Applied Intelligence (2024) 54:216–230
https://doi.org/10.1007/s10489-023-05207-x

Automatic filter pruning algorithm for image classification

Yifan Xue[1] · Wangshu Yao[1,2,3] · Siyuan Peng[1] · Shiyou Yao[1]

Accepted: 29 November 2023 / Published online: 7 December 2023

Abstract
Network pruning is an essential technique for compressing and accelerating convolutional neural networks (CNNs). Existing pruning algorithms primarily evaluate filter importance or similarity, and then remove unimportant filters or keep only one similar filter at each convolutional layer based on a global pruning ratio. These methods, ignoring the sensitivity of pruning among different convolutional layers, rely on a lot of manual experience and multiple experiments to obtain the optimal convolutional neural network structure. To this end, we propose an automatic filter pruning algorithm via feature map average similarity and reverse search genetic algorithm(RSGA), dubbed as AFPruner, which automatically searches for the optimal combination of pruning ratio for all convolutional layers, evaluates filter similarity by feature map average similarity and then prunes similarity filter. Our method is evaluated against several state-of-the-art CNNs on three different classification datasets, and the experimental results show that our algorithm outperforms most current network pruning algorithms.

Keywords Model compression · Convolutional neural network · Network pruning · Genetic algorithm · Feature map average similarity

1 Introduction

In the past decade, convolutional neural networks(CNNs) have exhibited remarkable success in various computer vision tasks, such as image classification [1], object detection [2], and semantic segmentation [3], owing to their exceptional ability for representation learning. However, these high-performance CNN models, such as VGG [4], ResNet [5], and DenseNet [6], require a large amount of memory and computational resources, which limits their deployment on edge computing devices, e.g., on mobile devices.

In recent years, the compression and acceleration techniques for deep models have become a hot research topic. Commonly used techniques include knowledge distillation, network pruning, and binary quantization. Network pruning refers to reducing the storage and computational cost of deep convolutional neural networks by removing redundant components, thus achieving model compression and acceleration. Current research focuses mainly on structured pruning, which involves removing unimportant filters from the original network based on appropriate pruning strategies. Li et al. [7] evaluates convolution kernels according to the l_1-$norm$ and removes unimportant kernels. Liu et al. [8] uses L_1 regularization of the channel scaling factors of the batch normalization(BN) layer to measure the importance of the channel, and then prunes the channels with smaller scaling factor values. HRank [9] first introduces the rank of the feature map as the evaluation criterion, which makes the whole pruning process more efficient.

In pruning process, the essential work is selecting appropriate pruning strategies to judge redundant components and determining the optimal pruning ratio. However, existing pruning strategies are oversimple and neglect the structural integrity and global correlation between layers of the CNN model. These strategies prune layer by layer using a uniform pruning ratio, which may result in some layers being overly pruned while others still exhibit structural redundancy, thus fail to obtain the optimal network model structure. In addition, the setting of the pruning ratio relies on manual experience and requires repeated test to find the optimal overall pruning ratio.

✉ Wangshu Yao
wshyao@suda.edu.cn

1 School of computer science and technology, Soochow University, Suzhou 21500, China

2 School of software, Soochow University, Suzhou 21500, China

3 Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu 21500, China

To address the above-mentioned problems, we propose an automatic filter pruning algorithm AFPruner based on feature map average similarity and reverse search genetic algorithm(RSGA), which automatically searches for the optimal combination of pruning ratio of each convolutional layer of CNN. AFPruner uses feature map average similarity to measure the similarity of filters, and then prunes similar filters at each layer according to the pruning ratio. In this way, we can obtain the optimal CNN model structure.

Our contributions are summarized as follows:

(1) We propose a reverse search genetic algorithm(RSGA), which enhances the diversity of the population using a reverse search strategy. At the end of each iteration, the algorithm replaces the worst two individuals with their reverse individuals. Additionally, we use clustering methods to group individuals and select individuals from different clusters for evolutionary operations, which further avoids the "prematurity" convergence.
(2) We use the reverse search genetic algorithm to search for the optimal combination of pruning ratio of each convolutional layer of CNN. Then, the similarity of the filter is described using the feature map average similarity, where the feature map average similarity is calculated by using the average Euclidean distance between the feature maps of multiple samples, and the filter is pruned according to the filter similarity. This avoids the effect of sample bias on filter similarity measurements. According to the similarity of the filters, we retain one similar filter per group at each layer of the network and prune the rest of the filters, achieving the purpose of model compression. This approach reduces manual intervention and repeated testing, and realizes automatic pruning.
(3) In this paper, experiments are designed on CIFAR-10, CIFAR-100, and ILSVRC-2012 datasets to verify the performance of the algorithm and compared with other methods. The results show that AFPruner achieves a good balance between performance and compression ratio, and is superior to most of the current pruning algorithms for CNN.

2 Related works

2.1 Pruning strategy

Due to weight pruning leading to non-structured sparse filters, it is hard to accelerate the pruned model with general hardware. Consequently, existing pruning research has mainly focused on structured filter pruning. The current filter pruning strategies can be roughly divided into the following two types:

Importance metric Common pruning methods usually use weight norm [7, 10, 11] to measure the importance of filters, and filters with smaller norm values contain less feature information and should be pruned first. Azadeh et al. [12] propose a genetic-based joint dynamic pruning and learning method and prune redundant filters adaptively during training, which not only considers the l_1-$norm$ of the filter, but also introduces the l_1-$norm$ of gradient to jointly measure the importance of filters. NISP [13] obtains the importance score of each filter by optimizing the reconstruction error of the final response layer. Attention mechanism [14] is also introduced in model pruning. Cheng et al. [15] apply Squeeze-and-Excitation(SE) blocks for each convolutional layer to predict the importance factor of each filter.

Similarity metric FPGM [16] proves that filters are not "smaller-norm-less-important". Furthermore, it calculates the geometric median of all filters in the layer to measure the similarity between filters, and then prunes the filters with the smallest distance sum to the geometric median. OSFP [17] also applies the idea of filter similarity, unlike FPGM [16], which explores the distance relationship between all filters of each convolutional layer. Similarly, Li et al. [18] judge whether a filter is replaceable by exploring the similarity relationship between feature maps. CLR-RNF [19] uses the idea of k-reciprocal nearest to measure similar filters. Some recent works [20–22] all introduce clustering to prune similar filters.

2.2 Pruning ratio

The current methods for setting pruning ratio can be broadly divided into two types: predefined and automatic. The predefined method is to manually set the same pruning ratio for all convolutional layers, and the pruning strategies mentioned above belong to this type. Previous studies [7, 23] have revealed that the different convolutional layers have different sensitivities to pruning. As a result, recent researches have focused on automatically obtaining the pruning ratio, i.e., setting different pruning threshold for all convolutional layers.

Yang et al. [24] leverages second-order information to prune filters with low sensitivity. AMC [25] combines reinforcement learning to determine the pruning ratio for each layer automatically. DSA [26] proposes a new differentiable pruning process, which optimizes the sparsity distribution of continuous space to find the pruning ratio of each layer. DMC [27] assigns a pruning ratio for each layer by applying a discrete gate to each channel.

Liu et al. [28] has shown that the essence of network pruning is finding the optimal structure. Based on this, recent works treat the pruning problem as an optimization problem. ABCPrunner [29] first shrinks the combinations of channels to a specific space, and then automatically obtains the pruning

ratio of each layer through the artificial bee colony algorithm(ABC). MetaPruning [30] proposes a two-stage pruning algorithm, which pretrains a PruningNet using meta-learning to predict the weight parameters of the pruned model, and then utilizes genetic algorithm to search for the optimal combination of pruning ratio. ACP [31] first completes the preliminary pruning of channels through hierarchical clustering of channels, and then uses particle swarm algorithm to search for the optimal pruned model. SST [32] limits the channel pruning ratio of each layer to four thresholds using two magnitude parameters and uses particle swarm optimization to search the combination of pruning ratio. IPSOPruner [33] removes the more sensitive layers during the pruning process and then uses an improved particle swarm optimization algorithm to search for the pruning ratio. AACP [34] uses hyperparameters to compress the pruning ratio combination space, and improves the traditional differential evolution algorithm to obtain the optimal pruning ratio for each layer. To solve the intractably huge combinations of pruned structure for convolutional networks, the above methods artificially constrain the search space, which may result in missing the optimal pruning ratio combination.

In addition to direct search pruning ratio, there are also works to sparse different layers of the network from the perspective of search weight parameters and pruning strategy. Wang et al. [35] guides channel pruning by sparing the distribution of BN layers, dynamically adjusting the importance of channels using a scale factor and bias factor (γ and β), and combines genetic algorithm to search for the optimal factor combination. LFPC [36] defines the search space as different pruning strategies to learn filter pruning criterion. According to the filter distribution of each layer, it adaptively selects a suitable pruning strategy for each layer to prune independently.

3 Methodology

This section will introduce the proposed automatic filter pruning framework based on reverse search genetic algorithm. In the convolutional layer of CNN, the important filters are able to extract discriminative local information [37]. Without destroying the network structure, pruning unimportant filters can effectively decrease the amount of parameters and computations of the model to accelerate the inference speed. For a pre-trained model of CNN with L convolutional layers, the model weights are $\left\{\mathrm{W}_i \in \mathbb{R}^{C_{i+1}\times C_i\times k_i\times k_i}, 1\le i\le \mathrm{L}\right\}$, where C_i, C_{i+1}, and k_i are the number of input channels, the number of output channels, and the size of the convolution kernel of the i-th convolutional layer, respectively. The j-th filter weights of the i-th convolutional layer is denoted as $\mathrm{W}_i^j \in \mathbb{R}^{C_i\times k_i\times k_i}$. $\mathrm{F}_i^j \in \mathbb{R}^{h_i\times w_i}$ is a feature map generated by the filter W_i^j after matrix operations. h_i, w_i are the height and width corresponding to the feature map of the i-th layer, respectively.

3.1 Definition of filter pruning

Most existing pruning methods prune filters(channels) of all layers with a predefined global pruning ratio. However, the pruned model may not be optimal due to the different pruning sensitivity of filters(channels) in different layers. In this paper, determining the optimal pruning ratio combination of each layer filter is regarded as a combination optimization problem. Specifically, we define the pruning problem as follows: given an unpruned model M with L convolutional layers, assuming that each layer of the network has c filters, the filter pruning problem can be formulated as:

$$\begin{aligned} &\min \mathcal{L}_{\text{test}}\ (\mathrm{M}^*, \mathrm{W}) \\ &\text{s.t. } \mathrm{M}^* = \arg\min \mathcal{L}_{\text{train}}\ (\mathrm{R}^*,\ \mathrm{W}) \end{aligned} \tag{1}$$

where M^* is the pruned model, $\mathrm{R}^* = [r_1, r_2, \ldots, r_{\mathrm{L}}]$ is the filter pruning ratio vector, with r_i denoting the filter pruning ratio of layer i-th, $i = 1, \ldots, \mathrm{L}$. W represents the weight parameter of the network, and $\mathcal{L}$ represents the cross-entropy classification loss of the network. This paper aims to obtain the pruned model M^* with the lowest training loss through searching the original network model. Furthermore, after determining the pruning ratio of each layer filters, fine-tuning is performed to minimize the testing error of the pruned subnetwork.

3.2 Filter similarity evaluation

During the search phase, we propose a method for evaluating filter similarity based on feature map average similarity. According to the discovered pruning ratio, AFPruner prunes the similarity filter at each layer.

The pruned model inherits the weight parameters of the original network, to evaluate its performance. Inspired by the previous work [38], we use the average euclidean distance(Other distances can also be used here, such as the cosine distance, which will be described detailedly in Section 4.5.) of s sample feature maps to measure the filter similarity in each convolutional layer, and the similarity calculation formula is as follows:

$$\begin{aligned} &dist\left(\mathrm{M}_i^l, \mathrm{M}_j^l\right) = \tfrac{1}{s}\textstyle\sum_{t=1}^{s}\sqrt{\sum_{n=1}^{\mathrm{N}}\left(m_{t,i}^n - m_{t,j}^n\right)^2} \\ &\text{s.t. } m_{t,i}^n = \text{ flatten }\left(\mathrm{M}_{t,i}^l\right), m_{t,j}^n = \text{ flatten }\left(\mathrm{M}_{t,j}^l\right) \end{aligned} \tag{2}$$

where $\mathrm{M}_{t,i}^l$ represents the i-th feature map generated by the t-th image in the l-th layer, flatten$(\cdot)$ converts the feature map matrix into a vector. In order to obtain more accurate

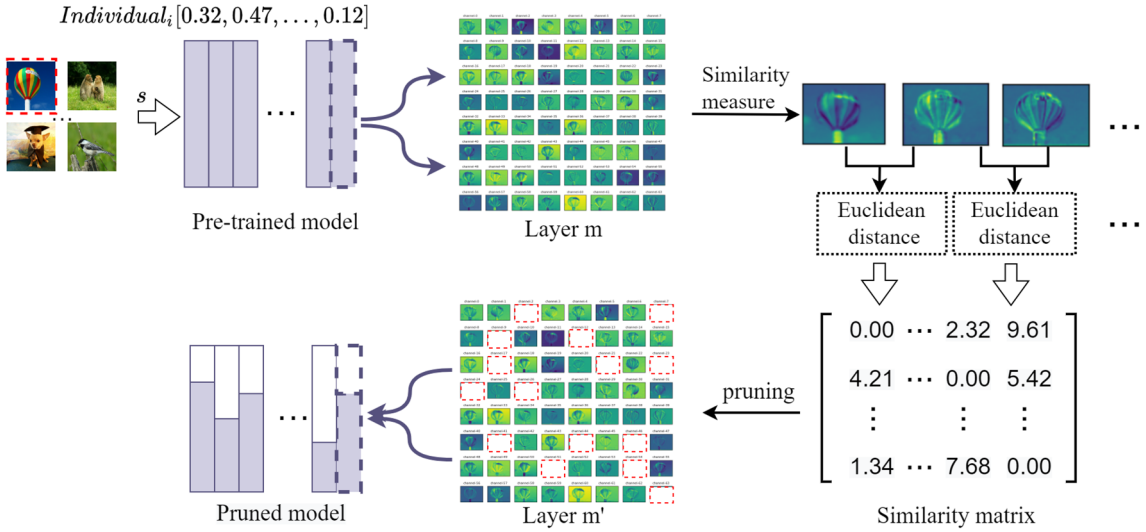

Fig. 1 For the evaluation of the pruned models, filters are selected by calculating the feature map average similarity of s sample. For the m-th layer of the network, we calculate the similarity matrix for all output feature maps and remove the filter corresponding to the feature map with the highest similarity based on the filter pruning ratio in the current layer structure vector

similarity, we uses the average Euclidean distance between the feature maps of s samples, which reduces the impact of sample bias. The smaller value of $dist(\mathrm{M}_i^l, \mathrm{M}_j^l)$ means that the two feature maps are more similar, and the specific process is shown in Fig. 1.

3.3 Framework of automatic filter pruning algorithm(AFPruner)

The proposed algorithm consists of four steps, as illustrated in Fig. 2.The entire algorithm flow is shown in Algorithm 1.

- **Population initialization**: Initialize the pruning ratio vector set by reverse search, and each vector represents an individual in the population (i.e., a pruned model M*), and the elements of each vector represent the filter pruning ratio of one layer.
- **Population evolution**: Selection, crossover, mutation, and evaluation of individuals to obtain next generation populations.
- **Reverse operation**: At the end of each iteration, the two individuals with the lowest fitness value are selected to generate two reverse individuals, and these two individuals are replaced with the generated reverse individuals. This process increases group diversity and avoids the "prematurity" convergence.
- **Model fine-tuning**: After the iteration of the reverse search genetic algorithm is completed, the individuals with the highest fitness in the population are selected as the optimal pruning ratio combination R*, and then the pruning method described in Section 3.2 is used to obtain the pruned model M*. Finally, the pruned model is fine-tuned to restore accuracy.

Algorithm 1 Automatic Filter Pruning(AFPruner)

Input: Original model M, W, Population Size: N, crossover probability: P_c, mutation probability: P_m, max search cycles: T
Output: The optimal pruned model M*
Initialize the individual R according to Eq.4
Initialize the population P according to Eq.5
for t = 0:T **do**
  fitness = Fitness(P^t)
  add Top2{fitness {P^t }} to P^{t+1}
  $\mathrm{P}_1^t, \mathrm{P}_2^t, \mathrm{P}_3^t$ = Kmeans(P^t)
  i = 2
  while i < N **do**
    $\mathrm{R}_{n1}^t, \mathrm{R}_{n2}^t$ =Selection($\mathrm{P}_1^t, \mathrm{P}_2^t, \mathrm{P}_3^t$)
    $\mathrm{R}_{n1'}^t, \mathrm{R}_{n2'}^t$ = Crossover($\mathrm{R}_{n1}^t, \mathrm{R}_{n2}^t$)
    $\mathrm{R}_{n1''}^t, \mathrm{R}_{n2''}^t$ = Mutation($\mathrm{R}_{n1'}^t, \mathrm{R}_{n2'}^t$)
    fitness = Fitness($\mathrm{R}_{n1}^t, \mathrm{R}_{n2}^t, \mathrm{R}_{n1''}^t, \mathrm{R}_{n2''}^t$)
    add Top2{fitness{$\mathrm{R}_{n1}^t, \mathrm{R}_{n2}^t, \mathrm{R}_{n1''}^t, \mathrm{R}_{n2''}^t$ }} to P^{t+1}
    i = i+2
  end while
  Replace the Bottom2{fitness {P^t }} with reverse operation
end for
R*= max fitness{P^T}
M*= Pruning(R*, M, W)
M*= Finetune(M*, W)
return the optimal pruned model M*

Fig. 2 AFPruner algorithm framework. The optimal pruned model is obtained through genetic algorithm, and then fine-tune to recover the model accuracy

3.4 The optimal combination search of prune ratio

To address the optimization problem described in (1), we propose the reverse search genetic algorithm (RSGA) for searching the optimal combination of pruning ratio at each convolutional layer. In this section, we provide a detailed description of our algorithm.

Fitness Previous works[29, 31–34] use the classification accuracy of the network model to evaluate the fitness of individuals, without considering the relationship between model compression ratio and classification accuracy. Our algorithm uses a combination of model classification accuracy and model compression ratio as an individual fitness function to achieve the balance between classification accuracy and compression ratio. Specifically, the fitness calculation is defined as follows:

$$\begin{aligned} &\text{Fitness} = w_1\,\text{Acc} + w_2\frac{\text{F}_o-\text{F}_k}{\text{F}_o} + w_3\frac{\text{P}_o-\text{P}_k}{\text{P}_o} \\ &\text{s.t.}\ \textstyle\sum_{i=1}^{3} w_i = 1 \end{aligned} \tag{3}$$

where Acc denotes the classification accuracy of the individual, F_o and P_o are the Flops and Params of the original network, respectively, F_k and P_k are the Flops and Params retained by the individual, respectively. w_1, w_2, w_3 are dynamically adjustable parameters.

Initialization Assuming the population size is N, we first randomly generate N/2 individuals to form a subpopulation P_1, and then perform the reverse operation to the subpopulation P_1 for generating a subpopulation P_2. Subpopulations P_1 and P_2 are combined to form the population P, which is mathematically expressed as formula 4.

$$\left.\begin{aligned} &\text{R}_i = [r_1, r_2, r_3, \ldots, r_{\text{L}}] \\ &\text{R}'_i = \left[r'_1, r'_2, r'_3, \ldots, r'_{\text{L}}\right] \\ &\text{s.t}\ r'_i = a + b - r_i \end{aligned}\right\} \rightarrow \begin{aligned} \text{P}_1 &= \left[\text{R}_1, \text{R}_2, \ldots \text{R}_{\text{N}/2}\right] \\ \text{P}_2 &= \left[\text{R}'_1, \text{R}'_2, \cdots \text{R}'_{\text{N}/2}\right] \end{aligned} \tag{4}$$

where L is the number of convolutional layers in the network model. The pruning ratio of the i-th layer of the network, denoted by r_i, is uniformly distributed between (a, b), where $r_i \in [a, b)$. In this paper, the value range of r_i is $[0, 1)$. R_i represents an individual, i.e. a filter pruning ratio vector. Using the aforementioned reverse initialization, the initial population can be better distributed in the search space, and N structure vectors as shown in Fig. 2(a) can be obtained. We use t to denote the generation of population evolution so that the initial population P can be expressed as:

$$\text{P} = \left\{\text{P}_1^0, \text{P}_2^0\right\} = \left[\text{R}_1, \text{R}_2, \ldots, \text{R}_{\text{N}/2}, \text{R}'_1, \text{R}'_2, \cdots \text{R}'_{\text{N}/2}\right] \tag{5}$$

Selection To maintain the diversity of individuals in the population, we first cluster the individuals into three subgroups using Euclidean distance by the K-means clustering algorithm. Then, individuals are selected for crossover and mutation operations from different subgroups in each generation, and the elitist preservation strategy is adopted to directly transfer the top two individuals with the highest fitness to the next generation. The specific process consists of three steps.

(1) The population P^t is clustered into three subgroups P_1^t, P_2^t, and P_3^t.
(2) Two different subgroups are randomly selected.
(3) The roulette wheel selection algorithm is used to select an individual from each of the two subpopulations, which constitutes a pair of individuals that are used for subsequent evolution operations, where the probability of individuals being selected is proportional to their fitness value. Assuming that the number of individuals in the subgroup is m, for each individual R_i in the subgroup P_j^t, $j = 1, 2, 3$, the selection probability s_i is calculated as follows:

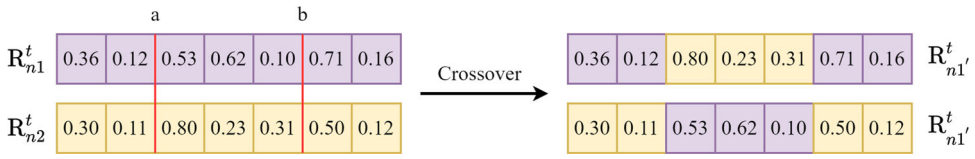

Fig. 3 Crossover process

$$s_i = \frac{f_i}{\sum_{k=1}^{m} f_k}, i = 1, 2, \ldots, m \tag{6}$$

Crossover We utilize the two-point crossover operation. Firstly, two crossover points are randomly selected based on the crossover probability. Then, the genes between the selected points of the two individuals are exchanged, generating two new individuals. The operation is illustrated in Fig. 3.

Mutation We employ the two-point mutation operation. After crossover, two mutated genes are randomly selected for each individual according to the mutation probability, and two mutation genes are chosen to be reinitialized. The mutation process is shown in Fig. 4.

Reverse operation To maintain the population diversity and avoid the "prematurity" convergence, we introduce the reverse operation in the genetic algorithm. At the end of each iteration, the two individuals with the lowest fitness are selected for the reverse operation, which generates two new individuals to replace the original two individuals in the population. By incorporating this reverse operation, the genetic algorithm can maintain population diversity to a certain extent, which is beneficial for algorithm convergence.

4 Experiments

To validate the effectiveness of our proposed method, we conduct experiments using state-of-the-art network models(VGGNet, ResNet) and compare with other algorithms on multiple datasets.

4.1 Datasets and experimental setting

We conduct experiments on publicly available datasets, including CIFAR-10, CIFAR-100, and ILSVRC-2012. The CIFAR-10 contains 60,000 (50,000 training images and 10,000 testing images) 32×32 color images in 10 different classes, with 6,000 images in each category. The CIFAR-100 is similar to CIFAR-10, except it has 100 classes, each containing 600 images. ILSVRC-2012 is one of the most commonly used subsets in the ImageNet, which contains 1,000 classes with 1.28 million training images and 50k validation images.

In experiments, we use Stochastic Gradient Descent(SGD) for fine-tuning with a momentum of 0.9. On CIFAR-10 and CIFAR-100, the initial learning rate is 0.01, the batch size is 128, and the weight decay is set to 5e-3. The final model is fine-tuned for 160 epochs and divided by 10 every 50 epochs. On ILSVRC-2012, batch size is set to 256, the weight decay is set to le-4 and 100 epochs are given for fine-tuning. The initial learning rate is set to 0.1 and divided by 10 every 30 epochs. In the process of evolutionary search, the pruned models are fine-tuned for 2 epochs to get more accurate classification accuracy. In the Algorithm 1, $\mathrm{N} = 10$, $\mathrm{T} = 10$, $\mathrm{P}_c = 0.7$, and $\mathrm{P}_m = 0.1$.

4.2 *w* parameter analysis

In our method, w_1, w_2 and w_3 are dynamically adjustable hyperparameters. w_1, w_2 and w_3 denote the percentage of accuracy, Flops compression ratio and parameter compression ratio of the pruned model, respectively. On CIFAR-10, we analyze the effect of these hyperparameters for the pruning results based on the pruning process of the ResNet56 model.

The results are shown in Table 1.

(1) From the first 5 rows of the table, it is clear that when the value of w_1 is tiny, the accuracy of the pruned model is low, but the compression ratio is high. As the value of w_1 increases, the accuracy of the pruned model becomes higher and higher, but the compression ratio becomes lower and lower. It can be seen that when w_1=0.7, the accuracy and compression ratio have achieved better results, so we initially choose the value of w_1 as 0.7.

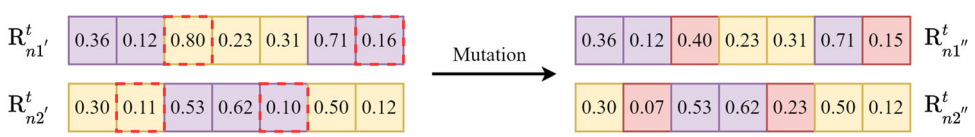

Fig. 4 Mutation process

Table 1 The influence of different w values for the compression ratio and accuracy of ResNet56 on CIFAR-10

Parameter setting	Pruned Acc/%	Flops.drop/%	Params.drop/%
w1=0.5, w2=w3=0.25	92.08	73.13	76.61
w1=0.6, w2=w3=0.20	92.92	63.03	73.19
w1=0.7, w2=w3=0.15	**93.34**	**50.81**	**59.15**
w1=0.8, w2=w3=0.10	93.48	41.23	37.32
w1=0.9, w2=w3=0.05	93.57	39.53	41.61
w1=0.72, w2=w3=0.14	93.31	50.76	58.32
w1=0.74, w2=w3=0.13	93.37	48.44	53.36
w1=0.76, w2=w3=0.12	93.41	45.89	47.92
w1=0.78, w2=w3=0.11	93.43	40.21	39.76
w1=0.7, w2=0.05, w3=0.25	93.23	48.57	56.95
w1=0.7, w2=0.10, w3=0.20	93.29	50.76	58.79
w1=0.7, w2=0.15, w3=0.15	93.34	50.81	59.15
w1=0.7, w2=0.20, w3=0.10	**93.45**	**53.67**	**52.64**
w1=0.7, w2=0.25, w3=0.05	93.39	57.35	46.14

(2) In order to seek for more optimal results, we make a more detailed analysis of w_1 within the range of [0.7-0.8]. The results are shown in the middle 4 rows of the table. It can be seen that as the value of w_1 increases, the accuracy of the pruned model is slightly improved, but the Flops and Parameters compression ratio decreases a lot. The reason is that the model accuracy is given more attention, and the compression ratio are ignored. Therefore, we finally choose $w_1 = 0.7$ for the subsequent analysis on w_2 and w_3.

(3) More attention is paid to the FLOPs compression ratio for network pruning, this is because it directly affects the computational speed of the model. From the last 5 rows of the table, it can be seen that $w_1 = 0.7$, $w_2 = 0.2$ and $w_3 = 0.1$ obtain the best model accuracy and compression ratio.

4.3 Algorithm analysis

4.3.1 Performance analysis

To validate the performance of the algorithm, we statistically analyze the individuals produced by different models on CIFAR-10. The reverse search genetic algorithm tends to retain good genes from parents, where crossover and mutation are central in the evolution process, i.e., a better individual is more likely to produce a better individual through mutation or crossover. Figure 5(a) supports our view that the average fitness of all the individuals is generally higher than that of the previous generation, which suggests that the overall quality of the individual has been improved through crossover and mutation. In addition, it is clear that the distance between individuals is larger after the reverse operation, including the average, maximum and minimum distances, with the solid line represents before the reversal operation and the dashed line represents after the reversal operation

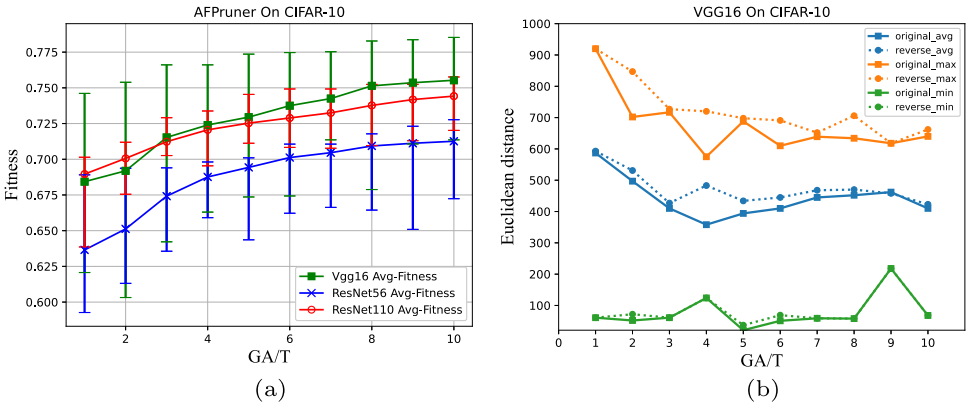

Fig. 5 (a) shows the average fitness over all individuals with respect to the generation number. The bars indicate the highest and lowest fitness in the corresponding generation. (b) shows the euclidean distances between individuals before and after the reversal operation in each gen-

Table 2 Accuracy and pruning ratio on CIFAR-10

Model	Top-1/Drop	Flops/Pruned	Params/Pruned
VGG16 Baseline	93.02/0.00	317.59M/0.00%	14.73M/0.00%
AFPruner(-)	93.67/-0.65	128.18M/59.64%	4.07M/72.36%
AFPruner(∗)	93.55/-0.53	114.52M/63.94%	3.52M/76.12%
AFPruner(+)	93.55/-0.53	102.42M/67.75%	2.57M/82.56%
ResNet56 Baseline	93.26/0.00	127.62M/0.00%	0.85M/0.00%
AFPruner(-)	93.53/-0.27	67.68M/46.97%	0.40M/52.47%
AFPruner(∗)	93.45/-0.19	60.81M/52.35%	0.39M/54.42%
AFPruner(+)	93.39/-0.13	54.42M/57.35%	0.36M/58.21%
ResNet110 Baseline	93.50/0.00	257.09M/0.00%	1.73M/0.00%
AFPruner(-)	94.25/-0.75	129.75M/49.53%	0.89M/48.36%
AFPruner(∗)	94.08/-0.58	116.59M/54.65%	0.79M/54.27%
AFPruner(+)	93.95/-0.45	106.44M/58.60%	0.69M/60.07%

ation as seen in Fig. 5(b). This further shows that the reverse operation increases the diversity of individuals, which is beneficial to discover better individuals.

4.3.2 Robustness analysis

In order to rule out randomness in the experimental results, we use statistical tests to record the results. Specifically, five experiments are conducted on different datasets for different models, and then average the results of the five experiments. The experimental results are shown in Table 2, 3 and 4, where "Model" is the model used in the experiment, "-" is the result with the smallest pruning ratio , "+" is the result with the largest pruning ratio, and "∗" is the result averaged over 5 trials. Four widely-used metrics are reported here, including accuracy, Flops, Params, and pruning ratio, where "Pruned" is the pruning ratio. For CIFAR-10 and CIFAR-100, Top-1 accuracy of pruned models are provided. For ILSVRC-2012, both Top-1 and Top-5 accuracies are reported.

From the table, it can be seen that after compressing the original model with an appropriate pruning ratio, the accuracy of the pruned models are all slightly better than the baseline. This is because common convolutional neural networks exhibit different degrees of redundancy, and removing the redundant parameters not only does not negatively affect the original network, but also improves the generalization performance of the model. AFPruner achieves nearly half of the compression ratio on both FLOPs and Params for different models, where VGG16 maximally compresses 67.75% FLOPs and 82.56% Params. This indicates that the VGG16 has greater parameter redundancy. In addition, from the results of multiple experiments, the accuracy of these pruning models are approximately the same, and the average results reflect the stability of our algorithm. Based on the above analysis, our pruning algorithm can effectively achieve the compression and acceleration of CNNs.

4.4 Compared with other methods

Since there are few pruning methods report results on CIFAR-100, we mainly compare our pruning results with other filter pruning methods on CIFAR-10 and ILSVRC-2012. In Section 4.4.1, the comparison of pruning ratio and accuracy are reported in Tables 5 and 6. Among the compared methods, ABCPruner [29], AACP [34] and SST [32] are all population intelligence optimization algorithms, AMC [25], HRank [9], FPGM [16], CSHE [21], Li et al.[18], FPSC [22] and CLR-RNF [19] are the state-of-the-art filter pruning methods. The

Table 3 Accuracy and pruning ratio on CIFAR-100

Model	Top-1/Drop	FLOPs/Pruned	Params/Pruned
ResNet56 Baseline	69.26/0.00	127.62M/0.00%	0.85M/0.00%
AFPruner(-)	70.07/-0.81	63.48M/50.26%	0.42M/50.36%
AFPruner(∗)	70.00/-0.74	59.74M/53.19%	0.40M/52.46%
AFPruner(+)	69.96/-0.70	57.20M/55.18%	0.41M/51.69%
ResNet110 Baseline	72.02/0.00	257.09M/0.00%	1.73M/0.00%
AFPruner(-)	72.06/-0.04	134.38M/47.73%	0.78M/54.68%
AFPruner(∗)	72.01/-0.01	120.45M/53.15%	0.76M/56.27%
AFPruner(+)	71.98/0.04	106.87M/58.43%	0.70M/59.64%

Table 4 Accuracy and pruning ratio on ILSVRC-2012

Model	Top-1/Drop	Top-5/Drop	FLOPs/Pruned	Params/Pruned
ResNet18 Baseline	69.66/0.00	89.08/0.00	1824.52M/0.00%	11.69M/0.00%
AFPruner(-)	68.06/1.60	88.51/0.58	1000.75M/45.15%	6.55M/43.96%
AFPruner(∗)	68.01/1.65	88.07/1.02	944.74M/48.22%	6.37M/45.47%
AFPruner(+)	67.29/2.37	87.35/1.74	879.05M/51.82%	5.67M/51.49%
ResNet34 Baseline	73.28/0.00	91.45/0.00	3679.23M/0.00%	21.90M/0.00%
AFPruner(-)	72.36/0.92	90.83/0.62	1959.56M/46.74%	10.57M/51.73%
AFPruner(∗)	72.17/1.11	90.60/0.85	1891.49M/48.59%	10.11M/53.82%
AFPruner(+)	71.98/1.30	90.09/1.36	1781.48M/51.58%	9.63M/56.05%
ResNet50 Baseline	76.01/0.00	92.96/0.00	4135.70M/0.00%	25.56M/0.00%
AFPruner(-)	74.87/1.14	92.31/0.65	2369.76M/42.70%	13.43M/47.46%
AFPruner(∗)	74.23/1.78	91.89/1.07	2060.41M/50.18%	12.00M/53.05%
AFPruner(+)	73.89/2.12	91.69/1.27	1887.53M/54.36%	10.68M/58.23%

results of these comparing methods are reported according to the original article. It is worth noting that the average values of the statistical tests are used for comparison here, not the optimal values. In Section 4.4.2, we further compare the time complexity of the algorithms in Table 7.

4.4.1 Accuracy and pruning ratio

Comparison on CIFAR-10 For ResNet56, it can be seen from Table 5 that most of the methods decrease the classification accuracy after pruning. In contrast, our method exceeds

Table 5 Performance comparison of ResNet56/110 on CIFAR-10

Method	Baseline	Top-1/Drop	FLOPs/Pruned	Params/Pruned
results with ResNet56				
Azadeh[12]	93.59	93.19/0.40	76.44M/40.10%	-
Li et al.[18]	93.26	92.42/0.84	62.95M/49.90%	0.48M/44.00%
AACP[34]	93.10	92.82/0.28	62.72M/50.00%	-
HRank[9]	93.26	93.17/0.09	62.72M/50.00%	0.49M/42.40%
AMC[25]	92.80	91.90/0.90	62.72M/50.00%	-
CSHE[21]	93.26	93.07/0.19	62.72M/50.00%	0.49M/42.40%
SST[32]	93.57	93.28/0.29	62.39M/51.11%	-
SFP [10]	93.59	93.35/0.24	59.40M/52.60%	-
FPGM[16]	93.59	92.89/0.70	59.40M/52.60%	-
FPSC[22]	93.59	93.14/0.45	59.40M/52.60%	0.39M/54.20%
ABCPruner[29]	93.26	93.23/0.03	58.54M/54.13%	0.39M/54.20%
CLR-RNF[19]	93.26	93.27/-0.01	54.00M/57.30%	0.38M/55.50%
AFPruner(Ours)∗	**93.26**	**93.45/-0.19**	**60.81M/52.35%**	**0.39M/54.42%**
results with ResNet110				
CSHE[21]	93.50	93.44/0.06	153.00M/39.50%	1.04M/38.70%
AACP[34]	93.30	93.76/-0.46	154.25M/40.00%	-
SFP[10]	93.68	93.86/-0.18	152.20M/40.80%	-
Li et al.[18]	93.50	92.99/0.51	134.88M/46.70%	0.74M/57.10%
Azadeh[12]	93.68	93.50/0.18	136.00M/47.10%	-
FPGM[16]	93.68	93.85/-0.17	122.63M/52.30%	-
HRank[9]	93.50	93.36/0.14	105.70M/58.20%	0.70M/59.20%
ABCPruner[29]	93.50	93.58/-0.08	89.87M/65.04%	0.56M/67.41%
CLR-RNF[19]	93.57	93.71/-0.14	86.80M/66.00%	0.53M/59.10%
AFPruner(Ours)∗	**93.50**	**94.08/-0.58**	**116.59M/54.65%**	**0.79M/54.27%**

Table 6 Performance comparison of ResNet18/34/50 on ILSVRC-2012

Method	Top-1/Drop	Top-5/Drop	FLOPs/Pruned	Params/Pruned
results with ResNet18				
Azadeh [12]	67.82/2.46	88.18/1.45	1454.14M/20.30%	-
SFP [10]	67.10/3.18	87.78/1.85	1058.70M/41.80%	-
ASFP [11]	67.41/2.82	87.89/1.62	1058.70M/41.80%	-
FPGM [16]	67.78/2.50	88.01/1.62	1058.70M/41.80%	-
FPSC [22]	67.43/2.85	87.99/1.64	1031.26M/43.30%	6.52M/44.20%
ABCPruner [29]	67.28/2.38	87.67/1.41	1005.71M/44.88%	6.60M/43.55%
AFPruner(Ours)∗	**68.01/1.65**	**88.07/1.02**	**944.74M/48.22%**	**6.37M/45.47%**
results with ResNet34				
Azadeh [12]	72.25/1.67	90.56/1.06	2899.23M/21.20%	-
SFP [10]	71.83/2.09	90.33/1.29	2170.77M /41.10%	-
ASFP [11]	71.72/2.20	90.65/0.97	2170.77M /41.10%	-
FPGM [16]	71.19 /2.13	90.70/0.92	2170.77M /41.10%	-
ABCPruner [29]	70.98/2.30	90.05/1.40	2170.77M/41.00%	10.12M/53.58%
AFPruner(Ours)∗	**72.17/1.11**	**90.60/0.85**	**1891.49M/48.59%**	**10.11M/53.82%**
results with ResNet50				
Azadeh	75.26/0.89	92.37/0.50	3068.69M/25.80%	-
SFP [10]	74.61/1.54	92.06/0.47	2406.98M/41.80%	-
ASFP [11]	74.88/1.27	92.39/0.48	2406.98M /41.80%	-
HRank [9]	74.98/1.17	92.33/0.54	2325.92M/43.76%	16.15M/36.67%
CHSE [21]	72.25/3.40	-	1985.23M/51.50%	18.23M/27.80%
FPGM [16]	74.13/2.02	91.94/0.93	1923.10M/53.50%	-
ABCPruner [29]	73.86/2.15	91.69/1.27	1890.60M/54.29%	11.75M/54.02%
AFPruner(Ours)∗	**74.23/1.78**	**91.89/1.07**	**2060.41M/50.18%**	**12.00M/53.05%**

Table 7 Time complexity on CIFAR-10

Method	Top-1/Drop	FLOPs/Pruned	Params/Pruned	Time Cost
result with VGG16				
Baseline	93.02/0.00	317.59M/0.00%	14.73M/0.00%	-
ABCPruner(ABC) [29]	92.90/0.12	106.25 M/66.20%	4.72 M/67.98%	2h31min
ACP(PSO) [31]	92.87/0.15	99.41 M/68.37%	4.35 M/70.43%	2h04min
Ours(RSGA)	**93.55/-0.53**	**101.46M/67.75%**	**2.57M/82.56%**	**1h36min**
result with ResNet56				
Baseline	93.26/0.00	127.62M/0.00%	0.85M/0.00%	-
ABCPruner(ABC) [29]	93.01/0.25	59.63 M/53.28%	0.40 M/52.96%	3h31min
ACP(PSO) [31]	92.98/0.28	59.03 M/53.74%	0.40 M/53.24%	2h10min
Ours(RSGA)	**93.41/-0.15**	**57.68M/54.79%**	**0.39M/54.33%**	**1h27min**
result with ResNet110				
Baseline	93.50/0.00	257.09M/0.00%	1.73M/0.00%	-
ABCPruner(ABC) [29]	93.86/-0.36	107.39 M/58.23%	0.73 M/58.01%	6h11min
ACP(PSO) [31]	93.73/-0.23	110.51 M/ 57.01%	0.86 M/50.49%	4h46min
Ours(RSGA)	**93.95/-0.45**	**106.44M/58.60%**	**0.69M/60.07%**	**2h45min**

Table 8 The influence of reverse operation

Reverse operation	ResNet56		ResNet110	
	Pruned Acc/%	Flops.drop/%	Pruned Acc/%	Flops.drop/%
✘	92.72	52.97	93.78	49.73
✔	**93.45**	**51.64**	**94.15**	**53.67**

Where "✘" means the operation is not used and "✔" means the operation is used

the baseline. Compared with ABCPruner [29], AMC [25], SST [32] and AACP [34], which also search for pruning ratio, our method outperforms these algorithms with similar pruning ratio. Compared with the newly released algorithms FPSC [22] and CSHE [21], our method improves accuracy by 0.19% when considering the pruning ratio of Flops (52.35% vs 52.60% vs 50.00%) and Params (54.42% vs 54.20% vs 42.40%), while the accuracy of FPSC [22] and CSHE [21] decreases by 0.45% and 0.15%, respectively.

For ResNet110, even removed 54.27% Params and 54.65% Flops, the accuracy of the pruned model is 0.58% higher than the baseline, which is slightly better than other methods. Among the compared methods, AFPruner outperforms CHSE [21] in all respects, CSHE [21] has the lowest pruning ratio and the classification accuracy of the pruned model decreased by 0.06%. Although the accuracy of ABCPruner [29] and CLR-RNF [19] is still improved compared to the baseline at high compression ratio, the improvement is not significant, which further illustrates that AFPruner can obtain the optimal pruning ratio.

Comparison on ILSVRC-2012 Table 6 shows the comparison of ResNet18/34/50 results on ILSVRC-2012. For ResNet18, AFPruner achieves the best results in terms of Flops, Params and Top-1 accuracy. In particular, compared with Azadeh [12], which also adopts genetic algorithm, our method compresses 48.22% of Flops and achieves 68.01% of Top-1 accuracy, while Azadeh [12] compresses 20.30% of Flops and achieves only 67.82% of Top-1 accuracy. For both ResNet34 and ResNet50, AFPruner still slightly outperforms existing methods with similar pruning ratio. Although Azadeh [12] slightly outperforms our method in terms of accuracy, its Flops compression ratio is only half of ours.

The comparisons of the experimental results on different datasets show that AFPruner achieves better performance in both the compression ratio of Flops and Params after pruning. Even though the accuracy metric is slightly lower than individual pruning algorithms on some models, its pruning ratio is higher than these algorithms, which indicates that AFPruner better considers the structural integrity and global correlation of CNN layers in the filter pruning process, and thus can better prune the redundant structure of the model. The experimental results show the effectiveness and superiority of the AFPruner.

4.4.2 Time complexity

We further compare the effective running time, including search time and fine-tuning time of the algorithms. Specifically, we focus on comparing the algorithms ABCPruner [29] and ACP [31], which all also use a population intelligence optimization approach to search pruning ratio. The results are reported in Table 7, where ABCPruner [29] employs an artificial bee colony(**ABC**) algorithm and ACP [31] employs particle swarm optimization(**PSO**) algorithm. To ensure the fairness of the comparison results, we use the same baseline for different algorithms on CIFAR-10. During the experiments, the same parameter configurations were set up for a total of 10 rounds of searching, with a final fine-tuning of 160 epochs, and all the experiments are performed on Tesla V100-SXM2-32GB. (Note: Because the original paper did not have running time, the data in Table 7 are not from the original paper, and are reproduced by ourselves using the same parameters).

As can be seen in Table 7, for the same model, our algorithm takes less time to run, and the final fine-tuned accuracy is higher.

Table 9 Cluster analysis of ResNet56 on CIFAR-10

clustering	reverse operation	Pruned Acc/%	Flops.drop/%
✘	✘	92.21	51.27
✘	✔	92.98	50.97
✔	✘	93.22	46.97
✔	✔	**93.45**	**51.64**

Table 10 Comparison of different similarity calculations on CIFAR-10

Model	Type	Pruned Acc/%	Pruned Flops/%
ResNet56	cos	93.32	51.64
	euclidean	**93.45**	**51.64**
ResNet110	cos	94.05	53.67
	euclidean	**94.15**	**53.67**
VGG16	cos	93.34	67.75
	euclidean	**93.55**	**67.75**

4.5 Ablation study

Reverse operation To demonstrate the effectiveness of the reverse operation during population iteration, we compare the improved genetic algorithm with the traditional genetic algorithm in Table 8. The result of the improved genetic algorithm is better because the reverse operation can increase the diversity of the population, which is more conducive to convergence.

Cluster analysis We take ResNet56 on CIFAR-10 as an example and analyze the influence of introducing clusters in Table 9. Intuitively, the results of adding clusters are better, indicating that the clustering operation can further improve the population diversity of the genetic algorithm and alleviate the "prematurity" convergence, thus effectively improving the convergence.

Similarity calculation In order to verify the effect of different similarity methods on the results, we use cosine distance instead of euclidean distance to calculate the similarity between feature maps. On CIFAR-10, the same individual is trained in different distances, and other parameters are consistent during the training process. As can be seen from Table 10, the results of euclidean distances outperform cosine distances in all three models. In addition, in the previous work FPSC [22], it is also proved that the euclidean distance is better than the cosine distance.

4.6 Visualization and analysis

We visualize some individuals generated during the search process of ResNet56 on CIFAR-10. Figure 6 shows the effect of hierarchical pruning on these individuals, and Table 11 shows the differences between them.

From Fig. 6, it can be found that the filter pruning ratio varies for different individuals in different layers. It also is indicated that our pruning algorithm adjusts the appropriate pruning ratio based on the pruning sensitivity of each layer during the search process. Additionally, for residual networks such as ResNet, excessive pruning of filters in the early stages leads to a decrease in accuracy, and the accuracy of pruned models is usually low when the residual blocks at the beginning or end of each stage (as shown in the figure 9/10/18/19 layers) are too pruned. Specifically:

(1) As shown in Table 11, the classification accuracy of individuals (a), (c), and (d) after fine-tuning is relatively low. Comparing with Fig. 6, it can be seen that individual (a) prunes too many filters in the early stages and the 9-th

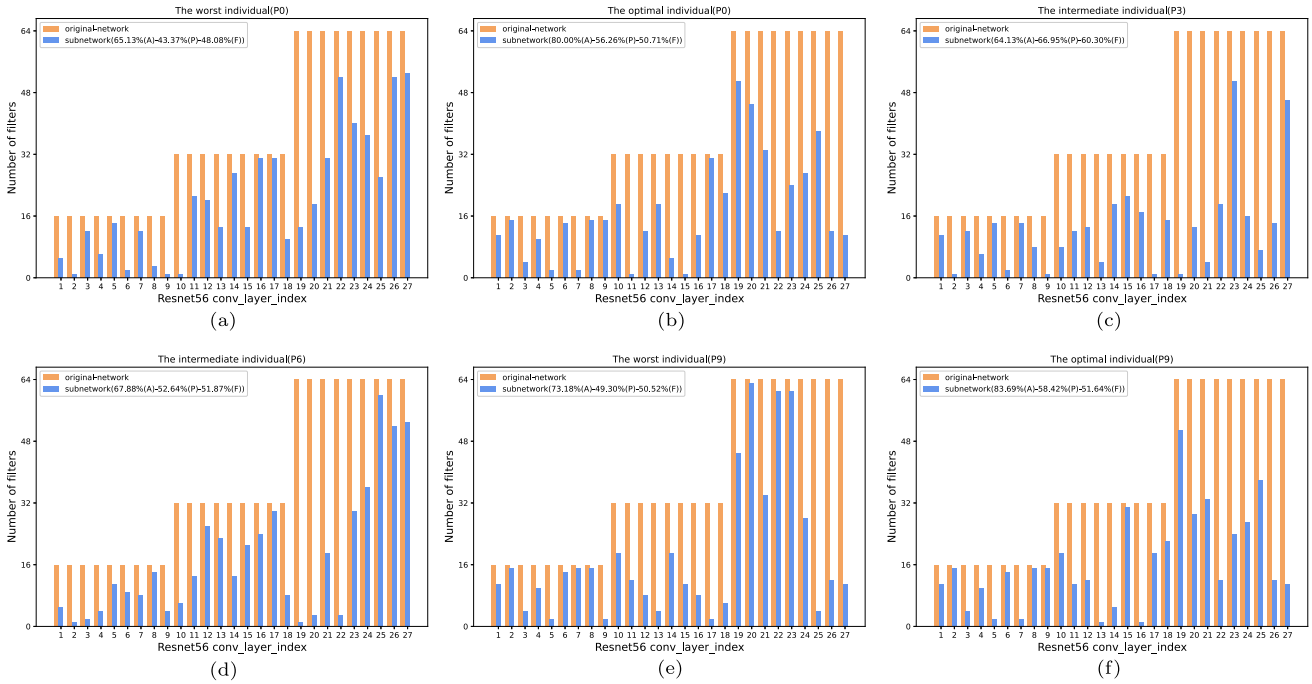

Fig. 6 Some individuals are in the iterative pruning process of ResNet56 on CIFAR-10. Red represents the number of filters in each layer of the original network, and purple represents the number of filters in each layer of the sub-network after pruning. A represents the test classification accuracy after two rounds of fine-tuning, P represents the param compression ratio, and F represents the Flops compression ratio. (a) is the worst individual in P_0; (b) is the optimal individual in P_0; (c) and (d) are the intermediate individuals in P_3 and P_6, respectively; (e) is the worst individual in P_9 ; (f) is the optimal individual in P_9 (that is, the optimal individual finally obtained)

Table 11 Comparison of some individuals generated during the search on CIFAR-10 for ResNet56

Individual	Acc/%	Param.drop/%	Flops.drop/%
The worst individual in P_0(a)	65.13	43.37	48.08
The optimal individual in P_0(b)	80.00	50.71	56.26
The intermediate individual in P_3(c)	64.13	66.95	60.30
The intermediate individual in P_6(d)	67.88	52.64	51.87
The worst individual in P_9(e)	73.18	49.30	50.52
The optimal individual in P_9(f)	83.69	58.42	51.64

and 10-th layers; the 10-th and 19-th layers of individual (c) are over-pruned; the 19-th layer of individual (d) is over-pruned;

(2) For the two contemporary optimal individuals (b) and (f), it can be seen that the 9/10/18/19-th layers are not over-pruned, which results in higher testing accuracy at the appropriate compression ratio.

(3) Furthermore, compared with the worst individual (a) in the initial population, the worst individual (e) in the final generation has a larger Flops and Param compression ratio, and its testing accuracy after fine-tuning is even better. This is because individual (e) preserves more filters in 10-th and 19-th layers, and evolves towards a better direction in seeking a balance between compression ratio and accuracy.

The reason for the above phenomenon may be that the first residual block in each stage downsamples the features and requires more filters for feature extraction to avoid information loss. In the subsequent pruning process, specific processing can be performed for these layers to guide network pruning more effectively.

5 Conclusion

This paper introduces a novel algorithm named automatic filter pruning algorithm via feature map average similarity and reverse search genetic algorithm(AFPruner) for automatically searching the optimal combination of pruning ratio in convolutional neural networks. We use reverse search and clustering to enhance population diversity and accelerate convergence, and utilize the average Euclidean distance between feature maps to calculate their similarity during the pruning process, thereby selecting filters. The experiment results show that AFPruner has excellent performance, achieves a high pruning ratio and model accuracy, and outperforms most current pruning algorithms. In addition, we also found that over-pruning has a significant impact on accuracy during feature downsampling, which suggests that specific treatment of these layers can be applied in subsequent pruning processes. In future research, we will consider the following methods: optimizing the search space for network pruning, leveraging faster optimization methods, and extending model pruning to additional tasks, such as object detection.

Acknowledgements This work was partially supported by Priority Academic Program Development of Jiangsu Higher Education Institutions(PAPD), Collaborative Innovation Center of Novel Software Technology and Industrialization.

Author Contributions Yifan Xue: Conceptualization, Methodology, Software, Investigation, Writing - original draft. Wangshu Yao: Conceptualization, Writing - review & editing, Project administration, Funding acquisition. Siyuan Peng: Validation, Resources, Investigation. Shiyou Yao: Visualization.

Data Availability Statements The datasets generated during and/or analysed during the current study are available in http://www.cs.toronto.edu/~kriz/cifar.html and https://image-net.org/.

Declarations

Competing Interests The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Krizhevsky A, Sutskever I, Hinton GE (2017) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
2. Ren S, He K, Girshick R, Sun J (2017) Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Trans Pattern Anal Mach Intell 39(6):1137–1149
3. Noh H, Hong S, Han B (2015) Learning deconvolution network for semantic segmentation. In: Proceedings of the IEEE International conference on computer vision, pp 1520–1528
4. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations (ICLR), pp 1–14
5. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
6. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708

7. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. In: International conference on learning representations
8. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision, pp 2736–2744
9. Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L (2020) Hrank: Filter pruning using high-rank feature map. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1529–1538
10. He Y, Kang G, Dong X, Fu Y, Yang Y (2018) Soft filter pruning for accelerating deep convolutional neural networks. In: International joint conference on artificial intelligence
11. He Y, Dong X, Kang G, Fu Y, Yan C, Yang Y (2020) Asymptotic soft filter pruning for deep convolutional neural networks. IEEE Trans Cybern 50(8):3594–3604
12. Famili A, Lao Y (2022) Genetic-based joint dynamic pruning and learning algorithm to boost dnn performance. In: 2022 26th International conference on pattern recognition (ICPR), pp 2100–2106
13. Yu R, Li A, Chen C-F, Lai J-H, Morariu VI, Han X, Gao M, Lin C-Y, Davis LS (2018) Nisp: Pruning networks using neuron importance score propagation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9194–9203
14. Hu J, Shen L, Albanie S, Sun G, Wu E (2020) Squeeze-and-excitation networks. IEEE Trans Pattern Anal Mach Intell 42(8):2011–2023
15. Cheng Y, Wang X, Xie X, Li W, Peng S (2022) Channel pruning guided by global channel relation. Applied Intelligence, pp 1–12
16. He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4340–4349
17. Sawant SS, Bauer J, Erick FX, Ingaleshwar S, Holzer N, Ramming A, Lang E, Götz T (2022) An optimal-score-based filter pruning for deep convolutional neural networks. Appl Intell pp 1–23
18. Li J, Shao H, Zhai S, Jiang Y, Deng X (2023) A graphical approach for filter pruning by exploring the similarity relation between feature maps. Pattern Recogn Lett 166:69–75
19. Lin M, Cao L, Zhang Y, Shao L, Lin C-W, Ji R (2022) Pruning networks with cross-layer ranking & k-reciprocal nearest filters. IEEE Transactions on neural networks and learning systems, pp 1–10
20. Ayinde BO, Inanc T, Zurada JM (2019) Redundant feature pruning for accelerated inference in deep neural networks. Neural Netw 118:148–158
21. Shao M, Dai J, Wang R, Kuang J, Zuo W (2021) Cshe: network pruning by using cluster similarity and matrix eigenvalues. Int J Mach Learn Cybern 13:371–382
22. Song K, Yao W, Zhu X (2022) Filter pruning via similarity clustering for deep convolutional neural networks. In: International conference on neural information processing. Springer, pp 88–99
23. Orseau L, Hutter M, Rivasplata O (2020) Logarithmic pruning is all you need. Adv Neural Inf Process Syst 33:2925–2934
24. Yang C, Liu H (2022) Channel pruning based on convolutional neural network sensitivity. Neurocomputing 507:97–106
25. He Y, Lin J, Liu Z, Wang H, Li L-J, Han S (2018) Amc: Automl for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision (ECCV), pp 784–800
26. Ning X, Zhao T, Li W, Lei P, Wang Y, Yang H (2020) Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In: Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III, Springer, pp 592–607
27. Gao S, Huang F, Pei J, Huang H (2020) Discrete model compression with resource constraint for deep neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1899–1908
28. Liu Z, Sun M, Zhou T, Huang G, Darrell T (2018) Rethinking the value of network pruning. In: International conference on learning representations
29. Lin M, Ji R, Zhang Y, Zhang B, Wu Y, Tian Y (2020) Channel pruning via automatic structure search. In: Proceedings of the international joint conference on artificial intelligence (IJCAI), pp 673–679
30. Liu Z, Mu H, Zhang X, Guo Z, Yang X, Cheng K-T, Sun J (2019) Metapruning: Meta learning for automatic neural network channel pruning. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 3296–3305
31. Chang J, Lu Y, Xue P, Xu Y, Wei Z (2022) Automatic channel pruning via clustering and swarm intelligence optimization for cnn. Appl Intell pp 1–21
32. Lian Y, Peng P, Xu W (2021) Filter pruning via separation of sparsity search and model training. Neurocomputing 462:185–194
33. Tmamna J, Ayed EB, Ayed MB (2021) Neural network pruning based on improved constrained particle swarm optimization. In: Neural information processing: 28th international conference, ICONIP 2021, Sanur, Bali, Indonesia, December 8–12, 2021, Proceedings, Part VI 28, Springer, pp 315–322
34. Lin L, Chen S, Yang Y, Guo Z (2022) Aacp: Model compression by accurate and automatic channel pruning. In: 2022 26th International conference on pattern recognition (ICPR), IEEE, pp 2049–2055
35. Wang Z, Li F, Shi G, Xie X, Wang F (2020) Network pruning using sparse learning and genetic algorithm. Neurocomputing 404:247–256
36. He Y, Ding Y, Liu P, Zhu L, Zhang H, Yang Y (2020) Learning filter pruning criteria for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2009–2018
37. Huang K, Wu S, Li F, Yang C, Gui W (2022) Fault diagnosis of hydraulic systems based on deep learning model with multirate data samples. IEEE Transactions on Neural Networks and Learning Systems 33(11):6789–6801
38. Li H, Ma C, Xu W, Liu X (2020) Feature statistics guided efficient filter pruning. In: Proceedings of the international joint conference on artificial intelligence (IJCAI)

Yifan Xue is pursuing a master's degree in software engineering at Soochow University. Her research interests include deep neural network compression and neural architecture search.

Siyuan Peng is pursuing a master's degree in software engineering at Soochow University. His research interests include computer vision and deep neural network compression.

Wangshu Yao (Member, IEEE) received the PhD degree in Department of Computer from Nanjing University, China, in 2005. He is currently an associate professor with School of Computer Science & Technology in Soochow University, Suzhou, Jiangsu, China. His main research interests include deep neural network compression, computer vision and machine learning.

Shiyou Yao is currently pursuing a master's degree in computer technology at Soochow University. His current research interests include neural architecture search and deep neural network compression.