A Filter Pruning Method of CNN Models Based on Feature Maps Clustering

Zhihong Wu 1,‡, Fuxiang Li 1,*, Yuan Zhu 1, Ke Lu 1,‡, Mingzhi Wu 2 and Changze Zhang 1

1 School of Automotive Studies, Tongji University, Shanghai 201804, China; zhihong.wu@tongji.edu.cn (Z.W.); yuan.zhu@tongji.edu.cn (Y.Z.); luke@tongji.edu.cn (K.L.); 2110214@tongji.edu.cn (C.Z.)
2 Nanchang Automotive Institute of Intelligence & New Energy, Tongji University, Nanchang 330052, China; wumingzhi@naiine.com
* Correspondence: 1710564@tongji.edu.cn

Abstract: The convolutional neural network (CNN) has been widely used in the field of self-driving cars. To satisfy the increasing demand, the deeper and wider neural network has become a general trend. However, this leads to the main problem that the deep neural network is computationally expensive and consumes a considerable amount of memory. To compress and accelerate the deep neural network, this paper proposes a filter pruning method based on feature maps clustering. The basic idea is that by clustering, one can know how many features the input images have and how many filters are enough to extract all features. This paper chooses Retinanet and WIDER FACE datasets to experiment with the proposed method. Experiments demonstrate that the hierarchical clustering algorithm is an effective method for filtering pruning, and the silhouette coefficient method can be used to determine the number of pruned filters. This work evaluates the performance change by increasing the pruning ratio. The main results are as follows: Firstly, it is effective to select pruned filters based on feature maps clustering, and its precision is higher than that of a random selection of pruned filters. Secondly, the silhouette coefficient method is a feasible method for finding the best clustering number. Thirdly, the detection speed of the pruned model improves greatly. Lastly, the method we propose can be used not only for Retinanet, but also for other CNN models. Its effect will be verified in future work.

Keywords: CNN; filter pruning; self-driving car

1. Introduction

In recent years, convolutional neural networks (CNNs) have been widely used in the field of self-driving car computer vision including obstacle detection, classification, segmentation, object tracking [1], and semantics. To achieve a better performance, the design of deeper and more complex CNN models has become a general trend.

While the deep neural network has enhanced learning ability and increased parameter quantities, it also has certain drawbacks. Storage space persistence: the model needs to train on a high-performance GPU server. Hardware cost: the neural network needs to load parameters in memory and complete the construction of the calculation graph when training evaluation and predicting. Calculation cost: convolutional neural networks need to perform convolution operations when processing data. Since parameters are usually stored in 32-bit floating-point data types in the model, multiplication operations take up a lot of computing resources when it comes to floating-point data. The training and use of the model are very time-consuming.

The simplification of the neural network structure mainly includes tensor factorization [2], sparse connection [3], quantization [4], channel pruning [5], and so on. The tensor factorization method can decompose the convolutional layer into several effective layers. However, this method does not reduce the number of channels. Moreover, the decomposition process introduces additional computational overhead. The sparse connection
method compresses the model by invalidating the connections between neurons or channels. Although this method can theoretically achieve a large speedup for the operation of the model, the sparse convolution layer has high requirements for hardware. Quantization is a relatively effective method that is widely used, but when dealing with deep CN networks such as GoogleNet, the accuracy is significantly reduced [6]. Compared with the previous methods, channel pruning directly reduces the number of convolutional kernels and feature maps, thus making the network layer numberless and the structure simpler. This method has an obvious acceleration effect on both CPU and GPU because it does not require additional computation or special hardware requirements. The idea of channel pruning is simple but deleting the channels in the previous layer directly changes the input of the next layer. In recent years, a training-based channel pruning project [7] has been proposed that focuses on applying sparse constraints to weights during the training process so that hyperparameters can be adaptively determined. However, each training process consumes a lot of time and resources. Moreover, this method is rarely used in CNN networks based on ImageNet datasets and experimental results are rarely mentioned.

There is some novel research about distributed computing that can be used to speed up CNN computing, and some related work can be found in [8,9]. This work is about using the compression of convolutional neural networks to accelerate computing. Many researchers have conducted relevant research in this area including weight pruning, quantization, and some other methods. However, the weight pruning method will introduce the operation of the spares matrix, and for the quantization method, the solution of the Hessian matrix is also a great difficulty. This paper chooses the filter pruning method to compress the neural network to accelerate the deep neural network. In this paper, a neural network compression method was proposed based on feature maps clustering. The basic idea is that by clustering, it gets to know how many features the input images have and how many filters are enough to extract all features. The main contributions are as follows: According to the corresponding relationship between feature maps and filters, the filters are clustered by feature maps clustering. K-Means and Hierarchical Clustering algorithms are used to cluster feature maps and a comparison between them is made. In the K-Means algorithm, the elbow and silhouette coefficient methods are used to determine the best clustering number. The changes in network performance before and after pruning are tested by experiments. The core of this thesis is the compression of convolutional neural networks. The many difficulties of this method include: (1) the definition of filters’ importance, (2) determining the number of pruned filters, and (3) explaining the effectiveness of the proposed network.

2. Related Work

In recent years, the deep neural network has received extensive attention and research. The powerful modeling ability of deep neural networks comes from their more complex structure and larger weight parameter amount, which makes it difficult for network models to run on mobile devices or embedded platforms with limited computing and storage resources [9]. Therefore, these network models need to be pruned or compressed to make them easier to be transplanted to the mobile terminal. This part will introduce four kinds of important algorithms in the field of network compression and acceleration and analyze their advantages and disadvantages.

LeCun et al. proposed an OBD (Optimal Brain Damage) method [10] for weight pruning; this method has some effect, but it cannot be applied to the general case because the Hessian matrix is not always a diagonal matrix, and the computational cost is relatively high. To solve the problem of the non-diagonal Hessian matrix, Hassib et al. proposed an OBS (Optimal Brain Surgeon) [11] method to find the optimal solution of the Hessian matrix by the Lagrange equation. Although this method has taken into account the general situation, it still has a high computation cost because of the requirement to find the inverse of the Hessian matrix. In a word, OBD and OBS methods use second-order Taylor expansion to select weights that need to be deleted and improve the generalization performance of the
model by pruning and retaining. However, these two methods inevitably require Hessian matrix operation, which significantly increases the memory and computation cost of the hardware device used for network fine-tuning.

The essence of network quantization is to compress the neural network by reducing the number of bits needed for weight storage. In the deterministic quantization method, the quantized value and the actual value have a one-to-one correspondence. Rounding should be the easiest way to quantify actual values. Courbariaux et al. [12] proposed deterministic quantification using the rounding function. However, during backpropagation, the error cannot be calculated by the same function because the gradient value is zero almost everywhere. So, a heuristic algorithm is needed to estimate the gradient of neurons. Rastegari et al. [13] improved the backward function. For the rounding function, Polino et al. [14] proposed a more general form. Gong et al. [15] first considered using vector quantization to quantize and compress neural networks. In a random rounding function, there is a one-to-one mapping between actual and quantized values. Generally speaking, taking a rounding function is a simple way to convert actual values into quantized values. However, the performance of the network may be significantly degraded when the parameters are quantized. Singular Value Decomposition (SVD) [16] is a popular low-rank matrix decomposition method. After the matrix is decomposed, the relatively small singular values are discarded and then the network parameters are fine-tuned. However, when the weight matrix in the network is high-dimensional, the SVD method cannot be effectively decomposed. At this time, it is necessary to use the Tucker decomposition method [17]. The low-rank matrix decomposition method can simplify the date scale and remove the noise points, which can effectively compress and accelerate the network model. However, it also has the disadvantages that data conversion is difficult to understand and the adaptability is poor.

Filter pruning refers to the direct deletion of unimportant filters in the neural network and the feature maps associated with it [18]. Compared with the weight pruning method of the neural network, filter pruning is a structured pruning method, which does not introduce additional sparse operations, so there is no need to use sparse libraries or any specific hardware devices. The number of pruned filters will directly affect the number of convolutional operations, and the large reduction in the number of convolutional operations will greatly increase the operational efficiency of the neural network. Although filter pruning has many advantages, it is rarely used in practical projects. One reason is that it is difficult to determine the filters that need to be pruned. The other is how to achieve a balance between the improvement of speed and a decrease in precision in the process of filter pruning. Hu et al. [19] proposed method beads on the percentage of zero activations to find the unimportant filters. Subsequently, experiments have proved that this method of evaluation and deletion of the filters in all layers at the same time will greatly reduce the accuracy of the neural network and it is difficult to restore its performance through retraining. Therefore, the filter needs to be pruned layer by layer. Li et al. [11] proposed a method for filter pruning based on the L1 norm of the filter matrix. Because L1 norms of filters from different convolutional layers are in different orders of magnitude, they cannot be compared together. Therefore, it is necessary to fine-tune the whole network after one layer is pruned by this method. One obvious disadvantage of this method is that it is difficult to achieve the best trade-off between pruning efficiency and network performance. Hu et al. [20] proposed a squeeze-and-extraction block at the 2017 ImageNet Image classification challenge competition referred to as SEBlock. SEBlock was originally used to improve the accuracy of image classification; its scaling factor reflects the network’s choice of feature channels. Wang et al. [21] proposed that the importance of the channel can reflect the importance of the filter because the channel and the filter are one-to-one correspondence. Hence, a soft attention mechanism can be added to the neural network to determine the importance of the filters through its scaling factors.
2.1. Weight Pruning

Weight pruning is the earliest method of network pruning. Its basic idea is to consider weights below a certain threshold in the network as unimportant weights and remove them from the network. Then, retrain the network to get the final weight based on the reserved sparse connection. The pruning process is shown in Figure 1.

\[
\begin{array}{cccc}
2.13 & 0.43 & 1.58 & 2.98 \\
3.92 & -1.35 & 0.96 & 1.42 \\
1.73 & 0.26 & 2.74 & -0.47 \\
0.56 & 2.09 & -3.48 & 1.39 \\
\end{array}
\]

\[
\begin{array}{cccc}
2.13 & 0.00 & 1.58 & 2.98 \\
3.92 & -1.35 & 0.00 & 1.42 \\
1.73 & 0.00 & 2.74 & 0.00 \\
0.00 & 2.09 & -3.48 & 1.39 \\
\end{array}
\]

**Figure 1.** Weight pruning based on the threshold.

2.2. Quantization Methods

The essence of network quantization is to compress the neural network by reducing the number of bits needed for weight storage. The process is shown in Figure 2.

**Figure 2.** Weight Quantization and Centroids Fine-tuning.

Figure 2 shows the process of weights and gradient quantization. The upper left corner is a \( 4 \times 4 \) weight matrix and the lower-left corner is a \( 4 \times 4 \) gradient matrix. The weights are all stored in a 32-bit float mode. To quantify the weight to 4-bits, it is necessary to divide the weights into four categories, which are represented by four different colors in Figure 2, and calculate the average value of each category. Then, it only needs to store an index value for every category in the value of the shared weight. In the process of weights updating, all gradients are grouped into the same category as the weights. The sum of each category is the gradients of the whole category. Finally, the shared weight subtracting the gradient is the final weight value. In general, quantization methods can be divided into two types: deterministic quantization and stochastic quantization.

2.3. Low-Rank Matrix Decomposition

The main idea of low-rank matrix decomposition is to compress and accelerate the neural network by decomposing the parameter matrix in the neural network into a low-
rank matrix product and eliminating redundancy in the convolution filter. There is the SVD method and tucker decomposition.

2.4. Filter Pruning

Filter pruning refers to the direct deletion of unimportant filters in the neural network and the feature maps associated with it [11]. The process is shown in Figure 3.

$c_i$ represents the number of input channels in the $i$th layer. $h_i$ and $w_i$ represent the height and weight of the input feature map, respectively. $c_i$ filters $F_{ij} \in R^{n_i \times h_i \times k \times k}$ can convert the feature maps $X_i \in R^{n_i \times h_i \times w_i}$ into $X_{i+1} \in R^{n_{i+1} \times h_{i+1} \times w_{i+1}}$ with convolution operations. In this process, the number of convolution operations is $n_{i+1}h_{i+1}w_{i+1}$. If one of the filters in $X_i$ is pruned, $n_i k^2 h_{i+1} w_{i+1}$ will be directly reduced, at the same time the feature maps of the filter output are deleted. $n_i$ refers to the filter size. In other words, the feature maps of the next convolutional layer input are also reduced. Then, $n_{i+2} k^2 h_{i+2} w_{i+2}$ convolution operations will also be reduced in the next layer. According to this rule, the subsequent convolutional layers will reduce at a large number of operations.

Compared with the weight pruning method of the neural network, filter pruning is a structured pruning method, which does not introduce additional sparse operations, so there is no need to use sparse libraries or any specific hardware devices. The number of pruned filters will directly affect the number of convolution operations and the large reduction in the number of convolution operations will greatly increase the operational efficiency of the neural network. Most of the existing filter pruning methods are based on the importance of filters. The importance of filter is based on the L1 norm. The greater the L1 norm value of the filter, the higher the importance of the filter is.

3. Framework of Retinanet

The authors of [22] proposed a one-stage detection model Retinanet. This model uses the Focal Loss function to solve the problem of extreme foreground-background class imbalance so that it can match the speed of previous one-stage detectors while surpassing the accuracy of all existing state-of-the-art two-stage detectors. The work will try to prune filters using Retinanet as an example and test it on the wider face dataset with the network before and after filter pruning to maintain a good balance between the accuracy and speed of the evaluation. The framework of Retinanet is shown in Figure 4.
The Retinanet network consists of three parts: Resnet [21], a network for feature extraction; FPN (feature pyramid network) [23], a feature pyramid network for fine processing of extracted image features; and subnets [24], used for classification and location.

3.1. Feature Extraction Network, Resnet 18

The feature map extraction process of Retinanet is divided into two parts. The first part is a bottom-up path, and the second part is a top-down path. In the field of deep learning, a linear convolutional neural network is usually used to extract image features such as AlexNet [25], LeNet [26], and VGG-Net [27]. The ability of this type of neural network to extract picture features tends to improve as the number of layers increases. However, the increase in network depth does not always improve the accuracy of the model and sometimes it may cause higher evaluation and training errors resulting in a rapid deterioration of accuracy after saturation. This phenomenon is called degradation in the field of deep learning. Resnet can avoid the shortcomings of linear convolution neural networks.

3.2. FPN

Before the appearance of FPN (feature pyramid network), most target detection algorithms only used top-level features to predict. However, this kind of method has its drawbacks because although the semantic information of image samples acquired at the high lever is richer than that at the low level, the target location is rougher than that at the low level. In addition, although an algorithm using a multi-scale fusion method has been proposed, it usually uses the fused features to predict. FPN fuses feature maps from different layers through top-down connections, bottom-up connections, and horizontal connections so that it performs better in small target detection.

3.3. Subnets

Faster R-CNN adopts the idea of a regional proposal network (RPN) [28]. The mapping point of the sliding window center in the original image is called the anchor. With the anchor as the center, proposed regions can be generated in different network layers of FPN. The Retinanet model follows this idea. When generating anchor boxes, nine different anchor boxes can be generated by using three scales and three aspect ratios.

4. Filter Pruning Based on Feature Maps Clustering

The filter is essentially a matrix of size $M \times N$. It is used to detect specific features in images. Different filters have different parameters. When the image is processed in the computer, it is in the form of $M \times N \times 3$ size. Suppose we only consider the grayscale of the image regardless of RGB, then the size of the image is $M \times N$. The filter processes the image by sliding through all the regions of the image from left to right, from top to bottom, and then dot-multiplying with all the same size areas in the image. After the sum of the products, a new filtered image, i.e., the feature map, is obtained.

When the filter processes the image, it dot-multiplies with each equal area in the image. If a certain area of the image is similar to the feature detected by the filter, the filter will be activated and get a high dot multiplication result when passing through the area. Conversely, if an area of the image is very dissimilar to the features detected by the filter, the filter will not be activated or the value obtained by dot multiplication will be very small.

It can be concluded that when the filter slides through the whole image, the higher the value obtained from an area of the image, the higher the correlation between the area and the undetermined features detected by the filter. The neural network can obtain sufficient features of the image through a large number of filters so that it can detect and locate the image.

The feature map is extracted by the filter and passed as input to the next convolutional layer. Zeiler et al. [5] used a de-convolution method to project the activated feature matrix back into the input pixel space and found that filters in different layers can extract features
of different levels. Chen et al. [29] demonstrated that different filters in the same layer can also extract features from different aspects. Therefore, the feature map can represent filters in some respects [30]. Inspired by these, the work hypothesizes that the redundant filters in convolutional neural networks may extract similar feature maps to that from other filters in the same layer when processing the same input.

Based on this assumption, we can determine how many similar feature maps are extracted by each convolutional layer using the clustering method, and then we can determine how many necessary features of the input image need to be extracted. Finally, we can select the filters to be pruned. The image clustering method used in this paper will be introduced below.

4.1. K-Means Method

The K-Means [31] algorithm is a common clustering method. Its basic idea is to select K points in the space as the center and classify the objects closest to them into one class. Through iteration, the points in each cluster center are updated one by one until the best clustering result is obtained. Since the matrix representing the feature map is 3D, it is difficult to represent it with a single point. It is necessary to reduce the dimension of the feature map and map it to a vector containing important components.

4.1.1. Feature Map Dimension Reduction

The method for a feature map’s dimension reduction mainly includes SVD (singular value decomposition) and PCA (principal component analysis) [13]. Similar to the SVD method, the basic idea of the PCA method is to project the original data onto a new coordinate system. If the variance of all data on a coordinate axis is the largest, this axis is recorded as A1. That is, the projection of data is most scattered in the direction of the A1 axis, which means that more information is retained. Then, A1 is the first principal component. Then, look at A2. If the covariance between A2 and A1 is 0, that is, the information of A2 and A1 does not overlap and the variance of data in this direction is as large as possible, then A2 is the second principal component. By analogy, it can find the third, fourth, and Nth principal components.

Usually, only the first two principal components of the data need to be preserved so that the high-dimensional image can be transformed into a two-dimensional vector.

4.1.2. Clustering

After the dimensional reduction of the feature maps, the K-Means algorithm can be used to cluster the feature maps. The specific algorithm steps are as follows:

1. Choose the initial centers of K classes appropriately. Generally, the class centers are initialized by guessing or in a random way.
2. For any sample, calculate its distance to K centers and classify the sample into the class, to whose center the distance is the shortest.
3. Calculate the average of all data points belonging to the same class, that is calculate the average of each dimension of the vector and use the average as the new class center.
4. Repeat steps 2 and 3 until convergence occurs, i.e., the centers of all classes do not change anymore.

Figure 5 is an example of the K-Means algorithm. Where A, B, C, D, and E represent different samples, two orange circles represent different centers. According to the distances between the samples and the centers, A, B, and C are clustered in one group, and D and E are clustered in the other group eventually.

The advantages of this algorithm are simple steps and fast operations. Most importantly, it is easy to be implemented and can be operated in parallel. However, this algorithm also has some drawbacks because the number of classes K should be set in advance and improper selection will lead to poor results. The key to the algorithm lies in the selection of initial centers and distance formula. Moreover, how to determine the clustering number K is also an important problem to be solved.
4.1.3. Determination of the Number of Classes

At present, there are many methods to determine the clustering number, two of which are selected in this thesis.

The first method is the Elbow method. This rule examines the relationship between the number of classes K and the cost function J. Cost function J is the sum of the distances from all data in classes to the center points. It can be expressed as:

\[ J = \sum_{i=1}^{k} \sum_{j}^{m} \text{dist}(x_{ij}, c_i)^2 \]  

(1)

where \( m \) represents the number of data points in the \( i_{th} \) class \( x_{ij} \) represents the \( j_{th} \) data point in the \( i_{th} \) class. For a class, the smaller the cost function value is, the closer the members in this class are. Conversely, the larger the function value is, the looser the structure within the class is. The cost function decreases with the increase in the number of classes.

Take Figure 6 as an example. When the number of clusters equals 3, the cost function is greatly improved, so 3 can be considered the best number of clusters. However, it is also possible that the cost function curve does not have a significant curvature change input. At this time, the second method is required to determine the optimal number of clusters.

The second method is the silhouette coefficient method. For a clustering task, the best clustering should be that the data within the class is as compact as possible, and the data
between classes is as far away as possible. The silhouette coefficient is an index to measure the degree of dispersion and intensity of a class. The formula is expressed as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$  \hspace{1cm} (2)

where \( i \) represents the \( i_{th} \) sample, \( b(i) \) denotes the mean distance from the \( i_{th} \) sample to the samples in the nearest class, \( a(i) \) represents the mean of the distance between samples in the same class, \( s(i) \) represents the silhouette coefficient of the sample. The value of \( s(i) \) is in the range of \((-1, 1)\). If it is close to 1, the classification of this sample is reasonable; if it is close to \(-1\), it means that the sample should be classified into other classes.

The average value of all samples’ \( s(i) \) is called the silhouette coefficient of the whole clustering results. The formula is as follows:

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i)$$  \hspace{1cm} (3)

In the above formula, \( N \) represents the total number of samples, and \( S \) represents the final silhouette coefficient. The silhouette coefficient is the reasonable and effective measure of the clustering result. In general, the larger the silhouette coefficient is, the better the clustering effect is.

Take Figure 7 as an example, the relationship between cost function and cluster number is shown in Figure 7a; there is no obvious elbow point in this curve. However, we can use the silhouette coefficient method as is shown in Figure 7b, when the clustering number is 3, the silhouette coefficient is the biggest, meaning that the classification of all samples is the most reasonable. Then, we can determine that the optimal clustering number is 3.

![Figure 7](image-url) An example for use of SC Method: (a) the relationship between cost function and cluster number; (b) the relationship between silhouette coefficient and cluster number.

### 4.2. HCA Method

The HCA (hierarchical clustering) method can divide the dataset into classes hierarchically, and the class generated later is based on the result of the former layer. Hierarchical clustering algorithms are generally divided into two types. Bottom-up Hierarchical Clustering [32]: each sample is considered as a class at the beginning, and each time, the two closest classes are merged into a new class according to certain criteria. According to this rule, until the end, all samples belong to one class. Top-down Hierarchical Clustering [32]: at first, all samples belong to one class and each class is divided into several classes according to certain criteria. According to this rule, until the end, each sample is a class.

In this thesis, a bottom-up clustering method is adopted. The process is shown in Figure 8. Suppose there are \( N \) samples to be clustered, the specific steps are as follows:

1. Classify each sample into a class and calculate the distance between every two classes, in other words, the similarity between different samples. In this thesis, the class-average method is used to measure the similarity between two classes, which is the average of the distance between two points in the two classes. For any two classes
$C_x, C_y$, their similarity is recorded as $L(C_x, C_y)$, and it can be calculated with the following formula:

$$L(C_x, C_y) = \frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \text{dist}(d_{xi}, d_{yj})$$  

(4)

where $n_x, n_y$ represent the number of samples in the two classes, respectively, $d_{xi}, d_{yj}$ represent the samples in two classes. $\text{dist}(d_{xi}, d_{yj})$ represents the Euclidean distance between two samples:

$$\text{dist}(d_{xi}, d_{yj}) = \| d_{xi} - d_{yj} \|^2$$  

(5)

2. Set a threshold to find two classes with closest distance among all the classes and the distance must be smaller than the threshold. If they exist, classify them into one class and reduce the total number of classes by 1. Otherwise, the classification process stops.

3. Recalculate the similarity between the generated new classes and the previous old classes.

4. Repeat steps 2 and 3 until all samples fall into one class, or when the distance between the two closest classes is greater than the specific threshold and the classification stops.

Figure 8. The hierarchical clustering algorithm flowchart.

4.3. Filter Pruning

This section will prune the convolutional layer with redundant filters. The idea is to cluster the output feature maps from each filter in the convolutional layer first, and then randomly select and save one sample in each class. Save the index of this sample and make its corresponding filter. The remaining filters in this layer will be pruned. The process is represented in Figure 9.

Figure 9. Filter Pruning Based on Feature Maps clustering.

In Figure 9, $M_i \in \mathbb{R}^{h_i \times w_i \times n_i}$ represents input feature maps, $h_i$ and $w_i$ represent the height and width of the feature maps in the $i_{th}$ layer, the original convolutional layer is recorded as $F_i \in \mathbb{R}^{k \times 1 \times n_i \times n_{i+1}}$. The output feature map is recorded as $M_{i+1} \in \mathbb{R}^{h_{i+1} \times w_{i+1} \times n_{i+1}}$, the number of convolutional operations for this convolution layer is $n_i n_{i+1} k^2 h_i w_i$. Let $n_f$ represent the cluster number of filters. Then, after cropping, $n_f$
filters will be preserved and the number of convolution operations in this layer will be reduced to \( n_i k^2 h_i w_i \).

5. Experiments and Results

5.1. Preparation in Advance

### 5.1.1. Dataset Preparation

The WIDER FACE dataset [33] is a face detection benchmark dataset, of which images are selected from the publicly available WIDER dataset. There are 32,203 images and 393,703 faces with a high degree of variability in scale, pose, and occlusion labeled as depicted in the sample images in Figure 10.

![Wider face Dataset Samples.](image)

In this paper, only the training set and the validation are used to verify the effectiveness of the algorithm by comparing the changes of the evaluation accuracy before and after pruning. Before using this dataset, the dataset must firstly be processed so that it can be correctly read into the Retinanet model. Considering the training speed and evaluation accuracy, this paper selected a part of the dataset for training and evaluation. In the training set, we removed the boxes with height and width less than 50, and in the validation set, we removed the boxes with height and width less than 100. After screening, the training set and validation set eventually included 9152 and 1282 images, respectively.

5.1.2. Training and Evaluation before Pruning

1. **Metrics:** In our experiments, since the face position is critical for future applications, for instance, detecting the face, IoU (intersection over union) for a bounding box that indicates the accuracy of the face position is adopted. The IoU can be calculated as

\[
\text{IoU} = \frac{A_o}{A_u} \tag{6}
\]

where \( A_o \) represents the area of the overlap for the ground truth bounding box and detected bounding box. \( A_u \) represents the area of the union for the ground truth bounding box and detected bounding box. To evaluate the performance of face detection, we use the mean average precision (MAP) to calculate the mean value of average precision (AP) for each query \( Q_i \). MAP is calculated as

\[
\text{MAP} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \text{AP}(Q_i) \tag{7}
\]

2. **Results:**

The training dataset is trained with the RetinaNet network for 80 epochs from scratch in Nvidia Titan-X GPU with i7-7700K CPU and 16 GB RAM. For tuning the parameters, the IoU threshold is set to 0.5, which is considered a good location predictor for iterating out the inaccurate predicted face position. When the training process finishes, the classification loss and the regression loss curves both converge as shown in Figure 11.
works, this paper prunes all filters, namely reduced related layer, in three convolutional layers one by one and evaluates them on the validation set. Then, we can measure the importance of each convolutional layer in FPN by comparing the decline of the feature maps in FPN layers are shown in Figure 13.

Figure 13. Examples of the feature maps in FPN layers.

The statistics for the Initial Network.

<table>
<thead>
<tr>
<th>Network</th>
<th>Parameters</th>
<th>GPU Time(s)</th>
<th>CPU Time(s)</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resnet18 + FPN</td>
<td>19195117</td>
<td>0.143</td>
<td>0.388</td>
<td>71.75</td>
</tr>
</tbody>
</table>

Table 1. The statistics for the Initial Network.

5.1.3. The Selection of Convolutional Layer

To verify the performance of the proposed filter pruning method, this paper takes the convolutional layers in the feature pyramid network as an example, because they are directly related to the output results. Therefore, this work focuses on the convolutional layers P5, P4, and P3 in FPN. To measure their effects on the performance of neural networks, this paper prunes all filters, namely reduced related layer, in three convolutional layers one by one and evaluates them on the validation set. Then, we can measure the importance of each convolutional layer in FPN by comparing the decline in the evaluation precision of the different pruned networks. The evaluation results are shown in Figure 12.

Figure 12. Precision Comparison of Pruning P3, P4, P5 layers.

When layer P5 is pruned, the precision declines from 71.75% to 38.9%, which is almost half of the original precision. In contrast, when layer P4 or P3 is pruned, the precision

![Loss curves in Training Process. (a) Classification Loss Curve. (b) Regression Loss Curve.](image-url)
has almost no drop. In other words, the removal of P5 leads to the biggest decline in precision. Hence, P5 is considered the most important layer and is selected to be pruned by our method.

5.2. Feature Maps Extraction

In general, the deep learning model is a “black box”, that is, the model-derived features are difficult to extract and present in a way that humans can understand. However, the features learned by convolutional neural networks are very suitable for visualization, largely because they are features of visual concepts. This paper chose to visualize the intermediate output of convolutional neural networks. In other words, for a given input, we present the output feature maps of each convolution layer and pooling layer in the network. Each channel corresponds to a relatively independent feature, so the correct way to visualize these feature maps of each convolution layer is to set a convolutional layer as an output layer and then plot the contents of each channel in this layer into a two-dimensional image. By this method, the examples of the feature maps in FPN layers are shown in Figure 13.

![Figure 13. Examples of the feature maps in FPN layers.](image)

5.3. Clustering Feature Maps of P5

5.3.1. The Output Feature Maps of P5

Since layer P5 has 256 filters for every input map, P5 will output 256 feature maps, as shown in Figure 14. These feature maps are all $25 \times 13 \times 256$ in size. Each of them is one feature of the input map. We can see that some of them are very similar. After clustering, the similar feature maps will be clear:

![Figure 14. The output feature maps of P5.](image)
5.3.2. The Clustering Number

At first, we cluster the feature maps using the K-means algorithm. In K-means, two components of features are selected by the SVD or PCA method. The clustering number \( k \) varies from 2 to 256. To find the best clustering number, we tried the silhouette coefficient method, and the result is shown in Figure 15. In the silhouette coefficient method, normally the greater the silhouette coefficient is, the better the clustering number for the K-Means algorithm is. As is shown in Figure 15, the optimal number of clustering is about 100.

![Figure 15](image_url)

Figure 15. The relationship between silhouette coefficient-clustering and number.

The HCA method is based on Euclidean distance, that is, the feature maps are clustered with their pixel matrices. This method is very similar to the magnitude-based pruning methods. Therefore, the number of clusters will vary with the threshold size. The larger the threshold is, the smaller the number of clusters is. This work sets the threshold as \( s \times (\text{max} - \text{min}) / \text{min} \), max and min are the maximal and minimal values of distance between feature maps. Because the pixel matrices of the feature maps are not particularly dispersed, the number of clusters is discrete in clustering. In our experiment, the change of the clustering number with the threshold and the change of the model precision with the clustering number is shown in Figure 16.

![Figure 16](image_url)

Figure 16. The change of clustering number with a threshold (a) and the change of model precision with clustering number (b).

As is shown in Figure 16, when the number of clusters is greater than 112, the prediction precision of the model decreases rapidly, so 112 is like the elbow joint in the elbow rule. According to the results introduced above, we choose 112 clusters for the comparative experiments. Because when we prune the filters, the first thing to be satisfied with is that the precision cannot be reduced too much. Therefore, the larger the clustering number is more appropriate. This work sets the number of initial centers in the K-Means method to 112 and the threshold in the HCA method to 0.049.
5.4. Pruning Network

5.4.1. Pruning P5

Through the K-Means and HCA methods, we can cluster the feature maps into different classes. From each cluster, we randomly sample a feature map and save the index of its corresponding filter. In order to compare the performance of the two methods, we prune the network with the same clustering number and compare the precision.

As is shown in Figure 17a, c, the precision of the network pruned by the HCA method is always higher than that of the network pruned randomly. In addition, the predicting precision of the HCA method has a smaller variance. The comparison between HCA and K-Means methods is shown in Figure 17b, d. Excluding one of the experiments, the precision of the network pruned by the HCA method is higher than that of the K-Means method. Moreover, the average value and variance of the former method are also better than the latter. As for the K-means method and the randomly pruning method, K-means also has a higher average detection precision.

![Figure 17](image_url)

**Figure 17.** The results of different methods of pruning the network. (a) The precision of the HCA pruned network compared with the result of the randomly pruned network. (b) The precision of the HCA pruned network compared with the result of the K-Means pruned network. (c) The average and variance of precision in HCA compared with the result of randomly pruned network. (d) The average and variance of precision in comparison with the result of the K-Means pruned network.

5.4.2. Pruning the Whole Network

By comparison, the HCA method is better. Therefore, we apply this method to the whole network, including Resnet18 and FPN. After pruning, we retrain the network with the corresponding cluster number. The performance of pruned models is reported in Table 2. The pruning ratio is calculated with pruned filters divided by all the filters in the network.

From Table 2 we can see that as the number of network pruning layers increases, the detection precision of the network also decreases. On the contrary, the computing speed of the model in the GPU and CPU increases constantly. In addition, it is also indicated that FPN has more impact on the CPU computation cost. In contrast, pruning filers in Resnet18 can save more GPU time.
Table 2. The statistics for the pruned network.

<table>
<thead>
<tr>
<th>Pruned Network</th>
<th>Pruning Ratio (%)</th>
<th>GPU Speed Up (%)</th>
<th>CPU Speed Up (%)</th>
<th>Precision Drop (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPN</td>
<td>29.3</td>
<td>1.1</td>
<td>43.5</td>
<td>0.97</td>
</tr>
<tr>
<td>FPN</td>
<td>37.6</td>
<td>6.8</td>
<td>54.2</td>
<td>3.23</td>
</tr>
<tr>
<td>FPN</td>
<td>40.2</td>
<td>9.2</td>
<td>56.1</td>
<td>6.16</td>
</tr>
<tr>
<td>FPN</td>
<td>41.1</td>
<td>10.8</td>
<td>56.9</td>
<td>9.02</td>
</tr>
<tr>
<td>FPN</td>
<td>41.4</td>
<td>12.8</td>
<td>57.2</td>
<td>11.21</td>
</tr>
<tr>
<td>Resnet18</td>
<td>59.9</td>
<td>15.8</td>
<td>10.9</td>
<td>10.04</td>
</tr>
<tr>
<td>Resnet18</td>
<td>62.5</td>
<td>20.4</td>
<td>13.7</td>
<td>12.02</td>
</tr>
<tr>
<td>Resnet18</td>
<td>63.7</td>
<td>21.3</td>
<td>18.7</td>
<td>17.42</td>
</tr>
<tr>
<td>Both</td>
<td>90.7</td>
<td>17.9</td>
<td>79.3</td>
<td>8.93</td>
</tr>
<tr>
<td>Both</td>
<td>95.7</td>
<td>30.6</td>
<td>78.7</td>
<td>12.93</td>
</tr>
<tr>
<td>Both</td>
<td>96.3</td>
<td>35.6</td>
<td>82.6</td>
<td>25.42</td>
</tr>
</tbody>
</table>

5.5. Compared with SSD Network

Finally, this work compares the pruned network to the existing lightweight network. SSD (Single Shot MultiBox Detector) [26] network is a typical lightweight network.

SSD has two structures, SSD300 and SSD512, which represent the size of the input picture. The size of the input image in SSD300 is 300 × 300. The size of the input image in SSD512 is 512 × 512. This work uses the same datasets to train and evaluate SSD300. The result is compared with filters in FPN pruned networks. The comparison is shown in Table 3.

Table 3. Comparison between SSD and Pruned Network.

<table>
<thead>
<tr>
<th>FPN Layers</th>
<th>Precision (%)</th>
<th>Speed(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>71.75</td>
<td>0.388</td>
</tr>
<tr>
<td>128</td>
<td>70.78</td>
<td>0.219</td>
</tr>
<tr>
<td>64</td>
<td>68.52</td>
<td>0.178</td>
</tr>
<tr>
<td>32</td>
<td>65.59</td>
<td>0.170</td>
</tr>
<tr>
<td>SSD</td>
<td>60.73</td>
<td>0.195</td>
</tr>
</tbody>
</table>

All of these experiments are performed on the CPU. Additionally, Speed means the time of model detection for one picture. It can be seen that the SSD model has a relatively high speed. However, when we prune FPN filters from 256 to 64, the Retinanet is better than SSD in terms of speed and precision. By pruning, we obtain a better lightweight network.

6. Conclusions

This paper studies the filter pruning method in a convolutional neural network and proposes a filter selection method based on feature maps clustering. Through experiment, we verify the feasibility of this method. It is effective to select pruned filters based on feature maps clustering, and its precision is higher than that of a random selection of pruned filters. Among the two methods adopted in this thesis, the HCA method is superior to the K-Means method.

The silhouette coefficient method is a feasible method to find the best clustering number. In general, when the number of layers is pruned to the same number as that of clusters, the detection precision of the model will not decrease much. The precision in the HCA method is higher and more robust. This work sets the number of initial centers in the K-Means method to 112 and the threshold in the HCA method to 0.049.

The detection speed of the pruned model improves greatly. The improvement of computing speed in CPU and GPU is mainly dependent on the structure of the model. Models that require more parallel operations can save more computational time in GPU after pruning. In contrast, models that require more serial operations can save more time on the CPU.
This paper also compared the pruned network with the existing lightweight network, SSD. By pruning, Retinanet exceeds the SSD in the precision and speed of the wide face dataset detection as shown in Table 3.

In addition to the above research results, there are also many areas that need to be improved. There are a lot of map clustering methods, and it is better if the method is more robust and easy to determine the best clustering number. It also needs a more complete theory to clarify the relationship among feature maps.

Author Contributions: Conceptualization, Z.W. and F.L.; methodology, F.L. and Y.Z.; software, C.Z.; validation, K.L., M.W. and C.Z.; formal analysis, K.L.; investigation, M.W.; resources, Y.Z.; data curation, K.L.; writing—original draft preparation, F.L.; writing—review and editing, F.L.; visualization, F.L.; supervision, Z.W.; project administration, Y.Z.; funding acquisition, M.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Nanchang Automotive Institute of Intelligence & New Energy. Tongji University (NAIT) under the project the Perspective Study Funding of Nanchang Automotive Institute of Intelligence and New Energy. Tongji University (TPD-TC202110-13).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References


