster than the adopted matrix computation approach in [19] and (b) prune the network for a given layer all at once for a given fraction, compared to the one-at-a-time filter pruning approach in [19], thus making the pruning method efficient.

3.3 Weight compensation for multiple channel pruning

Joo et al.[19] had proposed the weight compensation module for a single filter pruning for two purposes: (a) the change in weights of the pruned model will get adjusted by the updation of the $1x1$ convolution and (b) usage of $1x1$ convolution enables the pruning of any network, regardless of its architecture. We adopt this module and derive the compensated weights as per our framework for multiple channel pruning.

Consider the input and output of any $K \times K$ convolution layer to be $X = \{X_1, ..., X_m\}$ and $Y = \{Y_1, ..., Y_n\}$. Since the 1×1 convolution is built on the top of the $K \times K$ layer, Y goes as an input to the 1×1 convolution. Let the output of the 1×1 convolution layer be $Z = \{Z_1, ..., Z_n\}$, followed by $f \in R^{m \times n}$ and $g \in R^{n \times n}$ being the filter weights of $K \times K$ and 1×1 convolution layer respectively. We can formulate the above setup as:

$$Y_j = \sum_{i=1}^{m} X_i * f_{i,j} := X * f_{:,j} \tag{5}$$

$$Z_k = \sum_{j=1}^{n} Y_j * g_{j,k} := \sum_{j=1}^{n} X * f_{:,j} * g_{j,k} \tag{6}$$

Algorithm 1 : Filter Pruning-OMP (FP-OMP)

Input:
  Layer c, Number of filters N_c, filters $f_{:,j} \ \ \forall j \in \{1, 2, ..., N_c\}$, pruning fraction β
Initialize:
  Normalize $f_{:,j}$ such that $||f_{:,j}||_2 = 1$
  $R_j = f_{:,j} \ \ \forall j \in \{1, 2, .., N_c\}$ // Residual error
  $S = \phi$ // Set of selected filters
Output:
  S, Weight coefficients $\lambda_{j,l} \ \ \forall l \in S, \forall j \in \{1, 2, .., N_c\}$
Algorithm:
while $\|\lambda\|_0 \leq (N_c - \beta * N_c)$ **do**
  for i in S' **do**
    for j in $\{1, 2, .., N_c\}$ **do**
      Compute $Proj_{ij} = R_j . f_{:,i}$
    end for
    Absolute total projection $\xi_i = \sum_{j=1}^{N_c} |Proj_{ij}|$
  end for
  $ind = \max_i \xi_i$
  $S \longleftarrow S \cup \{ind\}$
  for j in $\{1, 2, .., N_c\}$ **do**
    $\vec{\lambda}_{j,:} = argmin_{\lambda_j} ||f_{:,j} - \sum_{l \in S} \lambda_{j,l} f_{:,l}||^2$
    $R_j \longleftarrow f_{:,j} - \sum_{l \in S} (\vec{\lambda}_{j,:} f_{:,l})$
  end for
end while

Now, let $f_{:,l} : l \in S$ be the selected filter weights and similarly, let $f_{:,l'} : l' \in S'$ be the pruned filter weights. Dividing Equation 6 into the two sets of filter weights, we can re-frame it as:

$$Z_k = \sum_{l \in S} X * f_{:,l} * g_{l,k} + \sum_{l' \in S'} X * f_{:,l'} * g_{l',k} \tag{7}$$

Following the above terminology, revisiting Equation 4, we can write it as:

$$f_{:,l'} = \sum_{l \in S} \lambda_{l',l} f_{:,l} + \epsilon_{l'} \quad ; \forall l' \in S' \tag{8}$$

Substituting Equation 8 in Equation 7, we rewrite Z_k as Z'_k in terms of retained filter weights $f_{:,l}$:

$$Z'_k = \sum_{l \in S} X * f_{:,l} * g_{l,k} + \sum_{l' \in S'} X * (\sum_{l \in S} \lambda_{l',l} f_{:,l} + \epsilon_{l'}) * g_{l',k} \tag{9}$$

The above can also be re-structured as:

$$Z'_k = \sum_{l \in S} [X * f_{:,l} * (g_{l,k} + \sum_{l' \in S'} \lambda_{l',l} * g_{l',k})] + \sum_{l' \in S'} X * \epsilon_{l'} * g_{l',k} \tag{10}$$

Once the pruning is performed, Equation 7 reduces to

$$\sum_{l \in S} X * f_{:,l} * g_{l,k} \tag{11}$$

and Equation 10 reduces to

$$\sum_{l \in S} [X * f_{:,l} * (g_{l,k} + \sum_{l' \in S'} \lambda_{l',l} * g_{l',k})] \tag{12}$$

Thus, the weight difference after pruning, for Z_k and Z'_k, are $\| \sum_{l' \in S'} X * f_{:,l'} * g_{l',k} \|$ and $\| \sum_{l' \in S'} X * \epsilon_{l'} * g_{l',k} \|$ respectively. Owing to the fact that $\epsilon_{l'} < f_{:,l'}$, the weight difference in using Z'_k is lesser than that of Z_k. Also, lower the difference in weights, better the approximation. Hence, we use Equation 12 for the weight compensation step to have a lesser weight difference and define the following step:

$$g'_{l,k} = g_{l,k} + \sum_{l' \in S'} \lambda_{l',l} * g_{l',k} \quad ; \forall k \in [1, n], \quad \forall l \in S \tag{13}$$

For the output channel pruning, Equation 13 is re-defined as

$$g'_{l,:} = g_{l,:} + \sum_{j \in S^c} \lambda_{j,l} * g_{j,:} \quad , \forall l \in S \tag{14}$$

while for input channel pruning, it is re-defined as

$$g'_{:,l} = g_{:,l} + \sum_{j \in S^c} \lambda_{j,l} * g_{:,j} \quad , \forall l \in S \tag{15}$$

3.4 Optimal filter search

One other caveat of the work of Joo et al.[19] was sub-optimality. In this section, we describe a variant of our proposed algorithm in Section 3.2. We intend to prune the filters from the entire network with C layers, whose inclusion do not lead to much change in entropy of the system. In order to find the optimal set of filters across the entire network, we discard the necessity of removal of uniform number of filters from each layer.

Algorithm 2 describes the approach. The algorithm proceeds by pruning a batch of filters from each layer followed by assessing the performance of the model in the pruned stage, and then putting back the removed filters. We keep a running record of the performance metric (*acc*) across all the layers. Eventually, the filters from a particular layer, on removing which gives the maximum accuracy compared to removing filters from other layers, are finally pruned from the network. This continues till we achieve the pruning criteria. Next, we empirically show the performance of our proposed frameworks *FP-OMP* and *FP-OMP Search*.

Algorithm 2 : Filter Pruning-OMP Search (FP-OMP Search)

Initialize:
  $K = 0$
  Pruning fraction : β
  Batch of filters : k
  Total number of layers : C
  Total number of filters across all the layers $\{1, 2, .., C\}$: N_C
Algorithm:
while $K < \beta * |N_C|$ **do**
  for c in C **do**
    Prune k filters from layer c using Algorithm 1
    Calculate entire network accuracy acc_c
    Put back the pruned filters in the layer c
  end for
  $ind = \max_c acc_c$
  Prune k filters from layer *ind* using Algorithm 1
  Fine tune for one epoch
  $K = K + k$
end while

4 EXPERIMENTAL RESULTS

In this section, we describe the experimental setup and the datasets used in Section 4.1. We compare the performance of the proposed methods with the state-of-the-art baselines in Section 4.2. We also do a detailed analysis of the working of the proposed method in Section 4.3.

4.1 Experimental settings:

Dataset Description: We use CIFAR-10, CIFAR-100, and TinyImagenet datasets for the task of image classification. CIFAR-10 has 10 classes comprising of 50k training set images and 10k test set images of resolution 32×32, with 5k training set and 1k test set images in each class. Similar to CIFAR-10, the CIFAR-100 dataset comprises of 100 classes with 500 training images and 100 testing images for each class. TinyImagenet has 200 class categories with a total of 0.1M images. For our experimentation, we resized its 64×64 images to 224×224.

Training Details: We run our experiments for ResNet-32 and ResNet-56 model architectures on CIFAR-10, CIFAR-100 and Tiny-Imagenet dataset with a pruning ratio of 50%. We use a pretrained model while pruning and all the other training settings are adapted from LRF [19]. Using the pretrained model, we add a warmup of 20 epochs before the pruning starts. Unlike the work of Joo et al. [19] where the network is fine-tuned for an epoch, after each filter removal from a layer, we fine-tune the model after pruning the entire β fraction of filters from each layer. Towards the end of pruning the entire model, the network is again fine-tuned for 300 epochs. The initial learnng rate for fine-tuning is set to $1e^{-2}$ with a decay of $1e^{-4}$. We also use a step scheduler that divides the learning rate by 10 at epoch 150.

4.2 Performance Comparison: Accuracy and Efficiency

ResNet on CIFAR10: We used ResNet-32 and ResNet-56 as the model architectures for our experiments. Table 1 shows the performance comparison with four other baselines. We report the baseline accuracy using the pre-trained model that gets used while pruning. Owing to the fact that the same pre-trained model get used for all baselines, the baseline accuracy stays the same. We can observe from the pruned accuracy, that the proposed methods (*FP-OMP* and *FP-OMP Search*) are having a higher accuracy compared to other baselines and eventually a higher accuracy drop from the baseline (pre-trained) model, thus denoting the efficacy of the method.

We report the reduction in parameter count and FLOPs which when reduces more, indicates a more pruned and efficient model. Parameter count refers to the number of parameters/weights across all the retained filters in all the layers, while FLOPs refers to the number of operations (in this case convolutions) within the retained filters in all the layers across the network. We observe a higher reduction in parameter count and FLOPs (**param ↓ , FLOPs ↓**) for FP-OMP compared to other baselines. FP-OMP Search leads to an accurate model but has a trade-off in reduction in parameter count

and FLOPs due to the non uniform pruning across all layers. A possible workaround can be increasing β for FP-OMP Search that can lead to more reduction in parameter count and FLOPs.

ResNet on CIFAR-100: We continue our experiments with the best performing baseline [19] and can observe in Table 2 that the proposed methods consistently perform better. We can observe that the difference in pruned accuracy with the baseline has increased compared to that in Table 1. This can be attributed to the nature of this dataset which is more complex than that of CIFAR-10, used in Table 1.

ResNet on TinyImagenet: We use ResNet-34 for the experiments on TinyImagenet. We can observe in Table 4 that our proposed method performs better compared to the baseline [19] and also has a much higher difference in pruned accuracy compared to [19]. Overall, from the Tables 1, 2 and 4, we can conclude that the difference in pruned accuracy with the baseline method [19], increases with increasing complexity of datasets and models, thus showcasing the robustness of our proposed framework.

Time Comparison: We compare the efficiency of the proposed method in terms of their running times in Table 3. We can observe that FP-OMP clearly takes lesser pruning time than LRF [19]. This happens due to the sparse approximation approach which is less expensive compared to the matrix computation method adopted by Joo et al. [19]. However, we notice an increase in pruning time for FP-OMP Search. This is due to the fact that we use a batch of 5 filters for the current set of experiments for FP-OMP Search (see *Algorithm 2*). Increasing the batch size will lead to pruning time reduction. The fine-tuning time remains the same across all methods, having minor changes with differing architectures.

4.3 Analysis of FP-OMP and FP-OMP Search

We analyse the working of the proposed methods in terms of their pruning quality. We show in Table 5 the pruning count using FP-OMP Search for ResNet32 on CIFAR-100. We maintain a global pruning ratio of 50% across the entire network and report the % of filters removed from each layer within each block of the model. Unlike LRF [19] or FP-OMP, where pruning is done uniformly across each layer, FP-OMP Search prunes the filters of a given fraction β, across the entire network, and retains the others.

In order to understand the intuition behind the filter choices for pruning using FP-OMP Search, we show a visualisation diagram of the feature maps for two layers, Layer 4(pruned by 31.25%) and Layer 10(pruned by 93.75%). We can observe visually that the feature map of Layer 4 in Figure 1 has a diverse set of filter outputs, that indicates its usefulness in capturing different features of the inputs. Consequently, our proposed method prunes only 31.25% of its filters (see *Table 5*), thus retaining around 68.75%. Similarly, Figure 2 shows the feature map outputs from Layer 10 that looks very similar, thus denoting its redundancy in filter outputs. We can observe the same in Table 5 that Layer 10 has 93.75% of its filters removed, thus retaining only about 6.25%. Thus, we can conclude that the pruning percentages yielded from FP-OMP Search correlate with the amount of information held by each filter in each layer.

We also observe the norm of difference in filter weights obtained from unpruned model (f_{up}) and pruned model (f_p) in Figure 3 using FP-OMP and FP-OMP Search. We can observe that for about 53% of the layers, FP-OMP Search has a lower norm difference than FP-OMP indicating better approximation of filter weights with non-uniform pruning ratio.

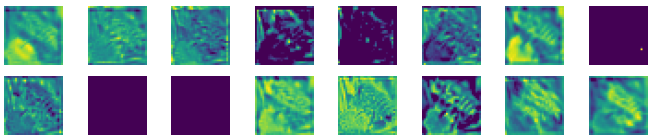

Figure 1: Visualisation of output feature map of ResNet-32 4^{th} layer on CIFAR-100

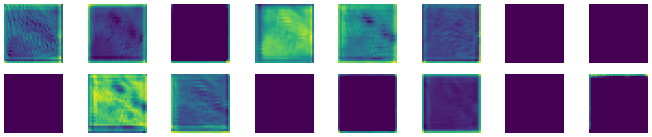

Figure 2: Visualization of output feature map of ResNet-32 10^{th} layer on CIFAR-100

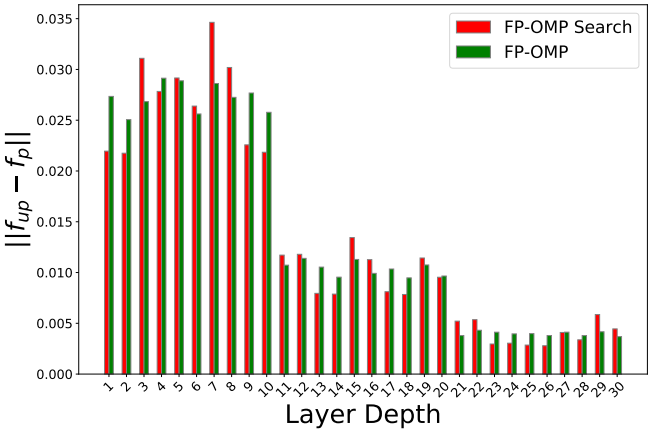

Figure 3: Norm of filter difference of Unpruned model and Model pruned using FP-OMP and FP-OMP Search

5 CONCLUSION AND FUTURE WORK

In this study, we propose the FP-OMP and FP-OMP Search algorithm, a fresh and efficient channel pruning technique. It is a novel pruning criterion that chooses the channel using a sparse approximation method. Regardless of kernel size, block type, or even architectures, it shows a good performance across all of them. Extensive experiments on the 3 datasets with 2 differing architectures prove our hypothesis. A lead that could be followed for enhancing the effectiveness of our suggested pruning strategy is to prune each layer in a different ratio to maximise network performance.

Table 1: Performance comparison of FP-OMP and FP-OMP Search for ResNet-32 and ResNet-56 on CIFAR10 for 50% pruned filters of the network. Acc ↓ , FLOPs ↓ and Param ↓ are the drops in accuracy, parameter count and FLOPs of pruned model compared to the baseline model (Baseline Acc). Higher the drop, better the method.

Models	Method	Baseline Acc	Pruned Acc	Acc ↓	Param ↓	FLOPs ↓
ResNet-32	**SFP [12]**	92.63%	92.08%	0.55%	-	41.5%
	LFPC [11]	92.63%	92.12%	0.51%	-	52.6%
	FPGM [13]	92.63%	91.93%	0.70%	-	53.2%
	LRF [19]	92.63%	92.66%	-0.03%	63.3%	62.55%
	FP-OMP	**92.63%**	**92.79%**	**-0.16%**	**63.3%**	**62.55%**
	FP-OMP Search	**92.63%**	**92.81%**	**-0.18%**	**58.58%**	**44.9%**
ResNet-56	**DCP [35]**	93.80%	93.79%	0.01%	70.3%	47.1%
	HRank [23]	93.80%	93.17%	0.63%	42.4%	50.0%
	SFP [12]	93.80%	93.26%	0.54%	-	52.6%
	FPGM [13]	93.80%	93.49%	0.31%	-	52.6%
	LFPC [11]	93.80%	93.24%	0.56%	-	52.9%
	GBN [33]	93.80%	93.43%	0.37%	42.5%	55.1%
	LRF [19]	93.80%	93.85%	-0.05%	63.3%	62.55%
	FP-OMP	**93.80%**	**94.03%**	**-0.23%**	**63.3%**	**62.55%**
	FP-OMP Search	**93.80%**	**94.08%**	**-0.28%**	**56.50%**	**43.32%**

Table 2: Performance comparison of FP-OMP and FP-OMP Search for ResNet-32 and ResNet-56 on CIFAR100 for 50% pruned filters of the network.

Models	Method	Baseline Acc	Pruned Acc	Acc ↓	Param ↓	FLOPs ↓
ResNet-32	**LRF [19]**	68.78%	68.78%	-0.07%	62.5%	62.54%
	FP-OMP	**68.78%**	**69.05%**	**-0.27%**	**62.5%**	**62.54%**
	FP-OMP Search	**68.78%**	**69.11%**	**-0.33%**	**50.72%**	**53.18%**
ResNet-56	**LRF [19]**	69.98%	70.07%	-0.09%	63%	62.92%
	FP-OMP	**69.98%**	**70.39%**	**-0.41%**	**63%**	**62.92%**
	FP-OMP Search	**69.98%**	**70.43%**	**-0.45%**	**50.72%**	**53.18%**

Table 3: Time comparison of different methods on ResNet for channel pruning on CIFAR10 and CIFAR100 dataset.

Models	Method	Pruning Time (hr)	Fine Tuning Time (hr)	Total Time (hr)
CIFAR10				
ResNet-32	**LRF**	0.58	3.43	4.01
	FP-OMP	**0.54**	**3.41**	**3.95**
	FP-OMP Search	13.63	3.45	17.08
ResNet-56	**LRF**	1.80	4.27	6.07
	FP-OMP	**1.55**	**4.25**	**5.8**
	FP-OMP Search	58.84	4.33	63.17
CIFAR100				
ResNet-32	**LRF**	0.60	3.38	3.98
	FP-OMP	**0.48**	**3.37**	**3.85**
	FP-OMP Search	12.97	3.39	16.36
ResNet-56	**LRF**	1.61	4.09	5.70
	FP-OMP	**1.57**	**4.07**	**5.64**
	FP-OMP Search	54.01	4.11	58.12

REFERENCES

[1] Olivier Bachem, Mario Lucic, and Andreas Krause. 2017. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476* (2017).

[2] Niv Buchbinder, Moran Feldman, and Roy Schwartz. 2014. Online submodular maximization with preemption. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*. SIAM, 1202–1216.

Table 4: Performance comparison of FP-OMP and FP-OMP Search for ResNet-34 on TinyImagenet for 50% pruned filters of the network.

Models	Method	Baseline Acc	Pruned Acc	Acc ↓	Param ↓	FLOPs ↓
ResNet-34	LRF [19]	64.18%	62.86%	1.32%	62.79%	60.76%
	FP-OMP	**64.18%**	**65.68%**	**-1.50%**	**62.79%**	**60.76%**
	FP-OMP Search	**64.18%**	**65.75%**	**-1.57%**	**51.67%**	**55.73%**

Table 5: Percentage removal of filters from each layer of ResNet32 on CIFAR100 dataset using FP-OMP Search method with overall 50% removal of filters from the ResNet32.

Output Channel											
Block 1	Layers	1	2	3	4	5	6	7	8	9	10
	Before/After	16/1	16/1	16/11	16/11	16/11	16/11	16/16	16/11	16/1	16/1
	Percent removed	**93.75**	**93.75**	31.25	31.25	31.25	31.25	0	31.25	**93.75**	**93.75**
Block 2	Layers	11	12	13	14	15	16	17	18	19	20
	Before/After	32/22	32/22	32/2	32/2	32/27	32/27	32/2	32/2	32/27	32/17
	Percent removed	31.25	31.25	**93.75**	**93.75**	15.62	15.62	**93.75**	**93.75**	15.62	46.87
Block 3	Layers	21	22	23	24	25	26	27	28	29	30
	Before/After	64/64	64/54	64/14	64/14	64/9	64/4	64/34	64/29	64/64	64/54
	Percent Removed	0	15.62	**78.12**	**78.12**	**85.93**	**93.75**	46.87	54.68	0	15.62
Input Channel											
Block 1	Layers	1	2	3	4	5	6	7	8	9	10
	Before/After	16/1	16/1	16/11	16/6	16/11	16/11	16/16	16/16	16/1	16/1
	Percent removed	**93.75**	**93.75**	31.25	62.5	31.25	31.25	0	0	**93.75**	**93.75**
Block 2	Layers	11	12	13	14	15	16	17	18	19	20
	Before/After	16/16	32/17	32/2	32/2	32/37	32/27	32/7	32/2	32/17	32/17
	Percent removed	0	46.87	**93.75**	**93.75**	0	15.62	**78.12**	**93.75**	46.87	46.87
Block 3	Layers	21	22	23	24	25	26	27	28	29	30
	Before/After	32/32	64/59	64/9	64/14	64/9	64/4	64/39	64/24	64/64	64/59
	Percent Removed	0	7.81	**85.93**	**78.12**	**85.93**	**93.75**	39.06	62.5	0	7.81

[3] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829* (2019).

[4] Soumi Das, Harikrishna Patibandla, Suparna Bhattacharya, Kshounis Bera, Niloy Ganguly, and Sourangshu Bhattacharya. 2021. TMCOSS: Thresholded Multi-Criteria Online Subset Selection for Data-Efficient Autonomous Driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 6341–6350.

[5] Soumi Das, Arshdeep Singh, Saptarshi Chatterjee, Suparna Bhattacharya, and Sourangshu Bhattacharya. 2021. Finding High-Value Training Data Subset Through Differentiable Convex Programming. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 666–681.

[6] Xuanyi Dong and Yi Yang. 2019. Network pruning via transformable architecture search. *Advances in Neural Information Processing Systems* 32 (2019).

[7] Ehsan Elhamifar and M Clara De Paolis Kaluza. 2017. Online Summarization via Submodular and Convex Optimization.. In *CVPR*. 1818–1826.

[8] P Thomas Fletcher, Suresh Venkatasubramanian, and Sarang Joshi. 2008. Robust statistics on Riemannian manifolds via the geometric median. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1–8.

[9] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems* 28 (2015).

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

[11] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. 2020. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2009–2018.

[12] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. 2018. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866* (2018).

[13] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4340–4349.

[14] Yihui He, Xiangyu Zhang, and Jian Sun. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*. 1389–1397.

[15] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* 2, 7 (2015).

[16] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* (2016).

[17] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[18] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*. PMLR, 448–456.

[19] Donggyu Joo, Eojindl Yi, Sunghyun Baek, and Junmo Kim. 2021. Linearly replaceable filters for deep network channel pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8021–8029.

[20] Krishnateja Killamsetty, Durga Sivasubramanian, Baharan Mirzasoleiman, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. 2021. GRAD-MATCH: A Gradient Matching Based Data Subset Selection for Efficient Learning. *arXiv preprint arXiv:2103.00123* (2021).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

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

[23] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. 2020. Hrank: Filter pruning using high-rank feature map. In

Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 1529–1538.

[24] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*. 2736–2744.

[25] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. 2019. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE/CVF international conference on computer vision*. 3296–3305.

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

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

[28] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating Training Data Influence by Tracing Gradient Descent. In *Advances in Neural Information Processing Systems*.

[29] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[30] Joel A Tropp and Anna C Gilbert. 2007. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory* 53, 12 (2007), 4655–4666.

[31] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1492–1500.

[32] Jinsung Yoon, Sercan Arik, and Tomas Pfister. 2020. Data valuation using reinforcement learning. In *International Conference on Machine Learning*. PMLR, 10842–10851.

[33] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. 2019. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *Advances in neural information processing systems* 32 (2019).

[34] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 9194–9203.

[35] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. 2018. Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems* 31 (2018).