



CorrNet: pearson correlation based pruning for efficient convolutional neural networks

Aakash Kumar¹ · Baoqun Yin¹ · Ali Muhammad Shaikh¹ · Munawar Ali¹ · Wenyue Wei¹

Received: 18 December 2021 / Accepted: 22 July 2022 / Published online: 3 August 2022
© The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2022

Abstract

Convolutional neural networks (CNNs) are quickly evolving, which usually results in a surge of computational cost and model size. In this article, we present a correlation-based filter pruning (CFP) approach to train more reliable CNN models. Unlike several available filter pruning methodologies, our presented approach eliminates useless filters according to the volume of information available in their related feature maps. We apply correlation to compute the duplication of information carried in the feature maps and created a feature selection scheme to obtain pruning approaches. Pruning and fine-tuning are cycled many times, producing slim and denser networks with similar accuracy to the original unpruned model. We practically calculate the success of our technique with various state-of-art CNN models on many standard datasets. Specifically, for ResNet-50 on ImageNet, our approach eliminates 44.6% filter weights and saves 51.6% Float-Point-Operations (FLOPs) with 0.5% accuracy gain and obtained state-of-art performance.

Keywords Correlation · Feature maps · FLOPs · ResNet · ImageNet

1 Introduction

In this paper, we describe the method of filter pruning for the efficient CNNs, can be achieved based upon correlation amongst the feature maps which are generated from corresponding filter. This sort of pruning benefits the model in many ways, it removes the redundant feature maps to reduce the model size, lower computational cost along with save number of FLOPs. Alternatively, in past few years, researcher have achieved sophisticated performance utilizing CNNs in several demanding tasks in different areas namely image recognition [1, 2], web search [3], speech recognition [4], NLP. Nonetheless, their nature of being computationally intensive, as CNN networks have gotten deeper, the memory footprint, power hungry, and necessity of floating-point operations (FLOPs) have also dramatically increased. Thus, reason behind this growth is the increment in parameter count and also convolution operations. The networks with higher capacity tend to possess substantial

inference costs specifically for the resource constrained devices namely mobile devices or the embedded sensors because these sorts of devices have limited computational and also power resources.

A substantial amount of research has been carried out to compress huge CNN networks or learn further efficient CNNs models directly. To mitigate the conflict of higher resource requirement of CNNs, researchers have presented numerous approaches regarding to compressing and accelerating CNNs in different models along with not having noticeable loss in accuracy. Among researches extensively utilized technique is pruning for network compression. This technique further classified into two sub categories namely weight pruning [5–7] other one is filter pruning [8–10]. However, weight pruning creates unstructured scarcities because it eliminates parameters directly in the filter [40]. This behavior results in unstructured memory access and that effects the overall efficiency. While filter pruning eliminating filters as a whole and leaves a structured model. Being a simple technique yet it is quite demanding because eliminating parameters from one specific layer can results in considerable changes in input of the subsequent layer. Thus, filter pruning efficient in contrast with weight pruning [25], which helps decrease size of model, minimizes

✉ Aakash Kumar
akb@mail.ustc.edu.cn

¹ Department of Automation, University of Science and Technology of China, Hefei 230026, People's Republic of China

computational cost and saves the number of FLOPs, which is the key focus of our proposed work.

Additionally, filter pruning methods involve some measure to compute filter importance in general. The number of past researches [2, 13] calculate the valuation of the filters by scale factors or L1-norm along with neglecting the volume of information contained by the feature maps. Moreover, in [7, 10] the authors showed that given that the output of a large number of feature maps by the middle layers of CNNs are most of the zeros or zero matrices, which exposes that not all the filters in the architecture are valuable.

Furthermore, we created a correlation-based feature selector (CSF), which will boost the efficiency of pruning through negate the setting of pruning rate layer to layer. The approach of pruning filters between multiple layers provides a general scenario of the effectiveness of the model resultant in a slimmer architecture. Further, we examine the behavior of CFS under several conditions and present that CFS can recognize efficient feature maps and eliminate less important feature maps. However, the data with the superior correlation amongst two variables replicating information which is similar to a loss in information.

Subsequently, numerous metrics can be employed to analyze the correlation amongst two random variables, for instance, correlation metric, range, standard deviation, and also mean. However, the requirement of the feature map selection algorithm based on statistical importance test is to consider that the features are mutually independent, but actually, there is always a definite correlation between the features [45]. Therefore, in this paper we utilized the Pearson correlation-based [11] technique to calculate correlation amongst two feature maps, where higher the correlation amongst two feature maps means greater the replication of information amongst two feature maps ought to be. Further, one feature map will be eliminated from the specific filter as described in the Fig. 1. Correlation performs a major part in data science as the selection of features and helps to find effective features according to the correlation.

2 Related work

In general, challenge of compressing CNNs has been extensively considered in literature, although interpretability is not an impelling aspect in large number of these studies. In the conventional way to compress CNNs, weights, but not filters are actually pruned and also quantized which commonly termed as “weight compression”. Few of these approaches are optimal brain surgeon [14], Deep compression [15], optimal brain damage [16], and the recent one is SqueezeNet [17].

The primary goal of filter pruning is basically to acquire an estimation regarding to importance of filters and then

eliminate unimportant filters [2, 18, 19]. Besides, thereafter at every pruning step, it essentially required re-training to cover the decline in accuracy. The authors [20] estimated the importance of the filter over a subset of training data which is based upon output feature map. In [14] they performed pruning based upon a greedy technique which assessed filter importance through verifying accuracy of the model right after pruning filter. However, feature maps or in other words pruning activation is utilized in [21] with the aim of acquiring more faster CNNs. This technique can also be perceived as eliminating filters in input on a particular location, but those filters most of the time be left in other locations which hardly ever results in any sort of filter compression. Thus, in our work, CFS selects less important filters based on the correlation value between successive corresponding feature maps. We apply Pearson correlation to calculate the correlation value between two feature maps. The CFS approach is different from other approaches and can tackle their weakness such as ignorance of spatial information in feature maps since we try to clip the feature maps by row and measure their spatial information with correlation to evaluate the usefulness of the corresponding filters. After we create a feature selection module to extract the output of each filter and calculate their correlation weights. These modules are put between each two successive feature maps of the filter of the pre-trained model, as illustrated in the Fig. 2. Those filters whose corresponding two feature maps correlation are the higher value will be pruned one of them. After the pruning, we fine-tune the pruned model to recover performance and can even obtain greater accuracy in several scenarios. Lastly,

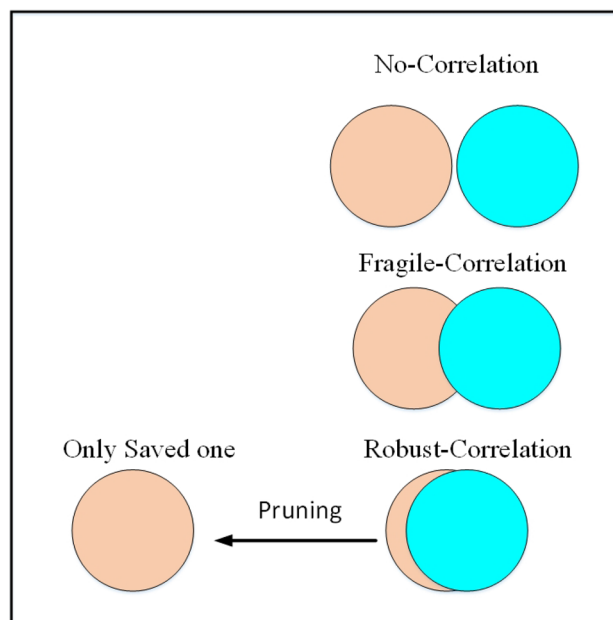


Fig. 1 Robustly correlated feature maps are chosen, and one of duplicate feature map is pruned

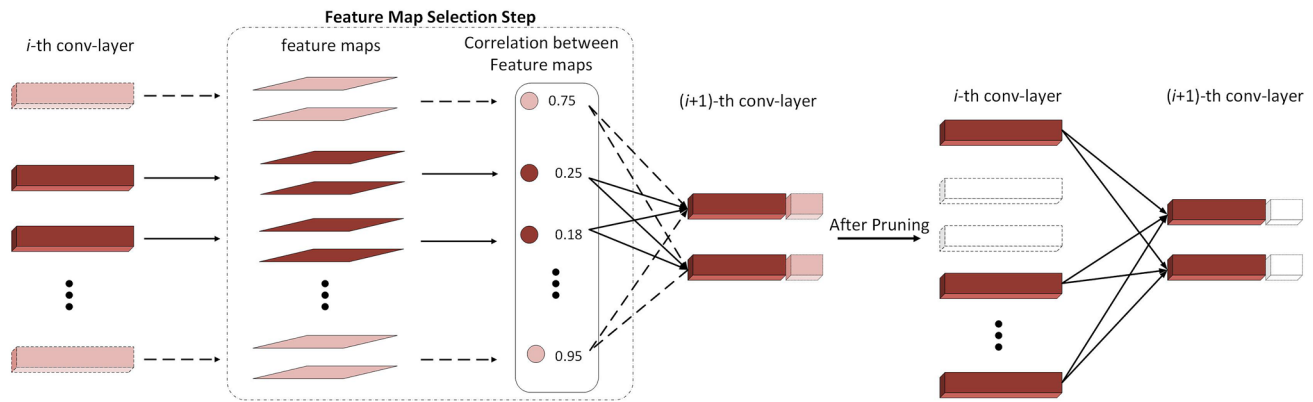


Fig. 2 This shows our introduced technique which prunes filters according to correlation score between two feature maps. It computes the correlation between each two adjacent feature maps of the pre-trained model (left side). For the i -th layer, the two output feature maps of convolutional filters are excerpted and input to the correlation module to measure their correlation. After those feature maps

with higher correlation scores show that they are duplicating the information and one of the feature maps and filter will be eliminated (right side). In the meantime, the related channels of each filter in $(i+1)$ -th layer will also be deleted to be reliable with input. Finally, all the filters of the convolutional layer are pruned layer by layer

the pruning and fine-tuning procedure are iterative for few times to achieve a slimmer model. Moreover, we also study the relationship between the pruning ratio and the number of filters to highlight the sharing of information in every convolutional layer of a CNN model.

To prune a fully trained network is fruitful in several ways in contrast to training a network from scratch containing lesser filters. Because there are many uncertainties such as architecture selection and how many filters are required to start with. Even though to tackle that issue many hyper-parameters optimization approaches presented [22–24], large number of feasible architecture along with filters that results in lofty computational cost into a conjugational way as in more model choice issues [25]. Additionally, contemporary outcomes recommend that since the CNNs with large-scale, the pruned network accuracy corresponds to faintly higher in contradiction to a certain network trained from scratch [10], for ResNet and also VGG. In contrary to larger-scale, small-scale CNNs, to train a particular network from scratch is most likely to accomplish similar accuracy considering pruned one. In ample amount of application regarding to transfer learning which based upon well trained networks, the algorithms can accomplish much higher accuracies in contrast to training from scratch provided same number of filters and architecture [21].

Many of researches focused on reducing complexity of multiplication as well as parameters. In [41] authors used Strassen Algorithm with the help of Pan's modification to reduce number of multiplications in CNNs. On the other hand, Li [42] presented Structured channel weight sharing to compress (SCWC) which uses distributive property to reduce number of multiplications in CNNs. The authors in [43] proposed two densely linked CNNs and named them

DenseDsc and Dense2Net. In DenseDsc, the reliable depth-wise separable convolution is applied to enhance the performance. For the Dense2Net, author applied group convolution to enhance the parameters performance.

Besides, a model presented in [26] familiarized sparsity to the model parameters and with that they also required support of sparse libraries to achieve anticipated compression results. Similarly, mentioned technique provides an inadequate compression rate on Total run time memory (TRM) and FLOPs. But these particular techniques deliver a better compression rate regarding to weights storage, with limited FLOPs. whereas pruning approach in [27] presented for filter importance possesses specific constraints as necessities that are not generally met. So, models sustain the redundancy which are compressed by these approaches since these approaches do not contemplate filter redundancy during pruning.

As another option, approaches based upon weight quantization [28, 29] have been utilized in previous researches for compression of models. In [30] performed compression relied upon float value quantization for the purpose of model storage. But in [7] conducted compressions though Binarization in which every float value is quantized up to binary value. However, researchers have also utilized Bayesian approaches considering network quantization. On contrary our proposed technique removes unimportant feature maps to help decrease the size of the model along with save the number of FLOPs and lower the computational cost.

We perform our approach on various famous datasets and different CNN models. For VGG16 on CIFAR100, we obtain approx. 59.6% of parameter pruning and 46.4% reduction of FLOPs with 0.17% accuracy loss. The model with less redundancy such as ResNet-50, we also approx.

44.6% parameter elimination, without loss of accuracy. In the coming sections, we will provide brief details of our correlation-based filter pruning approach.

3 Methodology

Initially, we present how to determine the correlation scores of two feature maps. After, our feature maps selection and filter pruning schemes are given. Lastly, the discussion of computational cost compression.

3.1 Calculate correlation scores of the two successive feature maps

The number of past researches [2, 10, 13] calculate the valuation of the filters through scale factors or L1-norm and neglect the volume of information contained by the feature maps. Some past studies [7, 10] have given that the output of a large number of feature maps by the middle layers of CNNs are most of the zeros or zero matrices, which exposes that not all the filters in the architecture are valuable.

To find the usefulness of filters, we apply correlation to calculate the information in feature maps. Correlation performs a major part in data science as the selection of features and helps to find effective features according to the correlation score. Considering that the output of dissimilar convolutional layers has a substantial difference in the amount of information, therefore, the feature maps weights are calculated in every layer individually. Particularly, to overcome the conditional outcome of a single image, we arbitrarily choose a large number of images from the training set to compute the average correlation weights of filters.

Suppose H_i is the height of the output feature maps and W_i is the width, m_i denoted as the number of filters in the $i - th$ convolutional layer, where one filter produces one feature map. Further, N shows the randomly selected images input into the model, value of N directly impact on the memory of the system, greater the value of N , more it consumes memory of the system. Moreover, It is worth noted that the value of N is same for all considered datasets. The lowest scores of final accuracy are obtained from N of 16, 32, 50, and 64. The highest results of final accuracy are demonstrated by the N of 256 and 512 instances. The N of 100, 128, 150, and 200 instances show the average outcomes of final accuracy. Therefore, the higher the value of N , greater the final accuracy we will get, additionally, much larger value can also effect on the final accuracy as shown in Fig. 3 (The

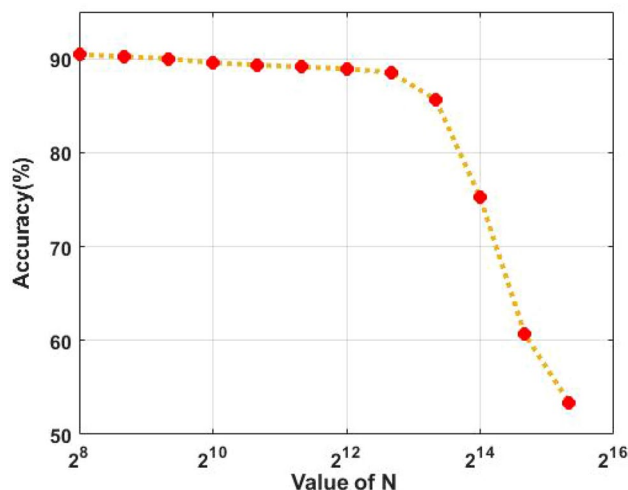


Fig. 3 Impact of N on VGG16 accuracy using ImageNet dataset

figure only highlights ImageNet dataset for VGG16 model). Let $X_{ij}^{(n)}$ set as $j - th$ matrix of output feature map of the layer i for the $n - th$ image, and $Y_{ik}^{(n)}$ is $k - th$ matrix of output feature map of the same layer i for $n - th$ image, and both the matrix converted into a vector of feature maps given below:

$$\hat{X}_{ij}^{(n)} = (x_{ij,1}^{(n)}, x_{ij,2}^{(n)}, \dots, x_{ij,L_i}^{(n)}) \tag{1}$$

$$\hat{Y}_{ik}^{(n)} = (y_{i,k,1}^{(n)}, y_{i,k,2}^{(n)}, \dots, y_{i,k,L_i}^{(n)}) \tag{2}$$

Where $L_i = H_i \times W_i$. Finally, we will normalize both $\hat{X}_{ij}^{(n)}, \hat{Y}_{ik}^{(n)}$ using the following equations.

$$P_{ij,l}^{(n)} = \frac{x_{ij,l}^{(n)} - \min_l \{x_{ij,l}^{(n)}\}}{\max_l \{x_{ij,l}^{(n)}\} - \min_l \{x_{ij,l}^{(n)}\}} \tag{3}$$

$$P_{ij}^{(n)} = (p_{ij,1}^{(n)}, p_{ij,2}^{(n)}, \dots, p_{ij,L_i}^{(n)}) \tag{4}$$

For $\hat{Y}_{ik}^{(n)}$

$$P_{i,k,l}^{(n)} = \frac{y_{i,k,l}^{(n)} - \min_l \{y_{i,k,l}^{(n)}\}}{\max_l \{y_{i,k,l}^{(n)}\} - \min_l \{y_{i,k,l}^{(n)}\}} \tag{5}$$

$$P_{i,k}^{(n)} = (P_{i,k,1}^{(n)}, P_{i,k,2}^{(n)}, \dots, P_{i,k,L_i}^{(n)}) \tag{6}$$

After, for the n – th image of i – th convolutional layer, the Pearson correlation of both j – th and k – th vector of feature maps are given as:

$$\rho(P_{ij}^{(n)}, P_{ik}^{(n)}) = \frac{\sum_{l=1}^{L_i} (P_{ij}^{(n)} - \overline{P_{ij}^{(n)}})(P_{ik}^{(n)} - \overline{P_{ik}^{(n)}})}{\sqrt{\sum_{l=1}^{L_i} (P_{ij}^{(n)} - \overline{P_{ij}^{(n)}})^2 (P_{ik}^{(n)} - \overline{P_{ik}^{(n)}})^2}} \tag{7}$$

Whereas $\overline{P_{ij}^{(n)}}$ is the mean of $P_{ij}^{(n)}$ and $\overline{P_{ik}^{(n)}}$ is the mean of $P_{ik}^{(n)}$. The value of $\rho(P_{ij}^{(n)}, P_{ik}^{(n)})$ range from -1 to 1 . If the value of $\rho(P_{ij}^{(n)}, P_{ik}^{(n)})$ is 0 then the $P_{ij}^{(n)}, P_{ik}^{(n)}$ both are independent, 1 otherwise. Feature maps that contain a greater correlation value are considered redundant feature maps, therefore, only those feature maps are selected, which contains the low redundancy between successive feature maps. The lowest Pearson’s correlation scores considering neighboring feature maps are forwarded to the selected feature map set. After will get the best feature maps and discarding the redundant feature maps. This process will continue layer to layer until will get all the effective feature maps from all layers. The algorithm 1 shows the steps based on our introduced method.

Algorithm 1: Pearson Correlation Based Pruning Algorithm

Data: Feature maps selection and elimination from the corresponding layer

Result: Compact Model

- 1 **Setting up Threshold Value ;**
 - 2 Calculate the Symmetrical Uncertainty (SU) score of each feature map put in ascending order in list **S** and the last superficial value set as threshold.
 - 3 **Redundancy Analysis ;**
 - 4 Take **X** as first feature map from the list **S**.
 - 5 Find and remove all feature maps for which **X** is approximately equivalent according to the Pearson correlation.
 - 6 Set the next remaining feature map in the list as **X** and repeat step 5 for all remaining feature maps in the **S** list.
-

3.2 Strategies for filter pruning

In the process to recognize the unimportant filters from a pre-trained network, the CFS is created and placed between every two successive convolutional layers of the network. As

illustrated in the Fig. 2, the output of the i – th convolutional layer is put into the correlation weights module to calculate the correlation score of each two consecutive feature maps through the algorithm explained in the above section. The higher correlation score shows there are duplications of information in these feature maps and one of them should be removed, further, the corresponding filter in the i – th convolutional layer is less valuable. After we can eliminate the feature maps with duplicate information by pruning all their incoming and outgoing connections. With that, all the unimportant filters of the i – th layer and feature maps for input to the next layer are eliminated, along with the corresponding channels of every filter in the next layer.

3.2.1 Setting up pruning threshold

It is important to set the pruning threshold according to the Symmetrical Uncertainty(SU) coefficient in each convolutional layer. Initially, Symmetrical Uncertainty score of each feature map in each layer are placed in ascending order. After they are gathered from the largest values until the set pruning ratio is surpassed. The last superficial value is applied as the threshold of the corresponding layer, and all the feature maps and concerned filters whose Symmetrical Uncertainty score are higher than the threshold will be eliminated. Finally, we achieved a slimmer architecture with fewer parameters, less run time memory, and storage.

3.2.2 Pruning by iteration and fine-tune process

After the pruning, we might experience some loss of accuracy temporarily, however, it can be greatly overcome through the following process of fine-tuning. In some cases, we can even obtain a greater accuracy than the baseline model. For the entire architecture pruning, past researches generally prune and fine-tune the filter layer to layer, or retrain the model after every pruning and fine-tuning process. Further, we reduce and fine-tune the model iteratively and there is no requirement to retrain the model from scratch again. This algorithm used as a prototype on LeNet shows that this is efficient. After few epochs, we can obtain a greater compression and even achieve a better classification accuracy.

3.2.3 Strategy to adjust residual models (ResNet)

The presented filter pruning approach can be effortlessly used on normal CNN models e.g. AlexNet and VGGNet. But, some modification schemes are needed when it is

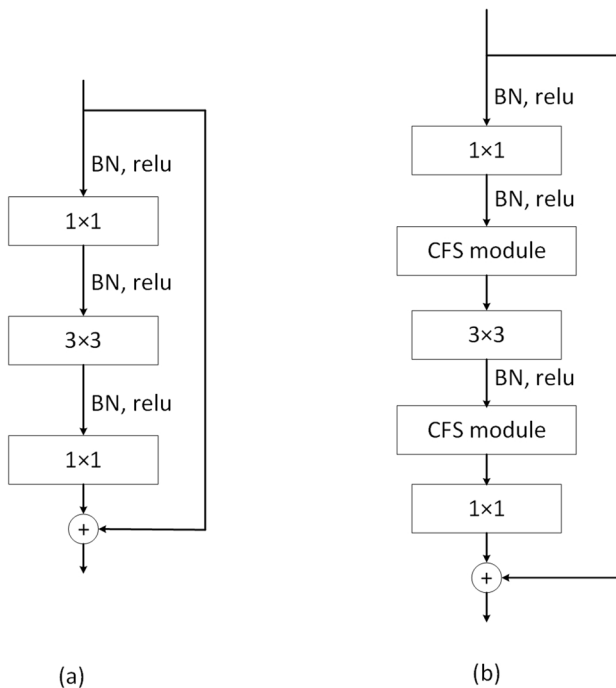


Fig. 4 Image of the method to prune cross layer architectures. **a** Original ResNet bottleneck block, **b** CFS module with bottleneck block. Before every convolutional layer, The Batch Norm layers is inserted, the ReLU is placed before CSF module

applied to reduce complex models as they have cross-layer connections for example ResNet. In ResNet, the output of the last convolutional layer and the identity mapping have a similar number of feature maps and similar in size, which creates hurdles to reduce them. As illustrated in the Fig. 4, our CFS are inserted between the first and second convolutional layers. In the bottleneck block of the third convolutional layer, we only eliminate the channels of every filter to make them compatible with the input feature maps and do not delete the number of filters, because the output of that have to align the identity maps and as we know that 1×1 filters have fewer parameters.

4 Computational complexity analysis and compression

As highlighted in Sect. 3.1, the input of the i -th convolutional layer is $H_{i-1} \times W_{i-1} \times m_{i-1}$ matrix of feature maps and generates a $H_i \times W_i \times m_i$ matrix, in which m_i and m_{i-1} the total number of feature maps. Let's suppose, parameterization of the i -th convolutional layer is $T_i \times T_i \times m_i \times m_{i-1}$, whereas T_i denote the spatial dimension of each filter. Generally, convolutions possess the computational complexity of $T_i \times T_i \times m_i \times m_{i-1} \times H_i \times W_i$. Now we set the pruning ratio of the i -th layer as p_i and the corresponding filter pruning

rate is denoted as \hat{p}_i . So that the i -th layer total filters will be decreased from m_i to $m_i(1 - p_i)$ and filters channels in the same layer are decreased from m_{i-1} to $m_{i-1}(1 - p_{i-1})$. Finally, we can obtain compression rate in computational complexity for this reduced layer as follows:

$$1 - \frac{T_i \times T_i \times m_{i-1}(1 - \hat{p}_{i-1}) \times m_i(1 - \hat{p}_i) \times H_i \times W_i}{T_i \times T_i \times m_{i-1} \times m_i \times H_i \times W_i} \quad (8)$$

$$= 1 - (1 - \hat{p}_i)(1 - \hat{p}_{i-1})$$

5 Experiments

We check our introduced CFS on various state-of-art models and datasets, such as CIFAR10 and CIFAR100 datasets used on the VGG16 model, on the other hand, we used the CIFAR10 dataset on ResNet50 and ResNet56 models. All the experiments are done with Tensorflow and Keras framework on NVIDIA Tesla P100 GPU using Google Colab Pro version.

5.1 Implementation settings

Initially, the architectures are trained from scratch to compute the classification accuracies and considered them as baselines. Before training, all the images are pre-processed before being fed into the model: all the images are randomly cropped into 32×32 size, and the horizontal flip is used with padding set to four. The mini-batch size is set to 100 for training and 1000 for test in the VGGNet model, on the other hand, the mini-batch size is set to 64 for train and 256 for test in the ResNet models. All the networks are trained and fine-tuned applying Stochastics Gradient Descent (SGD) for 160 iterations on both datasets. Further, we set the initial learning rate to 0.1, and after 50% and 75% iterations, we set the learning rate to 0.01 and 0.001 respectively. To avoid overfitting, we also applied weight initialization.

5.2 VGGNet pruning

According to [1], the convolutional layers of VGGNet are having different information attentiveness and robustness, therefore, we set separate pruning rates for every convolutional layer. Based on the architecture of VGGNet, the layers in the network are divided into three groups, (1) layers contain 512 filters, (2) layers contain 256 filters, and (3) layers consist of less than 128 filters. Further, we set different pruning rates p by each group, such as $1.5p$, p , and $0.5p$ respectively. Finally, we prune the VGG16 network iteratively with $r = 10\%$. Our introduced CFS obtains better results than the

Table 1 VGG16 Pruning for CIFAR10 dataset

Layer	Parameters	FLOPs	Pruned parameters (%)	Pruned FLOPs (%)	Activations (%)
conv1.1	2K	0.1B	65.6	58.6	56.9
conv1.2	37K	1.85B	83.5	46.9	42.2
conv2.1	74K	0.9B	92.7	62.9	51.5
conv2.2	148K	1.85B	65.5	65.2	34.4
conv3.1	295K	0.9B	95.7	40.5	40.1
conv3.2	590K	1.85B	89.6	36.9	51.8
conv3.3	590K	1.85B	96.5	57.9	51.2
conv4.1	1M	0.9B	89.9	47.2	41.8
conv4.2	2M	1.85B	97.7	64.2	28.3
conv4.3	2M	1.85B	94.8	52.4	37.1
conv5.1	2M	462.5M	98.6	55.6	41.4
conv5.2	2M	462.5M	95.6	47.9	56.1
conv5.3	2M	462.5M	97.7	70.1	55.4
fc1	103M	103M	96.9	60.8	32.1
fc2	17M	17M	85.8	44.9%	36.2
fc3	4M	4M	90.5	27.2	56.7
Total	138M	15.3B	89.7	52.4	44.6

approaches for filter pruning. After only a second epoch, the CFS approach can prune approx. 50% parameter and even obtain accuracy gain of 0.30%. After two more epochs, the saving of parameters can be approx. 90% and elimination of FLOPs can be 52.4% with a gain in accuracy of 0.05%. The Table 1 shows layer-wise pruning of parameters, FLOP saving and elimination of activation maps. Further, detailed illustration of results is given in the Table 2a.

We continue our training for VGG16 with the CIFAR100 dataset. The setting has been similar during the experiment process. This time our model can obtain a reduction of FLOPs approx. 47% in just three epochs as shown in the Table 2b. As compare to the CIFAR10, the pruning ratio is not greater, this is because the CIFAR100 has 10 times more classes and it requires more features to perform the classification task. Further, we trained VGG16 model with the ImageNet dataset, our method achieves 35.67% parameter pruning with 34.2% of FLOPs saved only on third epoch.

5.3 ResNet pruning

We have used two architectures of the ResNet family: ResNet50 and ResNet56 with the structure of bottleneck are used to check the performance of the presented algorithm. As mentioned in the summary of ResNet architectures, the

ReLU and Batch-Norm layers are inserted before every convolutional layer in bottleneck blocks. Since there are skip connections available in ResNet blocks and the useful information can be divided with the entire architecture, thus, we apply a similar pruning ratio for all layers for pruning. Furthermore, there are fewer redundant parameters in ResNet because the information is shared across the model via skip connections, therefore, we have to prune all ResNet models in single-shot pruning.

Firstly, we prune ResNet50 model using the ImageNet dataset. We can achieve nearly equal accuracy comparing to baseline architecture with approx. 40.5% parameters elimination and reduction of FLOPs approx. 48.2% with the pruning ratio is set to 20%. Further, for setting up pruning ratio to 30%, our CFS can also eliminate approx. 44.6% parameters with 51.6% FLOPs reduction, and 0.56% loss of accuracy as shown in Table 3a. It can be seen that the ResNet with bottleneck design contains less redundancy in calculations and parameters, unlike VGGNet.

We continue to apply our CFS approach to a deeper ResNet56 network for the CIFAR10 dataset, we enable the same settings as with ResNet50. Our algorithm can out-class the baseline model with a reduction of approx. 25.9% parameters and approx. 30.2% FLOPs with a gain in accuracy of 0.20%. when we try to increase the pruning ratio to 30%, our approach obtains about 52.2% of parameter elimination and saving about 55.6% of FLOPs with the only degradation of 0.31% accuracy as illustrated in Table 3b. Furthermore, we try our CFS algorithm on ResNet18 on ImageNet datasets, and our CFS approach prunes approx. 42.31% of pruning with 30.66% FLOPs saved as given in Table 3c.

6 Discussion

To widely analyze the effectiveness of our presented algorithm on the architecture, we evaluate the architecture accuracy of the different pruning rates and compression rates. As illustrated in the Figs. 5, 6, and 7 at the start the classification accuracy of pruned architecture increase over the baseline network and decreases as we increase the pruning rate. While the pruning rate is less than 25%, nearly 50% of the parameters are eliminated, and our approach obtains no loss of accuracy and sometimes it achieves marginally accuracy enhancement. This shows that our algorithm CFS can eliminate unimportant information and enhance the efficient set of features. This discussion only highlights the ResNet model.

Table 2 Pruning results of the VGG-16 model performed on CIFAR-10/100

(a) Prune results of VGG-16 model on CIFAR-10 datasets						
Method	Baseline (%)	Accuracy (top 5%) (%)	Param-baseline (%)	Parameters (%)	Pruned (%)	FLOPs saved (%)
Li [10]	93.25	93.30	1.5×10^2	5.4×10^6	64.0	34.2
Slimming [13]	93.60	93.80	–	–	88.5	51.0
Entropy [32]	93.72	93.97	1.5×10^2	3.5×10^5	76.4	49.5
Sai [33]	93.75	93.80	–	–	90.5	65.6
Kumar[34]	93.77	93.81	1.5×10^2	3.0×10^5	92.7	75.8
Yan[44]	99.47	99.25	1.5×10^2	–	62.38	42.56
Ours(epoch-1)	93.73	93.95	1.5×10^2	3.2×10^5	77.7	44.6
Ours(epoch-4)	93.73	93.82	1.5×10^2	1.1×10^5	89.7	52.4
(b) Prune results of VGG16 model on CIFAR-100 datasets						
Method	Baseline (%)	Accuracy (top 5%) (%)	Param-baseline	Parameters	Pruned (%)	FLOPs saved (%)
Slimming [13]	73.26	73.48	–	–	75.1	51.0
Entropy [32]	73.60	73.82	1.5×10^2	9.6×10^6	34.4	26.2
Kumar [34]	73.44	73.61	1.5×10^2	6.4×10^6	56.4	44.2
Ours(epoch-1)	73.72	73.65	1.5×10^2	7.6×10^6	35.7	27.6
Ours(epoch-2)	73.72	73.55	1.5×10^2	5.4×10^6	59.6	46.4
(c) Prune results of VGG16 model on ImageNet datasets						
Method	Baseline (%)	Accuracy (top 5%) (%)	Param-baseline	Parameters	Pruned (%)	FLOPs saved (%)
Yan-1 [44]	90.38	90.30	1.5×10^2	–	35.62	33.18
Yan-2 [44]	90.38	90.12	1.5×10^2	–	35.62	33.18
Ours(epoch-1)	90.35	90.23	1.5×10^2	–	23.8	19.2
Ours(epoch-3)	90.35	90.44	1.5×10^2	–	35.67	34.2

Bold shows the matched or better results than other state-of-art works present

7 Conclusion

In this article, we present an easy and efficient approach, which calculates the effectiveness of convolutional filters on the basis of duplication possess in the two successive feature maps generated through these filters. Our approach proposes correlation to compute the redundancy brought by the feature maps and measure the usefulness of related convolutional filters. A correlation feature selector is created to

designed pruning schemes. To overcome the dimensionality mismatch problem in the ResNet model in course of pruning, new pruning schemes are presented. Furthermore, this article also highlights the sharing of information in each convolutional layer with results reflects that filters in many layers contribute limited to the final accuracy of the model. Advanced experiments reflect the advantage of our method compared to the presently available approaches. Finally,

Table 3 Pruning results in ResNet models performed on CIFAR-10 and ImageNet

(a) Prune results of model ResNet-50 on ImageNet datasets

Method	Baseline (%)	Accuracy (top 5%) (%)	Param-baseline	Parameters	Pruned	FLOPs saved (%)
ThiNet-50 [35]	92.20	91.0	–	–	–	≈50
WAE [36]	92.20	90.40	–	–	–	46.8
He [2]	92.20	90.80	–	–	–	≈50
SSP [38]	92.20	90.40	–	–	–	≈50
CFP [39]	92.20	91.40	–	–	–	49.6
Yan [44]	92.86	92.09	–	–	37.26	38.78
Ours(20%)	92.30	92.45	–	–	40.5	48.2
Ours(30%)	92.30	92.35	–	–	44.6	51.6

(b) Prune results of model ResNet-56 on CIFAR-10 datasets

Method	Baseline (%)	Accuracy (top 5%) (%)	Param-baseline	Parameters	Pruned	FLOPs saved (%)
Li [10]	93.04	93.06	8.5×10^5	7.3×10^5	13.7	27.6
He [2]	92.80	91.8	–	–	–	50.0
He [40]	93.59	93.35	–	–	–	52.6
Entropy [32]	94.12	94.11	5.9×10^5	4.3×10^5	25.7%	29.4
Sai [33]	93.60	93.35	–	–	52.8%	52.1
Kumar [34]	94.11	93.37	6.5×10^5	2.9×10^5	51.3%	53.9
Ours(20%)	94.13	94.25	5.5×10^5	4.1×10^5	25.9%	30.2
Ours(30%)	94.13	94.20	5.5×10^5	2.7×10^5	52.2%	55.6

(c) Prune results of model ResNet-18 on ImageNet datasets

Method	Baseline (%)	Accuracy (top 5%) (%)	Param-baseline	Parameters	Pruned (%)	FLOPs saved (%)
Yan-1 [44]	89.08	89.20	–	–	42.27	30.61
Yan-2 [44]	89.08	89.16	–	–	42.27	30.61
Ours(20%)	89.13	89.16	–	–	40.12	30.05
Ours(30%)	89.13	89.11	–	–	42.31	30.66

Bold shows the matched or better results than other state-of-art works present

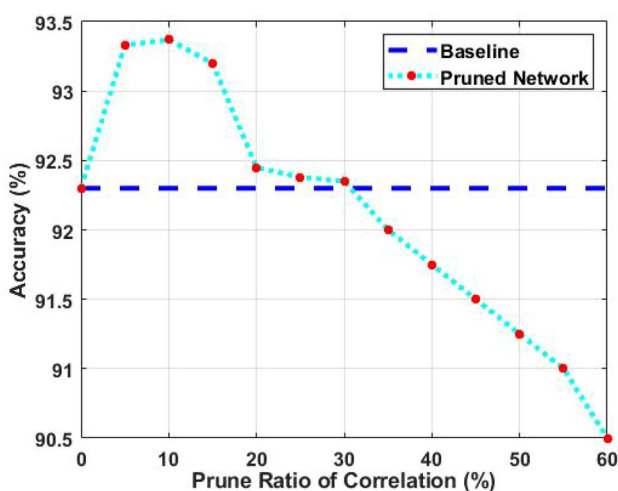


Fig. 5 Pruning results of ResNet50 on ImageNet regarding different correlation pruning ratios

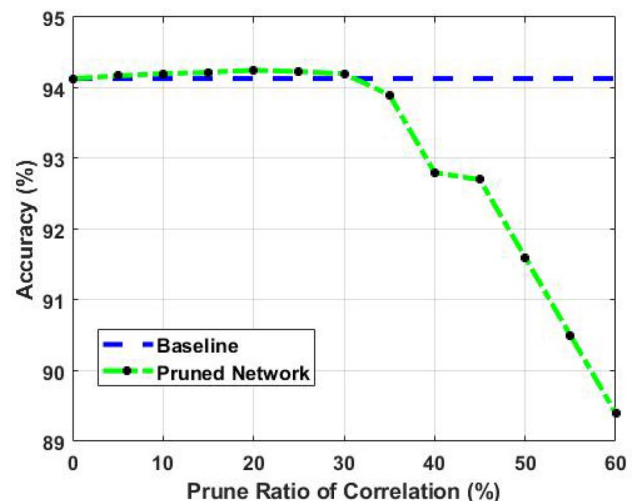


Fig. 6 Pruning results of ResNet56 on CIFAR-10 regarding different correlation pruning ratios

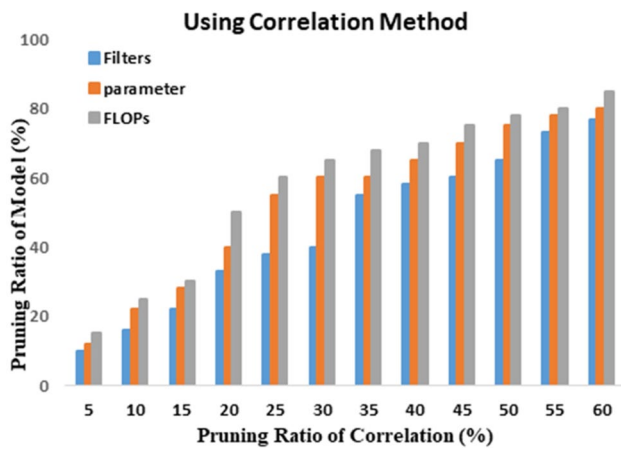


Fig. 7 Pruning comparison of FLOPs, parameters, and filters

for deeper and complex model ResNet56 on the CIFAR10 dataset, our presented approach can eliminate 52.2% of parameters along with 55.6% FLOPs reduced with 0.09% of accuracy gain, and this shows the greatness of the proposed algorithm.

Acknowledgements This work is supported by the National Natural Science Foundation of China under grant number 62133013 and sponsored by CAAI-Huawei MindSpore Open Fund. Chinese Academy of Sciences (CAS) and The World Academy of Sciences (TWAS) are highly acknowledged for the funds making this study possible.

References

- Hosseini B, Montagne R, Hammer B (2020) Deep-aligned convolutional neural network for skeleton-based action recognition and segmentation. *Data Sci Eng* 5(2):126–139
- He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1389–1397)
- Nguyen PQ, Do T, Nguyen-Thi A, Ngo TD, Le D, Nguyen TH (2016) “Clustering web video search results with convolutional neural networks,” 2016 3rd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), pp. 135–140, <https://doi.org/10.1109/NICS.2016.7725638>.
- Dahl GE, Yu D, Deng L, Acero A (2011) Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Trans Audio Speech Lang Process* 20(1):30–42
- Han S, Pool J, Tran J, Dall, WJ (2015) Learning both weights and connections for efficient neural networks. arXiv preprint [arXiv:1506.02626](https://arxiv.org/abs/1506.02626)
- Carreira-Perpinán MA, Idelbayev Y (2018) “learning-compression” algorithms for neural net pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 8532–8541)
- Liu B, Wang M, Foroosh H, Tappen M, Pensky M (2015) Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 806–814)
- Yu R, Li A, Chen CF, Lai JH, Morariu VI, Han X, Gao M, Lin CY, Davis LS (2018) Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9194–9203)
- He Y, Ding Y, Liu P, Zhu L, Zhang H, Yang Y (2020) Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2009–2018)
- Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint [arXiv:1608.08710](https://arxiv.org/abs/1608.08710)
- Press WH, Teukolsky SA, Flannery BP, Vetterling WT (1992) *Numerical recipes in Fortran 77: volume 1, volume 1 of Fortran numerical recipes: the art of scientific computing*. Cambridge University Press
- Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision* (pp. 2736–2744)
- Abbasi-Asl R, Yu B (2017) Structural compression of convolutional neural networks. arXiv preprint [arXiv:1705.07356](https://arxiv.org/abs/1705.07356)
- Han S, Mao H, Dally WJ (2015) Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint [arXiv:1510.00149](https://arxiv.org/abs/1510.00149)
- LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: *Advances in neural information processing systems 2 (NIPS 1990)*, vol 2, pp 598–605
- Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and < 0.5 MB model size. arXiv preprint [arXiv:1602.07360](https://arxiv.org/abs/1602.07360)
- Hu H, Peng R, Tai YW, Tang CK (2016) Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint [arXiv:1607.03250](https://arxiv.org/abs/1607.03250)
- Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2016) Pruning convolutional neural networks for resource efficient inference. arXiv preprint [arXiv:1611.06440](https://arxiv.org/abs/1611.06440)
- Jaderberg M, Dalibard V, Osindero S, Czarnecki WM, Donahue J, Razavi A, Vinyals O, Green T, Dunning I, Simonyan K, Fernando C (2017) Population based training of neural networks. arXiv preprint [arXiv:1711.09846](https://arxiv.org/abs/1711.09846)
- Fernando C, Banarse D, Blundell C, Zwols Y, Ha D, Rusu AA, Pritzel A, Wierstra D (2017) Pathnet: Evolution channels gradient descent in super neural networks. arXiv preprint [arXiv:1701.08734](https://arxiv.org/abs/1701.08734)
- Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2017) Hyperband: a novel bandit-based approach to hyperparameter optimization. *J Mach Learn Res* 18(1):6765–6816
- Reed R (1993) Pruning algorithms—a survey. *IEEE Trans Neural Netw* 4(5):740–747
- Chen W, Wilson J, Tyree S, Weinberger K, Chen Y (2015) Compressing neural networks with the hashing trick. In *International conference on machine learning* (pp. 2285–2294). PMLR
- He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4340–4349)
- Dubey A, Chatterjee M, Ahuja N (2018) Coresnet-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 454–470)
- Tung F, Mori G (2018) Clip-q: Deep network compression learning by in-parallel pruning-quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7873–7882)
- Miao H, Li A, Davis LS, Deshpande A (2017) Towards unified data and lifecycle management for deep learning. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)* (pp. 571–582). IEEE

29. Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) Xnor-net: Imagenet classification using binary convolutional neural networks. In European conference on computer vision (pp. 525–542). Springer, Cham
30. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770–778)
32. Li Y, Wang L, Peng S, Kumar A, Yin B (2019) Using feature entropy to guide filter pruning for efficient convolutional networks. In International Conference on Artificial Neural Networks (pp. 263–274)
33. Aketi SA, Roy S, Raghunathan A, Roy K (2020) Gradual channel pruning while training using feature relevance scores for convolutional neural networks. *IEEE Access* 8:171924–171932
34. Kumar A, Shaikh AM, Li Y et al (2021) Pruning filters with L1-norm and capped L1-norm for CNN compression. *Appl Intell* 51:1152–1160. <https://doi.org/10.1007/s10489-020-01894-y>
35. Luo JH, Wu J, Lin W (2017) Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE international conference on computer vision (pp. 5058–5066)
36. Chen T, Lin L, Zuo W, Luo X, Zhang L (2018) Learning a wavelet-like auto-encoder to accelerate deep neural networks. In Thirty-Second AAAI Conference on Artificial Intelligence
38. Wang H, Zhang Q, Wang Y, Hu H (2017) Structured probabilistic pruning for convolutional neural network acceleration. arXiv preprint [arXiv:1709.06994](https://arxiv.org/abs/1709.06994)
39. Singh P, Verma VK, Rai P, Namboodiri V (2020) Leveraging filter correlations for deep model compression. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 835–844)
40. He Y, Kang G, Dong X, Fu Y, Yang Y (2018) Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint [arXiv:1808.06866](https://arxiv.org/abs/1808.06866)
41. Ali M, Yin B, Kunar A, Sheikh AM, Bilal H (2020) “Reduction of Multiplications in Convolutional Neural Networks,” 2020 39th Chinese Control Conference (CCC), pp. 7406–7411, <https://doi.org/10.23919/CCC50068.2020.9188843>
42. Li Guoqing, Zhang Meng, Wang Jiuyang, Weng Dongpeng, Corporaal Henk (2022) SCWC: structured channel weight sharing to compress convolutional neural networks. *Inf Sci* 587:82–96
43. Li Guoqing, Zhang Meng, Li Jiaojie, Lv Feng, Tong Guodong (2021) Efficient densely connected convolutional neural networks. *Pattern Recognit* 109:107610
44. Zhaoyi Yan, Xing Peiyin, Wang Yaowei, Tian Yonghong (2020) “Prune it yourself: Automated pruning by multiple level sensitivity.” In 2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 73–78. IEEE
45. Pengtian Chen, Li Fei, Wu Chunwang (2021) “Research on intrusion detection method based on Pearson correlation coefficient feature selection algorithm.” In *Journal of Physics: Conference Series*, vol. 1757(1):012054. IOP Publishing

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.