Pruning CNN Filters via Quantifying the Importance of Deep Visual Representations

Ali Alqahtani[1,2], Xianghua Xie[1], Mark W Jones[1], and Ehab Essa[1,3]

[1] Department of Computer Science, Swansea University, Swansea, UK
[2] Department of Computer Science, King Khalid University, Abha, Saudi Arabia
[3] Department of Computer Science, Mansoura University, Mansoura, Egypt

Abstract. The achievement of convolutional neural networks (CNNs) in a variety of applications is accompanied by a dramatic increase in computational costs and memory requirements. In this paper, we propose a novel framework to measure the importance of individual hidden units by computing a measure of relevance to identify the most critical filters and prune them to compress and accelerate CNNs. Unlike existing methods, we introduce the use of the activation of feature maps to detect valuable information and the essential semantic parts to evaluate the importance of feature maps, inspired by novel neural network interpretability. A majority voting technique based on the degree of alignment between a semantic concept and individual hidden unit representations is proposed to quantitatively evaluate the importance of feature maps. We also propose a simple yet effective method to estimate new convolution kernels based on the remaining, crucial channels to accomplish effective CNN compression. Experimental results show the effectiveness of our filter selection criteria, which outperforms the state-of-the-art baselines. Furthermore, we evaluate our pruning method on CIFAR-10, CUB-200, and ImageNet (ILSVRC 2012) datasets. The experimental results show that the proposed method efficiently achieves a 50% FLOPs reduction on CIFAR-10, with only 0.86% accuracy drop on the VGG-16 model. Meanwhile, ResNet pruned on CIFAR-10 achieves a 30% reduction in FLOPs with only 0.12% and 0.02% drops in accuracy on ResNet-20 and ResNet-32 respectively. For ResNet-50 on ImageNet, our pruned model achieves a 50% reduction in FLOPs with only a top-5 accuracy drop of 0.27%, which significantly outperforms state-of-the-art methods.

Keywords: Deep learning · convolutional neural networks · filter pruning · model compression

1 Introduction

Convolutional neural networks (CNNs) have attained extraordinary levels of achievement in numerous recognition tasks, especially in computer vision (i.e., object detection ([13]), semantic segmentation, ([43]) and image classification ([25, 48, 17]), thereby becoming an indispensable method used for a variety of applications. CNNs have been shown to outperform all other techniques in image-processing tasks ([27]), where convolutional layers extract spatially related features, preserving the local structure of image data. The feature extractor benefits from its ability to discover, learn and perform automatic representation learning by transforming raw data into a more abstract representation. Their hierarchical representation allows models to learn features at multiple levels of abstraction, meaning complicated concepts can be learned from simpler ones. Units in the earlier layers of a network learn low-level features while units in later layers learn more complex concepts ([59]).

Due to greater quantities of data and advanced computing power, CNNs have turned into wider and deeper architectures, driving state-of-the-art performances in a wide range of applications. Despite their great success, however, CNNs have an enormous number of parameters, and their significant redundancy in parameterization has become a widely-recognized property ([9]). The over-parametrized and redundant nature of CNNs incur expensive computational costs and high storage requirements. To classify a single image, the VGG-16 model ([48]), for instance, requires more than 30 billion floating-point operations (FLOPs), and contains about 138 million parameters with more than 500 MB of storage space. This presents significant challenges and restricts many CNN applications. For example, deploying sizeable deep learning models to a resource-limited device leads to various constraints, as on-device memory is limited ([55]). Therefore, the necessity of reducing high computation costs and storage requirements becomes critical to allow CNNs greater applicability in a broader range of applications (e.g. mobile devices, autonomous agents, and embedded systems). Reducing the complexity of models while maintaining their powerful performance is always desirable. Recognizing the importance of networks units can help reduce model complexity by discarding less essential units.

Network pruning focuses on discarding unnecessary parts of neural networks, which reduces the massive computational costs and memory requirements associated with deep models. Pruning approaches can be applied to any part of deep neural networks, including fully connected layers ([28, 16, 15, 18, 39, 56]) and convolutional layers ([29, 56, 33, 41, 36, 21]). The idea of pruning has been studied since the early 1990s. Optimal Brain Damage (OBD) by [28] and Optimal Brain Surgeon (OBS) by [16] are considered pioneering works in network pruning, demonstrating that unnecessary weights can be eliminated from a trained network with little accuracy loss. Due to expensive computation costs, these methods are not applicable to today's deep models. Obtaining a subnetwork with far fewer parameters without compromising accuracy is the main goal of pruning algorithms. As the pruned version, a subset of the whole model, can represent the reference model at a smaller size or with fewer parameters. Thus, overparameterized networks can be efficiently compressed while maintaining the property of better generalization ([2]).

Recently, [15] introduced a simple pruning method to remove connections based on a predefined threshold. Han's framework relies on an iterative pruning procedure to obtain a sparse model. Nonetheless, such an unstructured sparse model requires a particular software/hardware accelerator which is not supported by off-the-shelf libraries. Moreover, the reliance on a predefined threshold is less practical and proves too inflexible for some applications. The random connectivity of non-structured sparse models can also cause poor cache locality and jumping memory access, which significantly limits the practical acceleration ([53]). In an attempt to confront these challenges, we consider filter-level pruning in our proposed method, whereby removing the unimportant filter in its entirety does not change the network structure so the method can be used with any off-the-shelf deep learning library, allowing for more compression and acceleration by other pruning techniques, such as the parameter quantization method ([55]). This procedure can also effectively reduce the memory requirements, as model compression focuses on reducing not only model parameters but also the intermediate activation; this has received little attention in previous works.

Most existing methods tend to focus on applying simple pruning techniques (e.g. statistical approaches) to compress networks rather than on discovering informative units for effective pruning. The fact that not all filters deliver essential information for the final prediction of the model motivates us to fundamentally rely on quantifying the importance of latent representations of CNNs by evaluating the matching/alignment between semantic concepts and individual hidden units to score the semantics of hidden feature maps at each convolutional layer. Our core aim is to evaluate neuron importance, which provides meaningful insight into the characteristics of the internal representations of neural networks. For the purpose of providing a clear understanding of the internal operation and work mechanisms of deep networks, several approaches have been developed to visually understand convolutional layers ([59, 38, 47]), interpret deep visual representations and quantify their interpretability ([4, 5]), as well as measure the influence of hidden units on the final prediction ([10, 42, 3]). The main focus of these methods is to understand the predictions of a model by analyzing the individual units and seeking an explanation for specific activation. Although these methods provide an intuitive process to determine criteria for neuron selection for effective pruning, most of the previously mentioned methods focus on gaining a better understanding of the network's behavior, with limited attention being paid to pruning methods.

In this paper, we propose a novel framework to compute a measure of relevance, identify the most critical filters and prune the unimportant filters to compress and accelerate CNNs, with only a small drop in model accuracy. Our proposed framework focuses on filter-level pruning based on evaluating the degree of alignment between a semantic concept and individual hidden unit representations. Quantifying the interpretability of deep visual representations of CNNs ([4]) determines the function of individual CNN's filters to deliver essential information, where a filter's feature map is more critical to the network when it represents more information. This fact reflects the contribution of input to deliver essential information for the final prediction of the model. Localization maps help identify the critical regions in the image to predict the concept used to explain individual network decisions ([46, 63, 61]). Motivated by this, we propose our pruning framework, where we evaluate the degree of alignment between a semantic concept and individual hidden unit representations. We introduce feature maps to detect valuable information and the essential semantic parts, which are fundamental factors in evaluating the importance of feature maps and determining individual CNN filters' function to deliver essential information with solid discriminative power for the model. We visually demonstrate that non-pruned channels are related to the concept of an object, closely matching the semantic segmentation of an object, while selected channels to be pruned are less informative and not correlated to the region of an object class across different images, which provide a convincing analysis of the motivation. Our work is considered a pioneering one that attempts to use the quantification of interpretability for more robust and effective CNN's pruning.

We propose a more accurate importance measure, a majority voting technique, to compare the degree of alignment values among filters and assign a voting score to evaluate their importance quantitatively. This mechanism helps to effectively reduce model complexity by eliminating the less influential filters and aims to determine a subset of the whole model that can represent the reference model with much fewer parameters. One significant advantage of filter-level pruning is that it does not lead to model structure damage; therefore, other pruning methods can be efficiently adopted for further compression.

To minimize the damage of the pruning procedure, we propose a simple yet effective method to estimate new convolution kernels based on the remaining, crucial channels. Our fundamental insight is that we introduce an optimization problem based on which the kernels can be estimated depending on the remaining feature maps inputs and the output of the pruned filter. This novel finding differentiates our kernels estimation method from a fine-tuning procedure, which is the most common technique applied by most of the existing methods to recover damaged accuracy.

In order to gather conclusive evidence to evaluate the effectiveness of our method, an experiment based on ablation analysis in trained models was carried out. By comparing our importance method with several state-of-the-art methods, we demonstrate that our approach substantially outperforms others in terms of filters' effective measurement, notably with larger compression ratios. We evaluate our pruning method on CIFAR-10, CUB-200, and ImageNet (ILSVRC 2012) datasets and two types of network architecture: plain CNN (VGG-16 ([48])) and residual CNN (ResNet-20/32/50 ([17])). The experimental results show that the proposed method efficiently achieves 50% FLOPs reduction on the VGG-16 model, with only 0.86% accuracy drop. Although ResNet is more compact and has less redundancy than VGG models, it can still reduce 30% FLOPs, with 0.12% and 0.02% accuracy drop on ResNet-20 and ResNet-32 respectively. For ResNet-50 on ImageNet, our proposed model is capable of reducing 30% FLOPs with only a 0.24% reduction in the original model's top-1 error and a 0.03% reduction in the top-5 error.

The rest of this paper is organized as follows. In section 2, we present related works, while we describe our proposed methodology in section 3. In section 4, we present our experimental results. Finally, concluding remarks are provided in section 5.

2 Related Work

Pruning approaches have received considerable attention as a way to tackle over-parameterization and redundancy. Consequently, overparameterized networks can be efficiently compressed and allow for the acquisition of a small subset of the whole model, representing the reference model with far fewer parameters ([8]). Pruning networks' redundancy always requires a more careful approach. There is no standard guidance for choosing the best network architecture; a model may need a certain level of redundancy during model training to guarantee excellent performance. There is therefore great necessity to decrease model size after its training ([36]).

Several methods have been proposed to prune non-informative parts from heavy, over-parameterized deep models while preserving reference model accuracy. [15] introduced a method to prune unimportant connections whose absolute values are smaller than a predefined threshold value. The threshold is calculated using the standard deviation of a layer's weights. The network is, thereafter, retrained to recover the dropped accuracy. However, a non-structured sparse model requires a particular software/hardware accelerator where additional sparse matrix operation libraries are adopted. Moreover, the reliance on a predefined threshold is less practical and proves too inflexible for many applications. Furthermore, [40] replaced the fully-connected layers with sparsely-connected layers by applying initial topology based on the Erdős–Rényi random graph. During training, fractions of the smallest weights are iteratively removed and replaced with the new random weights. Applying initial topology allows for the finding of a sparse architecture before training; however, this requires expensive training steps and obviously benefits from iterative random initialization. Random connectivity causes cache and memory access issues so the acceleration of even high sparsity models is very limited.

To overcome the weaknesses associated with the random connectivity of unstructured pruning, some strategies corresponding to group-wise sparsity-based network pruning were explored. [53] proposed the Structured Sparsity Learning (SSL) method, which imposes group-wise sparsity regularization on CNNs, applying the sparsity at different levels of their structure (filters, channels, and layers) to construct compressed networks. [26] also employed group-wise sparsity regularization, which has the effect of shrinking individual weights towards zero, meaning that they can be effectively ignored. [62] incorporated sparsity constraints on network weights during the training stage, aiming to build pruned DNNs. Although this proved successful in such sparse solutions, it results in the damage of the original network structure. The need therefore remains to adopt special libraries or use particular sparse matrix multiplication to accelerate the inference speed in real applications.

Compressing a network via a training process may present effective solutions. [12] presented an optimization method that enforces correlation among filters to converge at the same values to create identical filters, the redundant of which are safely eliminated during training. [19] proposed a filter pruning method which prunes convolutional filters in the training phase. After each training epoch, the method measures the importance of filters based on L2 norm, and the least essential filters are set to zero. [20] later iteratively measured the importance of the filter by calculating the distance between the convolution kernel and the origin or the geometric mean based on which redundant kernels are identified and pruned during training. [35] trained an auxiliary network to predict the weights of the pruned networks and estimate the performance of the remaining filters. [58] applied a training objective to compress the model as a task of learning a scaling factor associated with each filter and estimating its importance by evaluating the change in the loss function. AutoPruner ([37]) embedded the pruning phase into an end-to-end trainable framework. After each activation, an extra layer is added to estimate a similar scaling effect of activation, which is then binarized for pruning. [51] also used a training phase to learn scaling factors to help discover redundancy, where filters are pruned based on the learnable scaling factors. Furthermore, [51] used a training phase to learn scaling factors to help discover redundancy, where filters are pruned based on the learnable scaling factors' values. A significant drawback of iteratively optimized pruning is the extensive computational cost, as modern GPUs do not benefit from sparse convolutions. Pruning procedures based on iterative training often change the optimization function and even introduce hyper-parameters, making the training more challenging.

As in our work, filter-level pruning approaches have been widely studied in the community ([21, 29, 21, 41, 33, 36, 30]). The aim of these strategies is to evaluate the importance of intermediate units, where pruning is conducted according to the lowest scores. [21] evaluated the importance of filters based on the Average Percentage of Zero activations (APoZ) in their output feature maps. [29] put forward a pruning method based on the absolute weighted sum, where pruning is carried out according to the lowest scores. [33] also proposed a pruning method based on the mean gradient of feature maps in each layer, which reflects the importance of features extracted by convolutional kernels. Furthermore, [36] proposed the ThiNet method, which applies a greedy strategy for channel selection, pruning the target layer by greedily selecting the input channel that has the least increase in reconstruction error. The least-squares approach is applied to indicate a subset of input channels which have the smallest impact to approximate the output feature map. [30] recently introduced a filter pruning method based on the rank of feature maps, where the low-rank feature maps contain less information and can be safely pruned. [57] also computed nuclear-norm derived from singular values decomposition to quantify the importance of each filter. These methods tend to compress networks by adopting straightforward selection criteria based on statistical information. However, dealing with an individual CNN filter requires an intuitive process to determine selective and semantically meaningful criteria for filter selection, where each convolution filter responds to a specific high-level concept associated with different semantic parts. Most relevant to our current work is a CNN pruning method inspired by neural network interpretability. [56] proposed such a method based on layer-wise relevance propagation (LRP) by [3], where weights or filters are pruned based on their relevance score, combining the two disconnected research lines of interpretability and model compression.

Some works have utilized low-rank approximations for model compression and acceleration to achieve further speedup and obtain small CNN models ([49, 31, 54, 11]). Although such approaches are computationally expensive and cannot perform global parameter compression ([6]), they can be integrated with our proposed method to obtain more compressed networks for further improvement.

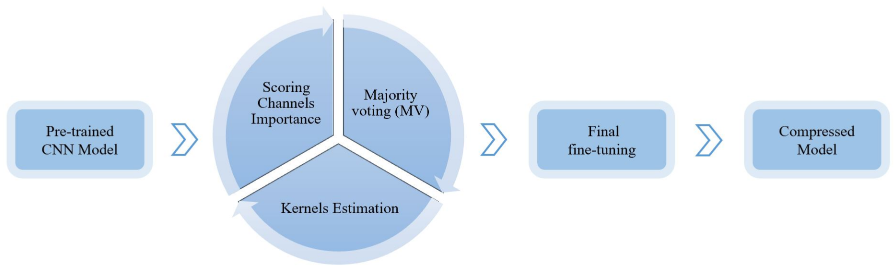


Fig. 1: The overall pipeline of our proposed framework. A pre-trained CNN model is pruned layer-by-layer through iteratively applying our proposed channels selection criteria, majority voting, and kernels estimation, followed by further final fine-tuning to recover the dropped accuracy.

3 Proposed Method

The paper mainly studies a filter-level pruning method to reduce model complexity and obtain a small subset of the whole model that can represent the reference model without performance degradation. In this section, our overall pruning framework is presented; our proposed method consists of three major parts, the first two being scoring channel importance via quantifying the importance of individual hidden representations **section 3.2** and assigning their voting scores **section 3.3**, on which we quantitatively evaluate the importance of filters in a specific layer, eliminating the less influential filters accordingly. Then, we introduce a kernels estimation method in **section 3.4**.

3.1 Overview of our Pruning Methodology

Our method prunes a pre-trained model layer-by-layer with a predefined compression rate. Given a pre-trained CNN model, we proposed a novel method to compute a measure of relevance that identifies the most critical units. Based on this, the less informative channels are pruned. Then, new convolution kernels are estimated based on the surviving channels, and a final fine-tune for the whole network is carried out. As we mainly concentrate on filter-level pruning, the pruned version of our model can be further pruned into an even smaller model by adopting other methods. The overall pipeline of our method is presented in Fig. 1. Here, the proposed method consists of four iterative steps as follows:

1. Channels Selection Criteria. In contrast to previous methods which benefit from the statistical information of layer i to lead the pruning of filters in the same layer, we benefit from the output feature maps of the previous layer $i-1$ to prune filters in the existing layer i. Based on the proposed novel method to score channels' importance, we aim to carefully select a set of channels in layer $i-1$ that are the most influential in the output feature map of layer i, as shown in Fig. 2.
2. Pruning. Unimportant channels and their corresponding filters in layer i are pruned, keeping the structure of the original network the same. This means that our pruning method assumes that only fewer informative filters and channels can approximate an output feature map of layer i. This procedure allows for unimportant filters to be neglected without harming the structure of the original network.
3. Kernels Estimation. To minimize damage from the pruning procedure, we introduce a method to estimate new convolution kernels based on part of the remaining, unpruned channels. The target number of filters is obtained by computing a partial convolution, which means that we only utilize the remaining channels of the output feature maps of layer $i-1$ to estimate optimal kernels that approximate the output feature map of layer i.
4. We iterate to the first step to prune the next layer.
5. To achieve a more accurate model, further final fine-tuning is carried out once all layers have been pruned.

3.2 Scoring Channel Importance Method

Our aim was to identify the most influential channels on CNNs based on which the crucial filters are detected. Measuring the importance of every individual convolutional channel allows for the determination of a subset of the channels whose patterns are the most substantial. Inspired by neural network interpretability, we propose a novel pruning framework based on evaluating the degree of alignment between a semantic concept and individual hidden unit representations. Network Dissection ([4]) was originally developed to quantify the interpretability of latent representations of CNNs that reflect the contribution of an input to deliver essential information for the final prediction of the model.

Every input image x is fed through a pre-trained model by applying forward passing through an optimized model to find the output of each feature map. Each layer has kernels that are convolved on an input example n or a feature

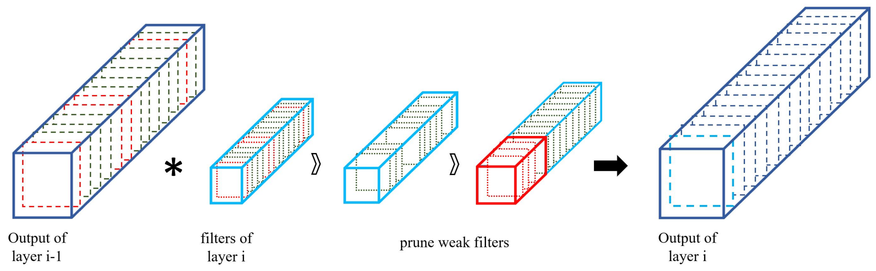


Fig. 2: Our pruning method. The initial state of a CNN feature map in a fully trained model. Green dotted boxes in the diagram indicate important channels. We identify several unimportant channels and their corresponding filters (the red dotted boxes), which contribute very little to the overall performance. These filters can be safely pruned, leading to a pruned model.

map of the internal layers, x, to produce an output corresponding to the n-th example. The activation at j-th feature map is then computed, where the output of the j-th unit in the i-th layer of CNN is defined as:

$$t_j^{(i)}x_{(n)} = \sigma\left(b_j^{(i)} + \sum_p w_p^{(i)} * t_p^{(i-1)}x_{(n)}\right), \tag{1}$$

where $x_{(n)}$ denotes the n-th data example at the input, σ is the activation function (e.g. sigmoid, ReLU), $b_j^{(i)}$ denoting the corresponding bias for the j-th unit in the i-th layer, $w_p^{(i-1)}$ is the weights of the p-th kernel in the i-th layer (existing layer), $t_p^{(i-1)}x_{(n)}$ is the output feature maps of the previous layer $i-1$ and $*$ denotes the 2D convolution operation.

To measure a channel's importance, the latent representation of every individual feature map is evaluated as a solution to a binary segmentation task of the visual concept in the input space. Determining the function of individual filters in a CNN and their ability to localize the meaningful semantic part aid to efficiently measure the importance of different feature maps, which is useful for effective pruning. After this the activation matrix $t_j^{(i)}x_{(n)}$ is calculated by Eq.(1), and the feature map of each internal convolutional channel j is obtained. Then, the distributions of individual feature maps j are computed, and it is based on this that the top quantile value is determined over every spatial location of the feature map. The top quantile value is used as a threshold of T to produce a binary matrix for each channel in the latent space. Here, the output feature map $t_j^{(i)}x_{(n)}$ is thresholded into a binary segmentation M, where all regions that exceed the threshold are selected. If a channel in hidden layers has feature maps that are smaller than the input resolution, they are scaled up to match the input resolution using bilinear interpolation. The interpolating function assigns each missing pixel by taking the weighted average of the nearest pixels. We evaluated the importance of every individual channel $M_j(t_j^{(i)}x_{(n)})$ by computing intersection over union score between their binary segmented versions against semantic segmentation of the input image $I(x_{(n)})$. Fig.3 summarizes the method of scoring channel importance by computing intersection over union (IOU) scores.

$$IoU_j = \frac{\left|M_j(t_j^{(i)}x_{(n)} > T) \cap I(x_{(n)})\right|}{\left|M_j(t_j^{(i)}x_{(n)} > T) \cup I(x_{(n)})\right|}. \tag{2}$$

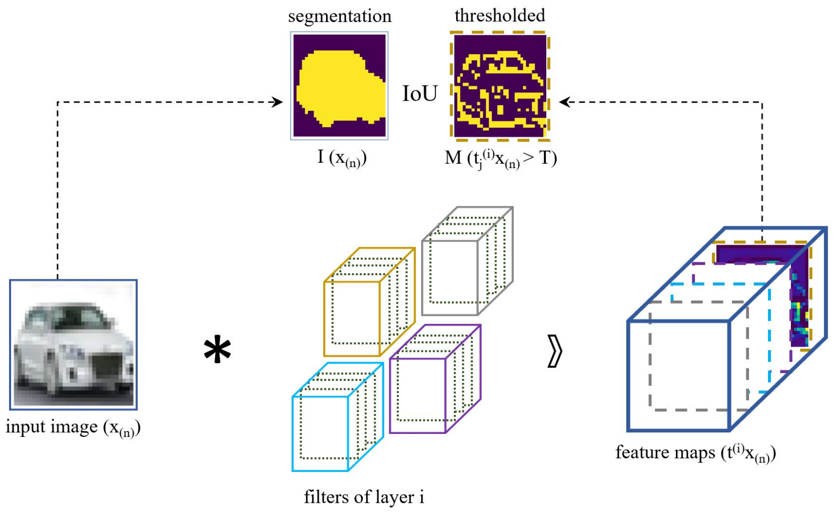


Fig. 3: Scoring Channel Importance Method. The activation matrix $t^{(i)}x_{(n)}$ is obtained by Eq.(1). The feature map of each internal convolutional channel j is collected. For each channel j, we determined the top quantile value and used it as a threshold T to produce a binary matrix for each channel $M_j(t_j^{(i)}x_{(n)} > T)$. We evaluated the importance of every individual binary segmentation for each channel $M_j(t_j^{(i)}x_{(n)} > T)$ against the semantic input segmentation $I(x_{(n)})$ by computing intersection over union scores (this figure is best viewed in color).

3.3 Majority voting (MV) Method

Feeding a set of the data through the network, each example is represented differently and has individual activation throughout all channels in the network. This can be viewed as random variables, and different input images can obtain different IoU scores for different channels. Unlike existing methods that use statistics, we propose a technique which utilizes a majority voting strategy to vote for crucial channels based on their IoU scores. The majority voting technique compares the IOU scores among all channels and assigns a voting score to quantitatively evaluate the channels' importance and gain more confidence regarding how much each channel contributes to the refined features. Our proposed method aims to compute a measure of relevance that identifies the most critical channels, where it only votes for a channel when all the instances agree, which is what majority voting refers to. After obtaining the IoU scores for each channel j, which correspond to an input example n, by Eq.(2), our method votes for l values with the highest IoU scores; this is defined as:

$$\upsilon_j^{(i)} x_{(n)} = \begin{cases} 1 & \text{if } \operatorname{argsort}(IoU)[1:l] \\ 0 & \text{Otherwise} \end{cases} . \tag{3}$$

As a result, a binary matrix is obtained with $J * N$ dimension in the i-th layer, where J is the number of channels and N the number of input examples. The obtained matrix determines how important a channel is for a given example, where 1 indicates the most influential and 0 indicates otherwise. Then, we sum over columns (examples) to score the number of times that the j-th channel is one of the top channels for given examples, voting for the crucial channels. This is given by the following form:

$$y_j^{(i)} = \sum_{n=1}^{N} \upsilon_j^{(i)} x_{(n)}. \tag{4}$$

$$\psi_j^{(i)} = y_j^{(i)} = \begin{cases} 1 & \text{if } \operatorname{argsort}\ (y_j^{(i)})[1:k*J] \\ 0 & \text{Otherwise} \end{cases} , \tag{5}$$

where k is the compression rate, and J are the channels. We set a k percentage of the J channels, which have the largest voting scores, to 1 and the remaining to 0. Here, k denotes the percentage of the largest index of y. For every layer, we determine a binary vector that indicates whether such channels are important or not, where 1 denotes that the channel is important and 0 denotes otherwise. The procedure of our majority voting (MV) method is summarized in Fig. 4.

3.4 Kernels Estimation Method

A binary vector by Eq.5 indicates k percent of the essential channels for every layer, based on which the non-important channels are pruned. Here, a certain number of channels with the lowest voting scores are pruned. This mechanism helps effectively reduce model complexity by eliminating the less influential channels and aims to obtain a subset of the whole model that can represent the reference model with much fewer parameters, whilst preserving the reference model accuracy.

Since there is no guarantee of preserving accuracy throughout the compression phase, a final fine-tuning or iterative layer-wise fine-tuning are the only techniques applied by most of the existing methods to recover damaged accuracy. A simple compression approach benefits from such valuable steps, especially

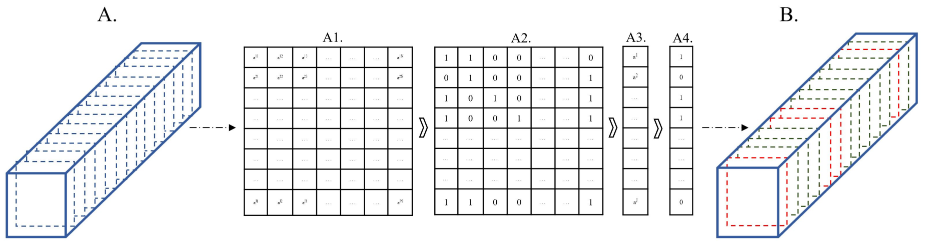


Fig. 4: Majority voting (MV) method. After obtaining IoU scores for each channel (Fig.3), we collect the IoU for each channel j and each input example n (A1). Our method votes for l percentage of highest IoU scores, and as a result, a binary matrix is obtained (A2), which determines how important a channel is for a given example, where 1 indicates the most influential and 0 indicates otherwise. Then, we sum over columns (examples) to score the number of times that the j-th channel is one of the top channels for the given examples, voting for the crucial channels (A3). We set a k percentage of the J channels, which have the largest voting scores, to 1 and the remaining to 0. Here, k denotes the percentage of the largest index of y. For every layer, we determine a binary vector which indicates whether such channels are important or not, where 1 denotes that the channel is important and 0 denotes otherwise (B) (this figure is best viewed in color).

when the selection criteria is straightforward and does not adequately measure the importance, due to the adoption of less efficient measurement standards.

To minimize the damage of the pruning procedure, we propose a simple yet effective method to estimate new convolution kernels based on the remaining unpruned channels. The new kernels can be estimated with only a small number of examples without further training, which is significantly faster to implement. This procedure does not require a multi-step process, in contrast to the fine-tuning procedure (e.g, building a new model, reloading the parameters of the pruned model, and freezing/unfreezing some of the layers), allowing for a fast and efficient process.

Here, we introduce a method to estimate a new convolution kernel based on the remaining, crucial feature map. The new convolution kernel is computed by partial convolution, which means that we will not convolve through all channel in the input feature map, as a subset of channels is pruned already. The target number of filters is obtained by utilizing the remaining channels of the output feature maps of the previous layer to estimate optimal kernels that approximate the output feature map of the existing layer. Therefore, we can prune the filter channels while minimizing the pruning effect.

Using the output feature maps of the previous layer and the convolved version of it, which is the output feature maps of the existing layer, we are able to cal-

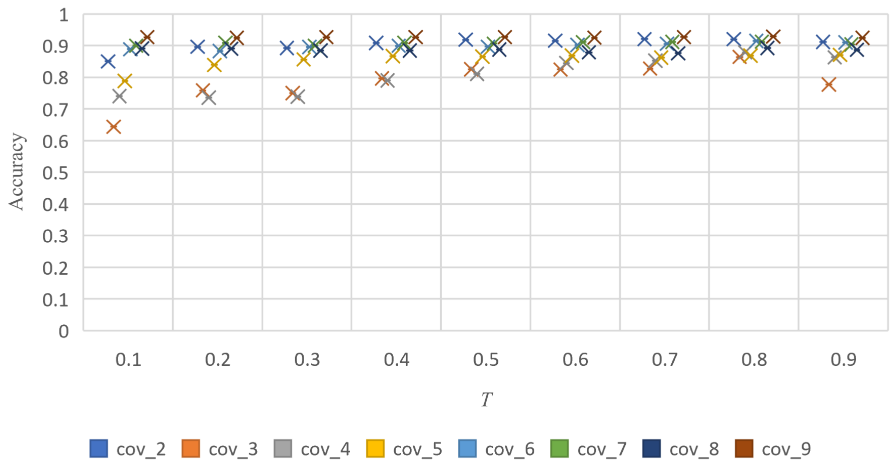


Fig. 5: Different settings to determine the top quantile level T in Eq.(2) at different layers of VGG16 on CIFAR-10.

culate the convolution kernel. This problem is considered a simple optimization problem in the spatial domain. Given an objective function as follows:

$$\frac{1}{2} \parallel h * x - y \parallel^2, \tag{6}$$

where h is the 2D convolution kernel, y is the convolution output feature maps, x is a given output feature maps of the previous layer and $*$ forms the 2D convolution operation. The gradient with respect to the convolution kernel h would be:

$$\frac{d\frac{1}{2} \parallel h * x - y \parallel^2}{dh} = \frac{1}{N} \sum_{i=1}^{n} x \otimes (x * h - y), \tag{7}$$

where N denotes the number of samples used in our estimation method, $\otimes$ denotes the correlation operation and $*$ denotes the convolution operation.

Algorithm 1 Channel pruning algorithm based on quantifying the importance of deep visual representations.

```
Input: a pre-trained Model, training set ($x$, $y$), and compression rate $r$.
Output: a pruned model.
for each layer do
    collect the receptive field of each feature map, Eq.(1).
    compute the $IOU$ score for each filter, Eq.(2).
    vote for top $lOU$ scores, Eq.(3).
    compute how many times a filter has been voted, Eq.(4).
    vote for $k$% of largest voting-score neurons, Eq.(5).
    prune the non-important filters.
    estimate new convolution kernels based on part of the remaining, unpruned channels, section. 3.4.
end
final fine-tuning of the pruned model
```

4 Experiments

In this section, we empirically study the performance of our proposed method. Pruning channels with efficient selection criteria along with the majority voting technique indicates that channels with larger voting scores are more important in network performance. We first apply the proposed method to prune two types of network architecture: plain networks (VGG-16 ([48])) and residual networks (ResNet-20/32/50 ([17])) on three different datasets: The CIFAR-10 dataset ([24]), Caltech-UCSD Birds (CUB-200) dataset ([52]), and ImageNet (ILSVRC 2012) dataset ([45]).The experimental results show that our method adds substantial compression and further reduces model complexity, with little reduction in model accuracy. In this section, we also compare our selection criteria for filter-level pruning with several baselines, and evaluate the change in loss caused by removing a set of filters. The experimental results show that our method substantially outperforms all other baselines. Finally, we empirically verify the validity of our kernels estimation (KE) method and compare it with the standard fine-tuning (FT) procedure. The proposed method was implemented using *Keras* ([7]) and *Tensorflow* ([1]) in *Python.*

Experimental Datasets

We evaluated our filter-level pruning method on three different datasets.

- **CIFAR-10** ([24]) : is an image dataset which consists of 60,000 images. Each example is a 32 x 32 color image, and is associated with a label from 1 of 10 different classes. Each class contains 6,000 images. The CIFAR-10 consists of 50,000 examples as a training set and 10,000 examples as a test set.
- **CUB-200** ([52]) : is a bird subcategories image dataset which contains 200 species of birds; 11,788 images are associated with a label from 1 of 200

classes, where each class has roughly 30 training images and 30 testing images. The CUB-200 contains 5,994 examples as a training set and 5,794 examples as a test set.

- **ImageNet (ILSVRC 2012)** ([45]) : is a large-scale dataset which consists of over 14 million labelled images. Each example is associated with a label from 1 of 1,000 different classes. The ImageNet consists of 1.28 million images as a training set and 50,000 images as validation images.

For CUB-200 and ImageNet, each image is resized to 256×256, then a 224×224 area is randomly cropped from each resized image. The classification performance is reported on the test set for both CIFAR-10 and CUB-200 datasets and on the standard validation set for the ImageNet dataset.

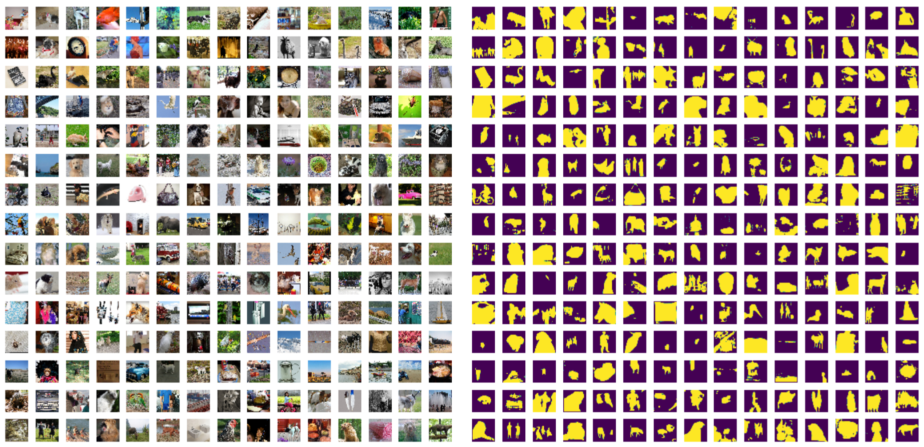


Fig. 6: Different images of the ImageNet and their segmentation results.

Inputs and Feature Maps Binary Segmentation

To collect instances for channel selection, we randomly selected ten images from each class in the training set to form our evaluation set. These selected samples were used to find the optimal channel subset via Algorithm 1. For semantic segmentation, we used the PSPnet model by [60] to segment the input images of CIFAR-10. For CUB-200, the segmentation masks were provided by Ryan Farrell[4]. We also used the segmentation masks provided by [14] for the ImageNet dataset. Fig. 6 illustrates different images of the ImageNet and their segmentation results. The proposed method evaluates every individual convolutional unit in a CNN as a solution to a binary segmentation task of the visual concept in the input space (Fig. 3). Feeding the selected instances through the network,

[4] http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

each example has an individual activation throughout all feature maps in the network. Each collected feature map is converted into a binary matrix using the top quantile value as a threshold T. An experiment based on different settings to determine the top quantile level T in Eq.(2) was carried out in order to gather conclusive evidence to carefully choose the top quantile value when producing a binary matrix for each channel in the given feature map. The comparative results are shown in Fig. 5. As a result, the top quantile value is determined such that $M_j(t_j^{(i)} x_{(n)} > T) = 0.8$ over every spatial location of the feature map. Therefore, the output feature maps of the previous layer $i-1$ are segmented into binary segmentation.

Implementation Details

To measure channels' importance, the feature maps' binary segmentation are evaluated against the semantic input segmentation by computing intersection over union (IoU) score. Given IoU scores for each channel, our results show that each example has different IoU scores, as each is represented differently and has individual activation throughout all channels in the network. Therefore, by using a set of data samples to find the optimal channel subset, the judgment of the selection criteria becomes more accurate. Empirical evidence comes from a comparison between a range of hyper-parameter settings of the l values in Eq.(3). Fig. 7 presents valuable evidence for choosing 0.2 as an appropriate value for the parameter l. The sensitivity of pruning channels for VGG-16 on CIFAR-10 was examined with minimum MV values, summed IoU scores, and maximum MV values. Fig. 8 shows the comparison of different IoU measured criteria and a reduction in the accuracy of different convolution layers, differentiating all three methods. Our proposed method (MV) votes for the highest IoU scores, compares these scores among all examples, and assigns a voting score to compute a measure of relevance that identifies the most critical channels. It only votes for a channel when all instances agree. Fig. 8a shows how the minimum voting scores indicate the most crucial channels. Pruning by smallest voting scores yields better performance than pruning by largest voting scores. As shown in Fig. 8c, pruning channels with the maximum MV values cause the accuracy to drop quickly as the pruning compression rate increases. However, from the comparison between pruning with minimum voting scores and the minimum IoU scores summed values, we can see that the accuracy of a pruned network with minimum MV scores adequately evaluates channel importance and demonstrates the best performance.

In each convolutional layer, the filters' channels with the smallest voting scores are pruned; consequently, filters and their corresponding channels on a batch normalization layer are also eliminated. After pruning the unimportant filters, we were able to minimize the pruning impact by applying our kernel estimation method. When unimportant filters are discarded, a new model with thinner filters is created. The weights of the modified layers, as well as the non-pruned layers, were transferred to the new model. After pruning all layers, a final fine-tuning for the whole pruned model was performed to recover the

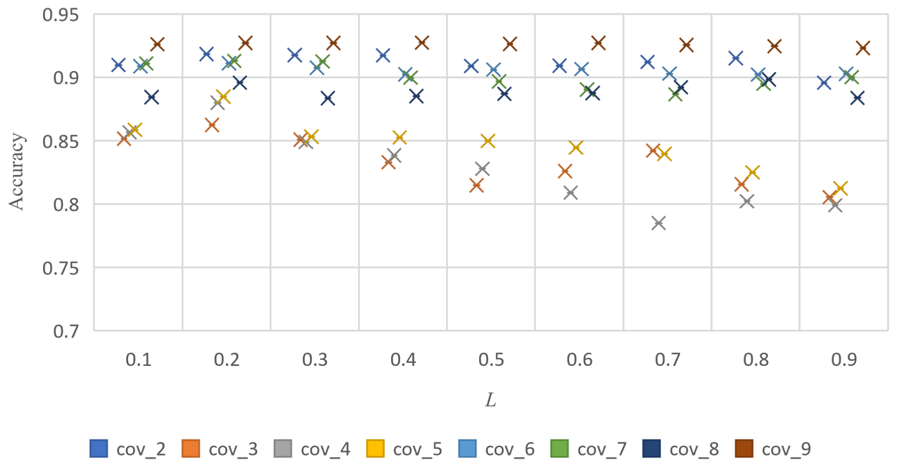
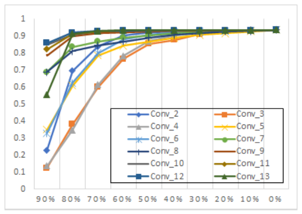
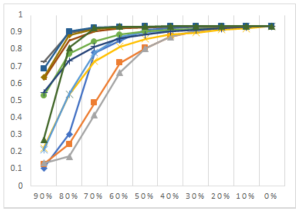


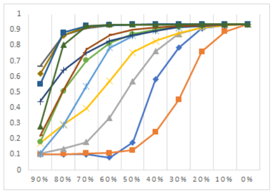
Fig. 7: Different settings to determine the optimal l value in Eq.(3) at different layers of VGG16 on CIFAR-10.



(a) Minimum MV values



(b) Minimum Summed *IOU* scores



(c) Maximum MV values

Fig. 8: Comparison of IoU pruning selection criteria. (a) and (c) compare our MV of IoU pruning selection criteria when pruning the lowest and highest MV scores. (b) Pruning filters based on assuming IoU scores.

overall dropped accuracy. During fine-tuning, the stochastic gradient descent optimizer was used, where each batch contained 32 randomly shuffled images. A data augmentation technique was applied using simple transformations such as flipping images horizontally. The entire network was pruned layer-by-layer. For both CIFAR-10 and CUB-200 datasets, we fine-tuned the pruned models for 40 epochs with a constant learning rate of 10^{-3} with a momentum of 0.9; a weight decay of 0.0005 was used. This was performed at the last round. For the ImageNet dataset, images were resized to 256×256, after which we randomly cropped them to 224×224. The pruned models were fine-tuned for only 20 epochs to reduce training time, where the learning rate changes from 10^{-3} to 10^{-5}. Other parameters were kept the same.

Compression Ratio

Deciding the number of filters which must be pruned from each layer as well as the best pruning ratios for different layers is a challenging task ([36]). It is also challenging to determine layer importance due to the fact that the performance of a CNN model is susceptible to specific layers and different layers have varying degrees of filter-level redundancy. Thus, we applied a fixed compression ratio to all layers in the pruned model for simplicity. In our experiment, we applied three different compression ratios: pruned-70, pruned-50, and pruned-30, where 70%, 50%, 30% of filters are preserved in each convolutional layer respectively. Reducing the complexity of models with larger compression ratios while maintaining their powerful performance is always desirable, as the inference speed is essential for some real-world applications. For example, the model used for self-driving vehicles must return fast predictions for safety purposes. Thus, the FLOPs of this kind of model should be reduced to fulfil the standard requirements.

4.1 VGG16 on CIFAR-10

In this section, we evaluated the performance of the proposed method on the most popular deep convolutional network: VGG-16 ([48]), a CNN architecture for large-scale image recognition proposed by Simonyan et al., initially designed for the ImageNet dataset. VGG-16 was modified by [34] to fit the CIFAR-10 dataset, achieving state-of-the-art results. VGG16 on CIFAR-10 consists of thirteen convolutional layers with a filter size of 3×3 with a stride of 1, and a pooling region of 2×2 without overlap. This is followed by two fully-connected layers, with the last layer consisting of 10 neurons. Due to the smaller input size, the dimensions of the fully-connected layers are shrunk, significantly reducing the number of parameters. Here, we adopted the model described in ([34]), adding a batch normalization layer ([22]) as well as a dropout layer ([50]) after each convolutional layer. The detailed architecture of the CNN model is presented in Table 1.

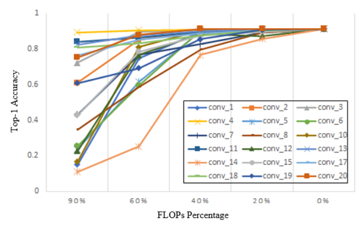
Table 2 shows the results of the pruned models for VGG-16 on CIFAR-10, VGG16-pruned-70, VGG16-pruned-50, and VGG16-pruned-30, in which 70%, 50%, and 30% of filters are preserved respectively in each convolutional layer. This also means that we assigned constant compression ratios of 30%, 50%, and 70%, respectively, for all layers. Fig. 14a shows that the convolutional layers with 512 feature maps have less impact on the dropping of model accuracy, as they can be pruned up to 70% without reducing the original accuracy. One definite explanation is that small dimensions of feature maps do not indicate meaningful spatial features for these convolutional layers. Our kernel estimation method can recover the pruning effect and help us safely prune the majority of the filters of such layers, see Fig. 14b. We observe that the first few layers have stronger negative effects and more synergistic filters compared with higher hidden layers due to hierarchically learned representations of deep networks. Therefore, an effective pruning method, as well as the reduction of FLOPs, mostly relies on the layer where pruning is applied within the network.

Table 1: VGG-16 on CIFAR-10 and three different pruned models. The number of remaining feature maps and the reduced percentage of FLOPs from each pruned model are shown.

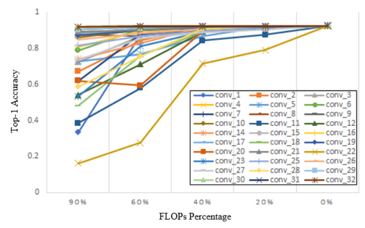
	VGG-16				VGG-16-pruned-A		VGG-16-pruned-50/B		VGG-16-pruned-C ([29])	
layer type	w_i x h_i	#Maps	#FLOP	#Params	#Maps	pruned%	#Maps	pruned%	#Maps	pruned%
Conv 2	32*32	64	3.80E+07	3.7E+04	38	40%	32	50%	32	50%
Conv 3	16*16	128	1.90E+07	7.4E+04	102	20%	64	50%	128	0%
Conv 4	16*16	128	3.80E+07	1.5E+05	102	20%	64	50%	128	0%
Conv 5	8*8	256	1.90E+07	2.9E+05	230	10%	128	50%	256	0%
Conv 6	8*8	256	3.80E+07	5.9E+05	205	20%	128	50%	256	0%
Conv 7	8*8	256	3.80E+07	5.9E+05	205	20%	128	50%	256	0%
Conv 8	4*4	512	1.90E+07	1.2E+06	410	20%	256	50%	256	50%
Conv 9	4*4	512	3.80E+07	2.4E+06	256	50%	256	50%	128	75%
Conv 10	4*4	512	3.80E+07	2.4E+06	205	60%	256	50%	128	75%
Conv 11	2*2	512	9.40E+06	2.4E+06	205	60%	256	50%	128	75%
Conv 12	2*2	512	9.40E+06	2.4E+06	205	60%	256	50%	128	75%
Conv 13	2*2	512	9.40E+06	2.4E+06	205	60%	256	50%	128	75%
FC	1	512	2.60E+05	2.6E+05	512	0%	512	0%	512	0%
FC	1	10	5.10E+03	5.1E+03	10	0%	10	0%	10	0%
Total			3.13E+08	1.47E+07		43.04%		50%		53.03%

Table 2: Performance of pruning VGG16 on CIFAR-10 using different pruning rates. The test accuracy is reported.

Model	Error(%)	#FLOPs	Pruned
VGG-16 ([48])	6.41	3.13E+08	-
VGG-16-pruned-70	**6.18**	2.20E+08	30%
VGG-16-pruned-A	6.37	1.37E+08	43.04%
VGG-16-pruned-50/B	7.27	1.57E+08	50%
VGG-16-pruned-C	7.00	1.66E+08	53.03%
VGG-16-pruned-30	9.10	9.42E+07	70%
VGG-16-pruned-A scratch-train	8.57	1.37E+08	43.04%
VGG-16-pruned-50 scratch-train	9.79	1.57E+08	50%



(a) ResNet-20



(b) ResNet-32

Fig. 9: Layer-wise pruning of ResNet-20/32 on CIFAR-10.

This observation motivated us to assign different compression rates to different layers based on our ablation study; thus, if the layer shows more sensitivity to pruning, the compression ratio decreases. The network pruning ratio is 43.04% and 53.03% for VGG16-pruned-A and VGG16-pruned-C, respectively.

Table 3: Performance of pruning ResNet-20/32 on CIFAR-10 using different pruning rates. The test accuracy is reported.

Model	Error(%)	#FLOPs	Pruned
ResNet-20 ([17])	8.75	8.16E+07	-
ResNet-20-pruned-70	8.87	5.71E+07	30%
ResNet-20-pruned-50	10.98	4.08E+07	50%
ResNet-20-pruned-30	14.67	2.45E+07	70%
ResNet-32 ([17])	7.51	1.38E+08	-
ResNet-32-pruned-70	7.53	9.68E+07	30%
ResNet-32-pruned-50	10.75	6.91E+07	50%
ResNet-32-pruned-30	13.73	4.15E+07	70%

The detail specification of such pruned models is presented in Table 1. Moreover, for both VGG16-pruned-A and VGG16-pruned-70 pruned models, we achieved 43.04% and 30% FLOP reduction, respectively, with no drop in the original model accuracy. We also trained a model from scratch with the same architecture as VGG-16-pruned-A and VGG-16-pruned-50, which allowed us to obtain the baseline accuracies for such networks and differentiate between training from scratch and pruning. Table 2 shows that VGG-16-pruned-A scratch-train and VGG-16-pruned-50 scratch-train present considerably worse results than our pruned models. Thus, a model may need a certain level of redundancy during model training to guarantee excellent quality performance. Hence, decreasing a model's size after training can be an effective solution.

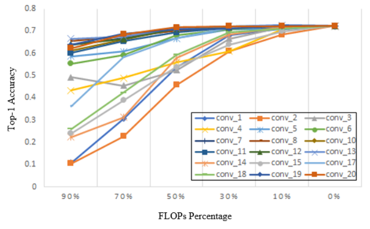
4.2 ResNet-20/ResNet-32 on CIFAR-10

The performance of our pruning method was also evaluated on the famous CNN architecture ResNet ([17]). The ResNets for CIFAR-10 have three stages of residual blocks, where 32×32, 16×16, and 8×8 are the sizes of their corresponding output feature maps. Each stage has an equal number of residual blocks. Identity shortcuts are directly used when the input and output comprise the same dimensions. When a feature map's size is down-sampled, the shortcut performs by 1×1 kernels. This procedure overcomes the issue which occurs when the shortcuts go across feature maps of two different sizes. As the input and output feature map sizes of this convolutional layer are different, we skipped those layers and pruned the remaining layers at each stage.

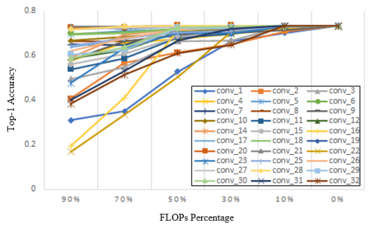
To investigate the abilities of our proposed method, we chose ResNet-20 and ResNet-32 to represent the ResNet family, which have the same designs, except for the number of layers and the depth of the network. In the initial experiment, we started with a trained Keras implementation with classification errors of 8.75% and 7.51% on the test set for ResNet-20 and ResNet-32, respectively. Fig. 9 shows the classification accuracy of ResNet-20 (left) and ResNet-32 (right) after pruning each layer using our proposed method. Unlike VGG-16, ResNet is more compact, and due to its reduced redundancy, pruning a large number of channels appears to be more challenging. It can be seen that some layers were more sensitive to pruning, such as layers 11 and 22 in ResNet-32 and 14 in ResNet-20. Similar to VGG-16, we found that deeper layers of the ResNet architecture were less sensitive to pruning than those in the earlier layers of the network.

Similar to VGG-16, we iteratively pruned ResNet-20/32 from the first block to the last. In the batch normalization layer, the channels corresponding to the pruned filters were also pruned. Within pruning iterations, we estimated new kernels using our proposed method and then applied final fine-tuning with a fixed learning rate of 10^{-3}, which was performed at the last round. A horizontal flip was applied for data augmentation. We pruned both models using three different compression rates, pruned-70, pruned-50, and pruned-30, where 70%, 50%, 30% of filters were preserved respectively in each block. Due to reduced redundancy and the more compact nature of the ResNet architecture, pruning a large number of filters is more challenging and affects the overall accuracy. However, we were able to prune 30% of both models with only 0.12% accuracy decrease on ResNet-20 and 0.02% accuracy decrease on ResNet-32. The results are presented in Table 3.

4.3 ResNet-20/ResNet-32 on CUB-200



(a) ResNet-20



(b) ResNet-32

Fig. 10: Layer-wise pruning of ResNet 20/32 on CUB-200.

We also evaluated the performance of the proposed method on the larger data: the CUB-200 dataset on ResNet architecture. Our focus was to reduce the number of convolutional channels in each filer and the approximated floating-point operations (FLOPs). In the first experiment, we began with an ImageNet pre-trained model to fine-tune the ResNet models, as fine-tuning is a common approach adopted in many recognition tasks, and the CUB-200 examples are not large enough to train such models from scratch; it seemed that the models overfit the training data and had poor generalization performance. In our implementation, a horizontal flip was applied for data augmentation. The Adam optimizer ([23]) was used, where each batch contains 32 randomly shuffled images. For

Table 4: Performance of pruning ResNet-20/32 on CUB-200 using different pruning rates. The test accuracy is reported.

Model	Error(%)	#FLOPs	Pruned
ResNet-20 ([17])	27.67	$2.95E+08$	-
ResNet-20-pruned-70	29.69	$2.07E+08$	30%
ResNet-20-pruned-50	32.95	$1.48E+08$	50%
ResNet-20-pruned-30	39.99	$8.86E+07$	70%
ResNet-32 ([17])	26.68	$1.82E+09$	-
ResNet-32-pruned-70	29.08	$1.27E+09$	30%
ResNet-32-pruned-50	35.48	$9.08E+08$	50%
ResNet-32-pruned-30	41.63	$5.45E+08$	70%

our experiment, we started with a learning rate of 0.001, a fixed momentum of 0.9, and a fixed weight decay of 0.0005. The learning rate was scheduled to be reduced after every 40 epoch.

We used our proposed method to prune unimportant filters of both ResNet-20 and ResNet-32 and convert a large model into a smaller one with a minor drop in model accuracy. Similar to CIFAR-10, we pruned both ResNet-20 and ResNet-32 models on CUB-200 using three different compression rates, pruned-70, pruned-50, and pruned-30, where 70%, 50%, 30% of filters are preserved in each block respectively. The results are shown in Table 4. Due to the small number of training examples, the accuracy of the pruning model ultimately could not be improved, and the final fine-tuning attains a limited contribution to completely recovering the accuracy of the reference model. Another issue is that the ResNet architecture has little redundancy, so pruning a large number of filters is more challenging and affects the overall accuracy.

Fig. 10 shows the classification accuracy of both ResNet-20 and ResNet-32 on CUB-200 after pruning each layer using our proposed method. Despite the compactness and reduced redundancy of ResNet models, our pruning method was able to compute a measure of relevance that identifies the less critical filters and prunes them accordingly, as illustrated in Fig. 10. In other words, it can be observed that our pruning method removes the unimportant part which does not appreciably contribute much to the final model performance. Fig. 10 also shows that many layers were minimally affected by the pruning of their unnecessary parts, especially when using a low compression rate. This also explains why the overall accuracy was not recovered completely when a small set of training examples was used for the final fine-tuning.

4.4 ResNet-50/ResNet-101 on ImageNet

To thoroughly validate our proposed method, we also evaluated its performance on a large-scale dataset: the ImageNet data ([45]) with ResNet-50 and ResNet-101 ([17]). In the initial experiment, we started with a pre-trained models in

Table 5: Performance of pruning ResNet-50 on ImageNet using different pruning rates. The classification errors (Top-1/5 Err.) are reported on the standard validation set, using the single central crop.

Model	Top-1 Err.(%)	Top-5 Err.(%)	#FLOPs	Pruned
ResNet-50 ([17])	25.1	7.9	3.86E+09	-
ResNet-50-pruned-70	25.34	7.93	2.44E+09	30%
ResNet-50-pruned-50	26.41	8.17	1.70E+09	50%
ResNet-50-pruned-30	30.95	10.99	1.10E+09	70%
ResNet-101 ([17])	23.6	7.2	7.6E+09	-
ResNet-101-pruned-70	25.28	7.25	4.80E+09	30%
ResNet-101-pruned-50	25.91	8.06	3.35E+09	50%

Table 6: Comparison among several state-of-the-art pruning methods on ResNet-50 and ImageNet. The Acc.↓ (%) denotes the accuracy drop between the baseline model and the pruned model.

Model	Top-1 Acc.↓ (%)	Top-5 Acc.↓ (%)	#Param.	#FLOPs	Pruned
C-SGD-70 ([12])	0.06	0.1	16.94E+06	2.44E+09	30%
FPGM ([20])	0.56	0.24	14.74E+06	2.55E+09	37.5%
SCOP ([51])	0.20	0.08	13.57E+06	1.85E+09	45.3%
C-SGD-50 ([12])	0.79	0.47	12.38E+06	1.70E+09	50%
AutoPruner ([37])	1.39	0.72	12.38E+06	1.70E+09	50%
DCP ([64])	1.06	0.61	12.38E+06	1.70E+09	50%
ThinNet-70 ([36])	1.27	0.09	16.94E+06	2.44E+09	30%
HRank ([30])	1.17	0.54	16.15E+06	2.30E+09	43.7%
ThinNet-50 ([36])	3.27	1.21	12.38E+06	1.70E+09	50%
SSR-L2 ([32])	3.65	2.11	12.38E+06	1.70E+09	50%
Weights Sum ([29])	4.31	2.42	12.38E+06	1.70E+09	50%
APoZ ([21])	4.25	2.41	12.38E+06	1.70E+09	50%
ResNet-50-pruned-70	0.24	0.03	16.94E+06	2.44E+09	30%
ResNet-50-pruned-50	1.31	0.27	12.38E+06	1.70E+09	50%

Keras Applications[5], which achieved classification errors of 25.1% in top-1 error and 7.9% in top-5 on ResNet-50 and 23.6% in top-1 error and 7.2% in top-5 on ResNet-101. The classification errors are reported on the standard validation set using the single central crop. The resized images are center-cropped to 224×224. To prune the ResNet-50, we followed the setting of ThiNet ([36]), where the first two layers of each residual block are pruned; this leaves the output block and the projection shortcuts consistent. The entire network was pruned from block 2a to 5c iteratively. The corresponding channels in the batch normalization layer were also pruned. Within pruning iterations, new kernels were estimated using the proposed method. After pruning, a final fine-tuning was performed

[5] https://keras.io/api/applications/

for 20 epochs at the last round. The model was pruned using three different compression rates, pruned-70, pruned-50, and pruned-30, where 70%, 50%, 30% of filters were preserved respectively in each targeted block. We were able to prune 30% of ResNet-50 with a 0.24% reduction in the original model's top-1 error and with only a 0.03% drop in the top-5 error. Similarly, we iteratively pruned ResNet-101 from the first block to the last using two compression rates, pruned-70 and pruned-50, where 70%, 50% of filters were preserved respectively in each targeted block. We achieved a 30% FLOP reduction, with little drop in the original model accuracy. The results are presented in Table 5, showing that significant performance degradation arises with an increased pruning rate. A much smaller model can be obtained at the cost of further accuracy reduction.

We compared our proposed approach with other state-of-the-art pruning methods. Table 6 presents the comparison results on ResNet-50 and ImageNet. For a fair comparison, most compared methods targeted the first and second layers of each residual block and adopted the same compression ratio of 0.7 and 0.5, where 70% and 50% of filters were preserved in each targeted layer respectively. In the first stage, we compared our proposed approach with other filter level pruning methods including ThinNet ([36]), HRank ([30]), SSR-L2 ([32]), Weights Sum ([29]), and APoZ ([21]). These methods evaluate the importance of intermediate units and prune them accordingly. Because the pruning pipeline of these filter level pruning methods is the same, it is a fair comparison, and our proposed approach has achieved better results. In top-1 error, the proposed approach surpasses the baseline methods ThinNet, SSR-L2, Weights Sum, and APoZ by 1.96%, 2.34%, 3% and 2.94% respectively, representing significant improvements on the ImageNet with a compression ratio of 0.5. In top-5 error, our method also outperforms ThinNet, SSR-L2, Weights Sum, and APoZ by 0.94%, 1.84%, 2.15% and 2.14% respectively, with the same compression ratio. Furthermore, our pruned model has shown better performance than HRank ([30]), reducing an extra 6.3% of its FLOPs. In other words, with more FLOPs reduction, our method surpasses the HRank by 0.27% in top-5 error. The relationship between a semantic concept and individual hidden unit representations is directly considered in our proposed approach, which can adaptively determine the function of individual CNN filters to deliver essential information and prune the lower impact filters on the global output.

We also compared our proposed approach with several training-based pruning methods including C-SGD ([12]), FPGM ([20]), SCOP ([51]), AutoPruner ([37]) and DCP ([64]). The results are summarized in Table 6. The pruning procedure of these methods is considered a single end-to-end trainable system, where evaluating channels' importance, pruning channels, and fine-tuning are performed jointly during an iterative training procedure. Although they achieve remarkable accuracy, their computational costs and memory requirements are increased as modern GPUs do not benefit from sparse convolutions. Pruning procedures based on iterative training often change the optimization function and even introduce many hyper-parameters, making the training more challenging. However, our pruned networks show a similar reduction in FLOPs with

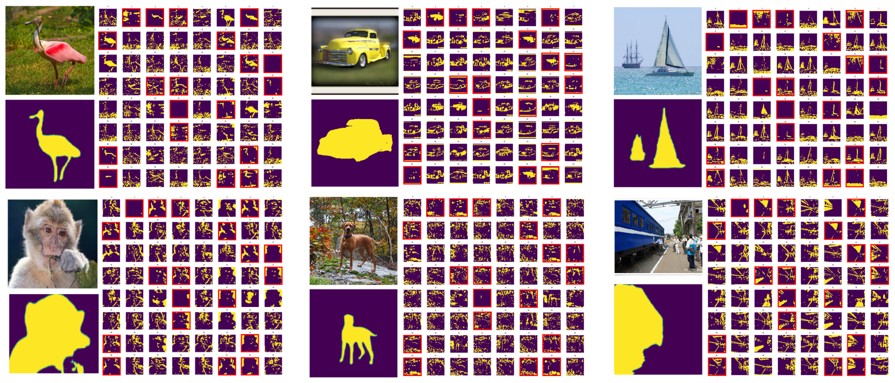


Fig. 11: Six different input images of ImageNet dataset and their semantic segmentation and visualization of feature maps binary segmentation of ResNet-50 first block (i.e. res2a). Feature maps with red borders are the eliminated channels in our pruned model (this figure is best viewed in color).

comparable accuracy. Note that we adopt the same compression ratios and target the same layer for ResNet-50. When comparing with AutoPruner ([37]), our method achieves a 0.08% increase in the top-1 error and a 0.45% increase in the top-5 error with similar FLOPs. Compared to the iteratively optimized pruning methods, our approach has several advantages. It is capable of pruning any CNN using a single forward pass without the need for a training process or back-propagation. For the forward pass, we only select a few images from each category to form our evaluation set used to find the optimal channel subset. Consequently, the small number of instances are used to find the optimal filter subset via Algorithm 1. After pruning all layers, we only fine-tune the pruned models once for a reasonable number of epochs to reduce training time.

4.5 Feature Map Visualization

We carried out a visual assessment to provide a convincing analysis of the motivation to fundamentally rely on evaluating the alignment between a semantic concept and individual hidden unit representations. Fig. 11 shows different examples from the ImageNet dataset with their semantic segmentation and visualization of feature maps binary segmentation for the first layer of the ResNet-50 first block (i.e. res2a). The channels with red borders correspond to the channels selected to be eliminated in our pruned model when the compression rate is set to 30%. Fig. 11 demonstrates that the selected channels are less informative (i.e. channels with title 2, 6, 23, 31, 35 and 56) compared to other channels that highly correlated to the region of an object class across different images. For instance, the channel's visualization with title 62 reveals that this particular feature map focuses on background rather than foreground objects. This demonstrates that our proposed method determines individual CNN filters' function to deliver essential information with strong discriminative power for the model. It can also be observed that non-pruned channels are related to the concept of an object, which closely matches the semantic segmentation of an object (e.g., pickup truck, spoonbill, and schooner). The apparent commonality among these channels is that representations are object classes appropriate with diverse visual appearances. Another remarkable appearance is that many channels represent parts of the object.

4.6 Comparison with Filter Selection Criteria

Implementation Details Classification performance was used to evaluate the impact of our filter selection criteria. An ablation study provides a scheme to evaluate the effectiveness of measuring filters' importance quantitatively. This procedure typically refers to the removal of some parts of the model and the study of its performance, as crucial filters capture meaningful information and contribute substantially to the model's final performance. We ablated non-informative filters by forcing their activation to be zero and computed the classification accuracy on the test-set. Quantifying the influence of the ablation on the classification performance allows for an impartial evaluation in order to distinguish the essential filters in a CNN and measure their importance, allowing for layer-wise comparison. This method not only enables the evaluation of filters' importance but can also detect the unimportant, redundant filters which can be safely pruned.

Different Filter Selection Criteria Several criteria have been developed to estimate the importance of a feature map or convolutional kernel in the CNNs. To evaluate the effectiveness of our evaluation criterion, we compared our filter selection method with several baseline methods, briefly explained as follows:

- **Random.** Filters are randomly ablated.
- **Weights sum ([29]).** Filters (i) with lowest absolute weights sum values are ablated: $\psi_i = \sum_i \mid \omega(i,:,:,:) \mid$.
- **Activation mean ([29]).** $\psi_i = \frac{1}{N} \sum mean(\tau(i,:,:))$, where τ is the activation values for filter i, and N denotes the size of the data. The feature maps with weak patterns and their corresponding filters and kernels are ablated.
- **Mean gradient ([33]).** $\psi_i = \frac{1}{N} \sum mean(\kappa(i,:,:))$, where κ is the calculated gradient for each filter channel i, and N denotes the size of the data.
- **LRP (Layer-wise Relevance Propagation) ([56]).** The LRP of each channel i is calculated as its importance score $\psi_i = \frac{1}{N} \sum \sum (LRP(i,:,:))$, where N denotes the size of the data; LRP calculates the summed relevance quantity of each channel in the network to the overall classification score, decomposing a classification decision into contributions for each channel.

All these baseline methods consider the higher scores as more critical, which is driven by the intuition that unimportant activation and filters have no influential outputs to the final prediction of a model.

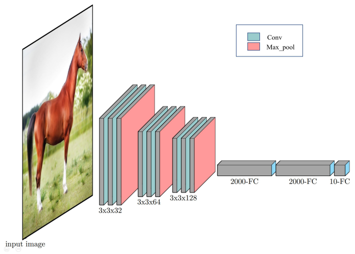
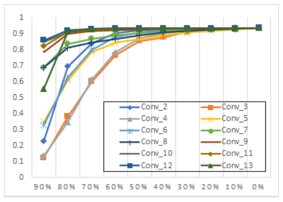
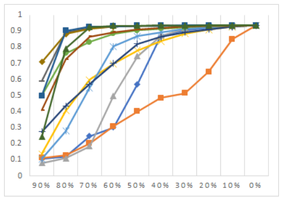


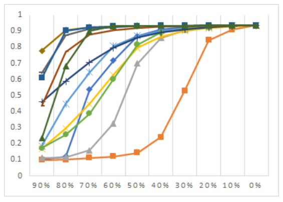
Fig. 12: The architecture of the CNN model.



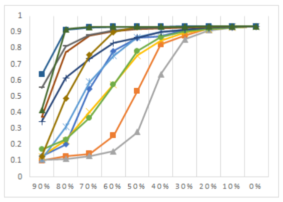
(a) Proposed Method



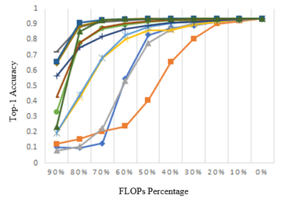
(b) Random



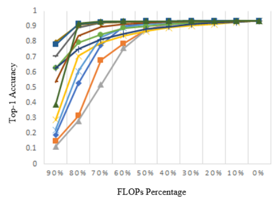
(c) Mean-Mean ([29])



(d) Weights Sum ([29])



(e) Mean Gradient ([33])



(f) LRP ([56])

Fig. 13: Comparison of different pruning methods for VGG-16 on CIFAR-10.

Table 7: Overall results of layer-wise pruning utilizing different filter selection criteria. The test accuracy is reported after ablating the unimportant filters. The results were conducted on a NVIDIA GeForce GTX 1080 GPU to prune the VGG-16 model on CIFAR-10 with a compression ratio of 0.5, where 50% of filters were preserved after pruning. For conv_2, the running time of identifying important filters is also reported (the model's forward time is approximately 9s).

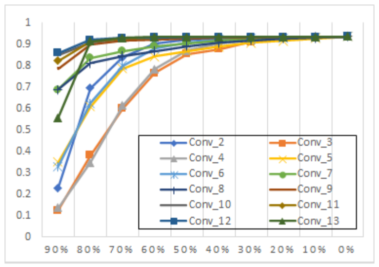
	Conv_2	Conv_3	Conv_4	Conv_5	Conv_6	Conv_7	Conv_8	Conv_9	Conv_10	Conv_11	Conv_12	Conv_13
Random	0.8290 (0.073ms)	0.408	0.7776	0.8592	0.8824	0.895	0.8605	0.9194	0.9307	0.931	0.932	0.9341
Weights-sum ([29])	0.8630 (2.015ms)	0.5333	0.2785	0.7558	0.8681	0.7834	0.8655	0.92	0.9218	0.9321	0.934	0.9343
Mean-mean ([29])	0.8686 (4.5s)	0.1433	0.6995	0.795	0.8707	0.8157	0.8587	0.9198	0.9323	0.9314	0.932	0.9339
Mean Gradient ([33])	0.5706 (11.1s)	0.4017	0.7448	0.7795	0.868	0.9052	0.821	0.9095	0.9313	0.9313	0.9331	0.9345
LRP ([56])	0.9186 (13.2s)	0.88	0.8698	0.8675	0.9067	0.9005	0.8815	0.9234	0.9316	0.9328	0.9332	0.9342
Our method	0.9199 (10.9s)	0.864	0.8784	0.8686	0.915	0.9059	0.8907	0.9286	0.932	0.9347	0.9346	0.9349

Overall Performance Comparison Table 7 and Table 8 summarize the comparison results for two network architectures: our proposed CNN architecture and VGG-16 with different pruning criteria on CIFAR-10. Our CNN architecture

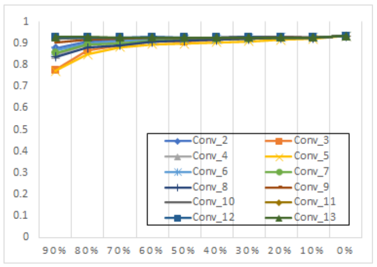
Table 8: Overall results of layer-wise pruning utilizing different filter selection criteria. The results are reported on our small CNN model on CIFAR-10 with a compression ratio of 0.5, where 50% of filters were preserved after pruning. The test accuracy is reported after ablating the unimportant filters.

	Conv_2	Conv_3	Conv_4	Conv_5	Conv_6
Random	0.7935	0.3556	0.6843	0.7096	0.7975
Weights-sum ([29])	0.8027	0.4500	0.7091	0.6852	0.7645
Mean-mean ([29])	0.7790	0.3761	0.7465	0.7133	0.7855
Mean Gradient ([33])	0.8471	0.2673	0.6149	0.6214	0.7437
LRP ([56])	0.8475	0.6122	0.7729	0.7466	0.8091
Our method (Sum-IoU)	0.8422	0.6709	0.7515	0.6988	0.8152
Our method (MV)	0.8599	0.6935	0.7858	0.7201	0.8239

consists of three convolutional blocks, where each block has two convolutional layers with a filter size of 3 x 3 with 32 kernels in the first block, 64 kernels in the second block, and 128 kernels in the third block. Each block ends with a max-pooling layer followed by three fully-connected layers consisting of 2,000, 2,000, and 10 neurons respectively. A standard Relu activation function was utilized. The detailed architecture of the CNN model is presented in Fig. 12. Using the ablation approach, the importance of the filters was evaluated by employing different selection criteria in a fully trained model. We compared our method with such baselines, and the results are reported in Fig. 13, where different compression rates are used. Table 7 and Table 8 also show different methods to measure the importance of filters, using fixed compression ratio= 0.5, where 50% of channels are preserved after pruning. The tables show layer-wise results for each layer, where we ablated layer-by-layer and calculated the accuracy for each layer separately. For random selection criteria, the mean value of three runs are reported.



(a) Pruning smallest value



(b) Prune and re-estimate new kernels

Fig. 14: Layer-wise pruning of VGG-16 on CIFAR-10. (a) Pruning filters with the lowest MV scores and their corresponding test accuracies on CIFAR-10. (b) Prune and estimate new kernels for each single layer of VGG-16 on CIFAR-10.

Table 9: Comparison of layer-by-layer pruning with fine-tuning (FT) and kernels estimation (KE) using the VGG-16 model on CIFAR-10 using 200, 500, and 1000 training examples. These results were conducted on NVIDIA GeForce GTX 1080 GPU. The test accuracy is reported before and after performing fine-tuning and kernels estimation. For conv13, the running time of both procedures is also reported.

	Conv_2	Conv_3	Conv_4	Conv_5	Conv_6	Conv_7	Conv_8	Conv_9	Conv_10	Conv_11	Conv_12	Conv_13
200 samples												
Before KE	0.9173	0.8536	0.8151	0.8084	0.8482	0.8219	0.7546	0.7988	0.7839	0.7769	0.8028	0.7707
After KE	0.9338	0.9265	0.9195	0.8923	0.8746	0.8619	0.8151	0.8028	0.7971	0.7958	0.7954	**0.7941** (5.86s)
Before FT	0.9173	0.8254	0.7362	0.705	0.6991	0.671	0.5714	0.6128	0.6163	0.5881	0.6001	0.6091
After FT	0.9203	0.8873	0.8499	0.7681	0.7401	0.6926	0.632	0.6317	0.6148	0.6034	0.6074	0.6044 (22.60s)
500 samples												
Before KE	0.9173	0.8498	0.8232	0.82	0.8494	0.8279	0.7699	0.8077	0.8055	0.806	0.8171	0.7979
After KE	0.9332	0.9271	0.9196	0.8942	0.8771	0.8647	0.829	0.8195	0.8157	0.8161	0.8145	**0.8138** (6.81s)
Before FT	0.9173	0.826	0.7395	0.7143	0.7102	0.7026	0.6341	0.6759	0.6765	0.6663	0.6609	0.6751
After FT	0.9239	0.8918	0.8559	0.7859	0.7647	0.7326	0.6817	0.6848	0.6808	0.6699	0.6725	0.6723 (22.74s)
1000 samples												
Before KE	0.9173	0.851	0.8165	0.8203	0.8556	0.8369	0.7935	0.8177	0.8127	0.8147	0.8238	0.809
After KE	0.9331	0.9275	0.9202	0.8958	0.8811	0.8685	0.8404	0.8303	0.8247	0.8245	0.8243	**0.8227** (8.81s)
Before FT	0.9173	0.8325	0.7507	0.7173	0.7499	0.7247	0.6506	0.6939	0.6963	0.6858	0.6888	0.6887
After FT	0.9235	0.8969	0.8622	0.8007	0.7777	0.7472	0.7011	0.709	0.6952	0.6915	0.6904	0.6948 (23.34s)

Our ablation study has shown that for both architectures, MV achieves higher classification performance when compared with other baselines. This demonstrates the robustness of our proposed method in identifying the essential filters. With VGG16, as shown in Table 7, our pruning method achieved the best results when using a compression ratio of 0.5 for each layer of the reference model. Fig. 13a demonstrates that the performance of the pruned model with our proposed method is relatively consistent, as model FLOPs reduce, especially when reaching a reduction of 60%. Our method delivers the best result among all baselines. On the other hand, it has less impact on the dropping of model accuracy while reducing FLOPs compared with the LRP-based method, which is also inspired by neural network interpretability. These results indicate an interesting potential research direction of combining the two fields of interpretability and model compression research.

Interestingly, ablating filters with random selection showed that the first few layers had stronger negative effects and more synergistic filters compared with higher hidden layers. It was also observed that the higher hidden layers were significantly redundant and more class-specific. This observation is consistent with a previous theoretical proposal by [44]. One reasonable explanation is that the neural networks hierarchically learn representations. Hence, the first layers are not relevant to a specific object. Still, they build feature representations of all input images that are joined to form more relevant object features in the later layers. By ablating these fundamental features, deeper layers fail to produce class-specific features and have a more negative impact on overall accuracy.

Although random selection is neither robust nor applicable in practice ([36]), it offers insight and demonstrates that the detection of principal filters is a critical approach when pruning redundant filters. The experiment empirically confirms that our importance method is sufficient, given that ablating filters with low values in the layers had a negligible impact on the overall accuracy compared to all baselines. As shown in Table 7 and Table 8, the experimental results for both networks show that the method substantially outperformed the baselines. Our proposed method to measure filters' importance helps not only to remove redundant nodes and compress the neuron network, but also to understand their inter-relationships and how said filters impact the model. The experiment confirms that selecting the right criteria to evaluate filters' importance throughout all layers can guarantee a successful pruning approach. In Table 7, we reported the running time of different filter selection approaches. The running speed of data-driven methods relies on model inference speed and dataset size. On a NVIDIA GeForce GTX 1080 GPU, it takes 9.186s to apply forward passing through the optimized VGG-16 model on CIFAR-10. Our proposed approach takes 10.9s to identify the important filters, which is faster than some competitive methods. For ResNet-50 on ImageNet, the time cost to estimate IoU scores and MV values is 422.3s of ResNet-50 first block (i.e. res2a).

4.7 Comparison of Kernel Estimation (KE) vs. Fine-Tuning (FT)

In order to gather conclusive evidence to evaluate the effectiveness of our kernels estimation (KE) method, an experiment based on the iterative layer-wise pruning process was carried out using the VGG-16 model on CIFAR-10. Thus, we were able to fairly compare our KE method with the standard fine-tuning (FT) procedure that is performed to preserve the original accuracy or recover the damage that might occur during the compression phase. After pruning each layer with a fixed compression ratio of 0.5, our KE method, as well as the FT procedure, were applied to improve the performance degradation. The experiments were performed on the training set using three different numbers of trained examples, i.e. 200, 500, and 1,000. We estimated new kernels and performed the fine-tuning using these settings with the same amount of iteration. The comparative results are shown in Table 9, which demonstrates that with a small number of examples, KE performed much better with lower run-time requirements. The approximate time needed to complete the process of each method is shown in Table 9. KE achieved higher classification performance, especially when the whole network was cumulatively pruned. Our experiment has shown that both KE and FT improved the accuracy after pruning each layer. These results demonstrate the necessity of adopting such steps to recover model accuracy which has been iteratively damaged. However, even though the strategy of iterative pruning with fine-tuning is the typical setting for CNN pruning, it incurs expensive computation costs, significant inference time, and high storage requirements. Such an iterative process requires considerable inference costs, including those related to the creation of a new model, loading of parameters, and retraining of the whole model.

5 Conclusion

In this paper, we have proposed a novel framework based on an effective channel-level pruning method, considering the power of novel neural network interpretability in evaluating the importance of feature maps. Based on the discriminative ability of interpretable latent representations, a majority voting technique is proposed to compare the degree of alignment values among filters and assign a voting score to quantitatively evaluate the importance of feature maps. The experimental results show the effectiveness of our filter selection criteria, which outperforms all other pruning criteria. It also allows for the identification of layers which are robust or sensitive to pruning, and this can be beneficial for further improving and understanding the architectures. We also propose a simple yet effective method to estimate new convolution kernels based on the remaining, crucial channels to accomplish effective CNN compression. The experimental results on CIFAR-10, CUB-200, and ImageNet (ILSVRC 2012) datasets demonstrate the effectiveness of our pruning framework in maintaining or even improving accuracy after removing unimportant filters. Our results also display the excellent performance of our proposed method. Moreover, our pruned model can be further pruned into even smaller models by adopting any existing model compression method. Our potential future work is to extend this framework and combine it with other pruning criteria to deeply explore the problem of CNN pruning from an interpretable perspective, aiming to link model compression and interpretability research.

Acknowledgement

This work is supported by EPSRC EP/N028139/1.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: Proceedings of the Symposium on Operating Systems Design and Implementation. pp. 265–283 (2016)
2. Arora, S., Ge, R., Neyshabur, B., Zhang, Y.: Stronger generalization bounds for deep nets via a compression approach. In: Proceedings of the International Conference on Machine Learning. pp. 254–263 (2018)
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Muller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7) (2015)
4. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6541–6549 (2017)
5. Bau, D., Zhu, J.Y., Strobelt, H., Zhou, B., Tenenbaum, J.B., Freeman, W.T., Torralba, A.: Gan dissection: Visualizing and understanding generative adversarial networks. In: Proceedings of the International Conference on Learning Representations (2019)

6. Cheng, Y., Wang, D., Zhou, P., Zhang, T.: Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Processing Magazine **35**(1), 126–136 (2018)
7. Chollet, F., et al.: Keras (2015), `https://github.com/fchollet/keras`
8. Denil, M., Shakibi, B., Dinh, L., Ranzato, M., De Freitas, N.: Predicting parameters in deep learning. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 2148–2156 (2013)
9. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 1269–1277 (2014)
10. Dhamdhere, K., Sundararajan, M., Yan, Q.: How important is a neuron? In: Proceedings of the International Conference on Learning Representations (2019)
11. Ding, H., Chen, K., Huo, Q.: Compressing cnn-dblstm models for ocr with teacher-student learning and tucker decomposition. Pattern Recognition **96**, 106957 (2019)
12. Ding, X., Ding, G., Guo, Y., Han, J.: Centripetal sgd for pruning very deep convolutional networks with complicated structure. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4943–4953 (2019)
13. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448 (2015)
14. Guillaumin, M., Küttel, D., Ferrari, V.: Imagenet auto-annotation with segmentation propagation. International Journal of Computer Vision **110**, 328–348 (2014)
15. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 1135–1143 (2015)
16. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal brain surgeon. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 164–171 (1993)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
18. He, T., Fan, Y., Qian, Y., Tan, T., Yu, K.: Reshaping deep neural network for fast decoding by node-pruning. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing. pp. 245–249 (2014)
19. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. In: Proceedings of the International Joint Conference on Artificial Intelligence. pp. 2234–2240 (2018)
20. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4340–4349 (2019)
21. Hu, H., Peng, R., Tai, Y.W., Tang, C.K.: Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. CoRR **abs/1607.03250** (2016)
22. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the International Conference on Machine Learning. pp. 448–456 (2015)
23. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (2015)
24. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical report, University of Toronto (2009), `https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf`

25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)
26. Lebedev, V., Lempitsky, V.: Fast convnets using group-wise brain damage. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2554–2564 (2016)
27. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (Nov 1998)
28. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 598–605 (1990)
29. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: Proceedings of the International Conference on Learning Representations (2017)
30. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1529–1538 (2020)
31. Lin, S., Ji, R., Chen, C., Tao, D., Luo, J.: Holistic cnn compression via low-rank decomposition with knowledge transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(12), 2889–2905 (2018)
32. Lin, S., Ji, R., Li, Y., Deng, C., Li, X.: Toward compact convnets via structure-sparsity regularized filter pruning. IEEE Transactions on Neural Networks and Learning Systems **31**(2), 574–588 (2019)
33. Liu, C., Wu, H.: Channel pruning based on mean gradient for accelerating convolutional neural networks. Signal Processing **156**, 84–91 (2019)
34. Liu, S., Deng, W.: Very deep convolutional neural network based image classification using small training sample size. In: Proceedings of the IAPR Asian Conference on Pattern Recognition. pp. 730–734 (2015)
35. Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.T., Sun, J.: Metapruning: Meta learning for automatic neural network channel pruning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3296–3305 (2019)
36. Luo, J., Zhang, H., Zhou, H., Xie, C., Wu, J., Lin, W.: Thinet: Pruning cnn filters for a thinner net. IEEE Transactions on Pattern Analysis and Machine Intelligence **41**(10), 2525–2538 (2019). https://doi.org/10.1109/TPAMI.2018.2858232
37. Luo, J.H., Wu, J.: Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. Pattern Recognition p. 107461 (2020)
38. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5188–5196 (2015)
39. Mariet, Z., Sra, S.: Diversity networks: Neural network compression using determinantal point processes. In: Proceedings of the International Conference on Learning Representations (2016)
40. Mocanu, D.C., Mocanu, E., Stone, P., Nguyen, P.H., Gibescu, M., Liotta, A.: Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature communications **9**(1), 2383 (2018)
41. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: Proceedings of the International Conference on Learning Representations (2017)
42. Na, S., Choe, Y.J., Lee, D.H., Kim, G.: Discovery of natural language concepts in individual units of cnns. In: Proceedings of the International Conference on Learning Representations (2019)

43. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1520–1528 (2015)
44. Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., Dickstein, J.S.: On the expressive power of deep neural networks. In: Proceedings of the International Conference on Machine Learning. pp. 2847–2854 (2017)
45. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115**, 211–252 (2015)
46. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 618–626 (2017)
47. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. CoRR **abs/1312.6034** (2013)
48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations (2015)
49. Sindhwani, V., Sainath, T., Kumar, S.: Structured transforms for small-footprint deep learning. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 3088–3096 (2015)
50. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(1), 1929–1958 (2014)
51. Tang, Y., Wang, Y., Xu, Y., Tao, D., Xu, C., Xu, C., Xu, C.: Scop: Scientific control for reliable neural network pruning. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 10936–10947 (2020)
52. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Technical report, California Institute of Technology (2011), `http://www.vision.caltech.edu/visipedia/papers/CUB_200_2011.pdf`
53. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 2074–2082 (2016)
54. Wen, W., Xu, C., Wu, C., Wang, Y., Chen, Y., Li, H.: Coordinating filters for faster deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 658–666 (2017)
55. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4820–4828 (2016)
56. Yeom, S.K., Seegerer, P., Lapuschkin, S., Wiedemann, S., Müller, K.R., Samek, W.: Pruning by explaining: A novel criterion for deep neural network pruning. Pattern Recognition p. 107899 (2021)
57. Yeom, S.K., Shim, K.H., Hwang, J.H.: Toward compact deep neural networks via energy-aware pruning. CoRR **abs/2103.10858** (2021)
58. You, Z., Yan, K., Ye, J., Ma, M., Wang, P.: Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 2133–2144 (2019)
59. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proceedings of the European Conference on Computer Vision. pp. 818–833 (2014)

60. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2881–2890 (2017)
61. Zhou, B., Sun, Y., Bau, D., Torralba, A.: Interpretable basis decomposition for visual explanation. In: Proceedings of the European Conference on Computer Vision. pp. 119–134 (2018)
62. Zhou, H., Alvarez, J.M., Porikli, F.: Less is more: Towards compact cnns. In: Proceedings of the European Conference on Computer Vision. pp. 662–677 (2016)
63. Zhu, J.Y., Krähenbühl, P., Shechtman, E., Efros, A.A.: Generative visual manipulation on the natural image manifold. In: Proceedings of the European Conference on Computer Vision. pp. 597–613 (2016)
64. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware channel pruning for deep neural networks. In: Proceedings of the Advances in Neural Information Processing Systems. pp. 875–886 (2018)