RESEARCH ARTICLE

PCA driven mixed filter pruning for efficient convNets

Waqas Ahmed¹, Shahab Ansari¹, Muhammad Hanif^{1*}, Akhtar Khalil²**1** Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology Topi, Swabi, KPK, Pakistan, **2** iFhaja Research Consultation, Peshawar, KPK, Pakistan

* These authors contributed equally to this work.

* muhammad.hanif@giki.edu.pk

Abstract

Deployment of the deep neural networks (DNNs) on resource-constrained devices is a challenging task due to their limited memory and computational power. In most cases, the pruning techniques do not prune the DNNs to full extent and redundancy still exists in these models. Considering this, a mixed filter pruning approach based on principal component analysis (PCA) and geometric median is presented. First, a pre-trained model is analyzed by using PCA to identify the important filters for every layer. These important filters are then used to reconstruct the network with a fewer number of layers and a fewer number of filters per layer. A new network with optimized structure is constructed and trained on the data. Secondly, the trained model is then analyzed using geometric median as a base. The redundant filters are identified and removed which results in further compression of the network. Finally, the pruned model is fine tuned to regain the accuracy. Experiments on CIFAR-10, CIFAR-100 and ILSVRC 2017 datasets show that the proposed mixed pruning approach is feasible and can compress the network to greater extent without any significant loss to accuracy. With VGG-16 on CIFAR-10, the number of operations and parameters are reduced to 18.56× and 3.33×, respectively, with almost 1% loss in the accuracy. The compression rate for AlexNet on CIFAR-10 dataset is 2.61× and 4.85× in terms of number of operations and number of parameters, respectively, with a gain of 1.2% in the accuracy. On CIFAR-100, VGG-19 is compressed by 16.02 X in terms of number of operations and 36× in terms of number of parameters with a 2.6% loss of accuracy. Similarly, the compression rate for VGG-19 network on ILSVRC 2017 dataset is 1.87× and 2.4× for operations and parameters with 0.5% loss in accuracy.

OPEN ACCESS

Citation: Ahmed W, Ansari S, Hanif M, Khalil A (2022) PCA driven mixed filter pruning for efficient convNets. PLoS ONE 17(1): e0262386. <https://doi.org/10.1371/journal.pone.0262386>

Editor: Nguyen Quoc Khanh Le, Taipei Medical University, TAIWAN

Received: May 25, 2021

Accepted: December 23, 2021

Published: January 24, 2022

Copyright: © 2022 Ahmed et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: ***** PA @ACCEPT ***** The data underlying the results presented in the study will be made available on demand.

Funding: The authors received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

Convolutional Neural Networks (CNNs) have achieved state of the art performance in many applications such as face recognition [1], object detection [2], semantic segmentation [3] and other classification tasks. Most of the CNN architectures are deeper and wider and contain a large number of parameters and operations which make them quite difficult to be deployed on power-constrained devices. In most of the cases, these networks are trained and deployed in

Table 1. Parameters overview of different CNN architectures.

S. No.	Network	Parameters (M)
1	VGG-19	143
2	VGG-16	138
3	AlexNet	61
4	SqueezeNet	1.24

<https://doi.org/10.1371/journal.pone.0262386.t001>

computationally-rich devices such as graphics processing unit (GPU) for training and multi-core systems for deployment. However, these networks are also being deployed on mobile devices for certain applications such as voice and face recognition etc. In contrast to GPUs and multi-core systems, mobile devices have very limited memory and computational power. The modern deep neural networks are computationally expensive and memory intensive and require more computational power for deployment and training, it has become a challenge to bring the advances in neural network technology to mobile devices. Consequently, much work has been done in recent years, focused on reducing the size of pre-trained neural networks, making them capable to be deployed on mobile devices for inferences [4, 5]. The latest architectures such as inception module [6] or residual connection [7] have millions of parameters which require extensive computation and storage power. Table 1 shows the number of parameters (in millions) for some of the recently proposed CNN architectures. These architectures produce state of the art accuracy and most of the designers start with pre-trained networks for transfer learning purposes. These networks are rarely evaluated on the given datasets and only the classifier is trained and fine-tuned. This usually results in redundancy in the network. Therefore, it is of great importance to devise deep neural network models with relatively low complexity and high accuracy.

One of the major approaches for the optimization of CNNs is pruning. Pruning is an effective technique to reduce the size of the network by removing redundant filters or weights without effecting the accuracy [8–11]. It has been observed that the filters of well-established CNN models contain redundancy and removing these filters do not cause any degradation in the accuracy of the model. Fig 1 shows the filters of first convolutional layer of AlexNet trained on ImageNet dataset [12]. The redundancy in the filters is clearly visible.

Pruning can be classified into two categories, i.e. filter pruning [13, 14] and weight pruning [10, 15]. In weight pruning, the weight values which contribute less are directly deleted whereas a whole filter which contributes less is deleted in the filter pruning. Whether it is

Fig 1. Visualization of first layer convolutional filters of AlexNet.

<https://doi.org/10.1371/journal.pone.0262386.g001>

weight pruning or filter pruning, most of the pruning methods [16–18] involve multiple iterations to identify a suitable threshold for pruning. They start with a pre-trained network and compress the network layers one by one to find out pruning threshold for each layer. This process consumes heavy computational resources and time. Moreover, the network is not fully pruned by any of these approaches alone and redundancy still exists in the network. In order to achieve more pruned network, a mixed approach of filter pruning is presented in this paper where the focus is on redundancy rather than importance of filter towards accuracy. The work of [19] is utilized to compress the network both by reduction in number of layers and reduction in number of filters per layer by using the principal component analysis (PCA). The compressed network is then trained, and some redundant filters are pruned based on the approach given by [20].

The rest of the article is organized as follow. The overview of two step pruning method is given in Section 2. Recent work related to compression of neural networks is discussed in Section 3. In Section 4, the methodology of mixed filter pruning approach is discussed in detail. The results of the proposed scheme are discussed in Section 5. In Section 6, we conclude our work with some future recommendations.

CNNs pruning

Neural network compression using PCA

PCA is used for network analysis to get the compressed design having a fewer number of layers and a fewer number of filters in each layer without any retraining iterations [19]. A pre-trained network is selected and the activations of all layers are analyzed using PCA. The number of filters in each layer of the network is determined by the principal components required to explain 99.9% cumulative variance. The number of layers is determined based on when the number of filters in all layers start contracting. A new network with pre-determined number of convolutional layers and number of filters per layer is constructed and trained to get optimized network.

Filter pruning via geometric median

As the centrality of data can be best expressed by geometric median, thus it can be utilized to find the filter which minimizes the summation of distances with other filters [20]. Geometric median is used to get common information of all the filters within a single layer. Then all the filters closed to geometric median are identified based on some threshold. As these filters can be represented by other filters in the layer, therefore pruning these has little impact on the network. These filters are then set to zero. The flow chart in Fig 2 shows the overall procedure of two step pruning of a convolutional neural network.

Fig 2. Flowchart of the proposed PCA driven mixed pruning technique.

<https://doi.org/10.1371/journal.pone.0262386.g002>

Related work

In recent years, a number of CNN models are represented having a large number of parameters and memory cost. These models have yield state of the art performance in many applications [21]. However, one of the major drawbacks of these models is over-parameterization which results in heavy computational costs. To tackle this problem, various techniques are proposed to get more robust but computationally efficient models. The work on the acceleration and optimization of CNNs can be divided into four main categories [22], i.e. knowledge distilling, compact convolutional filters, low rank factorization and quantization and pruning. In pruning, multiple techniques were developed to filter out and remove the redundant filters or weights. Some of the main techniques are discussed below.

Weight pruning

In this type of pruning techniques, the insignificant weights are identified using a predefined criteria and pruned. In [8], a deep compression technique is proposed containing three stages. They first pruned the network while retaining only important weights, then weight sharing was enforced by quantization methods and finally Huffman coding was applied to further reduce the size of the network. They successfully reduced the size of the VGG16 by 49×, however this method requires specialized hardware due to the formation of unstructured sparsity [23].

In [9], the authors suggested an approach for energy-efficient implementation of hardware for large scale neural networks. They have converted a neural network into an approximate neural network by analyzing the importance of a neuron. This was done by using the back propagation mechanism and then subsequently removing the least important connections. This technique is a good approach towards network compression however it requires repeated iterations and consumes huge amount of resources. Similarly, [16, 24] have implemented the weight pruning techniques in a similar manner which results in unstructured sparsity and requires repeated iteration to identify the weights that are less important.

Filter pruning

In this technique, the entire filter is removed if found insignificant towards the final accuracy of the network. In [25], authors have proposed a scheme to prune the convolutional kernels in neural networks by interleaving criteria-based pruning with fine tuning. They have approximated the changes in the cost function of network parameters by using Taylor expansion. The authors in [11] have proposed the method to prune the redundant filters and retrain the accuracy. They have used the L1-norm to estimate the importance of the filter and filtered out the unimportant filters thus leaving no unstructured sparsity in the pruned network. In [10], authors have removed the entire insignificant filters along with their feature maps to reduce the computational costs significantly. Their work is a good attempt to compress the network and they achieved up to 34× reduction in FLOPs for VGG16 on CIFAR10. Similarly, authors in [26] used the statistics of the next layer to prune the filters in the current layer. These methods do require iterative retraining which consumes heavy computational resources and time. In [25], authors have suggested analysis of all the layers together, however it too requires many repetitive iterations.

In [27], authors have removed the filters by introducing multiple losses to each layer. They selected the least important filters by using these losses and the process continues until a stopping criterion meets. This process is done for all layers and it takes multiple iterations.

Recently [28] proposed a filter pruning technique based on maximizing the number of zeroes in filters and eliminating the filters having more non-zero parameters. Their idea is

based on the fact that if there is any zero value out of two inputs to a multiplier then multiplication operation can be skipped. They are successful in replacing 82% of multiplications with zero-skip multiplications that do not switching energy in multiplication circuits. However, zero-skip multiplication operations are still performed and are not completely eliminated.

Weight matrices approximation

The third category of techniques to reduce the size and to accelerate the network is the approximation of weight matrices [29–31]. This is done either by quantization or low rank matrix approximation methods. In [32] and [18], authors have used techniques of quantization to reduce the size of the network. Quantization also involves multiple iteration which becomes a bottleneck to large networks. The authors in [33] have proposed to train the CNNs with binary weights and activations. They have divided the network into segments and trained all the segments in a sequential way. The idea of dividing the network into segments is due to the fact that a group can be reconstructed by combining the set of similar binary branches. However, this technique requires custom training methods. In [17], authors have proposed a similar technique to reconstruct the network having fewer parameters however the compression is mainly achieved by regularization. One of the major drawbacks of the regularization process is that it adds more hyper parameters into the process which takes more iterations to achieve optimized model.

Learning sparsity patterns

The techniques in this category of pruning usually modify the loss function for the model optimization [14]. The weights are detected that minimize the loss while satisfying the pruning criterion. In [34], authors have proposed a method to prune the network during training by using L_0 -norm regularization in neural networks. The scheme proposed to set the identified weights to zero by including a collection of non-negative stochastic gates. These methods do not guarantee structured sparsity in the network and takes longer training time. The authors in [35] have presented a non-structured architecture that contains a nucleus of connections at the start and the connections are optimized during the training process. This method requires more computational resources to optimize the network.

Apart from these four pruning categories, there are some methods that combine different techniques together. In [36], authors have combined weight pruning along with filter pruning to prune the network to the maximum extent. They have used information from the next layer to prune the filters in the current layer. But their method requires a huge amount of computation; first for identifying the insignificant filters and second for the fine tuning to regain accuracy.

Mixed filter pruning

Most of the pruning methods involve multiple iterations to identify a suitable threshold for pruning. They start with a pre-trained network and compress the network's layers one by one to find out pruning threshold for each layer. This process consumes heavy computational resources and time. Moreover, the network is not fully pruned by any of these approaches alone and redundancy still exists in the network. In order to achieve a more optimized network, a 2-step technique of filter pruning is presented in this section.

First, PCA is used to analyze the network to get the compressed design having fewer number of layers and fewer number of filters in each layer without any retraining iterations. A pre-trained network is taken, and the activations of all layers are analyzed using PCA. The number of filters in each layer of the network is determined by the principal components required to

explain 99.9% cumulative explained variance. The number of layers are determined based on contracting of filters in all layers. A new network with determined number of convolutional layers and number of filters per layer is constructed and trained to get optimized network.

Second, geometric median is used to get common information of all the filters within a single layer. Then all the filters are identified which are nearest to geometric median in that layer, based on some threshold. These filters can be represented by other filters in the layer, hence pruning them has little impact on the network. These filters are then removed from the network.

Step1: Pruning filters via PCA

PCA is used to analyze a pre-trained network without any retraining iterations to get an optimized design. The redundancy in the network is minimized in terms of width (number of filters per layer) and depth (number of layers). The PCA analysis is performed on a pre-trained network by analyzing activations of all the layers simultaneously. The width of the network is determined by the number of principal components required to explain 99.9% cumulative explained variance. Depth of the network is determined based on when the width of the layers starts contracting.

In complex data, difficulty in visualization and performing computations on data increases with the increase in the data dimensions. PCA is an analysis technique for the complex data in which only the significant dimensions are retained and the redundant dimensions are removed. For better understanding of the PCA, following terms are defined.

Variance is the measurement of variability in the data or how far the data points are with respect to each other. It can be computed as the average squared deviation from the mean as

$$var(x) = \frac{\sum (x_i - \bar{x})^2}{N} \quad (1)$$

Where x_i is the value of one observation x in the i^{th} dimension, $\bar{x}$ is the mean value of all observations and N is the total number of observations.

Covariance measures the degree to which corresponding elements from two ordered data-sets are moving in the same direction. Positive covariance of two variables x and y indicates that x increases with the increase in y . Negative covariance indicates x in decreasing with the increase in y . Zero covariance indicates that both variables are not related. Covariance can be computed as

$$cov(x, y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N} \quad (2)$$

x_i is the value of x in i^{th} dimension Where x_i and y_i represent the value of variable x and y in the i^{th} dimension; respectively. The $\bar{x}$ and $\bar{y}$ are the mean values of x and y ; respectively. N is the total number of values.

For a matrix A , an eigenvector is represented by x such that on the multiplication of A and x the direction of the resultant vector is the same as vector x scaled by λ . Mathematically,

$$Ax = \lambda x \quad (3)$$

Here, A is an arbitrary matrix, λ is an eigenvalue of A and x is the eigenvector corresponding to the eigenvalue.

A covariance matrix for a dataset having four-dimension a, b, c and d is shown as

$$C = \begin{bmatrix} V_a & C_{a,b} & C_{a,c} & C_{a,d} \\ C_{a,b} & V_b & C_{b,c} & C_{b,d} \\ V_{a,c} & C_{b,c} & V_c & C_{c,d} \\ V_{a,d} & C_{b,d} & C_{c,d} & V_d \end{bmatrix} \quad (4)$$

Where V_a represents variance along the dimension a and $C_{a,b}$ represents covariance along the dimensions a and b. For a matrix X having $m \times n$ dimensions where n is the number of data points having m dimension, the covariance matrix can be calculated as

$$C_x = \frac{(X - \bar{X})(X - \bar{X})^T}{n - 1} \quad (5)$$

Following are the major steps involved in computation of PCA:

1. Calculation of the covariance matrix X for the given data points.
2. Calculation of eigenvectors and corresponding eigenvalues.
3. Sorting of eigenvectors based on their eigenvalues in decreasing order.
4. For a certain threshold k , choose first k eigenvectors which are the new k dimensions.
5. Transformation of the original m dimensional data points into new k dimension.

PCA tries to retain high variance along the dimensions, i.e. data to be spread out by removing the correlated dimension. There are two major goals of PCA: (1) to find out linearly independent dimensions so that the data can be represented without any loss, and (2) the original dimension can be reconstructed by those linearly independent dimensions. Once the covariance matrix of the original dataset is calculated, our goal is to transform the original data points so the covariance matrix of the new transformed data points become diagonal matrix. For the m dimensional n data points, the size of covariance matrix is

$$X = m \times n \quad (6)$$

The k principal dimensions are chosen with respect to the k largest eigenvalues. Thus

$$P = k \times m \quad (7)$$

Then input data in reduced dimensions are given as

$$Y = PX \quad (8)$$

That is,

$$(k \times n) = (k \times m) \times (m \times n) \quad (9)$$

This shows n data points having k dimensions. In other words,

$$[new\ data]_{k \times n} = [top\ k\ eigenvectors]_{k \times m} \times [original\ data]_{m \times n} \quad (10)$$

Input data to PCA. It is observed that the filters present in the layers of deep convolutional neural networks are highly correlated and contains redundancy. These filters might be detecting the same feature hence making no contribution towards the network performance.

Fig 3. Flattened activation matrix as input to PCA.

<https://doi.org/10.1371/journal.pone.0262386.g003>

In order to reduce the redundancy present in the CNN, activations are used as feature values of the filters. The feature value of a filter is calculated by its convolution with the input patch. The input to PCA is a two-dimensional matrix representing the input sample as its row and corresponding feature value as its column. In the input matrix to PCA, a data point at the location $[i, j]$ is the activation generated by the convolution of i^{th} input patch with j^{th} filter. This input patch convolves with all the filter making a full row of features values.

Let A_L be the matrix that is obtained by the activations of layer L in a forward pass. The first input patch having size equal to the kernel size convolves with the kernel to produce first pixel of the output activation map. This input patch convolves with M filters to form a row in the output map having dimension $R^{1 \times 1 \times M}$. The next input patch is also convolved with all the filters to form another row in the output activation map. The resultant activation map will be having the dimension $A_L \in R^{N \times H \times W \times M}$, where N is the batch size, H is the height of activation map and W is its width. We can flatten this matrix as

$$A_L \in R^{N \times H \times W \times M} \rightarrow B_L \in R^{D \times M} \quad (11)$$

where $D = N \times H \times W$. The input matrix to the PCA is described in the Fig 3.

The flattening of the matrix results into the collection of $N \times H \times W$ samples in one forward pass. PCA is then performed on B_L . Singular Value Decomposition (SVD) is performed on mean normalized symmetric matrix to get M number of eigenvectors V_i and eigenvalues λ_i . Total variance T is the sum of diagonal elements in the covariance matrix and is equal to the sum of individual variance of the parameters. This is known as trace and is represented as $\text{tr}(B_L^T B_L)$. As $\text{trace } AB = \text{trace } BA$, hence $\text{trace } S = \text{trace } W(\Lambda WT) = \text{trace } (\Lambda WT)W = \text{trace } \Lambda$, i.e. trace of B_L is sum of its eigenvalues

$$\text{tr}(B_L^T B_L) = \sum_{i=1}^M \alpha_{ii}^2 \quad (12)$$

Or

$$\text{tr}(B_L^T B_L) = \sum_{i=1}^M \lambda_i \quad (13)$$

Here each eigenvalue is supposed to be explaining λ_i/T ratio of total variance. As all the eigenvalues are ordered in decreasing order, we can calculate how many cumulative eigenvalues are required to explain 99.9% of the explained total variance. These eigenvalues represent the significant dimensions in the layer and denoted by S_L which is given as

$$S_L = \frac{\sum_{i=1}^{\hat{M}} \lambda_i}{\sum_{i=1}^M \lambda_i} = 0.999 \quad (14)$$

The $\hat{M}$ significant dimensions are helpful in optimizing the depth and width of the network.

Optimizing width and depth of the network. The significant dimensions of the filter space is determined as the number of uncorrelated filters that are able to define 99.9% variance. As discussed above, these PCA matrices are flattened and analyzed by PCA to define the significant dimensions for each layer. These dimensions determine the number of filters per layer in the network. If we consider every layer as a transformation function, in which the expansion of input data into higher dimensions is performed until the data become linearly separable, it implies that the width of the network per layer must be a non-decreasing function. The results show that the number of significant dimension (width of the layer) per layer increase up to certain layer and then starts decreasing. This implies that the layer with lower number of significant dimensions than the preceding layer is contributing no useful transformation towards the final results.

Step2: Pruning filters via geometric median

Geometric median represents the central tendency in higher dimensions. We can compute the central tendency via geometric median as follow [37]. For a set of n points $a_1, a_2, \dots, a_n$ where every $a_i \in R^d$, find a point $x \in R^d$ that minimizes the sum of Euclidian distance to all the points.

$$x = \arg \min_x f(x) \quad (15)$$

where

$$f(x) = \sum_{i=1}^n |x - a_i| \quad (16)$$

For a particular layer L , geometric median can be used to get the common information pertaining to all the filters of L .

$$x^{GM} = \arg \min_x \sum_{j=1}^{N_j+1} |x - F_{i,j}|^2 \quad (17)$$

where $x \in R^{N_i \times K \times K}$.

After finding the geometric median, we can find the filter nearest to geometric median in that layer.

$$F_{i,j^*} = \arg \min_{F_{i,j}} |F_{i,j} - x^{GM}|^2 \quad (18)$$

where $j^* \in [1, N_{i+1}]$.

The filters represented by $F(i, j^*)$ are the redundant filters and can be pruned out without any negative effect towards networks output. Now we can determine which filter in the i th layer has the minimum summation of distances with other filters.

$$F_{i,x^*} = \arg \min_x \sum_{j \in [1, N_{i+1}]} |x - F_{i,j}|^2 \quad (19)$$

where $x \in [F_{i,1}, \dots, F_{i, N_{i+1}}]$.

Here, the filters represented by $F(i, x^*)$ are those filters which share the most common information which means that their information can be replaced by other filters. The effect of removing these filters will be negligible and network can regain the accuracy with fine tuning.

Results and discussion

The PCA driven mixed filter pruning approach was tested on three publicly available datasets and three neural networks. The CIFAR-10 and CIFAR-100 are online available on <https://www.cs.toronto.edu/kriz/cifar.html> and ILSVRC2017 is available on <https://image-net.org/challenges/LSVRC/2017/>. The CIFAR-10 is tested on VGG-16 and AlexNet while CIFAR-100 and ILSVRC2017 are tested on VGG-19. Mixed pruning is applied on trained networks and redundant filters are pruned out. The compressed models are fine tuned for some epochs to regain the accuracy. On the given datasets and networks, mixed pruning has achieved state of the art results. The models are compressed up to 4.85× in number of parameters and 18.56× in operations without any significant loss in accuracy up to 2.65%.

Methods

PCA driven mixed pruning approach is implemented using PyTorch [38] library and was trained on NVIDIA Tesla K80 GPU. The training scheme consists of four different levels:

1. Training the network on given data without any pruning.
2. Training with pruning in terms of depth and width contraction of network based on the approach given by [19].
3. Training with pruning approach given by [20].
4. Training with filter pruning based mixed approach.

The specifications of the networks and datasets used in the experiments are described in Table 2.

Table 2. Networks and datasets used in the experiments.

Sr. No.	Network				Dataset			
	Name	Conv Layers	FC Layers	Parameters	Name	Classes	Train Images	Test Images
1	AlexNet	5	3	60	CIFAR-10	10	50,000	10,000
2	VGG-16	13	3	138	CIFAR-100	100	50,000	10,000
3	VGG-19	16	3	143	ILSVRC2017	1000	1 Million	150,000

<https://doi.org/10.1371/journal.pone.0262386.t002>

Table 3. Configuration of VGG-16 on CIFAR-10.

Sr. No.	Type	Configuration
1	Initial Configuration	[64, 64, M, 128, 128, M, 256, 256, 256, M, 512, 512, 512, M, 512, 512, 512]
2	Significant Dimensions	[46, 48, M, 58, 118, M, 119, 229, 230, M, 223, 323, 372, M, 251, 257, 358]
3	PCA-Based Configuration	[48, 58, M, 118, 229, M, 323, 372]
4	Final Configuration	[30, 36, M, 72, 139, M, 165, 224]

<https://doi.org/10.1371/journal.pone.0262386.t003>

VGG-16 on CIFAR-10

The VGG-16 network with batch normalization was trained on CIFAR-10. The activations of a pre-trained VGG-16 on CIFAR-10 were given to PCA as input to detect the significant dimensions. PCA is used as a dimensionality reduction tool. The significant dimensions represent the filters in a convolutional layer that contributes significantly towards the output. The significant dimension for the VGG-16 network on CIFAR-10 dataset is summarized in Table 3. Based on these significant dimensions, the depth of the network is also compressed along with reduction in number of filters per layer. This phenomenon is shown in the Fig 4. It has been observed that the significant dimensions of the network expand up-to certain limit and then starts contracting. Therefore, it can be concluded that the layer having less or same number of filters than the preceding layer do not contribute towards any significant transformation of input data. In this way, the redundant layers are removed from the network. The final network is compressed to six layers only.

After obtaining the optimized architecture, the network is initialized randomly and trained. The trained network is pruned again based on geometric median. In this approach, the redundant filters are detected and removed. With a little fine tuning of 50 epochs, the network regained its accuracy. The final number of parameters and operations of the network are reduced by 3.33 X and 18.56 X respectively. This is shown on Table 4.

AlexNet on CIFAR-10

AlexNet is a smaller CNN architecture compared to VGG style networks and it is not much over-parametrized. For the analysis of redundancy in the network, the activations obtained by

Fig 4. Compressed architecture of VGG-16 on CIFAR-10.

<https://doi.org/10.1371/journal.pone.0262386.g004>

Table 4. Comparison of VGG-16 on CIFAR-10.

Model	Accuracy	FLOPs	Parameters
Original	93.57%	1×	1×
PCA-Based [19]	93.36%	7.78×	2.82×
FPGM [20]	93.23%	1.42×	1.62×
FPZ [28]	92.63%	5.55×	2.12×
Proposed	92.38%	18.56×	3.33×

<https://doi.org/10.1371/journal.pone.0262386.t004>

Table 5. Configuration of AlexNet on CIFAR-10.

Sr. No.	Type –	Configuration
1	Initial Configuration	[64, 192, 384, 256, 256]
2	Significant Dimensions	[58, 164, 329, 242, 200]
3	PCA-Based Configuration	[58, 164, 329, 242]
4	Final Configuration	[36, 99, 199, 146]

<https://doi.org/10.1371/journal.pone.0262386.t005>

convolution of input patches with the filters of every layer are given to PCA. As a dimensionality reduction tool, PCA is used to identify the significant elements from the input matrix which we call significant dimensions. These significant dimensions are then used to reconstruct the network based on the predetermined length and width. The elements of the network other than significant dimensions are considered redundant and can be discarded. By experiments, it has been observed that the redundancy in the newly constructed network based on the significant dimensions is still present. In order to prune the network further, geometric median based pruning technique is applied which further zeroized the redundant filters. The initial configuration, its significant dimensions and final configuration is shown in Table 5.

The layer wise significant dimensions are the key factor to determine the length of the network. It can be concluded that the layer having less or same number of filters than the preceding layer do not contribute towards any significant transformation of input data. In this way, the redundant layers are removed from the network. The final network contains four layers. This is shown in Fig 5. The network is initialized randomly based on the compressed

Fig 5. Compressed architecture of AlexNet on CIFAR 10.

<https://doi.org/10.1371/journal.pone.0262386.g005>

Table 6. Comparison of AlexNet on CIFAR-10.

Model	Accuracy	FLOPs	Parameters
Original	86.31%	1×	1×
PCA-Based [19]	86.64%	1.69×	4.33×
FPGM [20]	85.34%	2.35×	1.03×
Proposed	87.51%	2.61×	4.85×

<https://doi.org/10.1371/journal.pone.0262386.t006>

architecture obtained from PCA. This network is trained with the accuracy of 85.64% and the trained model is pruned again based on geometric median. Number of parameters in the final model are reduced by 4.85× and operations are reduced by 2.61×. The results are summarized in Table 6.

VGG-19 on CIFAR-100

VGG-19 is a deep convolutional neural network with 16 convolutional layers. This network is composed of sixteen convolutional layers, five pooling layers and three fully connected layers. VGG-19 is comparatively deeper network and it contains more redundancy. There might be a number of filters detecting the same feature and hence making no significant contribution towards the final out-put of the network. Therefore, this network is much overparametrized and can be compressed significantly. For the compression of this model, two step pruning method is proved to be effective. First, the activations of a pre-trained model are given to PCA for the analysis of significant dimensions. The redundant filters are identified by PCA by analyzing the activation maps of a pre-trained VGG-19 model.

The analysis shows that the significant dimensions obtained by PCA are fluctuating across the length of the network. Those layers which have fewer or equal numbers of filters than their preceding layers are treated as insignificant and can be removed from the network as shown in Fig 6.

Once the network is compressed according to the significant dimensions identified by PCA, it is initialized randomly and trained again on the same dataset. Once the required accuracy is obtained, the network is analyzed again for redundancy by using geometric median. To eliminate the redundant filters, filter pruning via geometric median is applied on the trained

Fig 6. Compressed architecture of VGG-19 on CIFAR 100.

<https://doi.org/10.1371/journal.pone.0262386.g006>

Table 7. Configuration of VGG-19 on CIFAR-100.

Sr. No.	Type	Configuration
1	Initial Configuration	[64, 64, 128, 128, 256, 256, 256, 512, 512, 512, 512, 512, 512]
2	Significant Dimensions	[46, 47, 58, 118, 120, 235, 229, 231, 235, 426, 246, 161, 67, 177, 383, 384]
3	PCA-Based Configuration	[45, 47, 58, 118, 120, 235, 426]
4	Final Configuration	[28, 29, 36, 72, 72, 142, 257]

<https://doi.org/10.1371/journal.pone.0262386.t007>

Table 8. Comparison of VGG-19 on CIFAR-100.

Model	Accuracy	FLOPs	Parameters
Original	70.42%	1×	1×
PCA-Based [19]	69.88%	14.41×	7.91×
FPGM [20]	69.21%	1.11×	2.66×
Proposed	67.77%	16.02×	36.42×

<https://doi.org/10.1371/journal.pone.0262386.t008>

network and some unimportant filters are set to zero. The final configuration of the network is shown in Table 7.

The results show significant improvement in the network efficiency in terms of number of operations and parameters without any significant loss in accuracy. The FLOPs are reduced by 16.02× and parameters are reduced by 36.4× with a total loss of 2.65% in accuracy. The results are shown in Table 8.

VGG-19 on ILSVRC2017

The final experiment was performed on ILSVRC2017 dataset with batch normalized VGG-19 network. By using the two-step pruning methods, the activations of a pretrained model are given to PCA for the analysis of significant dimensions. The redundant filters are identified by PCA by analyzing the activation maps of a pretrained VGG-19 model.

The compressed architecture on the network is analyzed again for redundancy by using geometric median and the redundant filters are zeroized. The summary for the filters in different stages is given in Table 9.

The compression in terms of number of parameters and number of operations is not as much significant as in case of CIFAR-100 dataset. This is because ILSVRC2017 is a huge dataset and most of the filters tend to learn some features. The Table 10 shows the results of this experiment.

Table 9. Configuration of VGG-19 on ILSVRC2017.

Sr. No.	Type	Configuration
1	Initial Configuration	[64, 64, 128, 128, 256, 256, 256, 512, 512, 512, 512, 512, 512]
2	Significant Dimensions	[6, 30, 49, 100, 169, 189, 205, 210, 400, 455, 480, 490, 492, 492, 492, 492]
3	PCA-Based Configuration	[6, 30, 49, 100, 169, 189, 205, 210, 400, 455, 480, 490, 492, 492, 492, 492]
4	Final Configuration	[5, 18, 31, 30, 103, 115, 124, 126, 240, 277, 288, 294, 296, 296, 296, 296]

<https://doi.org/10.1371/journal.pone.0262386.t009>

Table 10. Comparison of VGG-19 on ILSVRC2017.

Model	Accuracy	FLOPs	Parameters
Original	74.24%	1×	1×
PCA-Based [19]	74%	1.3×	1.14×
FPGM [20]	73.97%	1.42×	1.27×
Proposed	73.69%	1.87×	2.4×

<https://doi.org/10.1371/journal.pone.0262386.t010>

Conclusion

Deep neural networks have achieved high performance in different data intensive applications. However, their performance comes with a cost of high computational requirement. One possible solution is to compress the network using pruning. Most of the filter pruning techniques are iterative and consumes huge amount of time and computation to compress the network. These techniques have achieved good results however it is observed that the redundancy still exist in the networks and the networks are not fully pruned. A mixed pruning method without the involvement of any repetitive iterations is presented in this paper to remove the redundancy in the convolutional neural networks. First, the trained model is analyzed using PCA and its significant filters are detected by giving the activations as input to PCA. Based on these dimensions, a new network is initialized randomly with a smaller number of layers and smaller number of filters per layer. The newly formed model is trained and analyzed again for redundancy using geometric median. The identified redundant filters are then set to zero. The network is then fine tuned to regain its accuracy with an optimized number of layers.

The proposed network pruning method helps significantly in reducing the computational cost of the network but at the same also cause a minimal loss in accuracy. One of the possible future direction may be working on the pruning with improved accuracy.

Author Contributions

Conceptualization: Waqas Ahmed, Shahab Ansari, Muhammad Hanif.

Data curation: Akhtar Khalil.

Formal analysis: Shahab Ansari, Muhammad Hanif.

Investigation: Waqas Ahmed, Shahab Ansari.

Methodology: Waqas Ahmed, Shahab Ansari, Muhammad Hanif.

Project administration: Akhtar Khalil.

Resources: Akhtar Khalil.

Software: Waqas Ahmed, Akhtar Khalil.

Supervision: Shahab Ansari, Muhammad Hanif, Akhtar Khalil.

Validation: Waqas Ahmed, Shahab Ansari, Muhammad Hanif, Akhtar Khalil.

Visualization: Shahab Ansari, Akhtar Khalil.

Writing – original draft: Waqas Ahmed, Shahab Ansari, Muhammad Hanif.

Writing – review & editing: Waqas Ahmed, Shahab Ansari, Muhammad Hanif, Akhtar Khalil.

References

1. Khalajzadeh H., Mansouri M., and Teshnehl M. Face recognition using convolutional neural network and simple logistic classifier. *Soft Computing in Industrial Applications* 2014, 223, 197–207. https://doi.org/10.1007/978-3-319-00930-8_18
2. Girshick, R. B. Fast R-CNN. *CoRR* 2015. Available: <http://arxiv.org/abs/1504.08083>.
3. Su W., and Wang Z. Widening residual skipped network for semantic segmentation. *IET Image Processing* 2017, 11, 880–887. <https://doi.org/10.1049/iet-ipr.2017.0070>
4. Ullrich, K., Meeds, E., and Welling, M. Soft weight-sharing for neural network compression. *Soft Computing in Industrial Applications* 2017. arXiv preprint arXiv:1702.04008, 2017.

5. Wu, J., Leng, C., Wang, Y., Hu, Q., and Cheng, J. Quantized convolutional neural networks for mobile devices. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2016, 4820–4828.
6. Szegedy, C., Liu, W., Jia Y., Sermanet, P., Reed, S., Anguelov, D., et al. Going deeper with convolutions. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2015, 223, 1–9.
7. He, K. Zhang, X. Ren, S., and Sun, J. Deep residual learning for image recognition. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2016, 770–778.
8. Han, S. Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *arXiv preprint arXiv:1510.00149* 2015.
9. Venkataramani, S., Ranjan, A., Roy, K., and Raghunathan, A. AXNN: energy-efficient neuromorphic systems using approximate computing. *in IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2014 2015, 32–58.
10. Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710* 2016.
11. Ding, X., Ding, G., Han, J., and Tang, S. Auto-balanced filter pruning for efficient convolutional neural networks. *in Thirty-Second AAAI Conference on Artificial Intelligence* 2018.
12. Marcel, S., and Rodriguez, Y. TorchVision the machine-vision package of torch. *in Proceedings of the 18th ACM international Conference on Multimedia* 2010, 1485–1488.
13. Han S., Pool J., Tran J., and Dally W. Learning both weights and connections for efficient neural network. *in Advances in Neural Information Processing Systems* 2015, 1135–1143.
14. Carreira-Perpinan, M. A., and Idelbayev, Y. “Learning-Compression” algorithms for neural net pruning. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2018, 8532–8541.
15. Yu, R., Li, A., Chen, C. F., Lai, J. H., Morariu, V. I., Han, X., et al. NISP: Pruning networks using neuron importance score propagation. *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2018, 9194–9203.
16. LeCun Y., Denker J. S., and Solla S. A. Optimal brain damage. *in Advances in Neural Information Processing Systems* 1990, 598–605.
17. Alvarez J. M., and Salzmann M. Compression-aware training of deep networks. *in Advances in Neural Information Processing Systems* 2017, 856–867.
18. Jaderberg, M., Vedaldi, A., and Zisserman, A. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866* 2014.
19. Garg, I., Panda, P., and Roy, K. A low effort approach to structured CNN design using PCA. *IEEE Access* 2019.
20. He, Y., Liu, P., Wang, Z., Hu, Z., and 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* 2019, 4340–4349.
21. Khan A., Sohail A., Zahoor U., and Qureshi A. S. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review* 2020, 1–62.
22. Cheng, Y., Wang, D., Zhou, P., and Zhang, T. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* 2017.
23. Wang, Z. Sparsert: Accelerating unstructured sparsity on GPUs for deep learning inference. *arXiv preprint arXiv:2008.11849* 2020.
24. Hassibi B., Stork D. G., and Wolff G. Optimal Brain Surgeon: Extensions and performance comparisons. *in Advances in Neural Information Processing Systems* 1994, 263–270.
25. Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440* 2016.
26. Luo, J. H., Wu, J., and Lin, W. Thinet: A filter level pruning method for deep neural network compression. *in Proceedings of the IEEE international Conference on Computer Vision* 2017, 5058–5066.
27. Zhuang Z., Tan M., Zhuang B., Liu J., Guo Y., Wu Q., et al. Discrimination-aware channel pruning for deep neural networks. *in Advances in Neural Information Processing Systems* 2018, 875–886.
28. Woo Y.; Kim D.; Jeong J.; Ko Y.-W.; Lee J.-G. Zero-Keep Filter Pruning for Energy/Power Efficient Deep Neural Networks. *Electronics* 2021, 10, 1238. <https://doi.org/10.3390/electronics10111238>
29. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., et al. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* 2017.

30. Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and 0.5mb model size. *arXiv preprint arXiv:1602.07360* 2016.
31. Zhang, X., Zou, J., Ming, X., He, K., and Sun, J. Efficient and accurate approximations of nonlinear convolutional networks. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2015, 1984–1992.
32. Denton E. L., Zaremba W., Bruna J., LeCun Y., and Fergus R. Exploiting linear structure within convolutional networks for efficient evaluation. in *Advances in Neural Information Processing Systems* 2014, 1269–1277.
33. Zhuang, B., Shen, C., Tan, M., Liu, L., and Reid, I. Structured binary neural networks for accurate image classification and semantic segmentation. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 2019, 413–422.
34. C. Louizos, M. Welling, and D. P. Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312* 2017.
35. J. Liu, J., Gong, M., and He, H. Nucleus neural network for super robust learning. *arXiv preprint arXiv:1904.04036* 2019.
36. Yang W., Jin L., Wang S., Cu Z., Chen X., and Chen L. Thinning of convolutional neural network with mixed pruning. *IET Image Processing* 2014, 13, 5:779–784.
37. Fletcher, P. T., Venkatasubramanian, S., and Joshi, S. Robust statistics on Riemannian manifolds via the geometric median. in *IEEE Conference on Computer Vision and Pattern Recognition* 2008, 1–8.
38. Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., et al. PyTorch: An imperative style, high-performance deep learning library. in *Advances in Neural Information Processing Systems* 2019, 8024–8035.