IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 30, 2020, accepted January 10, 2021, date of publication January 13, 2021,
date of current version January 20, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3051504

Progressive Kernel Pruning Based on the Information Mapping Sparse Index for CNN Compression

JIHONG ZHU, YANG ZHAO, AND JIHONG PEI, (Member, IEEE)
College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China

Corresponding author: Jihong Pei (jhpei@szu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 62071303 and Grant 61871269, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2019A1515011861, and in part by the Shenzhen Science and Technology Projection under Grant JCYJ20190808151615540.

ABSTRACT Network pruning can effectively reduce a model's capacity and computational load, thereby making model deployment in mobile devices less difficult than that without pruning. To improve the pruning rate of the model while maintaining the kernel's feature extraction ability, this paper designs a progressive kernel pruning method for CNN model compression based on the proposed information mapping sparsity index. This method first prunes the kernels in the filter and then prunes the kernels in the convolution layer when the model reaches a certain compression ratio. The whole process is called progressive kernel pruning (PKP). For the kernel pruning process, this paper defines the information mapping sparse index (IMSI), which is used to measure the mapping ability of the convolution kernel related to the amount of information transferred by the convolution operation. When pruning the kernels of filters and layers, according to the IMSI, the kernels with the strongest mapping abilities are retained to transfer as much information as possible with the least number of kernels. Progressive kernel pruning can make use of the characteristic that the model is easy to optimize when kernel pruning in the filter, and it avoids having the model easily fall into local optima when kernel pruning in the layer directly. The experimental results on the CIFAR-10/100 and ImageNet datasets show that compared to the existing CNN model compression methods, the IMSI-based progressive kernel pruning method exhibits better compression performance in processing the model compression tasks that are currently popular. In particular, pruning VGG-16 on CIFAR-10 with our model achieves a compression ratio of 80.8x and an acceleration ratio of 14.8x, which are 5.8x and 4.2x higher than the best results at present, respectively, and the classification accuracy decreases by only 0.59% relative to that of the baseline.

INDEX TERMS Progressive kernel pruning, information mapping sparse index, convolutional neural network, compression.

I. INTRODUCTION

In recent years, deep learning has achieved great success in computer vision tasks, such as image recognition [1], [2], target detection [3], [4], semantic segmentation [5], [6], and target tracking [7]. A deep model usually has a substantially deep (and sometimes wide) structure with a large number of parameters, and this makes it difficult to deploy such a model on a mobile terminal.

Model compression and acceleration are conducive to the application of deep learning methods in various fields. Increasing the compression ratio of the model, which is achieved through network pruning, can easily solve the problem of deploying deep learning technology on a mobile terminal. Network pruning assumes that the convolutional network has many redundant parameters, and these parameters can be discarded by certain methods. After model pruning, the parameters of the model have high sparseness, and this reduces the model storage space requirements and computational complexity. The model structure consists of weights, convolution kernels, filters, and convolution layers. Generally, network pruning is divided into weight pruning, convolution kernel pruning, and filter pruning. Among them, weight pruning is a type of unstructured pruning, and

The associate editor coordinating the review of this manuscript and approving it for publication was Easter Selvan Suviseshamuthu.

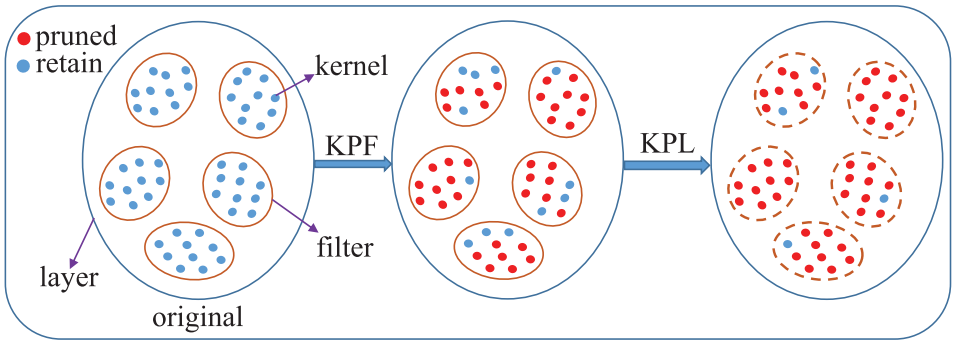


FIGURE 1. Illustration of the progressive kernel pruning. Kernel pruning is performed on the original model for each filter and then in the layers. From the original model to KPF and then to KPL, we call it progressive kernel pruning.

convolution kernel pruning and filter pruning are structured pruning methods.

The purpose of weight pruning is mainly to remove unimportant connections among the parameters with relatively small amplitudes [8]. The sparse models obtained from this pruning method are all irregular convolution kernels, which require special software/hardware for accelerated inference [9]. Structured pruning, including convolution kernel pruning [10]–[12], filter/channel pruning [13]–[22], and block pruning [23], removes structural weights; a pruned model that can achieve accelerated inference does not require specific software/hardware and is thus easy to deploy. In this paper, we mainly discuss convolution kernel pruning.

The convolution kernel is the smallest structural unit for feature extraction in a convolutional network. In structured pruning, the convolution kernel has the smallest pruning granularity, and these can be combined into a coarse granular structure, such as a filter or layer. The basis of coarse granularity pruning stems from the relationships between these granularities, for example, determining the importance of filters according to the filter norm [13], [14], [24], determining which filters need to be pruned by minimizing the reconstruction error of the feature output [16], [25], [26], and using sparse regularization training [8], [17], [20], [21], [27], [28].These coarse granularity pruning methods take the relationships between filters in the entire convolutional layer into account. In fact, even if the filter is considered to be redundant, some convolution kernels in the filter may still have an important role. This coarse granularity pruning method generally causes a drop in the classification accuracy of the model.

Kernel pruning is based on the relationships between the kernels in the filter/layer; granularity pruning is done using the metric method [12], the regularization method is used for sparse pruning [10], and the kernel sparsity and entropy index guide model compression [11]. The kernel is the smallest structural unit of feature extraction, and they can be combined into filters and layers. Thus, some retained kernels may have similar functions and are unable to achieve the optimal pruning effect.

Regarding the local relationships between the convolution kernels in the filter and the overall relationship between all convolution kernels in the convolution layer, this paper proposes a progressive kernel pruning (PKP) method from the local kernel pruning of each filter to the global kernel pruning of the convolution layer. The information mapping sparse index (IMSI) of the filter and the layer is defined according to the maximum mapping ability of the kernel. First, kernel pruning in the filter (KPF) is performed according to the IMSI, and then kernel pruning in the layer (KPL) is performed when the predetermined compression ratio is reached, as shown in Figure 1. The progressive pruning approach based on the IMSI proposed in this paper makes the retained kernel transmit as much effective information as possible. At the same time, the idea of progressive pruning, which considers the local relationships and global relationships between convolution kernels, can improve the convergence speed of pruning training and the pruning rate of the model while maintaining the performance of the model.

The innovations and contributions of this paper include the following: (1) A progressive kernel pruning method based on the IMSI is proposed. The method first prunes the convolution kernels with local relations by using the IMSI between the convolution kernels in each filter, and then it prunes the convolution kernels with global relations by using the IMSI between the convolution kernels in the whole convolution layer. Progressive kernel pruning exhibits the characteristics of better convergence and easier optimization than other methods when performing KPF, and it avoids the problem of easily falling into local optima when performing KPL directly. (2) In KPF and KPL, a new IMSI is defined to measure the relationships between convolution kernels. A certain number of retained kernels, selected according to the IMSI, can enable the maximum amount of effective information to pass through the filter. (3) The pruning process is easy to implement while achieving a high model compression ratio on the basis of maintaining network performance. In particular, pruning VGG-16 on CIFAR-10 achieves a compression ratio of 80.8x and an acceleration ratio of 14.8x, and the accuracy decreases by only 0.59%; these are the best results

known at present. When pruning from scratch, the performance is almost the same as that of the pretrained mode (pruning VGG-16 on CIFAR-10).

II. RELATED WORKS

Network pruning is an effective method for reducing model redundancy. Some of the pioneering work with regard to pruning includes [29] and [30]. With the development of deep learning, network pruning is mainly divided into unstructured pruning and structured pruning. Weight pruning [8], [31]–[35] is the major form of unstructured pruning, while structured pruning mainly includes kernel pruning [10]–[12] and filter/channel/layer/block pruning [13]–[25], [27], [28], [36]–[43].

A. WEIGHT PRUNING

Han *et al*. [8] maintained their original model accuracy after discarding the parameters with the smallest absolute values. In particular, their subsequent work [31] integrated weight pruning, quantification, and a Huffman code into the models, and this further reduced their capacity. Karen Ullrich *et al*. [32] introduced the method of soft weight sharing, which can achieve quantification and pruning together during the retraining process based on [31]. Xiao *et al*. [33] optimized a set of trainable auxiliary parameters to replace the original weights; the pruning process was highly robust to noise and insensitive to hyperparameters, and it achieved a high compression ratio. Wang *et al*. [34] and Liu *et al*. [35] both transferred weight to the DCT domain for pruning purposes and obtained excellent results.

B. KERNEL PRUNING

Lin *et al*. [10] applied sparse regularization to automatically identify useless kernel connections and pruned connections with small synaptic strength in the convolution layer to generate a compact model with kernel-level sparsity. When regularizing the synapse strength, only the sparseness of the kernel in the convolutional layer is considered. In addition, it is difficult to optimize the addition of regular terms. Li *et al*. [11] proposed kernel sparsity and entropy (KSE) indicators to guide model compression and performed kernel clustering based on these KSE indicators to achieve high-precision compression. However, compression can only be performed within the group, and the kernels generated after kernel clustering may need additional fine tuning. Mao *et al*. [12] explored pruning at different granularity levels, such as weight pruning, vector pruning, kernel pruning, and filter pruning.

C. FILTER PRUNING

Li *et al*. [13] and He *et al*. [24] proposed hard filter pruning and soft filter pruning methods, respectively. Lin *et al*. [19] proposed a global and dynamic filter pruning method. Liu *et al*. [15] added a channel-wise scaling factor to the batch normalization layer and then added L_1 regularization to make it sparse. Luo *et al*. [16] determined which channels needed to be pruned by minimizing the feature reconstruction errors. Zuo *et al*. [18] proposed a filter pruning method without damaging network capacity. Wen *et al*. [17] used the group lasso method for sparse regularization training (SSL) and explored the structural sparsity of different levels in channels, filters, filter shapes, and depths. Huang *et al*. [20] used the APG algorithm to make a mask sparse and achieved structured pruning by introducing a learnable mask. Lin *et al*. [21] proposed a structured sparse regularization method that contains two different regularization terms; this method can fully coordinate the global output and the local pruning operations to adaptively prune the filter. Zhu *et al*. [37] added decorrelation regularization to SSL [17], further reducing the correlation between filters. Wang *et al*. [23] proposed online ensemble distillation for pruning blocks/layers. Liu *et al*. [38] studied acceleration and pruning from the viewpoint of data differences and proposed instance feature sparse regularization, making the feature maps sparse. Molchanov *et al*. [39] proposed a sparse variational dropout method that expands the original variational dropout to adjust the dropout rate and tailor a model with a sparse solution. Lin *et al*. [41] proposed an optimal, structured network pruning method based on generative adversarial learning that uses unsupervised end-to-end training to prune redundant heterogeneous structures in the network. Ding *et al*. [22] utilized a long short-term memory (LSTM) to learn the hierarchical characteristics of a network and generate a global network pruning scheme. Ding *et al*. [42] proposed Centripetal SGD (C-SGD) optimization method, which can train several filters to collapse into a single point in the parameter hyperspace. Ding *et al*. [43] proposed Approximated Oracle Filter Pruning (AOFP), which continues to search for the least important filters in a binary search manner.

The kernel is the smallest structural unit for extracting features in a convolutional network. Direct kernel pruning avoids accidental pruning of the important role kernel during coarse-grained pruning. This paper proposes to directly measure the feature extraction ability of kernels and obtain different information mapping sparsity indexes according to the relationships between the kernels in different spaces, which can be used to guide the selection of appropriate numbers of retained kernels in the different spaces. Kernel pruning in a small space and kernel pruning in a large space are beneficial to pruning training.

III. THE METHOD OF PROGRESSIVE KERNEL PRUNING

This section introduces the proposed IMSI for filters and layers. We first analyze the principle of convolution kernel mapping to achieve the IMSI. The IMSIs of filters and layers are used as the basis for kernel pruning. Then, we propose the progressive pruning method for kernel pruning in the filters and in the layers.

A. INFORMATION MAPPING SPARSE INDEX OF A CNN

Convolution mapping in the convolution layer is the most basic operation for extracting features from a CNN.

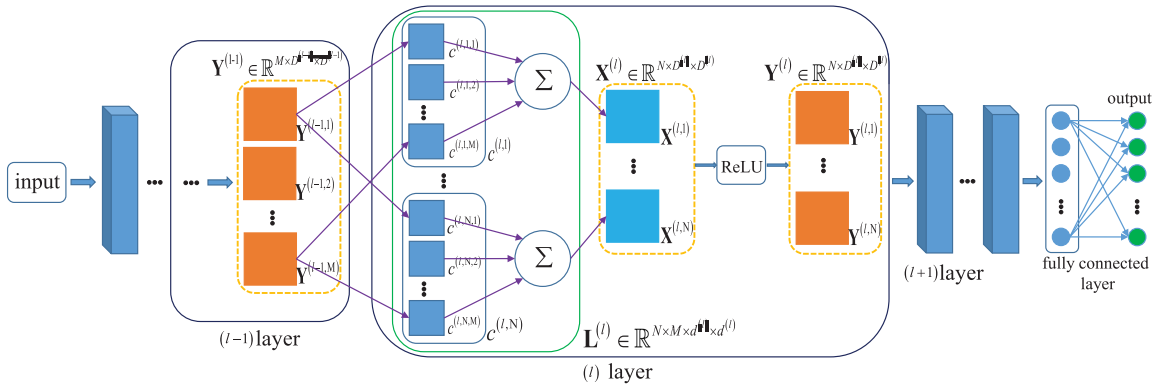


FIGURE 2. The simplified structure schematic of the CNN, which shows the mapping process of l-th layer in detail, where the input features $Y^{(l-1)}$ are the output of $(l-1)$-th layer, there are a total of M channels; The output characteristic in which convolution result $X^{(l)}$ passing through nonlinear mapping is $Y^{(l)}$, and there are N channels in total.

Therefore, the key to maintaining the performance of the pruned network is to retain a convolution kernel with strong feature extraction ability. This section analyzes the mapping of convolution layers in theory and proposes a measurement index that reflects the ability of convolution mapping. As shown in Figure 2, $\mathbf{Y}^{(l-1)} = [\mathbf{Y}^{(l-1,1)}, \cdots \mathbf{Y}^{(l-1,m)}, \cdots \mathbf{Y}^{(l-1,M)}] \in \mathbb{R}^{M\times D^{(l-1)}\times D^{(l-1)}}$ is the output of the $(l-1)$-th layer, which is the input for the l-th layer. In the l-th layer, the output is $\mathbf{Y}^{(l)} = [\mathbf{Y}^{(l,1)}, \cdots, \mathbf{Y}^{(l,N)}]$ containing N channels, which are computed by the activation function:

$$\mathbf{Y}^{(l,n)} = ReLU(\mathbf{X}^{(l,n)}), \quad n = 1, 2, \ldots, N \tag{1}$$

where $\mathbf{X}^{(l,n)} = \{\mathbf{X}^{(l,n,1)}, \cdots, \mathbf{X}^{(l,n,M)}\}$. $\mathbf{X}^{(l,n,m)}$ is got by the convolution operation

$$\begin{aligned}\mathbf{X}^{(l,n,m)} &= \mathbf{Y}^{(l-1,m)} * c^{(l,n,m)} \\ &= k^{(l,n,m)}\left(\mathbf{Y}^{(l-1,m)}\right) \\ &= \left(\mathbf{Y}^{(l-1,m)}, c^{(l,n,m)}\right)\end{aligned} \tag{2}$$

where $c^{(l,n,m)}$ is the mth convolution kernel of the nth filter $c^{(l,n)} = \{c^{(l,n,1)}, \cdots, c^{(l,n,M)}\}$ in the l-th layer. $\mathbf{X}^{(l,n,m)}$ is computed by a linear map determined by the convolution kernel $c^{(l,n,m)}$. By Riesz representation theorem [44], there is a bounded linear function $k^{(l,n,m)}(\cdot)$. The norm $\|k^{(l,n,m)}\|$ of the bounded linear function $k^{(l,n,m)}(\cdot)$ satisfies:

$$\|k^{(l,n,m)}\| = \|c^{(l,n,m)}\| \tag{3}$$

The input signal $\mathbf{Y}^{(l-1,m)}$ is mapped by the linear function $k^{(l,n,m)}(\cdot)$ to the output signal $\mathbf{X}^{(l,n,m)}$. The operator norm $\|k^{(l,n,m)}\|$ can be regarded as a metric of the mapping $\mathbf{Y}^{(l-1,m)} \to \mathbf{X}^{(l,n,m)}$. This metric reflects the capability of the mapping that the information in the feature map $\mathbf{Y}^{(l-1,m)}$ is mapped to the feature map $\mathbf{X}^{(l,n,m)}$ through the convolution kernel $c^{(l,n,m)}$. The output $\mathbf{Y}^{(l,n)}$ of the nth channel in layer l is the sum of M bounded linear functions.

$$\begin{aligned}\mathbf{X}^{(l,n)} &= \sum_{m=1}^{M} \mathbf{Y}^{(l,n,m)} \\ &= k^{(l,n,1)}(\mathbf{Y}^{(l-1,1)}) + \cdots + k^{(l,n,M)}(\mathbf{Y}^{(l-1,M)})\end{aligned} \tag{4}$$

From the nature of the bounded linear operator's norm:

$$\|\sum_{m=1}^{M} k^{(l,n,m)}\| \le \sum_{m=1}^{M} \|k^{(l,n,m)}\| = \sum_{m=1}^{M} \|c^{(l,n,m)}\| \tag{5}$$

Let

$$K^{(l,n)} = \sum_{m=1}^{M} \|k^{(l,n,m)}\| = \sum_{m=1}^{M} \|c^{(l,n,m)}\| \tag{6}$$

Then $K^{(l,n)}$ can be seen as a metric of mapping ability for the nth filter of the l-th layer. Since $\mathbf{Y}^{(l-1,1)}, \cdots \mathbf{Y}^{(l-1,m)}, \cdots \mathbf{Y}^{(l-1,M)}$ and $\mathbf{X}^{(l,n)}$ are feature maps with limited size, which have limited information capacity. $X^{(l,n)}$ can be understood as the superimposed result of M channels. Due to the limitation of the information capacity of the input and output channels, if as few convolution kernels as possible approach the upper limit of the information mapping ability, the structure of the CNN can be effectively simplified. For the filter $c^{(l,n)} = [c^{(l,n,1)}, \cdots c^{(l,n,m)}, \cdots c^{(l,n,M)}]$, the upper mapping ability of the kernel in this filter is

$$\|c^{(l,n,*)}\| = \max\{\|c^{(l,n,1)}\|, \cdots \|c^{(l,n,m)}\|, \cdots \|c^{(l,n,M)\|}\} \tag{7}$$

The information mapping sparse index (IMSI) $q^{(l,n)}$ of the nth filter $c^{(l,n)}$ is defined as

$$\begin{aligned}q^{(l,n)} &= \left\lfloor \frac{K^{(l,n)}}{\|k^{(l,n,*)}\|} \right\rfloor \\ &= \left\lfloor \frac{\|c^{(l,n,1)}\| + \cdots + \|c^{(l,n,m)}\| + \cdots + \|c^{(l,n,\mathrm{M})}\|}{\|c^{(l,n,*)}\|} \right\rfloor\end{aligned} \tag{8}$$

where $\lfloor\cdot\rfloor$ is a downward rounding function. $q^{(l,n)}$ reflects the number of convolution kernels needed for mapping by using the kernel with the upper mapping ability $\|c^{(l,n,*)}\|$ that the current convolution kernels can achieve. Therefore, IMSI $q^{(l,n)}$ of the filter $c^{(l,n)}$ can be used as the basis for kernel pruning in the filter $c^{(l,n)}$.

Considering the mapping ability of the convolution kernel in the whole convolutional layer, for the overall output

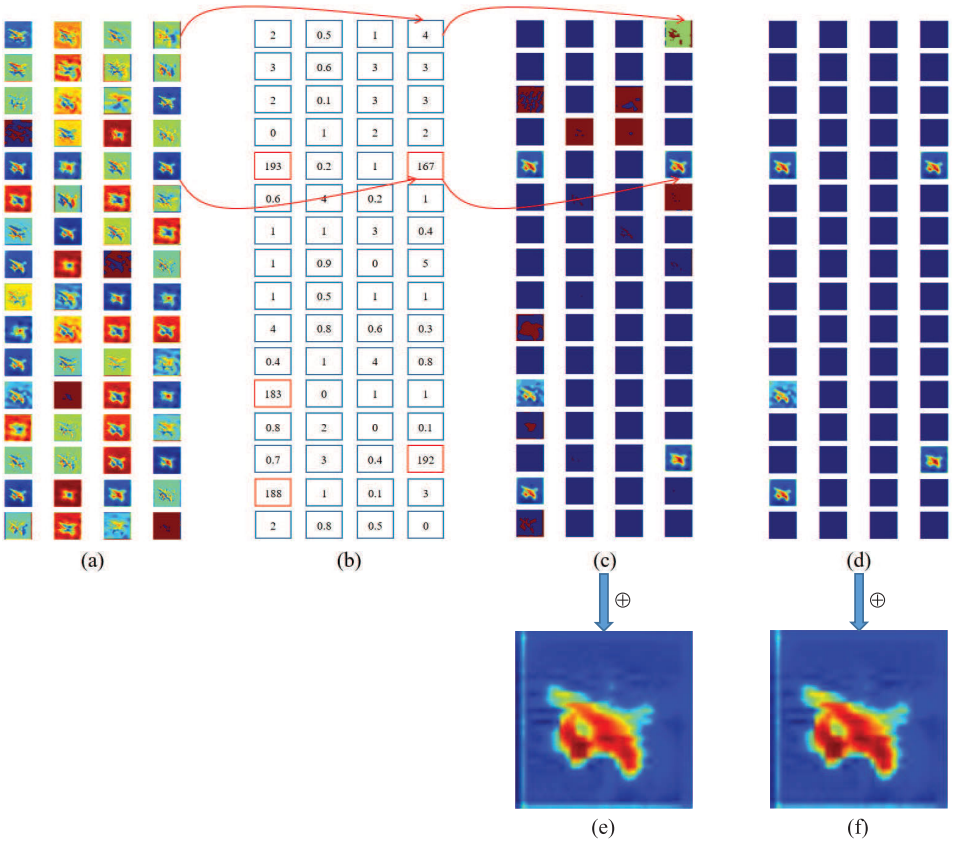


FIGURE 3. Schematic representation of the information mapping sparse index. (a) Input features; (b) Kernel norm ($\times 10^{-4}$), the red box are the retained kernel norm; (c) Output features corresponding to the convolution kernel (before pruning); (d) Output features corresponding to the convolution kernel (after pruning), the number of retained kernels is $q^{(l,n)} = 5$, the number of output effective features is 5, other features are no information; (e) The output feature of the filter before pruning, which is the accumulation of the features in (c); (f) The output feature of the filter after pruning, which is the accumulation of the features in (d).

$\mathbf{X}^{(l)} = [\mathbf{X}^{(l,1)}, \cdots, \mathbf{X}^{(l,N)}]$ of the l-th convolution layer, there are N filters:

$$c^{(l,n)} = [c^{(l,n,1)}, \cdots c^{(l,n,m)}, \cdots c^{(l,n,M)}], \quad n = 1, 2, \cdots N \tag{9}$$

which contain $N \times M$ convolution kernels. Then, the upper mapping ability that the convolution kernel $c^{(l,n,m)}$ of the whole l-th layer can currently reach:

$$\|c^{(l,*,*)}\| = \max\{\|c^{(l,1,1)}\|, \cdots \|c^{(l,1,M)}\|, \cdots, \|c^{(l,n,m)}\|, \cdots, \|c^{(l,N,M)}\|\} \tag{10}$$

The information mapping sparse index (IMSI) $q^{(l)}$ of the l-th layer can be defined as

$$q^{(l)} = \left\lfloor \frac{\sum_{n=1}^{N} \sum_{m=1}^{M} \|c^{(l,n,m)}\|}{\|c^{(l,*,*)}\|} \right\rfloor = \left\lfloor \frac{\sum_{n=1}^{N} K^{(l,n)}}{\|c^{(l,*,*)}\|} \right\rfloor \tag{11}$$

The IMSI $q^{(l)}$ of the l-th convolution layer reflects the number of convolution kernels needed for mapping using the upper mapping ability $\|c^{(l,*,*)}\|$ that the current convolution kernels can achieve. Therefore, IMSI $q^{(l)}$ can be used as the basis for the overall convolution kernel pruning in the whole convolution layer by considering the overall relationship in layer.

In order to show the working principle of information mapping sparse index, we take the 15-th filter of the Conv-2 in VGG-16 as an example. When kernel pruning in the filter, IMSI is used to select the retained kernels, and the changes of output features before and after pruning is observed. As shown in Figure 3. (a) is the input channel feature, which is the output feature of the previous layer; (b) is the norm value of the convolution kernel ($\times 10^{-4}$), which corresponds to the input feature; (c) is the features extracted by the convolution kernel before pruning. Most of the features have no information, some are noises, and a few have clear contours; (d) is the features extracted by the convolution kernel after pruning; the number of retained kernels is $q^{(l,n)} = 5$; the output feature contours corresponding to the retained kernels are all very clear, and other features have no information; (e) and (f) are the summation of the features in (c) and (d), which represent the output features of the filter before and after pruning, respectively. We can observe that (e) and (f) are almost the same. The model compression ratio is significantly

improved and the model performance is hardly affected, when the right number of convolutional kernels with stronger mapping ability is selected according to the IMSI guidance.

B. KERNEL PRUNING BASED ON OF IMSI IN THE FILTER

This section introduces the first stage of the proposed progressive pruning method, which is kernel pruning based on the IMSI in the filter. The filter in the convolution layer consists of multiple convolution kernels. As shown in equation (4), the output signal of each channel is obtained by accumulating multiple convolution results. The accumulation of signals can be regarded as a low-pass filter. In the filter, the information output by a kernel with poor mapping ability $\|k^{(l,n,m)}\|$ is not only less important than other information but also interferes with the output information of the filter due to its average effect. Therefore, pruning a kernel with a small mapping ability is conducive to effectively retaining the information output by a kernel with a stronger mapping ability, thereby making the output characteristics of the filter more significant. Therefore, KPF is proposed according to the IMSI of the filter in equation (8). Through the IMSI $q^{(l,n)}$ of the filter, the convolution kernels with the strongest mapping abilities are retained, and the mapping abilities of the retained kernels are adjusted by iterative training so the filter can approach its maximum mapping ability with as few convolution kernels as possible. For the filter $c^{(l,n)}$, the convolution kernels are sorted according to their mapping ability.

$$\|c^{(l,n,s_1)}\| \geq \cdots \geq \|c^{(l,n,s_p)}\| \geq \cdots \geq \|c^{(l,n,s_M)}\| \tag{12}$$

Let $p = q^{(l,n)}$, the set of the first p convolution kernels with stronger mapping ability retained:

$$P^{(l,n)} = \{c^{(l,n,s_1)}, c^{(l,n,s_2)} \cdots, c^{(l,n,s_p)}\} \tag{13}$$

The remaining $M-p$ convolution kernel $c^{(l,n,s_{p+1})}, \ldots c^{(l,n,s_M)}$ in the nth filter $c^{(l,n)}$ is pruned. When the mapping abilities of all convolution kernels in the entire filter are equal:

$$\|c^{(l,n,*)}\| = \|c^{(l,n,1)}\| = \cdots = \|c^{(l,n,M)}\| \tag{14}$$

there is

$$q^{(l,n)} = \left\lfloor \frac{\left\|c^{(l,n,1)}\right\| + \cdots + \left\|c^{(l,n,m)}\right\| + \cdots + \left\|c^{(l,n,M)}\right\|}{\left\|c^{(l,n,*)}\right\|} \right\rfloor = \frac{M\|c^{(l,n,*)}\|}{\left\|c^{(l,n,*)}\right\|} = M \tag{15}$$

In this case, the actual number of kernels retained is the total number M in the filter. When the mapping ability of the convolution kernel with the strongest mapping ability is greater than the sum of the mapping abilities of the remaining convolution kernels:

$$\left\|c^{(l,n,*)}\right\| = \left\|c^{(l,n,s1)}\right\| > \frac{1}{2}\sum_{m=1}^{M}\left\|c^{(l,n,m)}\right\| \tag{16}$$

there is

$$q^{(l,n)} = \left\lfloor \frac{\left\|c^{(l,n,1)}\right\| + \cdots + \left\|c^{(l,n,m)}\right\| + \cdots + \left\|c^{(l,n,M)}\right\|}{\left\|c^{(l,n,*)}\right\|} \right\rfloor = 1 \tag{17}$$

In this case, the number of kernels actually retained in the filter is 1. The number of kernels retained in the filter is gradually reduced from M to 1, so this can be regarded as a pruning method with a gradual increase in the pruning rate, allowing the information of the data to be gradually gathered into the retained kernel during the training process.

Algorithm 1 Kernel Pruning in Filter

1. Obtain the parameter $c_i^{(l,n,m)}$ of all convolution kernels at the i-th iteration of the model;
2. Calculate the IMSI$q^{(l,n)}$ of the filter according to Eq. (8);
3. According to Eq. (12), sort the convolution kernels in the filter according to the size of the mapping abilities;
4. Keep the weights of the first $q^{(l,n)}$ convolution kernels with stronger mapping abilities in the filter unchanged, and set the weights of other convolution kernels to zero;
5. Follow steps (2-4) for each filter in each convolutional layer;
6. Next iteration;

C. KERNEL PRUNING BASED ON THE IMSI IN THE LAYER

This section introduces the second stage of the proposed progressive pruning method, the kernel pruning based IMSI in the layer. When KPF, whether the kernel is pruned is determined by the relationships between the kernels inside the filter. The metric characteristics of the information mappings between different filters in the same layer are not considered. When the highest mapping ability of the best kernel in the filter is lower than those of the kernels in other filters and the overall mapping ability of the filter is lower than those of other filters, the information of this filter's output feature map is less valid than those of other filters. When the overall mapping ability of the filter is strong and the maximum mapping ability of the kernel in the filter is weaker than those of the kernels in other filters, the mapping abilities of the kernels retained in this filter by KPF are weak. Due to the low-pass filtering effects between the kernels in the filter, the feature map output by this filter is reduced and is less important. For the process of KPF, it is difficult to prune the filters and kernels that output the least important feature maps. For each filter, the information validity of its output feature map is mainly determined by the convolution kernel with the strongest mapping ability in the filter. Even if the overall mapping degree of the filter is small, if one of the convolution kernels in the filter has a strong mapping ability, the effectiveness of the feature map output by the filter is strong. For the whole convolution layer, the relative mapping ability of the kernel has a direct impact on the importance of the corresponding feature map output by the filter. Therefore, the information mapping sparse index IMSI $q^{(l)}$ of the convolution layer is introduced to prune the convolution kernels and the filters in the convolution layer.

The structure of KPL is similar to that of KPF. The role of the filter in the whole convolution layer is considered in KPL. According to equation (11), the convolution kernels with the strongest relative mapping abilities are selected by the IMSI $q^{(l,n)}$ of the layer. The mapping abilities of the retained kernels are adjusted by iterative training so the filter can reach its maximum mapping ability with as few convolution kernels as possible. $\|c^{(l,*,*)}\|$ is the current maximum mapping ability of a kernel in the l-th layer, and the actual mapping ability of the retained kernel is also strongest in the current layer, so the information output by the retained kernel is most likely to approach the upper limit of the corresponding information output of the filter. When the maximum kernel mapping ability $\|c^{(l,n,*)}\|$ in a filter is small, the retained value is low, and the whole filter is not retained in equation (11); this is conducive to further improving the compression ratio of the model.

Arrange all convolution kernels $c^{(l,1,1)}, \cdots, c^{(l,1,M)}, \cdots, c^{(l,n,m)}, \cdots, c^{(l,N,1)}, \cdots, c^{(l,N,M)}$ in l-th layer:

$$c_1^{(l)}, \cdots c_t^{(l)}, \cdots c_{NM}^{(l)} \tag{18}$$

where $t = n \times M + m$, $c_t^{(l)} = c^{(l,n,m)}$. All kernels are sorted according to the mapping ability:

$$\|c_{s_1}^{(l)}\| \geq \cdots \geq \|c_{s_p}^{(l)}\| \geq \cdots \geq \|c_{s_{NM}}^{(l)}\| \tag{19}$$

Let $p = q^{(l)}$, the set of the first p convolution kernels with stronger mapping ability retained:

$$P^{(l)} = \{c^{(l,s_1)}, c^{(l,s_2)} \cdots, c^{(l,s_p)}\} \tag{20}$$

The remaining $N \times M - p$ convolution kernel $c_{s_{p+1}}^{(l)}, \ldots c_{s_{NM}}^{(l)}$ in the l-th layer is pruned. When the mapping abilities of all convolution kernels in the entire layer are equal:

$$\|c^{(l,*,*)}\| = \|c_{s1}^{(l)}\| = \cdots = \|c_{sNM}^{(l)}\| \tag{21}$$

there is

$$\begin{aligned} q^{(l)} &= \left\lfloor \frac{\sum_{n=1}^{N}\sum_{m=1}^{M}\left\|c^{(l,n,m)}\right\|}{\left\|c^{(l,*,*)}\right\|} \right\rfloor \\ &= \frac{N \times M \times \|c_{s1}^{(l)}\|}{\left\|c_{s1}^{(l)}\right\|} \\ &= N \times M \end{aligned} \tag{22}$$

In this case, the actual number of kernels retained is the total number $N \times M$ in the filter. When the strongest mapping ability of the convolution kernel is greater than the sum of the mapping abilities of the remaining convolution kernels:

$$\left\|c^{(l,*,*)}\right\| = \left\|c^{(l,s1)}\right\| > \frac{1}{2}\sum_{m=1}^{NM}\left\|c^{(l,m)}\right\| \tag{23}$$

there is

$$q^{(l)} = \left\lfloor \frac{\sum_{n=1}^{N}\sum_{m=1}^{M}\left\|c^{(l,n,m)}\right\|}{\left\|c^{(l,*,*)}\right\|} \right\rfloor = 1 \tag{24}$$

In this case, the number of kernels actually retained in the layer is 1. Following pruning training, the number of retained kernels in the convolutional layer is gradually reduced from $M \times N$. Ideally, the number of retained kernels is 1. When the compression ratio of the model reaches its set value, the pruning process stops. The number of kernels retained in each layer is adaptively adjusted according to the network training situation. The IMSI $q^{(l)}$ is calculated by the retained kernel of the entire convolutional layer.

D. FILTER - LAYER PROGRESSIVE KERNEL PRUNING

This section integrates KPF and KPL to propose a progressive kernel pruning method. In the KPF stage, there are fewer convolution kernels in the filter than in the convolution layer, so the model converges more easily during pruning training. However, if the mapping ability of the retained kernel in the filter is small, the information transferred from the filter to the next layer of the network is small, so the retained structure has a certain degree of redundancy; this causes a problem in terms of the further simplification of the overall network structure. The KPL stage prunes the convolution kernel by considering the relationships between all convolution kernels in the convolution layer. If the mapping abilities of all convolution kernels in a filter are generally small, then all convolution kernels in the filter are pruned by the KPL process. Therefore, all convolution kernels of the filter, which are of relatively small importance in the process of KPF pruning, can be pruned, and the pruning rate is further improved. Since the convolution layer is composed of multiple filters, pruning by directly considering the importance of the kernels in the whole layer complicates the optimization of the process of obtaining pruning results. The probability of obtaining the local optimal solution increases during pruning training, and the network performance after pruning is greatly affected.

Therefore, this paper proposes a two-stage pruning method (PKP) based on the IMSI. This is a progressive kernel pruning method ranging from convolution kernel pruning in the filter to convolution kernel pruning in the convolution layer. KPL can be regarded as a progression of KPF. Due to the better convergence of KPF training than KPL training, the retained kernel in the filter can be fully trained to achieve the set model pruning rate. KPL training on this basis not only reduces the difficulty of pruning training but also further improves the pruning rate of the model.

Progressive pruning is divided into two stages, KPF and KPL. In the KPF stage, the compression ratio of the model continues to improve during the training process, and the accuracy of the model gradually increases. When the model compression ratio increases slowly, the number of convolution kernels in the filter stabilizes, and the retained kernels become fully trained. The KPF stage can progress to the KPL stage to further increase the compression ratio of the model. According to equations (8, 11), in KPF, the number of kernels retained in each filter is $q^{(l,n)}$, and for KPL, the number of kernels retained in each convolutional layer is $q^{(l)}$. Pruning is based on the use of a two-dimensional convolution kernel as

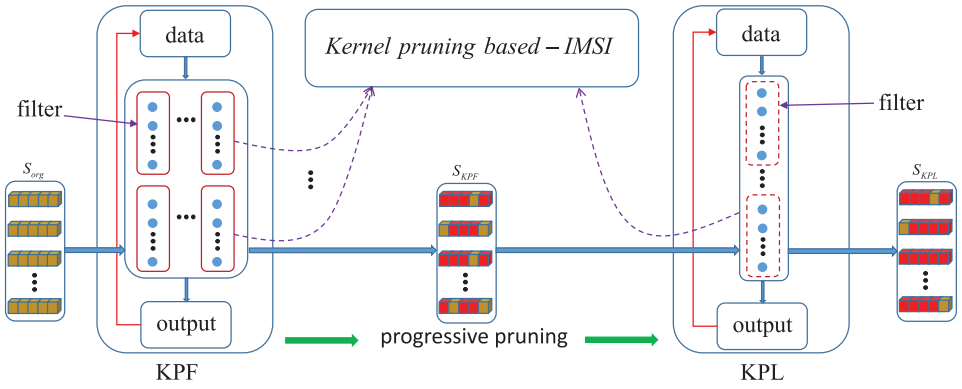


FIGURE 4. The progressive kernel pruning (PKP). In the KPF stage, kernel pruning for each filter, the number of retained kernels is IMSI $q^{(l,n)}$, when the CR can not increase, progressive to KPL stage, the number of retained kernels is IMSI $q^{(l)}$, which obtain a higher CR.

the smallest unit, and the number of convolution kernels in the l-th layer is $N \times M$. The parameters of the batch normalization layer are $2N^{(l)}$. In KPF, the compression ratio CR_{KPF} is as follows:

$$CR_{KPF} = \sum_{l=1}^{L}\sum_{n=1}^{N} \frac{N^{(l)} \times M^{(l)} \times k^{(l)} \times k^{(l)} + 2N^{(l)}}{q^{(l,n)} \times k^{(l)} \times k^{(l)} + 2N^{(l)}} = \sum_{l=1}^{L}\sum_{n=1}^{N} \frac{N^{(l)} \times (M^{(l)} + 2)}{q^{(l,n)} + 2N^{(l)}} \tag{25}$$

Therefore, the final compression ratio of the model is as follows:

$$CR_{KPL} = \sum_{l=1}^{L} \frac{N^{(l)} \times M^{(l)} \times k^{(l)} \times k^{(l)} + 2N^{(l)}}{q^{(l)} \times k^{(l)} \times k^{(l)} + 2N^{(l)}} = \sum_{l=1}^{L} \frac{N^{(l)} \times (M^{(l)} + 2)}{q^{(l)} + 2N^{(l)}} \tag{26}$$

Let the average compression ratio of the previous consecutive I_b iterations after completing the Ith iteration in the model pruning training is as follows:

$$\overline{CR}^{(I)} = \frac{1}{I - I_b} \sum_{i=I-I_b+1}^{I} CR^{(i)} \tag{27}$$

$CR^{(i)}$ is the compression ratio of the model after the ith iteration. Only when the compression ratio of the model increases very slowly can we switch from KPF to KPL. In the KPF stage, let $\Delta\xi^{(I)} = \overline{CR}^{(I)}_{KPF} - \overline{CR}^{(I-1)}_{KPF}$. When $\Delta\xi^{(I)}$ is small, the compression ratio of the model remains almost unchanged after repeated pruning. The model compression ratio cannot be significantly improved, and it is necessary to switch to KPL to further improve the model compression ratio.

The schematic diagram of the PKP based on IMSI is shown in Figure 4. The filter sets S_{org}, S_{KPF}, and S_{KPL} represent the filter sets of the l-th layer without pruning, after KPF, and after KPL respectively. Red cubes represent the pruned kernels. The soft kernel pruning method is adopted for training purposes, and the convolution kernel with the strongest mapping ability is selected according to the IMSI. First, kernel pruning is performed for each filter in each layer according to the IMSI $q^{(l,n)}$, using as few kernels as possible to approach the upper limit of the output information capacity of the filter. The aim of this process is to find a suitable kernel for retention by considering the relationships between the convolution kernels in the filter. Because a filter with a convolution kernel in the layer can be regarded as a subset of all convolution kernels in the layer, kernel pruning based on the IMSI $q^{(l)}$ in the layer can be regarded as the transition from the local subset to the overall set to find the best kernel for retention. Because most of the remaining kernels are well trained via the KPF process, it is easy to prune the convolution kernels in the whole convolution layer. This progressive convolution kernel pruning method has more obvious direction and purpose, simpler training, and easier optimization than other methods. The progressive kernel pruning algorithm is as follows:

The compressed model has a small number of kernels, and the network can be accelerated by sharing the convolution results. The model computational consumption is mainly convolution operations, therefore, the model theory acceleration ratio (AR) is calculated as follows:

$$AR \simeq \frac{\sum_{l=1}^{L} N^{(l)} \times M^{(l)} \times D^{(l)} \times D^{(l)} \times d^{(l)} \times d^{(l)}}{\sum_{l=1}^{L} q^{(l)} \times D^{(l)} \times D^{(l)} \times d^{(l)} \times d^{(l)}} \tag{28}$$

IV. EXPERIMENTS AND ANALYSIS

This section mainly introduces the experimental settings and the performance of model compression achieved by our method compared to those of the existing methods on classical networks and datasets. In this paper, VGG [1], ResNet [2], and MobileNet V2 [45] are pruned on CIFAR-10/100 [46], ImageNet [47] datasets. These models contain heavyweight (VGG-16, ResNet-18) and lightweight (ResNet-56/110, MobileNet V2) models and single-branch (VGG-16) and multi-branch (ResNet-18, MobileNet V2) models. The method is completed on PyTorch [48].

TABLE 1. Results for pruning VGG-16 and ResNet-18 on CIFAR-10.

Model	Method	Pretrained	Baseline Acc.(%)	Pruned Acc. (%)	$Acc.\downarrow$ (%)	Params CR(×)	FLOPs AR(×)
VGG-16	Lin *et al.* [10]	Y	93.34	**93.77**	−0.43	25.00	2.10
	AOFP [43]	Y	93.38	93.28	0.10	–	4.10
	Sparse VD [39]	Y	92.45	92.45	0.00	65.00	–
	SS-Auto [50]	Y	**93.70**	93.50	0.20	41.00	–
	Xiao *et al.* [33]	Y	92.40	92.18	0.22	75.00	–
	Auto Compress [49]	Y	**93.70**	92.72	0.98	61.10	10.60
	PKP(ours)	Y	93.35	92.76	0.59	**80.84**	**14.82**
	Auto Compress [49]	N	–	91.40	–	52.20	–
	PKP(ours)	N	–	**92.42**	–	**80.03**	**14.12**
ResNet-18	Lin *et al.* [10]	Y	93.55	**94.20**	−0.65	20.00	1.06
	SS-Auto [50]	Y	**94.10**	**94.20**	−0.10	24.60	–
	Auto Compress [49]	Y	93.90	92.98	0.92	80.80	14.30
	PKP(ours)	Y	93.36	92.72	0.64	**85.42**	**14.70**
	Auto Compress [49]	N	–	91.88	–	54.20	12.20
	PKP(ours)	N	–	**92.21**	–	**82.73**	**14.47**

Algorithm 2 Progressive Kernel Pruning

1. Prepare the pretrained model;
2. In the KPF stage:
   a. Pruning training;
   b. Select the retained kernel according to Equation (8);
   c. Calculate the model CR according to Equation (25);
   d. if $\Delta\xi_{KPF}^{(l)} \leq 0.01$:
      go to Step 3.
   else:
      return a.
3. In the KPL stage:
   e. Pruning training;
   f. Select the retained kernel according to Equation (11);
   g. Calculate the model CR according to Equation (26);
   h. if $\Delta\xi_{KPL}^{(l)} \leq 0.01$:
      obtain set $P^{(l)}$, CR, AR, and go to Step 4.
   else:
      return e.
4. Training retained kernels;
5. Obtain the performance of the model after pruning.

The filter in the first convolution layer of a given model contains 3 convolution kernels, which is not enough for pruning with KPF, so pruning is only performed with KPL. When pruning the fully connected layer, this paper regards each weight as a convolution kernel of size 1×1 and prunes in the same way as the progression from KPF to KPL. In this paper, the classification accuracy used is Top-1 accuracy, and the equation is as follows:

$$\text{accuracy}_{\text{Top-1}} = \frac{\sum TP_j}{Num_{total}} \tag{29}$$

where, j represents the j-th category, Num_{total} is the total number of test samples. True positives (TP) are examples correctly labeled as positives. False positives (FP) refer to negative examples incorrectly labeled as positive. True negatives (TN) correspond to negatives correctly labeled as negative and false negatives (FN) refer to positive examples incorrectly labeled as negative. Each experiment is conducted ten times and each result is presented as the mean. The model uses the same evaluation indexes of classification accuracy before and after pruning.

A. RESULTS ON CIFAR-10

1) VGG-16/ResNet-18

The results of pruning VGG-16 and ResNet-18 on CIFAR-10 are shown in Table 1. In VGG-16, the proposed PKP has the highest compression ratio (CR) and acceleration ratio (AR), reaching 80.84× and 14.82× respectively. The accuracy is decreases by 0.59% from the original, reaching 92.76%. The classification accuracy of [10] is slightly higher than the baseline, but the CR and AR are 25× and 2.1×, respectively. For the "pruning from scratch" mode, the CR and AR of the PKP are 80.03× and 14.12×, respectively, and the classification accuracy is 92.42%. Compared to the pretrained mode, the CR of the PKP is 0.72× lower, and the classification accuracy decreases by 0.31%. The CR and AR of the method [49] are 52.20× and 8.8×, respectively, when pruning from scratch, and the classification accuracy is 1.32% lower than that of the pretrained mode. Table 2 shows the kernel sparsity ratio in each layer. It can be observed that the higher the layer is, the higher the sparsity rate. For example, in Conv-12, the kernel sparsity rate is 99.92%, and the number of kernels actually retained is only 197. For the ResNet-18, the proposed PKP has the highest CR and AR, reaching 85.42× and 14.70×, respectively, and the classification accuracy is decreases by 0.64% from that of the baseline. When pruning from scratch, the CR of the PKP method is 82.73×, which is significantly higher than the 54.20× CR of the method in [49]. For VGG-16 and ResNet-18, the proposed PKP has the highest CR and AR and achieves the best performance. When pruning from scratch, it also exhibits excellent performance. Table 3 shows the PPV and TPR of VGG-16/ResNet-18. Regardless of the original model or the pruned model, the values between PPV are

TABLE 2. The kernel sparse rate in each layer (In VGG-16 for CIFAR-10).

Layer	Conv-1	Conv-2	Conv-3	Conv-4	Conv-5	Conv-6	Conv-7
Sparse Rate(%)	73.96	91.77	82.92	84.07	87.93	94.86	94.84
Layer	Conv-8	Conv-9	Conv-10	Conv-11	Conv-12	Conv-13	Full-1
Sparse Rate(%)	98.05	99.78	99.83	99.92	99.92	99.91	74.73

TABLE 3. The PPV and TPR of VGG-16/ResNet-18.

Model	Index	Pruned	category									
			0	1	2	3	4	5	6	7	8	9
VGG-16	PPV	N	0.926	0.972	0.919	0.874	0.924	0.894	0.964	0.963	0.963	0.948
		Y	0.920	0.959	0.920	0.839	0.930	0.882	0.953	0.966	0.956	0.958
	TPR	N	0.949	0.967	0.913	0.843	0.948	0.903	0.946	0.955	0.960	0.963
		Y	0.939	0.956	0.902	0.851	0.941	0.879	0.957	0.939	0.957	0.951
ResNet-18	PPV	N	0.940	0.968	0.930	0.853	0.946	0.874	0.956	0.967	0.945	0.959
		Y	0.917	0.954	0.921	0.852	0.931	0.880	0.956	0.948	0.959	0.956
	TPR	N	0.933	0.968	0.910	0.861	0.946	0.898	0.952	0.955	0.959	0.951
		Y	0.945	0.967	0.908	0.828	0.937	0.885	0.946	0.961	0.950	0.946

TABLE 4. Results for pruning ResNet-56/110 and MobileNetV2 on CIFAR-10.

Model	Method	Pretrained	Baseline Acc.(%)	Pruned Acc. (%)	$Acc.\downarrow$ (%)	Params CR(×)	FLOPs AR(×)
ResNet-56	PFEC [13]	Y	93.04	93.06	−0.02	1.15	1.38
	CP [21]	Y	92.80	91.80	1.00	–	2.00
	FPGM [14]	Y	93.59	92.49	1.10	1.67	1.90
	GAL [41]	Y	**93.97**	91.58	2.39	**2.93**	2.51
	OED [23]	Y	**93.97**	92.29	1.68	2.50	**3.09**
	FPDC [18]	Y	93.64	93.39	0.25	1.93	1.99
	KSE [11]	Y	93.03	92.88	0.14	2.40	2.50
	LSTM [22]	Y	93.04	92.93	0.11	1.76	1.90
	C-SGD [42]	Y	93.39	93.44	**−0.05**	–	2.55
	PKP(ours)	Y	93.60	**93.51**	0.09	2.51	2.54
ResNet-110	PFEC [13]	Y	93.53	93.55	**−0.02**	1.02	1.19
	FPGM [14]	Y	93.68	93.74	−0.06	1.67	1.90
	GAL [41]	Y	94.10	92.74	1.36	1.81	1.94
	OED [23]	Y	94.10	93.60	0.50	1.95	2.17
	FPDC [18]	Y	93.82	93.78	0.04	1.93	1.93
	PKP(ours)	Y	**94.27**	**93.82**	0.45	**2.32**	**2.18**
MobileNetV2	GAL [41]	Y	93.40	93.07	0.23	1.12	1.71
	OED [23]	Y	**93.40**	93.35	0.05	1.25	1.93
	PKP(ours)	Y	93.28	**93.41**	**−0.13**	**2.22**	**2.02**

relatively close, and the values between TPR are relatively close too. Because the classification accuracy of the model after pruning will be slight decrease, but the model in all categories of classification performance is relatively uniform, there is no large deviation, which also shows that the model in the high compression ratio and acceleration ratio, classification performance has better stability.

2) ResNet-56/110 AND MobileNetV2

The results for pruning ResNet-56/110 and MobileNetV2 on CIFAR-10 are shown in Table 4. In ResNet-56, the CR and AR of the PKP method are 2.51× and 2.54×, respectively. The classification accuracy is 93.51%, which is a decreases of 0.09% from that of the baseline and is the highest compared to those of the methods in [11], [13], [14], [18], [21]–[23], [41]. The CR of the method in [41] reaches 2.93×, but the classification accuracy is 91.58%, which is a decrease of 2.39%. The AR of the method in [23] reaches 3.09×, and the classification accuracy is 92.29%, which is a decreases of 1.68%. For ResNet-110, the proposed PKP has the highest CR and AR, reaching 2.32× and 2.18×, respectively, and the classification accuracy decreases by 0.45% from that of the baseline, reaching 93.82%; this is also the highest when compared to the classification accuracies of the methods in [13], [14], [18], [23], [41]. For MobileNetV2, the proposed PKP has the highest CR and AR, reaching 2.22× and 2.02× respectively, and the classification accuracy is increased by 0.13%, reaching 93.41%, which is also the highest. The proposed PKP can achieve the highest CR and AR for ResNet-56/110 and MobileNetV2 and has the best classification accuracy. This is because progressive pruning

TABLE 5. Results for pruning VGG-16 on CIFAR-100.

Model	Method	Pretrained	Baseline Acc.(%)	Pruned Acc. (%)	$Acc.\downarrow$ (%)	Params CR(×)	FLOPs AR(×)
VGG-16	SSL [17]	Y	**73.16**	73.18	−0.02	2.60	1.60
	Decorrelation [37]	Y	**73.16**	**73.21**	**−0.05**	3.90	2.10
	FPDC [18]	Y	73.22	72.49	0.73	1.95	1.98
	SS-Auto [50]	Y	72.90	72.20	0.70	17.50	–
	PKP(ours)	Y	72.92	72.77	0.15	**27.25**	**7.20**
	PKP(ours)	N	–	72.27	–	20.12	5.50

TABLE 6. Results for pruning ResNet-110 and MobileNetV2 on CIFAR-100.

Model	Method	Pretrained	Baseline Acc.(%)	Pruned Acc. (%)	$Acc.\downarrow$ (%)	Params CR(×)	FLOPs AR(×)
ResNet-110	OED [23]	Y	**74.36**	70.53	3.83	2.31	**3.23**
	FPDC [18]	Y	73.38	71.77	0.61	1.93	1.93
	PKP(ours)	Y	72.82	**72.21**	**0.61**	**2.42**	2.37
MobileNetV2	PKP(ours)	Y	76.40	75.39	1.01	2.11	1.91

TABLE 7. Results for pruning ResNet-18 on ImageNet.

Model	Method	Pretrained	Baseline Acc.(%) Top-1/Top-5	Pruned Acc.(%) Top-1/Top-5	$Acc.\downarrow$ (%) Top-1/Top-5	Params CR(×)	FLOPs AR(×)
ResNet18	DCP [40]	Y	69.64/88.98	67.35/87.69	2.29/1.38	2.00	–
	Auto Compress [49]	Y	−/−	−/−	−/0.10	**3.30**	–
	SFP [24]	Y	70.28/89.63	67.10/87.78	3.18/1.85	1.43	1.70
	FPGM [14]	Y	70.28/89.63	68.41/88.48	1.87/1.15	1.43	1.70
	PKP	Y	**70.46/89.56**	**70.06/89.50**	**0.40/0.06**	2.83	**2.00**
	PKP	N	–	68.67/88.50	−/−	2.54	1.81

not only takes the relationships between the local convolution kernels in the filter into account but also considers the overall relationships between the convolution kernels in the convolution layer, making it simple for the method to approach the global optimum.

B. RESULT ON CIFAR-100

1) VGG-16

The results for pruning VGG-16 on CIFAR-100 are shown in Table 5. In VGG-16, the proposed PKP has the highest CR and AR, reaching 27.25× and 7.20× respectively, the classification accuracy is decreases by 0.15%, reaching 72.77%. Although the CR of the method in [50] reaches 17.5×, the classification accuracy was only 72.20%, which was decreases by 0.7%. In method [37], the CR of the model is 3.90×, but the accuracy after compression is slightly higher than the baseline. However, when PKP reaches a 10.10× CR, the acc classification accuracy is increased significantly by 0.68%. From the scratch mode, the CR and AR of the PKP method are 20.12× and 5.50×, respectively, which is also significantly higher than that of the method [50]. Using the PKP method for pruning, the model accuracy will be significantly improved when the model CR is low. When the model CR is high, the classification accuracy decrease is not obvious, especially in the scratch mode, the model classification accuracy does not decrease much.

2) ResNet-110 AND MobileNetV2

The results for pruning ResNet-110 and MobileNetV2 on CIFAR-100 are shown in Table 6. In ResNet-110, the CR and AR of the PKP method are 2.42× and 2.37×, respectively. The classification accuracy is 72.21%, which is decreases by 0.61%, and is the highest compared to [18], [23]. In addition, in MobileNetV2, the proposed PKP has CR and AR are 2.11× and 1.91×, respectively, the classification accuracy is 75.39%, which is decreases by 1.01%. The results show that the PKP method selects the retained kernel according to the IMSI in the two-stage pruning process, which can reduce the false prune rate of the retained kernel and effectively improve the CR and AR of the model under the premise of ensuring performance.

C. RESULT ON IMAGENET

The results for pruning ResNet-18 on ImageNet are shown in Table 7. The CR of the method in [40] is 2.00×, the Top-1 classification accuracy decreases by 2.29%, and the Top-5 classification accuracy decreases by 1.38%. The CR and AR of the PKP method are 2.83× and 2.00×, respectively, the Top-1 classification accuracy decreases by 0.40%, and the Top-5 classification accuracy decreases by 0.06%. The classification accuracy of method [14], [24] decrease more. It can be observed that the PKP method is more robust in pruning training.

D. ABLATION STUDY

1) RESULTS ON KPF AND KPL

This section presents the experimental results of the KPF and KPL methods separately, as shown in Table 8. On CIFAR-10, the compression test for VGG-16 shows that the KPF provides better performance than KPL with the pretrained model.

TABLE 8. The pruning effect of using KPF and KPL, separately (Pruning VGG-16 on CIFAR-10).

Method	Pretrained	Baseline Acc.(%)	Pruned Acc. (%)	$Acc.\downarrow$ (%)	CR(×)
KPF	Y	93.35(±0.15)	92.97(±0.12)	0.38	39.03(±0.09)
KPL	Y	93.35(±0.15)	91.90(±0.75)	1.45	40.67(±0.54)

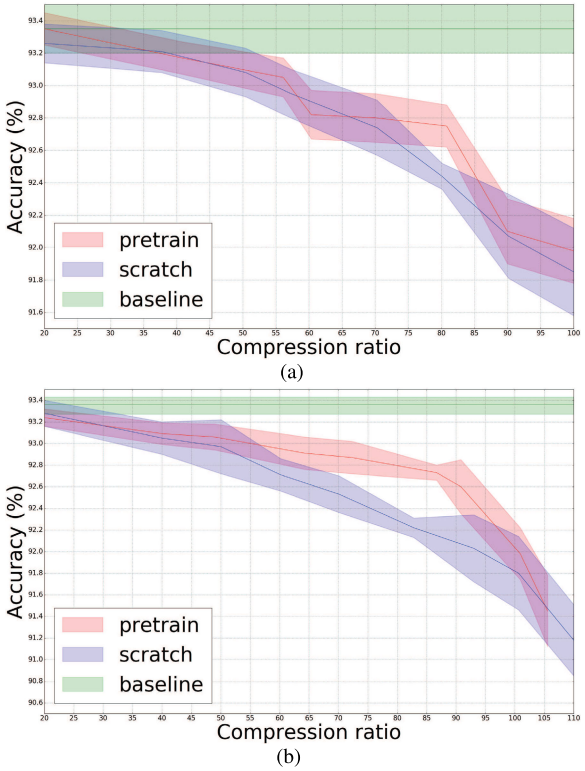


FIGURE 5. Model accuracy at different CRs. (Solid line and shadow denotes the mean values and standard deviation of three experiments, respectively.) (a) Pruning VGG-16 on CIFAR-10. (b) Pruning ResNet-18 on CIFAR-10.

While the CR reaches 39.03×, the classification accuracy decreases by only by 0.38%. However, if the maximum mapping abilities of kernels in the filter are very small, KPF cannot delete these kernels, making it difficult to further increase the CR. When the KPL method is used alone, the CR is 40.67×, while the classification accuracy decreases greatly (by 1.45%), because it is difficult to optimize all convolution kernels in the convolution layer without using KPF.

2) THRESHOLD SELECTION

In the KPF stage, if $I_b = 5$ and $\Delta\xi^{(I)}_{KPF} \leq 0.01$, the retained kernel in the filter is considered fully trained, so it is necessary to switch to KPL to further improve the model compression ratio.

3) ACCURACY CORRESPONDING TO DIFFERENT CRs

This section analyzes the changes in classification accuracy for different CRs, as shown in Figure 5. Solid line and shadow denotes the mean values and standard deviation of three experiments, respectively. For pruning VGG-16 on the CIFAR-10, the experimental results are shown in Figure 5(a). Under the pretrained mode, when the CR provided by the proposed PKP is less than 50×, the compressed model can basically reach the classification accuracy of the baseline through fine tuning. When $CR \geq 80\times$, the classification accuracy of the model decreases rapidly. When $CR = 95\times$, the classification accuracy of the model is 1% less than that of the baseline. When pruning from scratch, compared to the pretrained mode, when $CR < 70\times$, the accuracies of the two methods are similar. For Pruning ResNet-18 on CIFAR-10, the experimental results are shown in Figure 5(b). Under the pretrained mode, when $CR \leq 90\times$, the classification accuracy of the model decreases slowly. When $CR > 90\times$, the classification accuracy of the model decreases rapidly. When $CR = 105\times$, the accuracy is nearly 2% lower than that of the baseline. When pruning from scratch mode, when $CR < 50\times$, the classification accuracy of the model decreases slowly. When $CR > 50\times$, the decrease in accuracy is slightly faster. We believe that in the method using a pretrained model, the model is pruned under a better model. Although the CR is higher, it is easier to obtain results that are not much different from those of the baseline. It is important to select the best kernel when pruning from scratch. The method of progressive pruning training is also very important, showing that the method in this paper has excellent robustness.

E. STATISTICAL ANALYSIS OF RESULTS

In this paper, various types of models have been pruned on CIFAR-10/100 and ImageNet. Based on the experimental results, it can be observed that the proposed PKP can achieve high CR and AR, and the classification accuracy is not significantly reduced. Compared to the reference methods, the PKP exhibits some performance advantages, especially when pruning from scratch. Through progressive pruning, KPF is carried out first so that the mapping abilities of a few kernels can approach the upper limit of the information output capacity as closely as possible. During this process, the relationships between convolution kernels in each filter are considered for kernel pruning purposes; this approach is easy to optimize and can achieve better accuracy than those of other approaches. On this basis, KPL considers the relationships between convolution kernels in the convolution layer for kernel pruning. KPL is optimized on the basis of KPF. Compared with the direct use of KPL, the difficulty of optimization for KPF is significantly reduced, and better accuracy can be obtained. Therefore, the progressive kernel pruning method proposed in this paper can be applied to

the pruning of heavyweight/lightweight models and single-branch/multi-branch networks, respectively, and has good universality. By calculating the PPV and TPR of the subcategories, it is found that the classification performance of the pruned model has good stability under high compression ratio and acceleration ratio.

V. CONCLUSION

This paper proposes the PKP method based on the IMSI for CNN model compression. The proposed IMSI measures the mapping abilities of convolution operations. The filter IMSI is used to prune the kernels in each filter. Then, the layer IMSI is used to prune the remaining kernels in the entire layer. During the process of kernel pruning, the kernel with the strongest mapping ability is retained. The network passes as much effective information as possible during the convolution process. The proposed PKP can further improve the CR and AR of the model on the basis of KPF while avoiding the difficulty of optimizing the KPL method directly. Through the experimental results, it can be observed that the CR and AR of the proposed PKP are higher than those of other pruning methods, and the accuracy can be maintained at a comparable level to that of the baseline. Furthermore, it exhibits certain advantages in terms of performance, especially when pruning from scratch.

ACKNOWLEDGMENT

The authors are grateful to the reviewers for the precious suggestions and feedbacks for improving this article and to the editors for their meticulous processing.

REFERENCES

[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[6] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2359–2367.

[7] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," 2016, *arXiv:1604.03635*. [Online]. Available: http://arxiv.org/abs/1604.03635

[8] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[9] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.

[10] C. Lin, Z. Zhong, W. Wei, and J. Yan, "Synaptic strength for convolutional neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10149–10158.

[11] Y. Li, S. Lin, B. Zhang, J. Liu, D. Doermann, Y. Wu, F. Huang, and R. Ji, "Exploiting kernel sparsity and entropy for interpretable CNN compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2800–2809.

[12] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," 2017, *arXiv:1705.08922*. [Online]. Available: http://arxiv.org/abs/1705.08922

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," 2016, *arXiv:1608.08710*. [Online]. Available: http://arxiv.org/abs/1608.08710

[14] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4340–4349.

[15] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2736–2744.

[16] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 5058–5066.

[17] 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.*, 2016, pp. 2074–2082.

[18] Y. Zuo, B. Chen, T. Shi, and M. Sun, "Filter pruning without damaging networks capacity," *IEEE Access*, vol. 8, pp. 90924–90930, 2020.

[19] S. Lin, R. Ji, Y. Li, Y. Wu, F. Huang, and B. Zhang, "Accelerating convolutional networks via global & dynamic filter pruning," in *Proc. IJCAI*, 2018, pp. 2425–2432.

[20] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 304–320.

[21] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact ConvNets via structure-sparsity regularized filter pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 574–588, Feb. 2020.

[22] G. Ding, S. Zhang, Z. Jia, J. Zhong, and J. Han, "Where to prune: Using LSTM to guide data-dependent soft pruning," *IEEE Trans. Image Process.*, vol. 30, pp. 293–304, 2021.

[23] Z. Wang, S. Lin, J. Xie, and Y. Lin, "Pruning blocks for CNN compression and acceleration via online ensemble distillation," *IEEE Access*, vol. 7, pp. 175703–175716, 2019.

[24] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft filter pruning for accelerating deep convolutional neural networks," 2018, *arXiv:1808.06866*. [Online]. Available: http://arxiv.org/abs/1808.06866

[25] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9194–9203.

[26] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1389–1397.

[27] V. Lebedev and V. Lempitsky, "Fast ConvNets using group-wise brain damage," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2554–2564.

[28] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," 2017, *arXiv:1712.01312*. [Online]. Available: http://arxiv.org/abs/1712.01312

[29] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 598–605.

[30] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Proc. Adv. Neural Inf. Process. Syst.*, 1993, pp. 164–171.

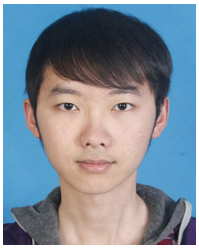
[31] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*. [Online]. Available: http://arxiv.org/abs/1510.00149

[32] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," 2017, *arXiv:1702.04008*. [Online]. Available: http://arxiv.org/abs/1702.04008

[33] X. Xiao, Z. Wang, and S. Rajasekaran, "Autoprune: Automatic network pruning by regularizing auxiliary parameters," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13681–13691.

[34] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, "Cnnpack: Packing convolutional neural networks in the frequency domain," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 253–261.
[35] Z. Liu, J. Xu, X. Peng, and R. Xiong, "Frequency-domain dynamic pruning for convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1043–1053.
[36] Y. He, X. Dong, G. Kang, Y. Fu, C. Yan, and Y. Yang, "Asymptotic soft filter pruning for deep convolutional neural networks," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3594–3604, Aug. 2020.
[37] X. Zhu, W. Zhou, and H. Li, "Improving deep neural network sparsity through decorrelation regularization," in *Proc. IJCAI*, 2018, pp. 3264–3270.
[38] C. Liu, Y. Wang, K. Han, C. Xu, and C. Xu, "Learning instance-wise sparsity for accelerating deep models," 2019, *arXiv:1907.11840*. [Online]. Available: http://arxiv.org/abs/1907.11840
[39] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2498–2507.
[40] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 875–886.
[41] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, and D. Doermann, "Towards optimal structured CNN pruning via generative adversarial learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2790–2799.
[42] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for pruning very deep convolutional networks with complicated structure," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 4943–4953.
[43] X. Ding, G. Ding, Y. Guo, J. Han, and C. Yan, "Approximated oracle filter pruning for destructive CNN width optimization," in *Proc. ICML*, 2019, pp. 1–11.
[44] J. Muscat, *Functional Analysis*, 1st ed. Cham, Switzerland: Springer, 2014, doi: 10.1007/978-3-319-06728-5.
[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.
[46] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 001, 2009.
[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[48] P. Adam, G. Sam, C. Soumith, C. Gregory, Y. Edward, D. Zachary, L. Zeming, D. Alban, A. Luca, and L. Adam, "Automatic differentiation in PyTorch," in *Proc. Neural Inf. Process. Syst.*, 2017, pp. 1–4.
[49] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic DNN structured pruning framework for ultra-high compression rates," in *Proc. AAAI*, 2020, pp. 4876–4883.
[50] Z. Li, Y. Gong, X. Ma, S. Liu, M. Sun, Z. Zhan, Z. Kong, G. Yuan, and Y. Wang, "SS-auto: A single-shot, automatic structured weight pruning framework of DNNs with ultra-high efficiency," 2020, *arXiv:2001.08839*. [Online]. Available: http://arxiv.org/abs/2001.08839

JIHONG ZHU received the B.S. degree in electronic information engineering from Gannan Normal University, China, in 2005, and the M.S. degree in signal systems and processing from Nanchang University, China, in 2010. He is currently pursuing the Ph.D. degree with the College of Electronics and Information Engineering, Shenzhen University, China. His research interests include deep learning, pattern recognition, and image processing.



YANG ZHAO received the B.S. degree in applied mathematics from Huanggang Normal University, China, in 2012, and the M.Sc. degree from the College of Mathematics and Statistics, Shenzhen University, China, in 2016, where he is currently pursuing the Ph.D. degree with the College of Electronics and Information Engineering. His research interests include pattern recognition and image processing.

JIHONG PEI (Member, IEEE) received the B.S. degree in electronics engineering from Beihang University, in 1989, and the M.S. degree in physical electronics and optoelectronics, and the Ph.D. degree in signal and information processing from Xidian University, in 1994 and 1998, respectively. He is currently a Professor with Shenzhen University and the Director of the Department of Electronic Engineering with the College of Information Engineering, Shenzhen University. His research interests include intelligent information processing, and video and image analysis. He is a member of the Technical Committee of Information Fusion in the Chinese Society of Aeronautics and Astronautics.