# Learning the number of filters in convolutional neural networks

## Jue Li

Institute of Big Data Science and Industry,
School of Computer and Information Technology,
Shanxi University,
Taiyuan, China
Email: 834573176@qq.com

## Feng Cao

School of Computer and Information Technology,
Shanxi University,
Taiyuan, China
Email: caof@sxu.edu.cn

## Honghong Cheng

Institute of Big Data Science and Industry,
Shanxi University,
Taiyuan, China
Email: chhsxdx@163.com

## Yuhua Qian*

Institute of Big Data Science and Industry,
School of Computer and Information Technology,
Shanxi University,
Taiyuan, China
Email: jinchengqyh@126.com
*Corresponding author

**Abstract:** Convolutional networks bring the performance of many computer vision tasks to unprecedented heights, but at the cost of enormous computation load. To reduce this cost, many model compression tasks have been proposed by eliminating insignificant model structures. For example, convolution filters with small absolute weights are pruned and then fine-tuned to restore reasonable accuracy. However, most of these works rely on pre-trained models without specific analysis of the changes in filters during the training process, resulting in sizable model retraining costs. Different from previous works, we interpret the change of filter behaviour during training from the associated angle, and propose a novel filter pruning method utilising the change rule, which can remove filters with similar functions later in training. According to this strategy, not only can we achieve model compression without fine-tuning, but we can also find a novel perspective to interpret the changing behaviour of the filter during training. Moreover, our approach has been proved to be effective for many advanced CNN architectures.

**Keywords:** model compress; filter pruning; filter correlation; filter behaviour interpretable.

**Reference** to this paper should be made as follows: Li, J., Cao, F., Cheng, H. and Qian, Y. (2021) 'Learning the number of filters in convolutional neural networks', *Int. J. Bio-Inspired Computation*, Vol. 17, No. 2, pp.75–84.

**Biographical notes:** Jue Li is currently pursuing her Master's degree at the Institute of Big Data Science and Industry, Shanxi University. Her research interests include associations mining in big data, deep learning and machine learning.

Feng Cao received his PhD at the School University of Chinese Academy of Sciences, China. He is a master tutor of the School of Computer and Information Technology, Shanxi University. His research interests include deep learning and machine learning.

Honghong Cheng received her BS degree at the School Mathematical Sciences from Shanxi University, China, in 2012. She is a PhD candidate at the Institute of Big Data Science and Industry, Shanxi University. Her research interests include associations mining in big data and machine learning.

Yuhua Qian received his MS and PhD in Computers with Applications at the Shanxi University, in 2005 and 2011, respectively. He is a Professor at the Key Laboratory of Computational Intelligence and Chinese Information Processing of the Ministry of Education, China. He is actively pursuing research in artificial intelligence, granular computing, machine learning and deep learning.

This paper is a revised and expanded version of a paper entitled 'Learning the number of filters in convolutional neural networks' presented at CGCKD 2020, Taiyuan, China, 31 October 2020.

## 1 Introduction

In recent years, convolutional neural networks (CNNs) have made significant progress in the most of computer vision tasks (Singh et al., 2018a; Schroff et al., 2015; He et al., 2016; Long et al., 2015; Zhu et al., 2019). Despite this progress, high computing costs and storage space limit the deployment of the model on some mobile devices. Therefore, designing a small CNN with high performance is a breakthrough method to solve this bottleneck. However, currently higher performance models are often accompanied by higher complexity. For example, VGG-16 (Simonyan and Zisserman, 2014), has as many as 138 million parameters and consumes up to 500 MB storage space. To store the intermediate results of the model, it requires more than 16 billion floating point operations (FLOPs) and 93 MB of additional runtime memory, which puts a heavy burden on low-end mobile devices. Ioannou et al. (2015) proved that the convolutional layer can be compressed and accelerated. For instance, Han et al. (2015) directly deleted weight values of filters. However, weight pruning is an unstructured pruning method and cannot use the current efficient BLAS library, which makes efficiency in computational cost still low. In contrast, compared with weight pruning, filter pruning makes the model structured sparsity and more effective memory usage, thus making full use of the BLAS library to achieve more realistic acceleration. Therefore, the filter pruning is more convenient in accelerating the networks.

Filter pruning methods remove unimportant filters according to different criteria. But most filter pruning methods (Han et al., 2015; He et al., 2017; Luo et al., 2017; Dong et al., 2017; Yu et al., 2018; Luo and Wu, 2017; Ye et al., 2018) are based on the original trained model. Once the filters are pruned, it takes a long time to fine-tune to restore its reasonable performance, which however brings low training efficiency and often takes more training time than the traditional training schema. And the over dependence on the retraining process seriously decreases the rationality of the conventional filter significance identification. Ye et al. (2018) showed that the retraining process actually rebuilds the CNN models. Therefore, to explore the internal structure of the model, rather than merely restoring the accuracy of the model by increasing the training time, it is urgent to qualitatively interpret the filter behaviour and identify real model redundancies.

The internal changes of the convolutional filters are still in a 'black box' state. Yosinski et al. (2015) utilised CNN visualisation techniques to interpret the convolutional filter functionality. In this paper, different from other methods, we explore the regular pattern of filters change in the training process from the perspective of correlation, and interpret the filter's self-evolution during training. We discover that most filters with strong correlation are always in a stable state in the later period of CNN training. Thus, we propose a novel filter pruning approach named filter pruning via CKA (FPC) (Kornblith et al., 2019) during the training process without basis of the pre-trained model, which prunes highly correlated filters in the later period of training and has advanced performance.

We verify the effectiveness of our approach on some CNNs and use several image recognition datasets. On CIFAR-10 dataset, FPC reduces 56.2% FLOPs on ResNet-56 with only 0.69% accuracy drop and also achieves 40.8% FLOPs reduction on ResNet-110 with even 0.03% accuracy improvement. On the CIFAR-100 dataset, with 52.6% and 52.3% FLOPs reduction, our method can accelerate ResNet-56 and ResNet-110 with tiny accuracy drops. In addition to the typical large network such as ResNet, we have also performed experiments on small networks such as Alexnet. And results show that FPC has better performance on Alexnet compared with SFP (He et al., 2018b) and FPGM (He et al., 2019a), especially when the pruning rate is relatively high.

The major contribution of this paper can be summarised as follows:

- We closely study the change law of filters via centred kernel alignment (CKA) in the training process, and discover that most filters with strong correlation are always in a stable state in the late training period.
- We use CKA to determine the strength of the relationship between filters. It could measure any form of dependency over correlation, which is different from other indexes.
- We propose FPC to prune the most replaceable filters that contain redundant information in the later training

process, which can achieve better performance than other methods.

The rest of the paper is organised as follows. Section 2 introduces the related work of model compression. The FPC algorithm is introduced in detail in Section 3, including interpreting the behavioural change of the filters and how to get a small and accelerated model. At last, FPC algorithm evaluation and conclusions are given in Section 4 and Section 5.

## 2 Related work

### *2.1 Filter pruning*

#### *2.1.1 Pruning criterion*

Most recent work on pruning criterion can be roughly divided into two categories:

1 *Just think about the importance of a single filter.* Early work (Ye et al., 2018; He et al., 2018b; Li et al., 2016) utilised in 'smaller-norm-less-important' criterion. This criterion thought that filters with small normals were less critical, thus, they used $L_1$ norm or $L_2$ norm pruning unimportant filters and achieved reasonable performance. Luo et al. (2017) thought that it was difficult to judge the importance of filters by the size of weight value, and it was possible to prune some useful filters through this method. Therefore, a pruning method based on entropy value was proposed to determine the importance of filters. And moreover, APoZ (Hu et al., 2016), Taylor (Molchanov et al., 2016), ThiNet (Luo et al., 2017), and so on to be proposed to evaluate the importance of each filter.

2 *Consider the correlation between filters.* What we have found is that in the recent years, researchers have begun to focus on the synergy between the filters (He et al., 2019a, 2019b; Zhuo et al., 2018; Singh et al., 2018b; Wang et al., 2018, 2019; Qin et al., 2018). They think correlated filters will have the same effects on the network prediction, and the redundant filters can be further discarded. For instance, Singh et al. (2018b) used person criterion to identify the filter pair with the greatest correlation and removed one of the filters from each such pair.

#### *2.1.2 Pruning rules*

Most filter pruning work relies on pre-trained models. After pruning, they will fine-tune to restore the original accuracy, which takes a lot of time. In order to solve this problem, many automatic pruning methods are proposed. For instance, Liu et al. (2017), according to the specific performance index of the model, automatically obtained the reduction factor corresponding to the redundancy elimination strategy. He et al. (2018b, 2019a) and Zhuo et al. (2018) applied a soft filter pruning method, which could train a model from scratch and get better results. For the pruning ratio per layer, different works have different ways. Li et al. (2016) performed sensitivity analysis for each layer, which pruned filters layer by layer, and retrained the model before pruning the next layer to make the weights adapt to the changes in the pruning process. He et al. (2019a) pruned the same ratio for each layer which could automatically prune the filters. Liu et al. (2017) divided network into three parts: first, middle and last, and the corresponding pruning ratio was adjusted step by step. It is found that the convolutional layer of the first few layers is relatively redundant, while the convolutional layer of the second half of the model plays a more critical role in the prediction process.

### *2.2 Other methods*

Most previous work for model compress could divide into four categories:

- *Knowledge distilling* (Hinton et al., 2015; Kim et al., 2018) using the output of the teacher network as a soft label to train a student network is a popular model compression method at present.
- *Low-rank decomposition* (Zhang et al., 2015; Tai et al., 2015) approximates network weights with several lower rank matrices.
- *Quantisation* (Rastegari et al., 2016): generally speaking, the parameters of the neural network model are represented by 32-bit floating point numbers. In fact, it is not necessary to retain such a high accuracy. It can be quantified, such as 0~255 for the original 32 bits, by sacrificing accuracy to reduce the space required for each weight.
- *Automatic neural structure search* combined with some optimisation algorithms for model search has also attracted attention in recent years. For instance, Baker et al. (2016) proposed MetaQNN, which models the network architecture search as a Markov decision process and uses the RL method to generate a CNN architecture. In Real et al. (2017), evolutionary algorithms were introduced to solve NAS problems and had been proven to achieve high accuracy starting from a simple initial condition on the CIFAR-10 and CIFAR-100 datasets.

## 3 Method

### *3.1 Motivation for 'black box'*

As is known to all, neural networks are still a 'black box' now. In order to design a network efficiently, many researchers actively explore the internal structure of neural networks. Recently, some researchers have used association indicators to explore the internal structure of neural networks. Li et al. (2015) proposed a specific method of probing representations via mutual information: training

multiple networks and then applying correlation analysis to comparing and contrasting their individual, learned representations at the level of neurons or groups of neurons. Raghu et al. (2017) combined CCA with SVD to measure the intrinsic dimensionality of layers, they showed that the increase in the depth of the model did not lead to a corresponding increase in the learned features, due to several layers learning representations in correlated directions. Morcos et al. (2018) used CCA to find that the representation in the hidden layer of the neural network contained two signal components, one was stable during training, corresponding to the performance curve, and the other was an unstable noise component. Kornblith et al. (2019) applied CKA, which could measure meaningful correlations between representations whose dimension is greater than the number of data points, to establishing correspondences between layers of different network architectures, and verified that wider networks learn more similar representations. All these explorations measure the activation value of the neural network. This aspect shows that the correlation index can relatively accurately explore the internal representation of the neural network, and at the same time, it can also peep out the strong redundancy between the features generated by the neural network. We think that the weights corresponding to the correlated activation values are also correlated. So based on these findings, we ask some questions-how correlated are the filters within a single layer? How does the correlation change between filters as the training time increases and can we open the 'black box' state of the filters in training from the correlated angle?

### 3.2 Correlations between filters

We assume $N_i$ and $N_{i+1}$ denote the number of input channels and output channels for the $i^{\text{th}}$ layer, respectively. $N_{i+1}$ also represents the number of filters for the $i^{\text{th}}$ convolution layer. $L$ is the depth of the convolutional network. $\mathcal{F}_{i,j}$ represents the $j^{\text{th}}$ filter of the $i^{\text{th}}$ layer. $\mathbb{R}^{N_i \times K \times K}$ is the dimension of filter $\mathcal{F}_{i,j}$ and $K$ represents the kernel size of the network. The $i^{\text{th}}$ layer of the network $W^i$ could be marked as $\{\mathcal{F}_{i,j}, 1 \le j \le N_{i+1}\}$. The connection tensor of the deep CNN network can be parameterised by $\{W^i \in \mathbb{R}^{N_{i+1} \times N_i \times K \times K}\}$.

We apply CKA to measuring the correlations between filters. For each filter, we can constitute a two-dimensional matrix ($N_{i+1}$, $K \times K$). Correlation of any two filters $\mathcal{F}_{i,j}$ and $\mathcal{F}_{i,h}$ on the same layer could be calculated as follows:

$$\begin{aligned} &\left\langle \left(\mathcal{F}_{i,j}\left(\mathcal{F}_{i,j}\right)^T\right), \left(\mathcal{F}_{i,h}\left(\mathcal{F}_{i,h}\right)^T\right)\right\rangle \\ &= tr\left(\mathcal{F}_{i,j}\left(\mathcal{F}_{i,j}\right)^T \mathcal{F}_{i,h}\left(\mathcal{F}_{i,h}\right)^T\right) \\ &= \left\|\left(\mathcal{F}_{i,h}\right)^T \mathcal{F}_{i,j}\right\|_F^2 \end{aligned} \tag{1}$$

$$\begin{aligned} &\frac{1}{\left(N_{i+1}-1\right)^2} tr\left(\mathcal{F}_{i,j}\left(\mathcal{F}_{i,j}\right)^T \mathcal{F}_{i,h}\left(\mathcal{F}_{i,h}\right)^T\right) \\ &= \left\|\operatorname{cov}\left(\left(\mathcal{F}_{i,h}\right)^T \mathcal{F}_{i,j}\right)\right\|_F^2 \end{aligned} \tag{2}$$

Equations (1) and (2) are extended by the Hilbert-Schmidt independence criterion (Gretton et al., 2005) to reproduce the inner product of the kernel Hilbert space. Let $K_{i,j} = k(w_j^i, w_j^j)$ and $L_{i,j} = l(w_h^i, w_h^j)$, where $k$ and $l$ are two kernels. There, we use RBF kernel $k(w_j^i, w_j^j) = \exp(-\| w_j^i - w_j^j \|_2^2 / (2\sigma^2))$. To ensure that the correlation index is constant for the isotropic calibration, we choose the bandwidth $\sigma$ as part of the median distance. The empirical estimator of HSIC is:

$$HSIC(K, L) = \frac{1}{\left(N_{i+1}-1\right)^2} tr(KHLH) \tag{3}$$

where $H$ is the centring matrix $H_{N_{i+1}} = I_{N_{i+1}} - \frac{1}{N_{i+1}}$. As a test statistic, HSIC tests whether two sets of variables are independent. It can detect any existing dependence with high probability, as the sample size increases. Normalise HSIC to keep the isotropic scaling unchanged, then the structure of CKA is constructed as follows:

$$CKA(K, L) = \frac{HSIC(K, L)}{\sqrt{HSIC(K, K) HSIC(L, L)}}. \tag{4}$$

### 3.3 Behavioural change of the filters

In order to explore the behavioural change of the filters during the training process, we design a small vanilla ConvNet which has three convolutional layers and one fully connected layer as a toy example. It is implemented on the CIFAR-10 dataset. Every convolutional layer has 32 filters. We use CKA to find the correlation between the filters of the same layer. Then, we visualise the changes of the filters during the training process under different layers and different epochs, which we can see in Figure 1. It shows a self-evolution process of the filters from random initialisation learning to finally obtaining recognition ability. In the early training, the differences between the filters are small, and the learning features are relatively single. As the number of training iterations increases, the correlation between the filters gradually decreases, and the differences gradually increase, indicating that as the number of training iterations increases, the filters capture features in all directions and the learning ability becomes stronger. This explains why researchers are starting to focus on orthogonal initialisation of weights. For example, Xiao et al. (2018) proposed a filter orthogonal initialisation algorithm, which gave the filters the ability to capture features in all

dimensions from the beginning, greatly improved the learning speed. Take a closer look at Figure 1, it can be found that when the network is trained to 10%, the correlation trend of the filters for the first convolutional layer is about the same as when it is trained to 100%, but the trend in the second and third layers are not obvious. It indicates that early layers converge faster. Further exploration, we use CKA to measure the correlation of weights at each layer between the different stages in the training process and when it reaches the final convergence state, which shows in Figure 2. It shows that the deeper the number of layers, the slower the convergence speed. Raghu et al. (2017) used SVCCA to measure the similarity between the features of each layer in the training process and the features they finally trained, and also found that the layers close to the input converge faster, which shows that the neural network converges from the bottom up. Moreover, we also find that after we train to 30%, the correlation trend between the filters is relatively stable, and the most highly correlated filters are always in the stable state. It indicates that increasing the training time of the model does not always lead to a corresponding increase in differences between filters in the late training time. Some filters have been learning features of the same dimension. Under these circumstances, we can prune the highly correlated filters in the late training process.

To further prove our idea that the highly correlated filters are always in the stable state in the later stages of CNN training. We can prune one of the correlative filters with strong correlation between two and replace one with the other, which is shown in Figure 3. For example, suppose we have three filters, each of which is a three-dimension vector: $A$ = (2, 2, 2), $B$ = (2, 2.1, 2), $C$ = (0.6, 0.4, 0.2). We will find that $A$ and $B$ are statistically correlated; they have the same contribution to the network, so pruning anyone of $A$ or $B$ is reasonable. But $C$ is different from them so it cannot be replaced. We experiment on the Alexnet model with CIFAR-10 dataset as a toy example. Under normal training, the classification accuracy of the Alexnet model converges to 87.36%. Then at the same initialisation, the neural network is retrained with the same learning rate and epochs. But when accuracy reaches 10%, 30%, 50% and 80%, we respectively prune filters with different pruning ratios and then observe the final accuracy, which we can see in Figure 4(a). The pruning rate is set the same for each layer. We find that pruning the filters at high precision has less effect on Alexnet performance than at low. It implies that pruning the filters at an early training stage undermines the self-evolution of the model. We further observe the final accuracy of pruning different rates of filters on Alexnet after its performance reaches 50%, which is shown in Figure 4(b).

Interestingly, we find that the final accuracy of pruning the filter when Alexnet performance reached 70%, 80% is almost identical. So it verifies that in the later stage of CNN training, the most highly correlated filters are in a stable state, while also indicates that improvement of the network performance in the later stage is enhanced by the most highly differentiated filters with greater differences. Comparing the changes in the filter correlation when training reaches 30% and 100% in Figure 1, it can be found that the correlation of most filters with small correlation becomes smaller.

### *3.4 FPC algorithm*

Through the above analysis, we find that the highly correlated filters in the post-training period do not greatly improve the performance of the model. Therefore, we can remove some similarly-functioning filters in the later stage of the training to reduce the complexity of the model. Not only can this approach ultimately result in a smaller, faster model, but it can also reduce the amount of computation during training. In summary, The FPC is summarised in Algorithm 1.

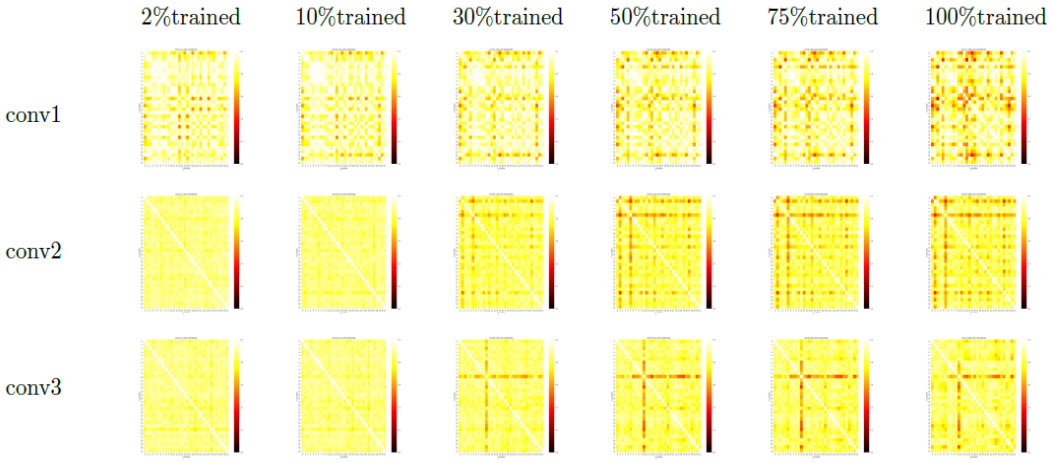
**Algorithm 1** Algorithm description of FPC

**Input:** training data: $X$
**Given:** pruning rate $P_i$
**Initialise:** model parameter $W = W^i$, $0 \le i \le L$
**for** $epoch = 1$; $epoch \le epoch_{\max}$; epoch + + **do**
  Update the model parameter W based on X
  **if** accuracy close to convergent accuracy **then**
    **for** $i = 1$; $i \le L$; $i$ + + **do**
      Calculate $s = \mathrm{CKA}(\mathcal{F}_{i,j}, \mathcal{F}_{i,h})$
      Sort s
      Find $N_{i+1}P_i$ filters
      Zeroise selected filters
      Selected parameters are not updated
    **end for**
  **end if**
**end for**
Obtain the compact model $W^*$ from $W$
**Output:** the compact model and its parameters $W^*$

### *3.5 Computation complexity analysis*

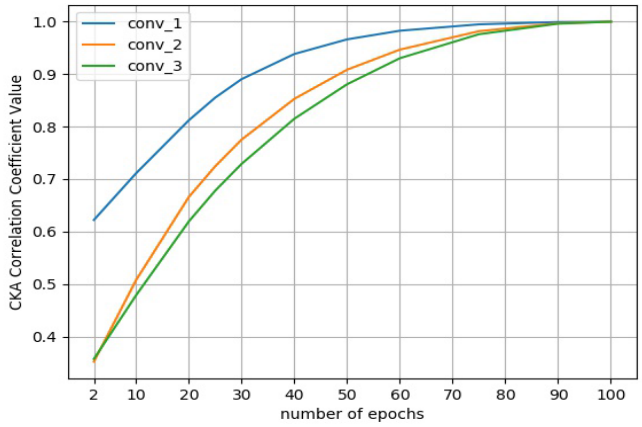
We analyse the acceleration process of the model from theory. For the $i^{\text{th}}$ layer, suppose $N_i$ and $N_{i+1}$ denote the input channels and output channels, respectively. $H_i \times W_i$ represents a feature map. The pruning rate of the $i^{\text{th}}$ layer is set to $P_i$. When the $i^{\text{th}}$ layer is pruned, $N_{i+1}P_i$ filters are reduced, and $N_{i+1}P_i$ feature maps are correspondingly reduced. For the $(i + 1)^{\text{th}}$ layer, the output dimension of $N_{i+1}P_i \times H_{i+1} \times W_{i+1}$ feature maps become the input of the $(i + 1)^{\text{th}}$ layer. Setting the pruning rate to $P_{i+1}$, the remaining amount of calculation is $[N_{i+1}(1 - P_i) \times N_{i+2}(1 - P_{i+1}) \times H_{i+2} \times W_{i+2}]$. Therefore, the reduced calculation ratio is $[1 - (1 - P_i)(1 - P_{i+1})]$, which greatly accelerates the inference speed of the model.

**Figure 1** Changes in the filters between different layers and different epochs (see online version for colours)



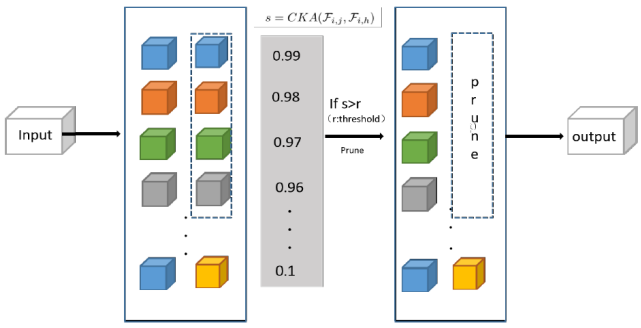

Notes: We use CKA to measure the correlation between any two filters in each layer to form a symmetrical correlation matrix. The darker the colour, the less the correlation. We find that with the increase of training time, the differences between the filters increase.

**Figure 2** CKA correlation between the weights of different epochs and the final convergence weights per layer (see online version for colours)



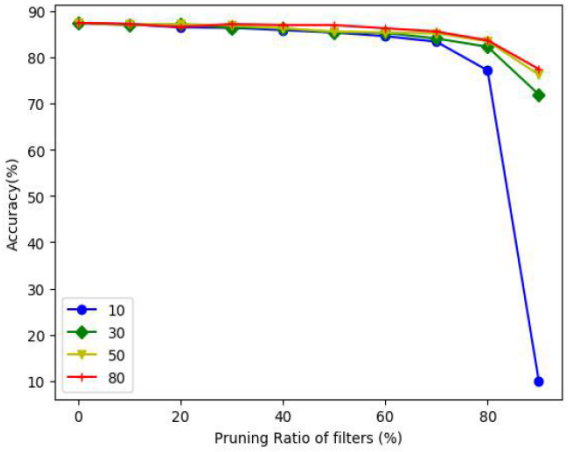

Note: For example, the correlation between the weight of the first convolution layer after 2 epochs and the weight after 100 epochs is 0.62.

**Figure 3** Measure the correlation between any two filters via CKA (see online version for colours)

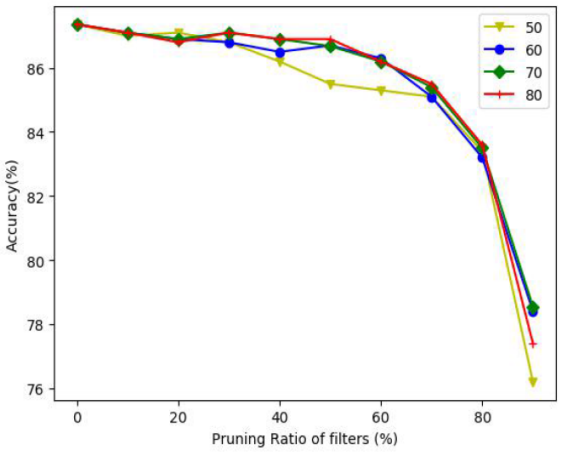



Note: We measure the correlation between any two filters through CKA, and then rank their values from large to small, if the value is greater than a certain threshold $r$ ($r$ is determined by the pruning ratio $P_i$) of each layer, prune one of them.

**Figure 4** Changes in performance of Alexnet with pruning filters respectively in different pruning ratio when the performance reached different accuracy (see online version for colours)




(a)




(b)

## 4 Experiment

### *4.1 Benchmark datasets and experimental setting*

We evaluate FPC for Alexnet and ResNet on two benchmarks: CIFAR-10 and CIFAR-100. Both CIFAR datasets (Krizhevsky et al., 2009) contain 60,000 32 × 32 colour images, in which 50,000 training images and 10,000 testing images are included. The CIFAR-10 dataset is categorised into ten classes, and the CIFAR-100 is categorised into 100 classes.

Our models are implemented on the Pytorch framework. In particular, For CIFAR-10 dataset, following the Alexnet as the base network structure, we train the network for 100 epochs with the constant learning rate of 0.01 as the pre-trained model. For fair comparison, Alexnet are retrained with the same epochs and learning rate, in the later stages of training (accuracy is about 70%), filters are pruned with 10%, 20%, 30%, …, 90% rate respectively, then we compare our final results with the accuracy of pre-trained model and recent advanced filter pruning accelerated method. In order to prove the generality of the FPC algorithm, it is also evaluated on the larger network such as ResNet-56 and ResNet-110. For ResNet-56, 110, firstly, the network is trained for 60 epochs with the constant learning rate of 0.1, and then trained 60 epochs with the learning rate of 0.01. Finally, it continues to be trained 80 epochs with the learning rate of 0.001 to achieve convergence accuracy. Similarly, we prune the filters with different pruning rates in the later stage of training and observe the performance changes. For CIFAR-100 dataset, testing FPC on the ResNet-56, 110 network set as above also brings good results. For the pruning step, we just need to prune all convolution layers with the same pruning rate at same time, which is the same as He et al. (2018b, 2019a). FPC is compared with previous acceleration algorithms, e.g., PFEC (Li et al., 2016), CP (He et al., 2017), AMC (He et al., 2018a), SFP (He et al., 2018b), MFP (He et al., 2019b), MIL (Dong et al., 2017) and FPGM (He et al., 2019a). Not surprisingly, our FPC method achieves the better result. (The method with * in Table 2 is implemented with the code in its original article, without * is a comparison with the results in the original paper).

### *4.2 Evaluation protocol*

FLOPs are the number of floating point operations per second, which is a widely-used measure to test the model computational complexity. We follow the criterion in Molchanov et al. (2016), the formulation is described as below.

For convolutional layers:

$$FLOPs = 2HW\left(C_{in}K^2 + 1\right)C_{out} \tag{5}$$

where $H$, $W$ are the size of the input feature map, $K$ is the size of the kernel, and $C_{in}$ and $C_{out}$ are the input and output channels, respectively. For fully connected layers:

$$FLOPs = \left(2C_{in} - 1\right)C_{out} \tag{6}$$

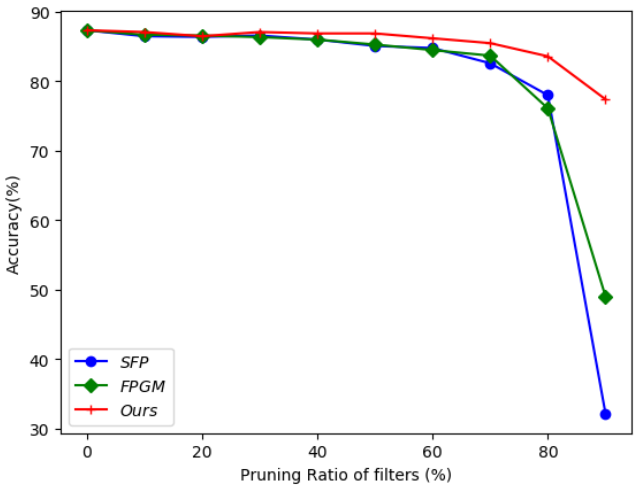
where $C_{in}$ and $C_{out}$ are the input and output channels.

### *4.3 Results on CIFAR-10*

#### *4.3.1 Alexnet base network*

For the CIFAR-10 dataset, we test our FPC on Alexnet, and set the pruning rate to 0.1, 0.2, …, 0.9, respectively. Figure 5 explains the comparison between our method and SFP (He et al., 2018b) and FPGM (He et al., 2019a). As we can see, our approach achieves better performance than others. When pruning 10% and 20% filters respectively, the performance of our method is not obviously different from SFP, FPGM. But after 20%, the SFP and FPGM algorithms cause the model's performance to decrease significantly, and the FPC has only a small change, especially after 80%. In addition, we prune 60% of the filters, and the performance decreases by less than 1%, which greatly reduces the complexity of the model. These results demonstrate that small networks also have great redundancy and our method is suitable for pruning small networks.

**Figure 5** Compare our method with SFP and FPGM (see online version for colours)




#### *4.3.2 ResNet-56, 110 base network*

We also evaluate our method on two deeper networks to show its strength. As a deep and powerful network, ResNet is our primary choice. We utilise ResNet-56, 110 as our base networks. The results are illustrated in Table 1. For ResNet-56, we set two different pruning rates: 40%, 50%. Compared with recent filter pruning methods, our FPC achieves advanced performance. For example, CP (He et al., 2017) with fine-tuning accelerates ResNet-56 by 50% speedup ratio with 1% accuracy drop, whereas FPC without fine-tuning attains 52.6% speedup ratio with only 0.69% accuracy loss. Compared with SFP (He et al., 2018b), when pruning 52.6% FLOPs of ResNet-56, our FPC has only 0.69% accuracy drop, which is much less than SFP (He et al., 2018b) (1.33%). For ResNet-110, we set 30% and 40% pruning rates. FPC achieves the same speedup with SFP (He et al., 2018b), but its accuracy exceeds by 0.65%. That is all thanks to the effectiveness of our methods.

**Table 1** Comparison of pruning ResNet-56, 110 on CIFAR-10

| *Depth* | *Method* | *Fine-tune?* | *Baseline acc. (%)* | *Accelerated acc. (%)* | *Acc.↓ (%)* | *FLOPs* | *FLOPs↓ (%)* |
|---|---|---|---|---|---|---|---|
| 56 | PFEC | × | 93.04 | 91.31 | 1.75 | 9.09E7 | 27.6 |
| | CP | × | 92.80 | 90.90 | 1.90 | - | 50.0 |
| | CP | √ | 92.80 | 91.80 | 1 | - | 50.0 |
| | AMC | √ | 92.80 | 91.90 | 0.90 | - | 50.0 |
| | SFP | × | 93.59 | 92.26 | 1.33 | 5.94E7 | 52.6 |
| | MFP | × | 93.59 | 92.76 | 0.83 | 5.94E7 | 52.6 |
| | Ours (40%) | × | *93.64* | *92.95* | *0.69* | *5.94E7* | *52.6* |
| | Ours (50%) | × | *93.64* | *92.39* | *1.25* | *4.62E7* | *63.2* |
| 110 | MIL | × | 93.63 | 93.44 | 0.19 | - | 34.2 |
| | PFEC | × | 93.53 | 92.94 | 0.61 | 1.55E8 | 38.6 |
| | PFEC | √ | 93.53 | 93.30 | 0.2 | 1.55E8 | 38.6 |
| | SFP | × | 93.68 | 93.38 | 0.30 | 1.50E8 | 40.8 |
| | Ours (30%) | × | *94.0* | *94.03* | *–0.03* | *1.50E8* | *40.8* |
| | Ours (40%) | × | *94.0* | *93.70* | *0.3* | *1.21E8* | *52.3* |

Notes: In 'fine-tune?' column, '√' and '×' represent whether to use the pre-trained model as initialisation or not, respectively. The 'Accu.Drop' indicates how much the accuracy of the pruned model decreases compared to the baseline model, so a negative number means that the pruned model is more accurate than the baseline model.

**Table 2** Comparison of pruning ResNet on CIFAR100

| *Depth* | *Method* | *Fine-tune?* | *Baseline top-1 acc. (%)* | *Accelerated top-1 acc. (%)* | *Baseline top-5 acc. (%)* | *Accelerated top-5 acc. (%)* | *Top-1 acc.↓ (%)* | *Top-5 acc.↓ (%)* | *FLOPs↓ (%)* |
|---|---|---|---|---|---|---|---|---|---|
| 56 | SFP* | × | 72.56 | 70.18 | 92.5 | 91.8 | 2.38 | 0.7 | 52.6 |
| | FPGM* | × | 72.56 | 70.53 | 92.5 | 92.13 | 2.03 | 0.37 | 52.6 |
| | Ours (10%) | × | 72.56 | *72.71* | 92.5 | *92.72* | *–0.15* | *–0.22* | 14.7 |
| | Ours (40%) | × | 72.56 | *70.86* | 92.5 | *92.46* | *1.7* | *0.04* | *52.6* |
| 110 | FPGM* | × | 74.03 | 72.24 | 92.71 | 92.57 | 1.79 | 0.14 | 52.3 |
| | SFP* | × | 74.03 | 72.71 | 92.71 | 92.61 | 1.32 | 0.1 | 52.3 |
| | Ours (10%) | × | 74.03 | *74.21* | 92.71 | *93.01* | *–0.18* | *–0.3* | 14.6 |
| | Ours (40%) | × | 74.03 | *73.16* | 92.71 | *92.74* | *0.87* | *–0.04* | *52.3* |

Note: Fine-tune? and Accu.Drop have the same meaning with Table 1.

### *4.4 Results on CIFAR-100*

*ResNet-56, 110 base network*, for CIFAR-100 dataset, we evaluate our approach on ResNet-56, 110. As we can see in Table 2, compared with recent two method SFP (He et al., 2018b) and FPGM (He et al., 2019a), our FPC achieves better performance. For example, FPC without fine-tuning attains the same acceleration with He et al. (2019a) on ResNet-110, but its top-1 accuracy exceeds by 0.92% and top-5 accuracy exceeds by 0.17%. SFP (He et al., 2018b) and FPGM (He et al., 2019a) both pruning filters from beginning, but they will restore the model's capacity in the next training, which will not reduce the amount of calculation during the training process. But once our FPC prunes filters during the training, model capacity will no longer restore, which greatly improves computing efficiency. Moreover, pruning a small number of filters will also improve the generalisation ability of the model. For instance, FPC prunes 10% filters on ResNet-56, where top-1 accuracy improves 0.15 and top-5 accuracy improves 0.22. It can be seen that our FPC brings great convenience.

**Table 3** CKA compare with other filter pruning criterion

| *Criterion* | *Acc($P_i$ = 0.5)* | *Acc($P_i$ = 0.6)* |
|---|---|---|
| Stand (baseline) | 87.3 | 87.3 |
| Ortho | 85.4 | 84.4 |
| Person | 86.0 | 84.6 |
| Random | 86.1 | 85.1 |
| CKA | *86.9* | *86.4* |

### *4.5 Compare with other criterion*

This work prunes redundant filters through CKA (Kornblith et al., 2019) which can detect any existing dependencies. Prakash et al. (2019) and Singh et al. (2018b) use ortho and person criteria to evaluate the strength of filter correlation, respectively. But they can only find linear correlations, this

is a limitation for them. We combine ortho, person and random criterion with our pruning method and experiment on Alexnet with CIFAR-10. Setting the pruning ratio to 0.5 and 0.6, respectively. From Table 3, it shows that CKA works better than other indicators.

## 5 Conclusions and future work

Most filter pruning methods need fine-tuning to reduce performance degradation. These pruning methods because the black box of neural networks brings some repeated efforts. In order to solve this problem, in this work, we analyse the evolution of the filter during the training process though CKA. The following points were found:

1 The learning ability of the filter is proportional to the training time.

2 In the early training, the features learned by the filters are relatively single.

3 Convolution layer converges from bottom to top.

4 In the later stage of training, most of the filters with strong correlation are in a stable state, and the performance of the model in the later period is mostly improved by the difference of the filters with strong differences.

According to the discovery 4, we propose a model acceleration algorithm FPC, which can prune highly correlated filters in the later stage of training.

FPC achieves the advanced performance in several benchmarks. However, this work only explores the correlation changes of filters at same layer, it does not pay attention to the interaction between filters at different layers, therefore, this is a direction for future efforts, and we also try to combine FPC with other acceleration algorithms, e.g., quantisation and knowledge distilling, to improve the acceleration performance to a higher stage.

## Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and suggestions. This work is supported by the National Key Research and Development Program (No. 2018YFB1004300), National Natural Science Fund of China (Nos. 61672332, 61872226), Shanxi Key Research and Development Program (International Scientific and Technological Cooperation) (201903D421003), and Research Project for Overseas Returnees in Shanxi Province (No. 2017023).

## References

Baker, B., Gupta, O., Naik, N. and Raskar, R. (2016) *Designing Neural Network Architectures Using Reinforcement Learning*, arXiv preprint arXiv: 1611.02167.

Dong, X., Huang, J., Yang, Y. and Yan, S. (2017) 'More is less: a more complicated network with less inference complexity', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.5840–5848.

Gretton, A., Bousquet, O., Smola, A. and Schölkopf, B. (2005) 'Measuring statistical dependence with Hilbert-Schmidt norms', *International Conference on Algorithmic Learning Theory*, pp.63–77.

Han, S., Pool, J., Tran, J. and Dally, W. (2015) 'Learning both weights and connections for efficient neural network', *Advances in Neural Information Processing Systems*, pp.1135–1143.

He, K., Zhang, X., Ren, S. and Sun, J. (2016) 'Deep residual learning for image recognition', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.770–778.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L-J. and Han, S. (2018a) 'AMC: AutoML for model compression and acceleration on mobile devices', *International Conference on Machine Learning*, pp.815–832.

He, Y., Kang, G., Dong, X., Fu, Y. and Yang, Y. (2018b) *Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks*, arXiv preprint arXiv: 1808.06866.

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

He, Y., Liu, P., Zhu, L. and Yang, Y. (2019b) *Meta Filter Pruning to Accelerate Deep Convolutional Neural Networks*, arXiv preprint arXiv: 1904.03961.

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

Hinton, G., Vinyals, O. and Dean, J. (2015) *Distilling the Knowledge in a Neural Network*, arXiv preprint arXiv: 1503. 02531.

Hu, H., Peng, R., Tai, Y-W. and Tang, C-K. (2016) *Network Trimming: A Data-driven Neuron Pruning Approach Towards Efficient Deep Architectures*, arXiv preprint arXiv: 1607.03250.

Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R. and Criminisi, A. (2015) *Training CNNs with Low-rank Filters for Efficient Image Classification*, arXiv preprint arXiv: 1511. 06744.

Kim, J., Park, S. and Kwak, N. (2018) 'Paraphrasing complex network: network compression via factor transfer', *Advances in Neural Information Processing Systems*, pp.2760–2769.

Kornblith, S., Norouzi, M., Lee, H. and Hinton, G. (2019) *Similarity of Neural Network Representations Revisited*, arXiv preprint arXiv: 1905.00414.

Krizhevsky, A., Hinton, G. et al. (2009) *Learning Multiple Layers of Features from Tiny Images*, Computer Science Department, University of Toronto, Tech. Rep.

Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P. (2016) *Pruning Filters for Efficient ConvNets*, arXiv preprint arXiv: 1608.08710.

Li, Y., Yosinski, J., Clune, J., Lipson, H. and Hopcroft, J.E. (2015) 'Convergent learning: do different neural networks learn the same representations?', *FE@ NIPS*, pp.196–212.

Liu, C., Wu, Y., Lin, Y. and Chien, S. (2017) *A Kernel Redundancy Removing Policy for Convolutional Neural Network*, arXiv preprint arXiv.

Long, J., Shelhamer, E. and Darrell, T. (2015) 'Fully convolutional networks for semantic segmentation', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.3431–3440.

Luo, J-H. and Wu, J. (2017) *An Entropy-based Pruning Method for CNN Compression*, arXiv preprint arXiv: 1706.05791.

Luo, J-H., Wu, J. and Lin, W. (2017) 'Thinet: a filter level pruning method for deep neural network compression', *Proceedings of the IEEE International Conference on Computer Vision*, pp.5058–5066.

Molchanov, P., Tyree, S., Karras, T., Aila, T. and Kautz, J. (2016) *Pruning Convolutional Neural Networks for Resource Efficient Inference*, arXiv preprint arXiv: 1611.06440.

Morcos, A., Raghu, M. and Bengio, S. (2018) 'Insights on representational similarity in neural networks with canonical correlation', *Advances in Neural Information Processing Systems*, pp.5727–5736.

Prakash, A., Storer, J., Florencio, D. and Zhang, C. (2019) 'Measuring statistical dependence with Hilbert-Schmidt norms', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.10666–10675.

Qin, Z., Yu, F., Liu, C. and Chen, X. (2018) *Functionality-oriented Convolutional Filter Pruning*, arXiv preprint arXiv: 1810. 07322.

Raghu, M., Gilmer, J., Yosinski, J. and Sohl-Dickstein, J. (2017) 'SVCCA: singular vector canonical correlation analysis for deep learning dynamics and interpretability', *Advances in Neural Information Processing Systems*, pp.6076–6085.

Rastegari, M., Ordonez, V., Redmon, J. and Farhadi, A. (2016) 'XNOR-Net: ImageNet classification using binary convolutional neural networks', *European Conference on Computer Vision*, pp.525–542.

Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V. and Kurakin, A. (2017) 'Large-scale evolution of image classifiers', *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70, pp.2902–2911.

Schroff, F., Kalenichenko, D. and Philbin, J. (2015) 'Facenet: a unified embedding for face recognition and clustering', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.815–823.

Simonyan, K. and Zisserman, A. (2014) *Very Deep Convolutional Networks for Large-scale Image Recognition*, arXiv preprint arXiv: 1409.1556.

Singh, B., Najibi, M. and Davis, L.S. (2018a) 'SNIPER: efficient multi-scale training', *Advances in Neural Information Processing Systems*, pp.9310–9320.

Singh, P., Verma, V.K., Rai, P. and Namboodiri, V.P. (2018b) *Leveraging Filter Correlations for Deep Model Compression*, arXiv preprint arXiv: 1811.10559.

Tai, C., Xiao, T., Zhang, Yi., Wang, X. et al. (2015) *Convolutional Neural Networks with Low-rank Regularization*, arXiv preprint arXiv: 1511.06067.

Wang, D., Zhou, L., Zhang, X., Bai, X. and Zhou, J. (2018) *Exploring Linear Relationship in Feature Map Subspace for ConvNets Compression*, arXiv preprint arXiv: 1803.05729.

Wang, W., Fu, C., Guo, J., Cai, D. and He, X. (2019) *COP: Customized Deep Model Compression Via Regularized Correlation-based Filter-level Pruning*, arXiv preprint arXiv: 1906.10337.

Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S.S. and Pennington, J. (2018) 'Dynamical isometry and a mean field theory of CNNs: how to train 10,000-layer vanilla convolutional neural networks', *International Conference on Machine Learning*, pp.5389–5398.

Ye, J., Lu, X., Lin, Z. and Wang, J.Z. (2018) *Rethinking the Smaller-norm-less-informative Assumption in Channel Pruning of Convolution Layers*, arXiv preprint arXiv: 1802. 00124.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T. and Lipson, H. (2015) 'Understanding neural networks through deep visualization', *International Conference on Machine Learning – Deep Learning Workshop 2015*, p.12, p.7, p.8.

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

Zhang, X., Zou, J., He, K. and Sun, J. (2015) 'Accelerating very deep convolutional networks for classification and detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp.1943–1955.

Zhu, F., Zhu, L. and Yang, Y. (2019) 'Sim-Real joint reinforcement transfer for 3d indoor navigation', *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp.11388–11397.

Zhuo, H., Qian, X., Fu, Y., Yang, H. and Xue, X. (2018) *SCSP: Spectral Clustering Filter Pruning with Soft Self-adaption Manners*, arXiv preprint arXiv: 1806.05320.