Received September 26, 2019, accepted November 22, 2019, date of publication December 2, 2019,
date of current version December 13, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2956976

A Channel Pruning Algorithm Based on Depth-Wise Separable Convolution Unit

KE ZHANG[1], KEN CHENG[2], JINGJING LI[1], AND YUANYUAN PENG[1]
[1]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China

Corresponding author: Jingjing Li (lijin117@yeah.net)

This work was supported in part by the Sichuan Science and Technology Program under Grant 2019YFG0405 and Grant 2019YFG0141, and in part by the 6th Innovation and Entrepreneurship Leading Talents Project of Dongguan.

ABSTRACT Deep learning has made significant progress in many fields such as image identification, speech recognition and natural language processing, especially in the field of computer vision. The better performance of the neural network often built on deeper, wider network structure, more network parameters and more storage and often computational expensive. As a result, it is hard to deploy neural network to mobile and embedded devices. Therefore, compressing of convolutional neural networks is very necessary and practical. In this paper, we propose a channel pruning algorithm for depth-wise separable convolution units and introduce a new channel selection algorithm based on information gain and a method for quickly recovering network performance after pruning. The proposed method is implemented on MobileNet and validated on several popular datasets. The experimental results show that our method can achieve better experimental results on several image classification datasets, and also achieve good detection results on the PASCAL VOC image detection dataset.

INDEX TERMS Deep learning, channel pruning, convolutional neural networks, depth-wise separable convolution unit.

I. INTRODUCTION

Convolutional neural networks have caught a lot of attention since Alex presented AlexNet [1] in 2012 and won the ILSVRC2012 (Large Scale Visual Recognition Challenge 2012). In recent years, convolutional neural networks have achieved extremely high accuracy in various computer vision tasks such as image recognition, image detection and image segmentation. It has become the dominative method for solving computer vision problems.

Convolutional neural networks rely on tens of millions of parameters in the network to participate in the calculation, and have some disadvantages, such as complex network structures, large computational complexity, slow training speed, and difficult to deploy on embedded devices. For instance, AlexNet has five convolutional layers and three fully connected layers, with a total of 60 million network parameters. It cost more than two days to train on the ImageNet [2] dataset with a single NVIDIA K40. As the number of network model layers and parameters increases, reducing their size and computational cost is critical, especially for real-time applications such as online learning and reinforcement learning. Furthermore, in recent years, VR, AR and smart wearable devices have developed rapidly, and researchers are urgent to solve the problem of deploying large-scale learning systems to mobile devices. Achieving this goal requires integrated solutions of many fields, including but not limited to machine learning, optimization methods, computer architecture, data compression, indexing and hardware design.

Related works [3], [4] have shown that the computational complexity of convolutional neural networks is mainly concentrated in the convolutional layers. For the problem of feature redundancy in the convolutional layers, some researchers [5], [6] proposed channel pruning to remove the less important channels in the convolutional layers to compress the network structure. These methods work well in convolutional neural networks stacked by conventional convolutional layers with parametric redundancy but not fit in efficient convolutional structures such as depth-wise separable convolutional units. Therefore, this paper proposes a two-stage channel pruning method for depth-wise separable convolution units,

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad E. H. Chowdhury.

which uses a parameter reconstruction method to quickly recover the network performance, and introduces a new channel selection algorithm based on information gain. In summary, the main contributions of this paper can be listed as follows:

(1) We propose a novel channel pruning method based on information gain and achieve state-of-the-art results on standard benchmarks.

(2) We report a parameter reconstruction method with linear regression algorithm so that it could fix parameters and make up the accuracy degradation caused by pruning in a short time.

II. RELATED WORK

A. APPROACHES OF MODEL COMPRESSION

In recent years, scholars have made many progress in network compression and acceleration, and proposed varieties of lightweight algorithms, such as network pruning [6]–[8] network quantization [6], [9]–[15], low-rank approximations [17], knowledge distillation [18] and neural network architecture design [3], [4], [21], [22].

Han et al. introduced a simple pruning strategy: if the weight of a connection is lower than threshold, it will be regarded as a low contribution connection and be removed, and then with some fine-tuning operations, the accuracy of the neural network will be preserved [6]. This pruning process can be implemented iteratively until we derived a sufficiently sparse network. However, compared to the unpruned network, the pruned network may not have a big advantage in terms of speed unless there are special software or hardware solutions to the sparsity storage and calculation. Wei Wen came up with the Structured Sparsity Learning (SSL) [7] method to solve this problem. They add sparsity regularizations into different structures (like filters, channels, and etc.) of the network, as a result, partial weights get to 0 and be removed, and then a hardware-friendly structured sparsity will be gotten. Compared to the original model, this method won't introduce sparseness, but it will bring additional consumption in training process. Liu et al. tried to fix this issue by applying L1 regularization to the scaling parameter in batch normalization directly [8], but it turns out that the training from scratch is still time-consuming.

Network quantization [9] attempts to cut down the number of bits occupied by weights to achieve network compression. Gong *et al.* [9] and Chen *et al.* [10] achieved this goal by applying weight quantification and k-means clustering Vanhoucke et al. obtained a 4× speedup [11] without accuracy lost by applying 8-bit quantification to network. Besides, S. Gupta et al. even implemented 16-bit floating-point quantization in training process, and get a distinct cost degradation of memory and floating-point computation with tiny loss in accuracy [12]. To reduce the size of the model further, W. Chen et al. advocated to divide weights into different groups so that weights in a group could share a common value [13]. In addition, Han et al. achieved a compression of VGG-16 model from 552MB to 11.3MB without any degradation of accuracy [6]. Courbariaux et al. gave a definition of Binarized Neural Networks (BNNs) with whose weights are all quantified to $+1$ or -1, and achieved state-of-the-art performance on MNIST and CIFAR-10 datasets [14]. Rastegari et al. introduced XNOR-Net [15] which perfectly speeds network up yet performs badly on large-scale dataset like ImageNet to further cut down the computational cost by the way of the binarized weights and the binarized inputs.

Singular-value Decomposition (SVD) is a prevalent means to low-rank matrix decomposition. Denton et al. just use this method to implement matrix approximation, and indirectly reduce the computational consumption by redundancy of the matrix [17]. Though this approach could achieve a 3× parameter compression without computational cost loss, it is usually used in fully connected layers while most calculations of neural network existed in the convolutional layers. In the convolutional layers, parameters are usually stored in a multidimensional matrix whose dimension is typically as a form like this: $d \times h \times w \times c$, and d is the output dimension, h and w is the height and width of the convolution filters respectively, and c is the input dimension of the convolutional kernels. The essence of the low rank decomposition is to decompose the parameter matrix into a series of small matrices combinations with linear algebra so that the combination of small matrices is basically consistent with the original convolutional layer in discriminative ability, so it can be inferred that we could preserve sufficient accuracy and greatly reduce memory cost of parameters by this way. Low-rank approximation will be carried out layer by layer from shallow convolutional layers to deep ones, and when a low-rank decomposition finished, the parameters of current layer will be fixed and fine-tuned with a reconstruction error.

Caruana et al. creatively introduced Knowledge Transfer (KT) to compress network models, which aims at making a better use of the existing knowledge [18]. The small, compressed network with soft tags of the strong models they trained could achieve similar output to the original network for shallow networks, but it is not suitable for the deep networks. Furthermore, Knowledge Distillation (KD), which is defined by Ba et al. in order to compress a deep and wide network into a small network, achieved a similar performance to the original model by imitating the output of original network [19]. High deployment demand for embedded devices stimulates the development of neural network architecture design. For example, GoogleNet uses the Inception module rather than a simple stack of layers to reduce the computational cost [20], and ResNet [4] achieves an excellent image recognition performance by introducing bottleneck architectures. On the basis of that, ShuffleNet [21] combines pointwise group convolution with channel shuffle, and achieves a great acceleration on ResNet. MobileNet [4] achieves state of the art compression effect with depth-wise separable convolution. However, thought these methods have made a lot of contributions to network acceleration, they neglect to consider the size reduction.

B. DEPTH-WISE SEPARABLE CONVOLUTION UNIT

The main layers for computation in a convolutional neural network are the convolutional layers. The calculation principle of the convolutional layers is as follows:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

If we assume that I is the input layer, K is the convolutional kernel, $D1$ is the width and height of the input feature map, $D2$ is the size of convolutional kernel, and M is the dimension of the input feature map, then we can get that the size of I is $D1 \times D1 \times M$, the size of convolutional kernel is $D2 \times D2 \times M \times N$, and I will get an output feature map with a dimension of $D1 \times D1 \times N$ through the convolutional kernel, so the computational complexity is $O(D1 \times D1 \times D2 \times D2 \times M \times N)$, and the space complexity of convolution layers is $O(D2 \times D2 \times M \times N)$. The total computational complexity and space complexity of a convolutional neural network is the sum of the computational complexity and space complexity of each convolutional layer.

Xception [22] mentioned that the convolutional kernel can be seen as a three-dimensional filter: channel dimension + space dimension (corresponding to the width and height of the feature map), The convolution operation actually implements a joint mapping of channel correlation and spatial correlation. The Inception module has an assumption that the combination of spatial features and the combination of channel features of the convolutional layers can be performed separately and can achieve better results in this case. And then a depth-wise separable convolution can decomposed conventional convolution into a depth-wise convolution and a 1×1 pointwise convolution [4].

Assume that the size of input feature map is $D1 \times D1 \times M$, the size of output feature map is $D1 \times D1 \times N$, and the size of convolution kernel is $D2 \times D2 \times M \times N$. According to the previous calculation, the computational cost of the traditional convolution is: $D1 \times D1 \times D2 \times D2 \times M \times N$. If the calculation is performed with a depth-wise separable convolution, it's computational cost is the sum of the computational cost of the depth-wise convolution and the 1×1 convolution: $D1 \times D1 \times D2 \times D2 \times M + M \times N \times D1 \times D1$.

The ratio of the computational cost of the conventional convolution to the depth-wise separable convolution is:

$$\frac{D1 \times D1 \times D2 \times D2 \times M + M \times N \times D1 \times D1}{D1 \times D1 \times D2 \times D2 \times M \times N} = \frac{1}{N} + \frac{1}{D2^2}$$

Generally, we use a 3×3 convolutional kernel, and the value of 1/N is negligible. It can be seen that the computational complexity of the depth-wise separable convolution is reduced to about 1/9 of the conventional convolution.

From the above analysis, the depth-wise separable convolution structure of the 1×1 convolution greatly reduces the computational cost in the forward. Moreover, in 3 MobileNet [4], about 95% of the multiply-and-accumulate operations come from the 1×1 convolution (95% of the number of parameters), and the large use of 1×1 pointwise convolution means that some highly optimized Matrix multiplication algorithms (such as GEMM) can be implemented to improve computational efficiency.

C. CHANNEL PRUNING

Channel pruning has been widely used in model compression of convolutional neural networks. In early work, channel pruning was considered as a way to effectively reduce network complexity and reduce overfitting. For instance, reference [6] pruned the best performing model so far, and reduced network complexity without losing accuracy. The general steps for network pruning are as follows:

(1) Use common methods to train a complete convolutional neural network.

(2) Sort the weights of each layer by absolute values.

(3) Select the weights whose absolute value is below a certain threshold and remove them.

(4) Re-train the pruned network to achieve the best performance of the model before pruning.

Generally, this method can greatly reduce the number of network parameters in the fully connected layers. The weights pruning can reduce the amount of parameters of the network, but it also has the following disadvantages:

(1) Those pruning methods are only used for the fully connected layers, and the fully connected layers are often the part which has most redundant parameters. As a result, in practical applications, we tend to discard the fully connected layers and replace them with the average pooling layers. Therefore, in the existing convolutional neural networks, the convolutional layers often account for most of the computational complexity and the most part of time. In general, the algorithms described above are capable of achieving faster speed or less storage capacity, yet they rarely achieve significant acceleration while compressing the entire network.

(2) Weight pruning is prone to sparse connections, but the sparse neural network structure is not as efficient as the original neural network with tight connections.

Due to the above problems, more and more scientists have begun to turn their attention to the channel pruning. Channel pruning [5] is another way of weight pruning. Unlike the removal of a single neuron connection in neuron pruning, channel pruning removes less important channels from the entire convolutional layers. This method has the following advantages over neuron pruning:

(1) The entire channel is removed, so sparse connections will not be introduced.

(2) The speed in forecasting will be greatly improved.

The ordinary channel pruning of convolution layers is shown in Figure 1. Using a triplet $<Li; Wi; *>$ to represent the i_{th} depth-wise separable convolution unit, so that $Li \in R^{c\times h\times w}$ is the input vector, $Wi \in R^{d\times c\times k\times k}$ is a set of convolution filters, and the * stands for convolution operations. Note that each filter corresponds to one channel of the activation layer (as shown by the yellow layer in the middle of Figure 1), so the discriminative ability of each channel filter is

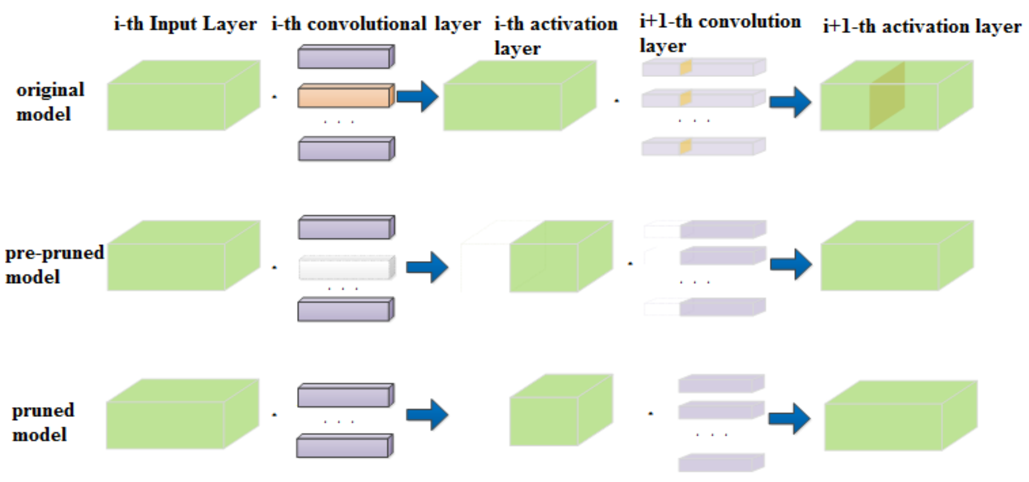


FIGURE 1. The ordinary channel pruning of convolution layers.

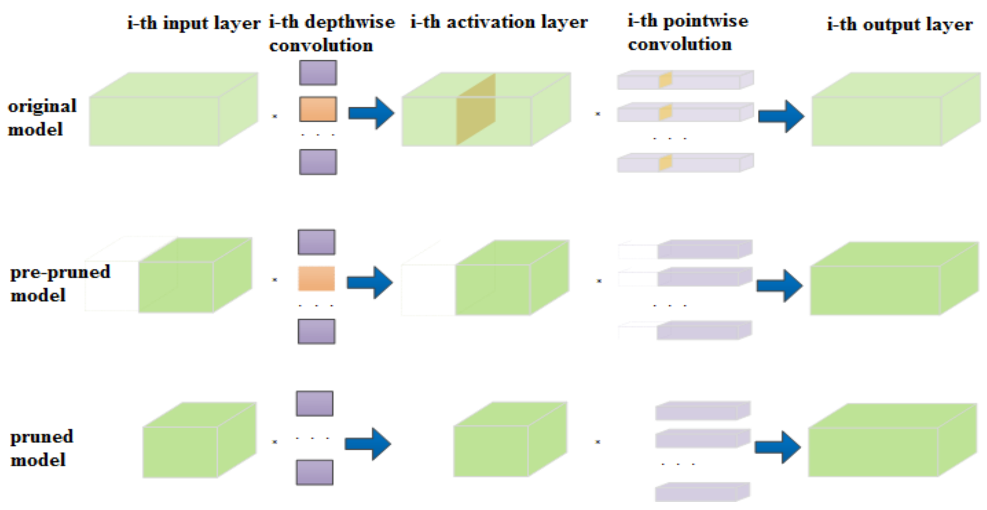


FIGURE 2. The channel pruning method based on the depth separable convolution unit.

closely related to its corresponding activation layer. A simple strategy for calculating the discriminative ability of a filter is to calculate the Average Percentage of Zeros (APoZ) of each channel of the activation layer: the higher APoZ, the lower importance of the filter, and if the ApoZ is below a certain threshold, then remove it.

III. OUR APPROACH

A. FRAMEWORK

As mentioned in the previous section, the depth-wise separable convolution unit consists of a 3×3 depth-wise convolutional layer and a 1×1 pointwise convolution. The main computational cost of this unit is concentrated in the 1×1 pointwise convolution. If we prune the channels derived by the 3×3 depth-wise convolutional layer, the number of input channels of the 1×1 convolution layer will be significantly reduced, thereby reducing the computational complexity of the 1×1 pointwise convolution. From another point of view, in the depth separable convolution, the 3×3 depth-wise convolutional can be regarded as the process of feature extraction, which extracts the features of each channel of the feature layers. When the features are extracted, the features will be combined through the 1×1 pointwise convolution to get useful features, which is similar to the feature engineering in traditional machine learning. The introduction of channel pruning after deep convolutional layers can be seen as introducing a feature selection process, retaining the more important features in image classification, detection or segmentation tasks, and filtering out less important features. The channel pruning method based on the depth separable convolution unit is explained in Figure 2.

Using a triplet $<Li, Di, Pi>$ to represent the i_{th} depth-wise separable unit, Li is the input unit, Di is the depth-wise convolutional layer in the depth-wise separable convolution structure, and Pi is the pointwise convolution in the structure, which is the pointwise convolution part. The activation layers in figure 2 refer to the activation layers next to the Di layers,

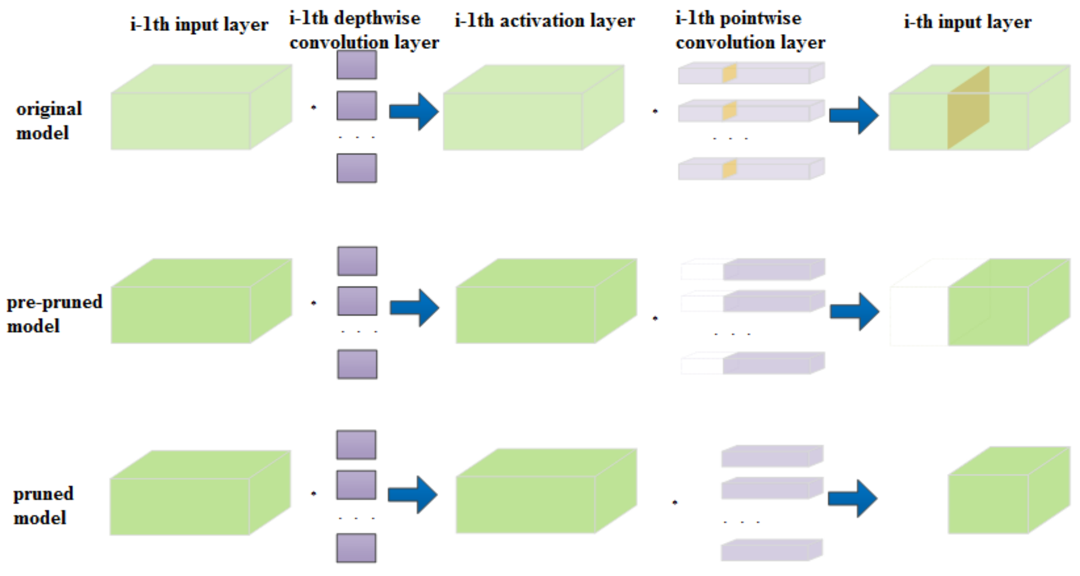


FIGURE 3. **The effect on the previous layer when the current layer pruned channels.**

which is the input unit of the pointwise convolution. The channel pruning aims to cut off the less important channels in the activation layers, and remove the corresponding convolutions in *Di* and *Pi* at the same time.

Each filter in *Di* corresponds to a channel in the activation layer, and the importance of the feature channels is often evaluated by some indicators. The light color layers in the activation layer represent the less important output channels that should be removed. At the same time, the corresponding convolution filters in *Di* and the convolution filters with the channels as the input in *Pi* will be removed together, and the output dimension of the depth-wise separable convolution unit will not change. According to the calculation principle of the depth-wise separable convolution, the channel j of the activation layer is convoluted by the j_{th} channel of the input unit and the 3×3 filter of the j_{th} channel of the deep convolution layer, so once the j_{th} channel of activation layer is removed and the j_{th} channel of the input layer should also be removed. For the convolution filters which should be removed, the weights of their corresponding convolutional layer and their corresponding learning rate are set to 0 during the pruning process, so it will not be participated in the calculation and inference process. Specifically, it is shown in dotted lines in figure 2. As mentioned above, the computational cost of the whole module is $D1 \times D1 \times D2 \times D2 \times M + M \times N \times D1 \times D1$, and now some channels are removed by the channel pruning. It should be noted that the pruning process does not reduce the output dimension of the convolution unit, but only cuts off the less important feature layers and its corresponding weights, so the current computational cost, i.e., $D1 \times D1 \times D2 \times D2 \times M \times (1\text{-}\varepsilon) + M \times N \times D1 \times D1 \times (1\text{-}\varepsilon)$ is much lower than that of the original convolutional neural network, and the output dimension size N does not change.

As shown in Figure 3, the pruning of the i_{th} depth-wise separable convolution unit will affect the $i\text{-}1_{th}$ convolution unit. Assume that the j_{th} channel of the input layer *Li* of the i_{th} depth-wise separable convolution unit is removed, and *Li* is generated by the pointwise convolution *P(i-1)* of the (i-1)$_{\text{th}}$ depth separable convolution structure, so the 1×1 convolution of *P(i-1)* can also be removed. It can be known that the channel pruning based on the depth-wise separable convolution unit has better discriminative ability than the ordinary convolution unit, especially when dealing with the multi-layer pruning, the layer-to-layer influence can be avoided as much as possible and the pruning can be more thorough.

A complete process of channel pruning based on depth-wise separable convolution units is shown in Figure 4. First, get a trained MobileNet model by following complete training steps. Then, for the ith depth-wise separable convolution unit <*Li*, *Di*, *Pi*>, the importance of all the output channels of the deep convolutional layer *Di* is obtained according to an importance evaluation criterion, and then the convolution channels will be sorted in order of importance, and those channels with the least importance and their corresponding convolution filters in *Pi* and *Di* and the input channel in *Li* will all be removed. Moreover, the convolution filters in the previous layer <*L(i-1), D(i-1), P(i-1)*> of the depth separable convolution unit that corresponding to *Li* also needs to be removed, so that the pruning process for the i_{th} deep separable convolution unit will be finished. And when the pruning of all convolution units are completed, the network fine-tuning method can be used to restore the performance of the model loss after pruning.

B. CHANNEL SELECTION ALGORITHM BASED ON INFORMATION GAIN

The study of channel importance evaluating criteria is a major research hotspot in the community. At present, the mainstream importance evaluation methods are as follows.

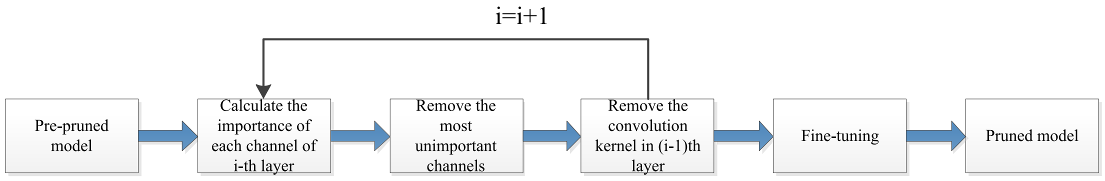

i=i+1
Pre-pruned model
Calculate the importance of each channel of i-th layer
Remove the most unimportant channels
Remove the convolution kernel in (i-1)th layer
Fine-tuning
Pruned model

FIGURE 4. The overall flow of channel pruning.

(1) Evaluated according to the APoZ indicator. The higher APoZ, the more units not activated, and the lower the importance.

(2) Evaluated by calculating the sum of the absolute values of the weights of the channel filters (i.e., $\sum |W(i,j)|$) [26]. The smaller the value, the smaller the activation threshold of the feature map generated by the channel filter is, and the channel is more tend to be removed.

(3) Some scholars believe that if the weights are used as the standard clipping, some useful filters may be cropped, so J. Luo proposed a method for evaluating whether filters are important based on entropy [27].

(4) Yang believes that the previous pruning methods do not take the bandwidth and energy consumption of the model into account, and cannot pruning the model to the maximum extent from the perspective of energy utilization. Therefore, he proposed a pruning method based on energy-efficiency [28].

(5) Some scholars adopt the idea of genetic algorithm to randomly prune the neural network, and select the model with good effect in use [29].

(6) Some scholars believe that if the activation of a certain neuron in a layer has a strong correlation with the activation of a certain neuron in the previous layer, then this neuron has a strong discriminative effect on the activation of the next layers [5].

(7) Hu considers the pruning problem as a combinatorial optimization problem [29]. Choosing an optimal group B from the lots of weight parameters to minimize the loss of cost function of the cropped model.

These channel selection algorithms are mostly based on the mathematical significance of the channel in convolutional layers to indicate its importance. In the depth-wise separable convolution unit, each convolutional layer derived by depth-wise convolution can be regarded as a feature, so the importance of each channel can be evaluated by the common methods of evaluating feature importance.

Firstly, we will give a definition of channel importance evaluation standard based on entropy, and on the basis of this standard, we will introduce a new Information-Gain-based channel importance evaluation standard, and then we will explain its ideas, calculation principles and specific implementation process.

In information theory, entropy is defined as the expectation of a random variable. The greater the entropy, the greater the uncertainty of the variable, and the greater the amount of information contained in the random variable. Information entropy is a concept used in information theory to measure the amount of information. The more ordered a system, the lower the information entropy; conversely, the more chaotic a system is, the higher the information entropy is, so information entropy can also be used as a measure of the degree of system ordering.

The calculation formula of information entropy is: $H(x) = -\sum_{k=1}^{n} p(k) \times \log_2 p(k)$.

Algorithm 5-1 The Entropy Calculation Flow of Channels of Convolution Layers

```
Require: dataset $D$; depth-wise separable convolution unit $<L_i, D_i, P_i>$;
1. Get the output $O_i$ of activation layers with dataset $D$ and the depth-wise separable convolution unit $< L_i, D_i, P_i >$
2. Randomly select 20 points on the spatial dimension
3. Select a channel j and calculate the information entropy of the 20 points
4. Average all the entropies derived to get the average entropy $E_j$
5. $j$++
6. return $E_j$
```

For input layer *Li*, when the feature extraction is performed by the depth-wise convolution *Di*, the nonlinearity will be introduced by the Rectified Linear Unit (ReLU), and then the activation layer *Ai* will be derived. A simpler way of calculating the entropy of each channel of the activation layer is presented in Alg. 5-1. Firstly, select a dataset, and for each sample of the dataset, get the output of each channel convoluted by the depth-wise convolutional layer *Di*. Considering the efficiency of the channel selection process, for the entropy calculation of each channel output, we randomly select only 20 points to balance the efficiency and the costs since 20 works good for our experiments. For each point, divide the value range into 10 intervals, count the number of values in each interval, and use this as a basis to calculate the probability of the value of each position of the channel, and then the information entropy of the position can be calculated. Then the sum of the entropies of the various locations is the total information entropy of the channel, which represents the amount of information for this output channel. Combined with the definition of information entropy, the formula for calculating the entropy of a single channel is shown as follows:

$$H(x) = -\sum_{j=1}^{20}\sum_{k=1}^{10} p_j^k \times log_2 p_j^k$$

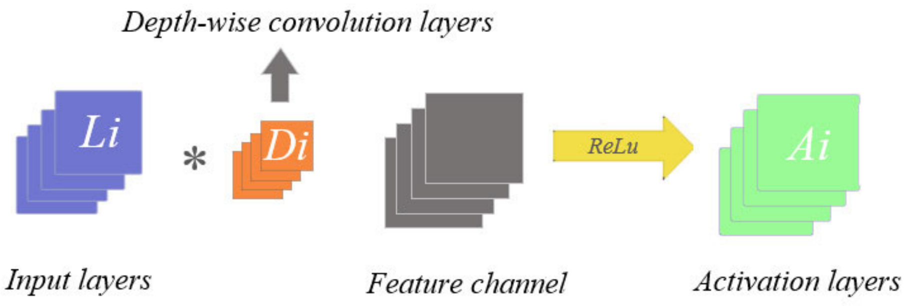


FIGURE 5. The generation process activation layers.

where p_j^k is the probability that the j_{th} point of the activation layer falls on the interval k, and $H(x)$ is the information entropy of the convolution channel. In fact, from the perspective of APoZ, if an activation layer is not important enough, most of the activation values in the layer are 0, and the information entropy of the layer will be small. Therefore, the entropy-based approach is a method used to assess the importance of channel [27]. Specifically, the smaller the information entropy $H(x)$ corresponding to the activation layer Ai, the lower its importance, and the corresponding depth wise convolution Di is more tend to be removed.

Information gain is a concept often mentioned in the decision tree algorithm [30], which is a tree algorithm that classifies instances based on features. And its basic principle is very simple: from the multiple features of the dataset sample, select the appropriate features, and recursively divide the dataset according to the different feature values, so that each sample has a relatively correct classification process. Appropriate features are critical to the decision tree splitting, and information gain is a common criterion for feature selection.

The information gain is defined as the difference in entropy of pre-divided and the divided dataset that divided by a certain feature. For the dataset D to be divided, the amount of information (entropy) in the data is definite, but the entropy after division by feature selection is uncertain. The smaller the entropy value, the smaller the uncertainty of the subset obtained by dividing this feature. The bigger difference in entropy values between the pre-divided and the divided datasets, the greater amount of information contained in the feature. Therefore, the importance of each feature channel can be evaluated by the magnitude of the information gain.

As shown in Figure 5, the input layer Li performs feature extraction through the spatial convolution layer Di, and then introduces nonlinearity through a Rectified Linear Unit (ReLU) to get the activation layer Ai. The activation layer Ai holds the output Oi of the entire depth-wise separable convolution unit through the 1×1 pointwise convolution. In order to calculate the information gain of each channel of the activation layer Ai, we can calculate the difference between the Oi information entropy before and after the operation. The specific calculation steps are as shown in Algorithm 5-2:

Algorithm 5-2 The Information Gain Calculation Flow of Channels of Convolution Layers

```
**Require:** dataset $D$; depth-wise separable convolution unit $<L_i, D_i, P_i>$;
1. Get the original information entropy (original_entropy) of output layer
$O_i$ with dataset $D$
2. Set all features of channel j in $A_i$ layer to 0 and get a new activation layer $A_i'$
3. Convolve $A_i'$ and the $1 \times 1$ pointwise convolution layer, and get a new
ouput $O_i'$
4. Calculate new information entropy(new_entropy) for $O_i'$
5. Get $G_j$ By calculate the difference between new_entropy and original_entropy
6. $j$++
7. return $G_j$
```

According to the steps of Algorithm 5-2, the information gain of each channel of the feature layer Ai can be derived. Then, according to the information gain, the feature channel with less information gain under the prune ratio and its corresponding 3×3 depth-wise convolution and 1×1 pointwise convolution should be removed together. The parameter reconstruction then will be performed with linear regression, and the pruning of a single depth-wise separable convolution unit could be completed.

1) PARAMETER RECONSTRUCTION BASED ON LINEAR REGRESSION ALGORITHM

The channel pruning algorithm based on the depth-wise separable convolution unit reduces parameters and network computation complexity of the convolution layers by removing the less important feature channels of the depth-wise convolution output and their corresponding convolutional kernels. It can be seen from the previous analysis of network pruning that the method does not reduce the dimension of the output layer Oi of the convolution unit, so the linear regression algorithm can be used to reconstruct the parameters of the corresponding separable convolution unit in the channel pruning so that the Oi' layer of the network after pruning

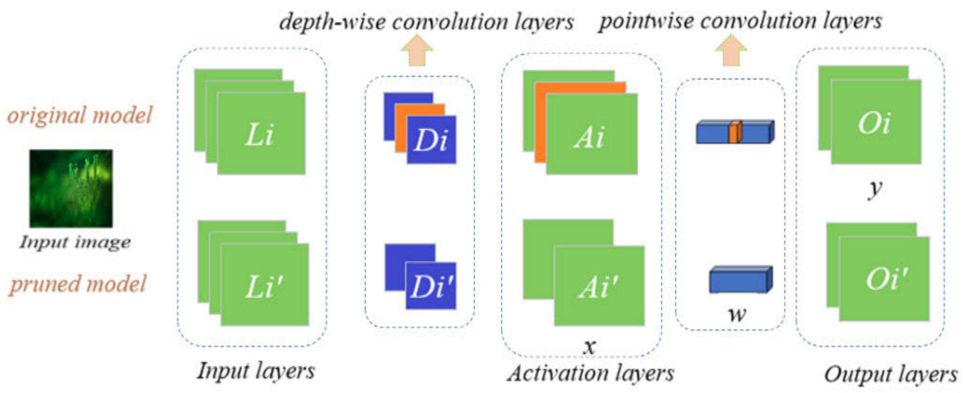


FIGURE 6. Parameter reconstruction with linear regression algorithm.

can fit the output of the *Oi* layer before pruning as much as possible.

As shown in Figure 6, using the output *Oi* of the i_{th} depth separable convolution unit <*Li*, *Di*, *Pi*> before pruning to direct the output *Oi'* of the convolution unit <*Li'*, *Di'*, *Pi'*> after pruning. The specific method is to use the linear regression algorithm to minimize the square error between the two outputs. The square error is defined as follows:

$$J(\theta) = \frac{1}{2n} \sum_{m}^{n} (\theta^T x^i - y^i)^2$$

Linear regression algorithm is a regression analysis that uses the least squares function of a linear regression equation to model the relationship between one or more independent and dependent variables. This function is a linear combination of one or more model parameters called regression coefficients (the arguments are all first power). The case of only one independent variable is called simple regression, and the case of more than one independent variables is called multiple regression. For our parameter reconstruction, it needs to derived a group of new weights to get a $H \times W \times C$ shaped output *Oi* (H and W is the height and width of the output layer, and C represents the dimension of the output layer), and it often has many variables. In order to reduce the time to reconstruct each convolutional layer, this chapter only randomly selects m points in the output layer as the dependent variable y, and selects the output of the previous layer of channel pruning as x, then the squared error function is used as the loss function in the reconstruction process, in which n is the size of the training set, θ is the reconstructed convolution parameter to be calculated. Every time we pruned a layer, we would apply linear regression to that layer with less channels to get a group of new parameters to fit the original output.

Using linear regression after channel pruning of each convolution unit to reconstruct the parameters in the pointwise convolution is equivalent to the layer-by-layer fine-tuning training to reduce the network error, but it is more computationally efficient.

In summary, we followed the process as shown in Figure 7 to prune MobileNet model.

IV. EXPERIMENT

To verify the effectiveness of the algorithm, we conduct experiments on multiple datasets. The experiments are carried out under the Tensorflow [30] deep learning framework.

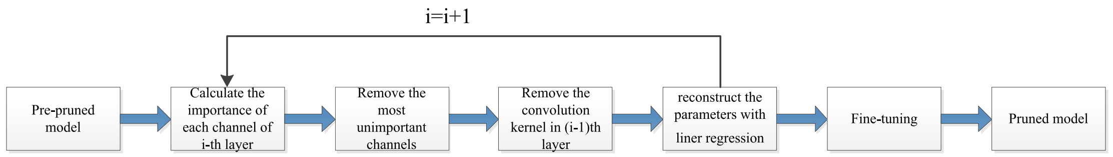
A. IMAGE CLASSIFICATION ON CIFAR-10

1) DATASET PREPARATION

The original image size is 32 × 32. Enlarging the image to 224 × 224 can improve the performance of the model partly and increase the accuracy of the model, but it will greatly increase the training time. The purpose of this chapter is to prove that the proposed pruning algorithm can significantly reduce the size and complexity of the model with less accuracy in model loss. In order to ensure that the experiments can be completed quickly, the original images will not be enlarged in the experiment, but the images will be preprocessed as follows. Before training, four pixels of zero padding are applied around each picture, and the original picture of 32 × 32 is filled to a size of 40 × 40, and then the picture is randomly cropped to a size of 32 × 32. Finally, we randomly flip the image in the direction of left and right, so that the picture for model training is derived. In the test process, the original test data is used as input, and no pre-processing is performed.

2) IMPLEMENTATION DETAILS

In this experiment, an NVIDIA GTX GPU was used for the calculation. The loss function in the training process is the cross entropy cost function, and the optimal solution of the parameters is Adam [32]. The initial learning rate was set to 0.06, and a total of 100 epochs were run. At the 30th, 60th, 80th, and 90th epoch, the learning rate was multiplied by 0.1, and a batch was set to train 64 pictures, a total of 78125 iterations. The experiments in this chapter are designed to compare the performance differences between the



i=i+1
Pre-pruned model
Calculate the importance of each channel of i-th layer
Remove the most unimportant channels
Remove the convolution kernel in (i-1)th layer
reconstruct the parameters with liner regression
Fine-tuning
Pruned model

FIGURE 7. Flow of network pruning with parameter reconstruction.

TABLE 1. The size of each layer of MobileNet.

layer	channels	stride	FLOPs
Depthwise_sep_conv_1	64	1	1048576
Depthwise_sep_conv_2	128	2	1048576
Depthwise_sep_conv_3	128	1	2097152
Depthwise_sep_conv_4	256	2	1048576
Depthwise_sep_conv_5	256	1	2097152
Depthwise_sep_conv_6	512	2	1048576
Depthwise_sep_conv_7	512	1	2097152
Depthwise_sep_conv_8	512	1	2097152
Depthwise_sep_conv_9	512	1	2097152
Depthwise_sep_conv_10	512	1	2097152
Depthwise_sep_conv_11	512	1	2097152
Depthwise_sep_conv_12	1024	2	1048576
Depthwise_sep_conv_13	1024	1	2097152

original network and the pruned network, so no pre-training models were used during the experiment.

According to the above data pre-processing and model training methods, the MobileNet model achieved a training accuracy of 0.9703 and a test accuracy of 0.81 on the CIFAR-10 [31] standard dataset. The FLOPs and the pb file size saved by the model are 22482944 and 15M respectively, which means a fairly small model. The layer size and floating point computational cost of each depth separable convolution layer of the model are shown in Table 1.

3) RESULTS AND ANALYSIS

In this experiment, we perform experiments according to the process shown in Figure 7. In the pruning process, the channel importance evaluation criteria based on weight sum [26], APoZ [5], information entropy [27] and information gain are used to select unimportant channels for pruning. After the single depth-wise separable convolution unit is pruned, the linear regression algorithm is used to reconstruct the parameters. After the whole model is pruned, the network is fine-tuned according to the first training process of the network.

After pruning, the FLOPs, model size, forward calculation speed of each prune ratio is shown in Table 2.

It can be seen that as the prune ratio increases, more convolution filters with less importance in the model are filtered out, the floating-point operations per second (FLOPs) of the model is significantly reduced, and the model size

TABLE 2. Change of size and speed after pruning.

Prune ratio	FLOPs	Model size (MB)	Forward calculation speed (ms)
0	22482944	15	4.19
0.5	5652675	7.3	1.08
0.6	3888406	5.9	0.99
0.7	2497353	4.5	0.69
0.8	1465704	3.1	0.51
0.9	831403	1.7	0.37

is greatly reduced. At the same time, since the number of convolution operations is greatly reduced, the forward speed of the model is significantly improved. When the prune ratio reaches 0.7, the FLOPs after pruning have dropped to about 1/10 of the initial model, the model size is about 1/3 of the initial model, and the operation speed is only 1/6 of the previous one (within 1 ms), which is basically suitable for use in mobile or embedded devices.

Comparing the effect of different channel selection methods is to compare how much information the remaining convolutional layer channels retained after the channel pruning when compared to the original convolutional layer. The more information retained, the less performance loss between the pre-pruned and pruned model. Therefore, this group of experiments firstly compared the test accuracy changes before fine-tuning the pruned network under the three evaluation criteria of APoZ, information entropy and information gain. The curve of variation is shown in Figure 8:

It can be known that no matter which pruning method is adopted, the training accuracy rate reduce rapidly, but the pruning method based on information gain preserves the model performance as much as possible compared with the other two methods. When the prune ratio is not more than 0.6, 70% or better the test accuracy can be guaranteed, which indicates that the information gain based pruning method retains as many important feature layers as possible and removes the less important features in the depth separable convolution unit as compared to the other two methods. The fine-tuning training can improve the accuracy of the network to some extent, since Figure 9 shows that the accuracy of the model with fine-tuning is much better than cases of without fine-tuning.

It can be seen that when the prune ratio is no more than 0.65, the test accuracy of the model with the information gain-based pruning method is improved compared with the original model, which proves that the convolutional layer pruning method can properly alleviate the overfitting of

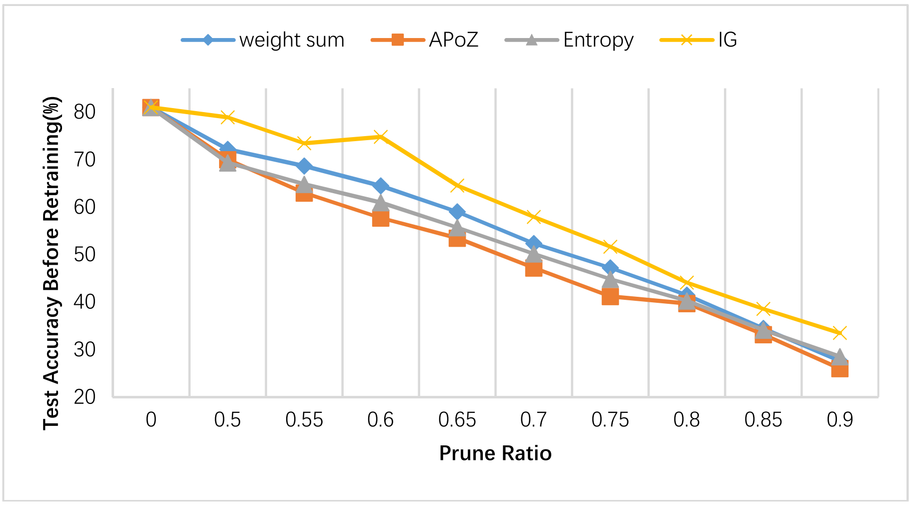


FIGURE 8. The change of test accuracy of each pruning method before retraining.

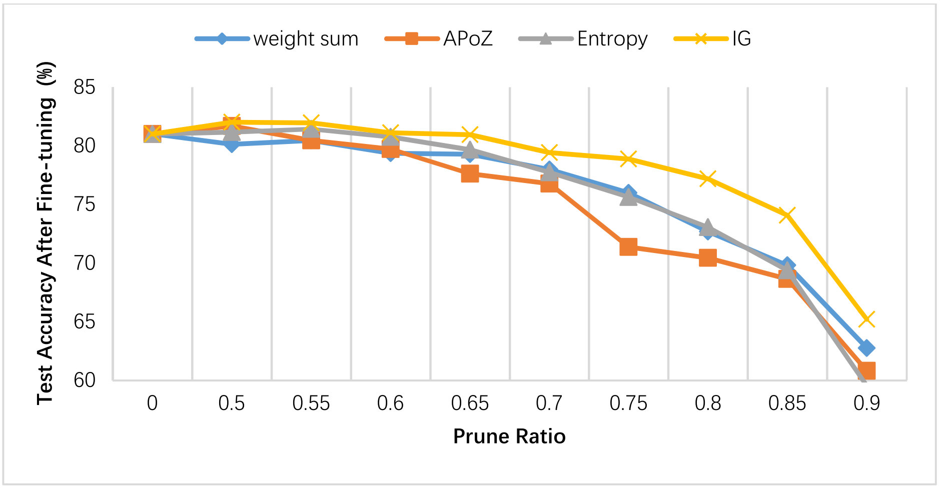


FIGURE 9. The test accuracy changes after each pruning method is retrained.

model. And the figure also shows that after the retraining, the information gain-based pruning method can still get better test accuracy, especially in the case of large prune ratio.

Now let's evaluate the performance of the model after pruning from another angle. In the last 10,000 iterations of the model retraining, the training accuracy rate will be calculated every 100 iterations, and then those training accuracy rates will be averaged to get the final average training accuracy of the model. As shown in Figure 10, the channel pruning method based on information gain has more obvious advantages on the training set, which indicates that the method retains the useful information in the original model as much as possible, and fits the training data well.

In a convolutional neural network, the shallow convolutional layers often contain more image feature information which is important for image classification tasks. Therefore, the following pruning strategy can be adopted, that is, for shallow network of the depth-wise separable convolution unit uses a relatively small prune ratio, while the deeper convolution layer selects a larger prune ratio.

According to the above strategy, we use the prune ratio of 0.75 in the 2nd, 4th, 6th, 7th, 8th, 9th, 10th, 11th, and 13th depth-wise separable convolution unit which has more channels, while in other convolution units take a prune ratio of 0.5, and the model FLOPs is obtained as 2666496.0. Compared with the model derived by the prune ratio of 0.7, the computational complexity is similar, but the performance of the model is greatly improved. The specific performance is shown in Table 3, which shows the method of pruning according to different proportions of channels further compressed the neural network while maintaining the performance of the model.

4) THE ROLE OF LINEAR REGRESSION RECONSTRUCTION MODEL

The idea of the parameter reconstruction method is to reconstruct the parameters by linear regression so that the pruned input X fits the previous convolutional layer output Y as much as possible. The squared error is used to measure the difference between the new convolutional layer output and the

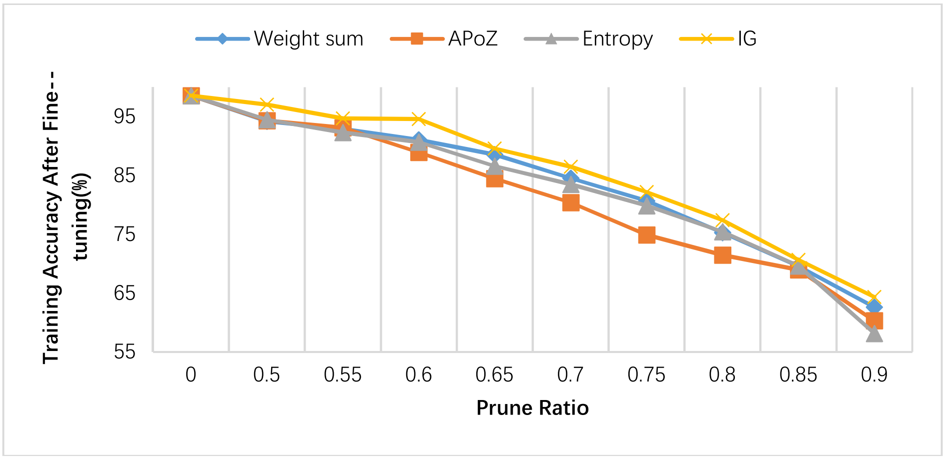


FIGURE 10. Changes in training accuracy of each pruning method.

TABLE 3. Changes in the accuracy of the pruning model.

Pruning strategy	Accuracy_before_Finetune	Accurac_after_Finetune	Ave_Accuracy
Weight sum	59.35%	59.35%	85.46%
APoZ	58.33%	79.38%	85.38%
Entropy	60.18%	78.93%	85.86%
IG	63.23%	79.73%	87.79%

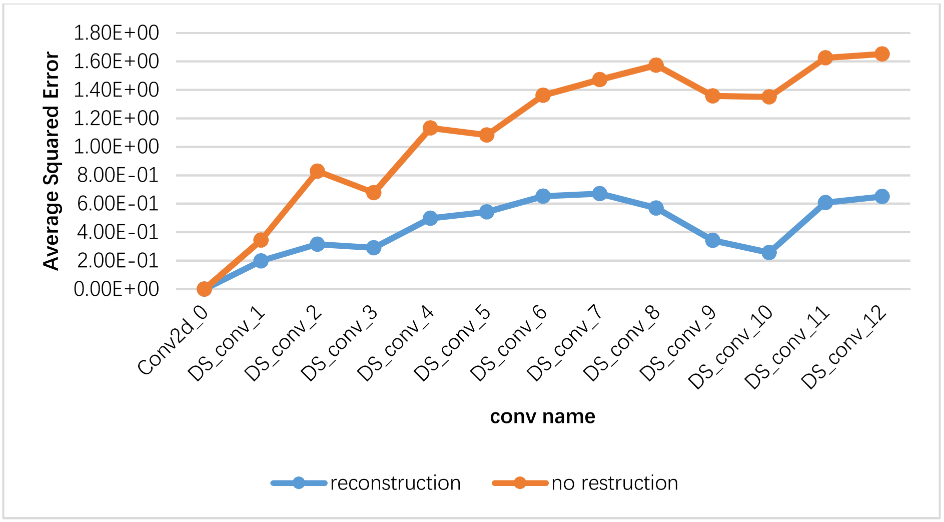


FIGURE 11. Squared error curve for each layer.

original convolutional layer output. The calculation formula is as follows:

$$J(\theta) = \frac{1}{2n} \sum_{m}^{n} (\theta^T x^i - y^i)^2$$

We use Figure 11 to analyze the variation of the average squared error of each layer in the layer-by-layer pruning process of neural network.

In the layer-by-layer pruning process, the square error generated by the fore layer is added to the current layer. If the linear regression method is not used for parameter reconstruction, as shown by the orange curve in the above figure, the square error will appear an increasing trend as the number of layer increases. But with the introduction of the linear regression method, although the square error of the deep layers of the network is still larger than that of the shallow layers, the overall square error tends to be stable. In the

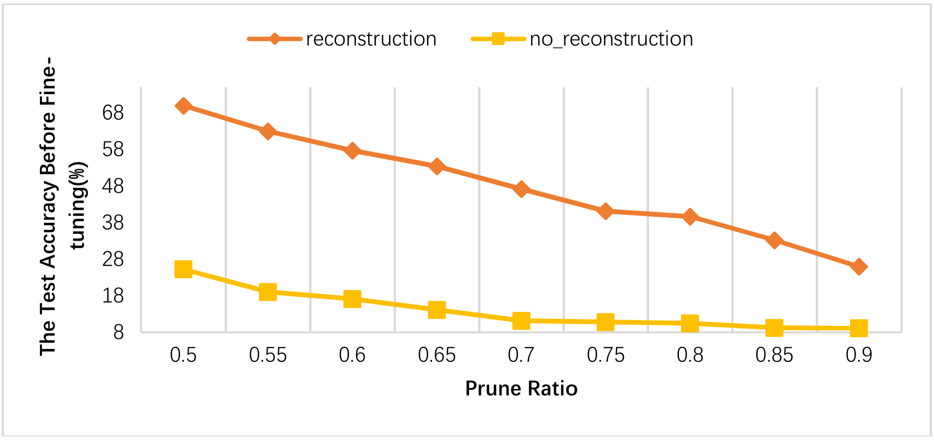


FIGURE 12. Changes in test accuracy before fine-tuning training.

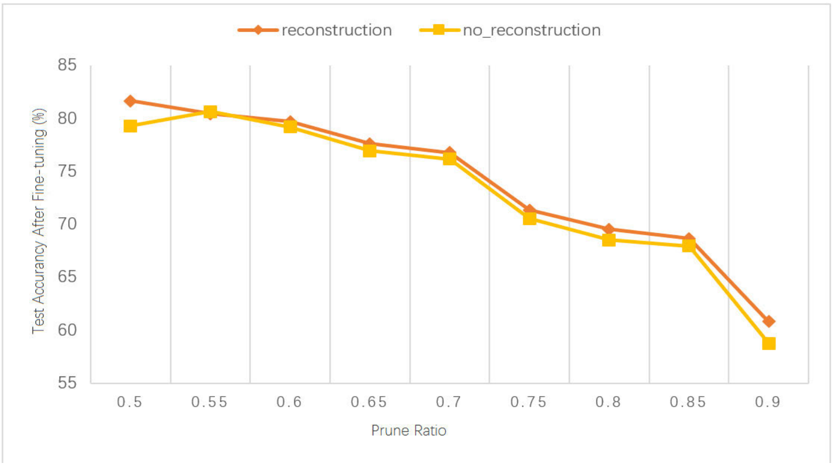


FIGURE 13. Changes in test accuracy after fine-tuning.

8th and 10th deep separable convolution units, the squared error is reduced because the pointwise convolution has a higher dimension in these convolutional units and has a better ability to fit the output. Therefore, it can be seen that in these convolutional units, the parameter reconstruction method can achieve better results with a lower square error. The most direct effect of the squared error reduction for each layer is the test accuracy of the model before fine-tuning training, as shown in Figure 12.

After channel pruning, the accuracy of unreconstructed MobileNet dropped to around 10% when the prune ratio was greater than 0.7, and the model performance was severely lost. However, the MobileNet reconstructed with linear regression algorithm could still maintain a proper ability to accurately classify after pruning, and the model could still retain a more than 50% accuracy when the prune ratio is not bigger than 0.7. In order to observe the influence of the parameter reconstruction method on the test accuracy of the final model, fine-tuning the reconstructed model to obtain the test accuracy is as Figure 13.

It can be seen that the parameter reconstruction method greatly improves the accuracy of the network fine-tuning training. However, with the progress of fine-tuning training, the gap between the two settings is gradually reduced. When the prune ratio is not so big, the method of parameter reconstruction is less obvious for improvement of the final test accuracy of the network. The reason for the conjecture may be that the network was also trained with the gradient descent method during the network fine-tuning training process, and the functions of this two progress were repeated. However, the reconstruction method is not without merit. In other more complicated tasks, linear regression method reconstruction of the model middle layer parameters may also be a feasible method.

B. LARGE-SCALE CLASSIFICATION ON ILSVRC12

1) DATASET

The ImageNet [2] project is a large visualization database for visual object recognition software studies. More than 14 million image URLs are manually annotated by ImageNet in order to indicate objects in the image, and at least one million images provide a bounding box. ImageNet contains more than 20,000 categories; a typical category, such as "balloon" or "strawberry", contains hundreds of images. It is by far

TABLE 4. Variation of FLOPs in different prune ratio.

Prune_ratio	0	0.1	0.2	0.3	0.4	0.5
FLOPs	306575360	219796121	173276061	132893542	97724761	68277101
FLOPs(%)	100	71.69	56.52	43.35	31.88	22.27

the most authoritative dataset of image processing task, often used as a benchmark dataset in experiments.

Since 2010, in the annual ImageNet Large Scale Visual Recognition Competition (ILSVRC), research teams have to evaluate their algorithms on a given dataset and competed for higher accuracy in several visual recognition tasks. ILSVRC is designed to "track the PASCAL Visual Object Classes (PASCAL VOC) established in 2005", which includes only about 20,000 images and 20 target categories. ILSVRC only uses 1000 image categories or a trim list with of it, and 90 out of 120 varieties are classified by the full ImageNet architecture. ILSVRC-12 is the dataset used in the 2012 ILSVRC competition, which training set contains 1.2 million images and the validation set contains 150,000 images belonging to 1000 categories.

Due to the large number of categories, the indicators for evaluating ILSVRC image classification are Top-1 accuracy and Top-5 accuracy. Generally, Top-5 accuracy is used as the evaluation standard.

2) IMPLEMENTATION DETAILS

Due to the complexity of the image classification task of the ILSVRC-12 dataset, the larger prune ratio has a greater impact on the performance of the model. Therefore, we select the prune ratio of 0.1, 0.2, 0.3, 0.4, and 0.5 to pruning the network. Similarly, the workflow to prune the network is shown in Figure 7.

A complete training of the ILSVRC-12 dataset requires 100 repetitions of the 1.2 million images in the training set. Assuming a training batch size of 64, the entire training process requires 2 million iterations. And it will take about 200 hours to train once on the NVIDIA GTX 1070Ti. Due to hardware limitations, this group of experiments used the open source, already trained MobileNet as the initial network. The pre-training model achieved 69.1% Top-1 accuracy and 88.2% TOP-5 test accuracy on ILSVRC12. Based on the pre-training model, only pruning and parameter reconstruction are performed, and subsequent fine-tuning training is not performed. The effects of three channel selection methods (APoZ, information entropy and information gain) are compared by comparing the test accuracy of the reconstructed network.

3) RESULTS AND ANALYSIS

The variation of the model FLOPs with the increase of the prune ratio during the experiment is shown in Table 4:

As shown in Figures 14 and 15, under the Top-1 and Top-5 evaluation criteria, the information gain-based pruning method achieves the best results at various prune ratios, especially when the prune ratio is relatively large. When the prune ratio is small than 0.3, although the model is not fine-tuned and retrained, the information gain-based pruning model can still achieve more than 60% of Top-1 accuracy and more than 80% of Top-5 accuracy. As shown in Table 4, the FLOPs of the model are only half of the initial model when the prune ratio is equal to 0.3. In other words, only through model pruning and linear regression reconstruction, one can compress MobileNet more than double with a model loss of less than 5%, which requires only 20 minutes for a typical computer. However, when the prune ratio exceeds 0.3, the test accuracy of the model after pruning begins to decrease drastically, and the network after pruning needs to be fine-tuned for a long time to restore the model performance.

C. DETECTION ON PASCAL VOC

1) DATASET

The PASCAL Visual Object Classes is a world-class computer vision competition that began in 2005. The content is different every year, from the initial classification to the subsequent image detection, image segmentation, human body layout, motion recognition, etc. The capacity and variety of datasets are also constantly increasing and improving. In this section, we mainly use this dataset for image detection. The dataset contains 20 categories of targets, namely airplanes, bicycles, birds, boats, cats, dogs, etc., with a total of 9,963 high-quality images, of which 22,136 are training set and 4,952 are test set. And for image detection tasks, the dataset will even provide the image with the category and location information of the identified object.

2) IMPLEMENTATION DETAILS

For this group of experiments, we attempt to use MobileNet as the basic network to achieve target detection based on the SSD. The network structure diagram is shown in Figure 16.

Firstly, the size of input image is 300 × 300, removing the last classification layer of MobileNet, and only left the previous convolution units as the feature layers. In order to enrich the extracted features, the additional eight different scale convolution layers were added after MobileNet. The output feature maps of the eleventh and thirteenth depth-wise separable convolution units conv14_2, conv15_2, conv16_2, conv17_2 are taken as input features for subsequent target detection. In this section, a total of three sets of target detection experiments are set up, corresponding to MobileNet networks with prune ratios of 0, 0.3 and 0.5 respectively.

To speed up model training, the first set of experiments used MobileNet that has been trained on the ILSVRC-12 dataset as the pre-training model, while the second and third

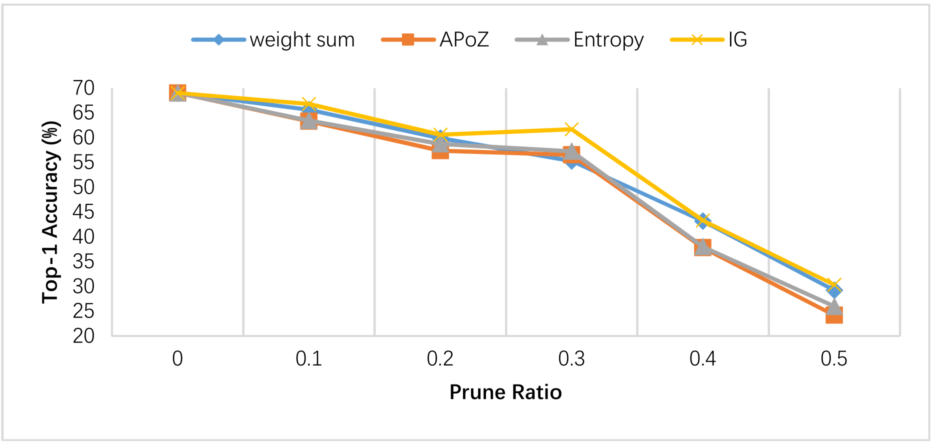


FIGURE 14. Top-1 Accuracy of Models with Different Prune ratios.

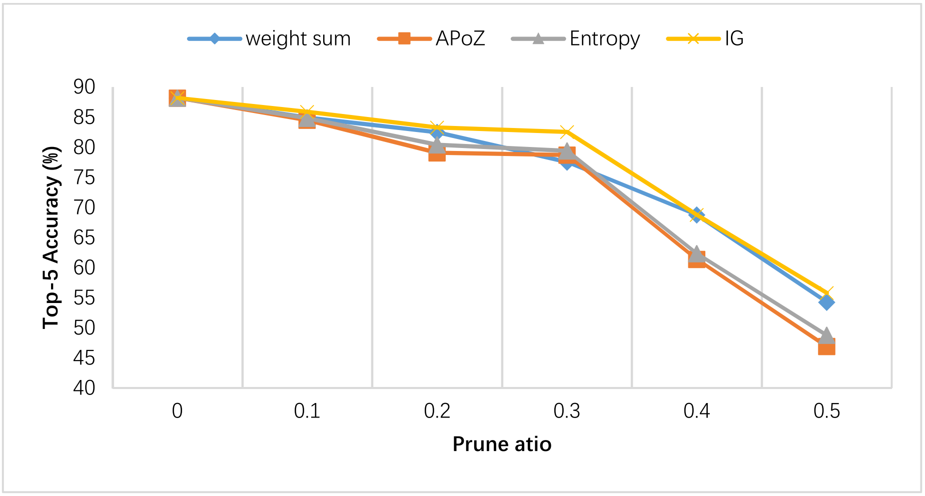


FIGURE 15. Top-5 Accuracy of Models with Different Prune ratios.

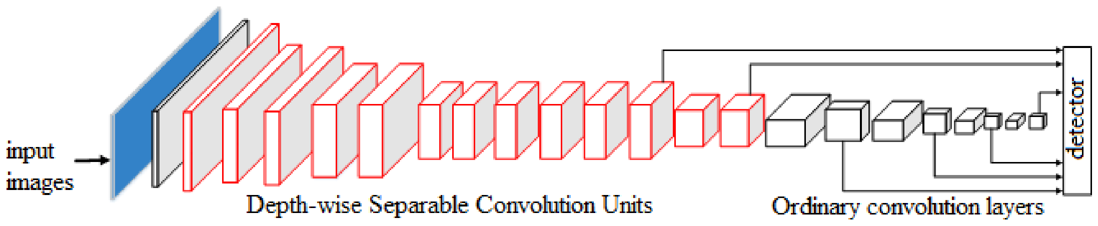

input
images
Depth-wise Separable Convolution Units
Ordinary convolution layers
detector

FIGURE 16. MobileNet-SSD structure.

sets of experiments uses the information gain pruning criteria for channel pruning, and the model parameters after reconstruction will be used as pre-training models in the training process. We still use a NVIDIA GTX 1070Ti to train on the TensorFlow deep learning framework. And the loss function is weighted by the target location loss function and the target recognition loss function. We used the SGD optimization algorithm with the momentum parameter set to 0.9 to train the network, and got a total of 120,000 iterations. As a result, the learning rate drops to 0.0001 and 0.00001 respectively when the network iterates to 80k and 100k with the initial learning rate setting to 0.001.

3) RESULTS AND ANALYSIS

After the pruning, the changes of FLOPs and mAP of the model are shown in Table 5:

The experimental results show that the channel pruning method based on the information gain criterion preserves the

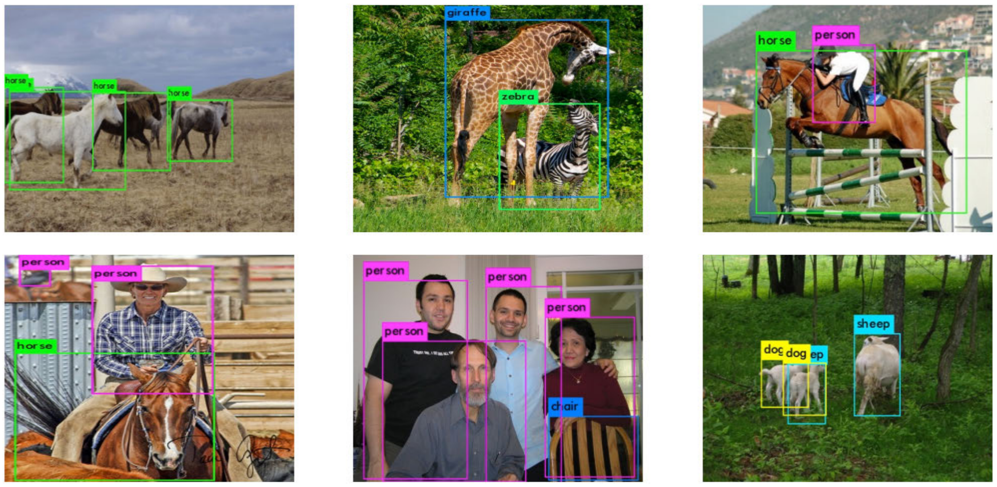


FIGURE 17. Results of image detection.

TABLE 5. Changes of FLOPs and map after pruning.

	FLOPs	mAP(%)
Original model	616575360	72.7
Pruned(0.3)	294857653	72.5
Pruned(0.5)	94378269	71.3

important information in the model as much as possible. The pruned neural network not only maintains the original image classification accuracy, but also has an obvious effect in the more complex image detection tasks. Besides, the detection model keeps small loss of performance when the computational cost is greatly reduced. Some image detection results are shown in Figure 17. It is foreseeable that the channel pruning method can be widely used in other computer vision related tasks such as image segmentation, motion recognition, and image rendering, which is also our future research direction.

V. CONCLUSION AND FUTURE WORK

This paper selects the MobileNet, a lightweight network model widely used in the industry, as the basic network for network compression. A new pruning process with a new information gain-based channel selection method and a new parameter reconstruction method are designed according to the special structure of the depth separable convolution unit. Then, we carried out experimental verifications on CIFAR10 and ILSVRC12 image classification datasets. The proposed channel selection method retains more important information in the model than the channel selection method of APoZ and information entropy, and has a higher accuracy. In addition, when we use the pruned MobileNet as the basic model of SSD, we also get good results on the PASCAL VOC image detection dataset.

The existing neural network compression methods mainly compress the redundancy in the neural network. It can be seen from the experiments that when the prune ratio is small, the performance of the neural network does not decrease too much with retraining. But when the prune ratio is bigger than a certain threshold, the performance of the model begins to decrease greatly, and the accuracy also significantly decreases as the prune ratio increases. Therefore, we can envisage whether there is a neural network of the right size for a particular problem, which can solve this problem with almost no redundancy in the network. It needs us to have a more precise definition of this problem, and we need some means to calculate how many parameters and network layers this problem required at least to be resolved. This is a difficult but feasible direction, which is how to automatically learn the most appropriate architecture through the neural network for the complexity of different tasks without artificial design. Only in this way we can get closer to the real "artificial intelligence".

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* Boston, MA, USA: MIT Press, 2012, pp. 1097–1105.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," presented at the CVPR, Jun. 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," presented at the CVPR, Jun. 2016. [Online]. Available: https://arxiv.org/abs/1512.03385

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: https://arxiv.org/abs/1704.04861

[5] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," presented at the CVPR, Oct. 2017. [Online]. Available: https://arxiv.org/abs/1707.06342

[6] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* Boston, MA, USA: MIT Press, 2015, pp. 1135–1143.

[7] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.* Boston, MA, USA: MIT Press, 2016, pp. 2074–2082.

[8] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," presented at the ICCV, Oct. 2017. [Online]. Available: https://arxiv.org/abs/1708.06519

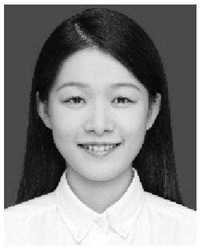
[9] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," 2014, *arXiv:1412.6115*. [Online]. Available: https://arxiv.org/abs/1412.6115
[10] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," presented at the ICML, 2015. [Online]. Available: https://arxiv.org/abs/1504.04788?context=cs.LG
[11] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," presented at the Proc. Deep Learn. Unsupervised Feature Learn. NIPS Workshop, 2011, vol. 1. [Online]. Available:https://static.googleusercontent.com/media/research.google.com/zh-CN//pubs/archive/37631.pdf
[12] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," presented at the ICML, 2015. [Online]. Available: https://arxiv.org/abs/1502.02551
[13] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing convolutional neural networks," presented at the KDD, 2016. [Online]. Available: https://www.kdd.org/kdd2016/papers/files/rpp0534-chenA.pdf
[14] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830
[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," presented at the ECCV, 2016. [Online]. Available: https://arxiv.org/abs/1603.05279
[16] C. Tai, T. Xiao, Y. Zhang, X. Wang, and E. Weinan, "Convolutional neural networks with low-rank regularization," presented at the ICLR, 2016. [Online]. Available: https://arxiv.org/abs/1511.06067v1
[17] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Adv. Neural Inf. Process. Syst.* Boston, MA, USA: MIT Press, 2014, pp. 1269–1277.
[18] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil, "Model compression," presented at the ACM SIGKDD, 2006.
[19] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proc. Adv. Neural Inf. Process. Syst.* Boston, MA, USA: MIT Press, 2014, pp. 2654–2662.
[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," presented at the CVPR, 2015. [Online]. Available: https://arxiv.org/abs/1409.4842
[21] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," presented at the CVPR, Jun. 2018. [Online]. Available: https://arxiv.org/abs/1707.0108
[22] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," presented at the CVPR, 2017. [Online]. Available: https://arxiv.org/abs/1610.02357
[23] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," presented at the ICLR, 2017. [Online]. Available: https://arxiv.org/abs/1608.08710
[24] J.-H. Luo and J. Wu, "An entropy-based pruning method for CNN compression," 2017, *arXiv:1706.05791*. [Online]. Available: https://arxiv.org/abs/1706.05791
[25] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," presented at the CVPR, 2017. [Online]. Available: https://arxiv.org/abs/1611.05128
[26] Y. Hu, S. Sun, J. Li, X. Wang, and Q. Gu, "A novel channel pruning method for deep neural network compression," 2018, *arXiv:1805.11394*. [Online]. Available: https://arxiv.org/abs/1805.11394
[27] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," presented at the ICLR, 2017. [Online]. Available: https://arxiv.org/abs/1611.06440
[28] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Burlington, VT, USA: MKO, 2011.
[29] R. Hechtnielsen, "Theory of the backpropagation neural network," in *Neural Networks for Perception*, vol. 2. San Diego, CA, USA: Harcourt, 1989, pp. 65–93.
[30] M. Abadi *et al.*, "TensorFlow: A system for large-scale machine learning," presented at the OSDI, 2016. [Online]. Available: https://arxiv.org/abs/1605.08695
[31] A. Krizhevsky, V. Nair, and G. Hinton. *The CIFAR-10 Dataset*. Accessed: 2009. [Online]. Available: http://www.cs.toronto.edu/kriz/cifar.html
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," presented at the 3rd Int. Conf. Learn. Represent., 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

KE ZHANG received the B.S. and M.S. degrees from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2002 and 2006, respectively, and the Ph.D. degree from the School of Computer Science and Engineering (SCSE), UESTC, in 2010. He is currently an Associate Professor with the SCSE, UESTC. His research interests include the Internet of Things, data fusion, and big data. He is also a Senior Member of the China Computer Federation (CCF).

KEN CHENG received the B.E. degree in network engineering from the University of Electronic Science and Technology of China (UESTC), in 2016, where he is currently pursuing the M.A.Eng. degree.

His research interests include the Internet of Things, bigdata, and neural networks.

JINGJING LI received the M.Sc. and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China, in 2013 and 2017, respectively. He is currently an Associate Professor and a National Postdoctoral Program for Innovative Talents Research Fellow with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He has great interest in machine learning, especially transfer learning, subspace learning, and recommender systems.



YUANYUAN PENG received the B.E. degree in Internet of Things engineering from Southwest Petroleum University, in 2019. She is currently pursuing the M.A.Eng. degree with the University of Electronic Science and Technology of China (UESTC).

Her research interests include blockchain, the IoE, and compression and acceleration in neural networks.