Applied Soft Computing 113 (2021) 107860

Contents lists available at ScienceDirect

Applied Soft Computing

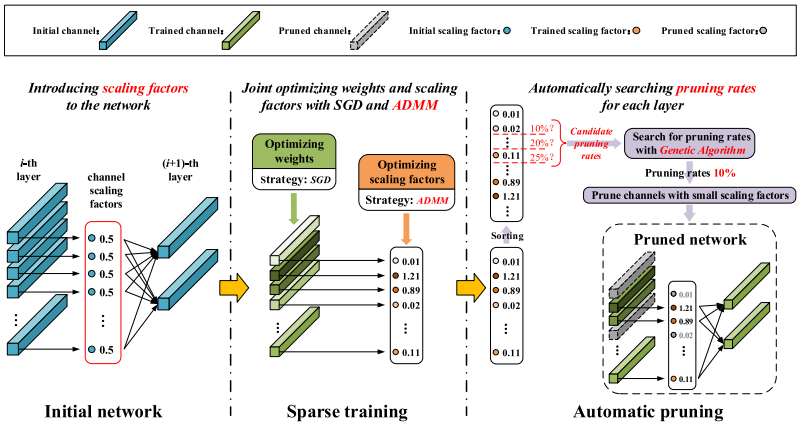
journal homepage: www.elsevier.com/locate/asoc

RFPruning: A retraining-free pruning method for accelerating convolutional neural networks

Zhenyu Wang, Xuemei Xie [*,1], Guangming Shi [2]

School of Artificial Intelligence, Xidian University, Xi'an, Shaanxi 710071, PR China
Pazhou Lab, Guangdong, Guangzhou 510330, PR China

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 27 December 2020
Received in revised form 14 July 2021
Accepted 22 August 2021
Available online 3 September 2021

Keywords:
DCNN
Network pruning
Retraining-free

ABSTRACT

Network pruning has been developed as a remedy for accelerating the inference of deep convolutional neural networks (DCNNs). The mainstream methods retrain the pruned models, which maintain the performance of the pruned models but consume a great deal of time. While the other methods reduce the time consumption by omitting to retrain, they lose the performance. To resolve the above conflicts, we propose a two-stage ***R**etraining-**F**ree* pruning method, named *RFPruning*, which embeds the rough screening of channels into training and fine-tunes the structures during pruning, to achieve both good performance and low time consumption. In the first stage, the network training is reformulated as an optimization problem with constraints and solved by a sparse learning approach for rough channel selection. In the second stage, the pruning process is regarded as a multiobjective optimization problem, where the genetic algorithm is applied to carefully select channels for a trade-off between the performance and model size. The proposed method is evaluated against several DCNNs on CIFAR-10 and ImageNet datasets. Extensive experiments demonstrate that such a retraining-free pruning method obtains 43.0% $\sim$ 88.4% compression on model size and maintains the accuracy as the methods with retraining while achieving 3$\times$ speed up in pruning.

1. Introduction

Benefited from the powerful capacity of feature extraction, deep convolutional neural networks (DCNNs) have achieved significant advances in various computer vision tasks. However, the powerful learning capacity of DCNNs relies on expensive storage

* Corresponding author at: School of Artificial Intelligence, Xidian University, Xi'an, Shaanxi 710071, PR China.
E-mail address: xmxie@mail.xidian.edu.cn (X. Xie).
1 Senior Member, IEEE.
2 Fellow, IEEE.

https://doi.org/10.1016/j.asoc.2021.107860
1568-4946/

and computational resources, which restrict applications of DCNNs on mobile devices. This issue drives the research of network acceleration algorithms [1–4], where network pruning is one of the most effective methods.

In general, network pruning is aimed to remove the unimportant network parameters and can be divided into two categories, i.e., unstructured pruning [5,6] and structured pruning [7–9]. The unstructured pruning methods achieve high compression ratios because arbitrary weights in the network may be pruned. However, they destroy the form of the weight matrix and need special hardware to store the pruned weights, which limits their application in real hardware implementation. The structured pruning methods reduce the number of parameters by pruning whole channels or filters of the network. By this means, the size of the weight matrix is compressed and the structure of the matrix is preserved. Therefore, the structured pruning methods are more flexible for most hardware implementations and receive increased attention.

The framework of most existing structured pruning methods [10,11] contains three stages: training, pruning, and retraining. As revealed by Z. Liu et al. [12] and X. Ding et al. [13], such a framework exists two major drawbacks: (1) The optimization process, which requires iterative pruning and retraining, is computation-intensive and time-consuming. (2) The pruned models, at the retraining stage, are easily trapped into bad local minima. For overcoming the two drawbacks, some works (e.g., GAL [14] and VCNNP [15]) have proposed to omit the retraining stage and directly obtain the target pruned models from the trained models. However, the compression ratio and accuracy they achieved are inferior to those of the methods with retraining. Thus, it is a challenge for both good performance and fast deployment.

In this paper, we propose a retraining-free structured pruning method, named *RFPruning*, for fast and efficient network pruning. To directly achieve the good-performing pruned models from the original models, we embed the rough screening of channels into training and fine-tune the structures during pruning. In the training stage, the network training is reformulated as an optimization problem with the sparse constraint, which is abstracted as l_0 norm in the loss function. Considering that l_0 norm is not differentiable, a sparse-learning approach based on the Alternating Direction Method of Multipliers (ADMM) algorithm is proposed to minimize the loss function. Through such an optimization, the unimportant channels are roughly identified and invalidated, which prevents the significant channels from being pruned by mistake and makes the models prunable. In the pruning stage, the task of obtaining the optimal pruned model from the trained model is regarded as a multiobjective optimization problem, where the targets of optimization are the accuracy and compression ratios. For a trade-off between performance and model size, the genetic algorithm is chosen to fine-tune the structure of pruned models, which is implemented by searching suitable pruning rates for each layer. The contributions of this work can be summarized as follows:

- A retraining-free pruning framework, which contains sparse-learning and automatic searching, is proposed for both the fast pruning process and good model performance.
- An ADMM-based sparse-learning technique is embedded into training to roughly identify and invalidate insignificant channels during training, which prevents important channels from being removed by mistake and makes the DCNN models prunable.
- A genetic algorithm-based pruning strategy is developed to fine-tune the network structure for a trade-off between performance and model size. This strategy automatically obtains the optimal pruning rates for models, while has no requirement of additional optimized networks like other automatic strategies (e.g., GAL [14] and MetaPruning [16]).
- Extensive experiments on CIFAR-10 and ImageNet demonstrate that the proposed retraining-free pruning method works as well as the competing methods with retraining, on several popular DCNN models, including VGGNet [17], ResNet [18], GoogLeNet [19], and DenseNet [20].

The organization of this paper is as follows. The relevant literature is reviewed and discussed in Section 2. The proposed method is introduced in detail in Section 3. In Section 4, we report and analyze the experimental results. Finally, this paper is concluded in Section 5.

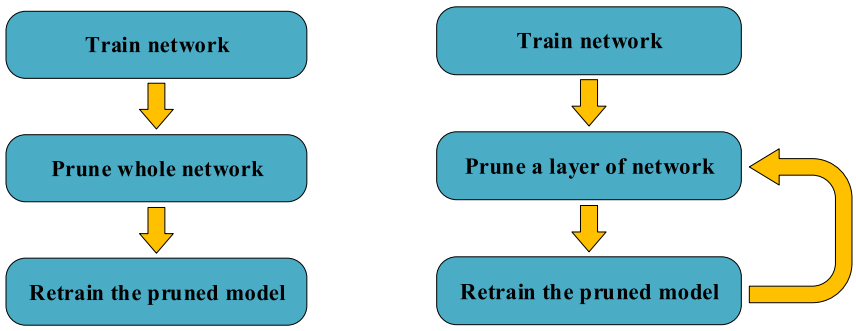
2. Related work

Unstructured pruning: The unstructured pruning methods can be traced back to Optimal Brain Damage [21] and Optimal Brain Surgeon [22], which pruned network parameters according to the Hessian matrix of weights. SNIP [5] proposed to prune network parameters based on the derivatives of weights. Deep Compression [6] and H. Song et al. [23] evaluated the absolute value of weights and pruned the small ones. These methods often achieved high compression ratio because they can prune arbitrary weights in a network. Since parameters of the pruned model are stored in a sparse matrix format with indexes, such methods are only suitable for special hardware while degrading the performance in general process platforms such as GPU.

Structured pruning: The structured pruning methods [10,11,24] overcame the limitation of the unstructured pruning methods. Generally, structured pruning compresses models by removing some channels from the network. Such a scheme maintains the regularity of the weight matrix. Existing structured pruning methods mainly focused on designing channel evaluation criteria, such as l_p norm of channels [25], Geometric Median [26], APoZ [27], HRank [28], the entropy of channels [29], and fisher information [30]. All these methods required retraining the pruned models to recover the accuracy, which is inefficient. Recently, some works [15,31] have proposed new pruning frameworks to improve the efficiency of structured pruning. SFP [31] proposed soft-filter-pruning to zero weights in redundant channels during training. C. Zhao et al. [15] designed a Bayesian-based pruning scheme, which gradually pruned channels during training according to the distribution of weights.

AutoML: The concept of automated machine learning (AutoML) has been investigated in many deep-learning works. The application of AutoML in network pruning frees researchers from complex hyperparameter determination, such as the pruning rates. AMC [32] automatically generated the pruning rates for each layer via reinforcement learning [33]. GAL [14] utilized the framework of GAN [34] to guide the process of pruning. AutoCompress [35] designed a heuristic search technique to searching the pruning rates for the target network. MetaPruning [16] introduced meta-learning to network pruning and designed a PruningNet to generate weights for various pruning structures. Most of these methods required an extra network whose optimization is also challenging.

Weight quantization: Quantization aims at approximating the weights of networks with less bit width. HashNet [36] proposed recoding the weights with hash encoding, which reduced the storage requirements while had the same computational complexity as the original networks. S. Anwar et al. [37] achieved good performing quantified models by minimizing the errors of fixed-point weights. BinaryNet [38] and XnorNet [39] replaced the floating-point operations with binary operations, which significantly accelerated the inference of DCNNs but caused a serious



(a) Global pruning (b) Progressive pruning

Fig. 1. Frameworks of existing pruning methods.

accuracy drop. DoReFa-Net [40] quantified not only the weights but also the gradients, so the inference and training were both accelerated. INQ [41] introduced an incremental quantitative method, which improved the precision of the quantified models.

Lightweight network designing: Different from the methods (pruning and quantization) that compress standard networks, some researchers construct compact networks from scratch. MobileNet [42] compressed the number of parameters by replacing the widely used convolution with a depth-wise separable convolution. ShuffleNet [43] proposed building a compact network with pointwise group convolution and designed channel shuffling to help information flow. GhostNet [44] analyzed the relevance between feature maps and proposed achieving more feature maps with a simple linear operation.

3. Proposed method

This section describes the proposed pruning method in detail. First, the motivations of the proposed method are introduced in Section 3.1. Second, the overall framework of the method is described in Section 3.2. Third, the training approach is introduced in Section 3.3. Fourth, the automatic pruning strategy is proposed in Section 3.4. Finally, the special operations for pruning networks with cross-layer connections are illustrated in Section 3.5.

3.1. Motivations of the proposed method

As presented in Section 2, the structured pruning technique, which removes redundant channels from the networks, has been receiving increased attention due to its advantages in hardware deployment. The main frameworks of existing structured pruning methods are categorized into global pruning and progressive pruning. The global pruning framework is shown in Fig. 1(a). All layers in a network are pruned together in such a framework and then the pruned models are retrained to recover the accuracy. Fig. 1(b) presents the framework of progressive pruning, where the networks will be pruned and retrained multiple times. At each time, only a layer will be compressed. However, both frameworks require much time to retrain the pruned models. As revealed by Z. Liu et al. [12] and X. Ding et al. [13], the retraining scheme is computation-intensive and may trap the pruned models into bad local optimum. If the pruned models with little performance drop can be directly obtained from the trained models, the efficiency of pruning will be improved and the bad convergence caused by retraining will be avoided.

However, it is challenging to directly obtain the pruned model, which maintains the accuracy, from the trained model. The main reason is that the trained models used for pruning are lack sparsity, which results in the unfriendliness of these models to pruning. In the training stage, most exiting structured pruning methods train the networks without constraint on their channels. As a result, the discriminate information used for the final judgment is distributed to all channels. Thus, the movement of channels has a significant impact on the performance of the network. Most pruning methods mainly focus on the pruning criteria while ignoring the training process and the sparsity of the model. So the pruned models of these methods have serious performance drops which have to be recovered by retraining.

Fortunately, sparse-learning is an efficient approach to enhance the sparsity of DCNN models. In general, sparse-learning is used to optimize the sparse coding matrix under the constraint of the sparse norm. In the optimized coding matrix, only a few coefficients record the information while the others are zero. The DCNN can also be considered as a coding matrix, which encodes the inputs to the features. If we regard the channels as the coefficients in a coding matrix and consider the constraint of channels, the training of DCNN becomes the constrained optimization of the coding matrix. By solving this constrained optimization problem with sparse-learning approach, we will obtain the models with channel-level sparsity, where the unimportant channels are automatically invalidated. These invalid channels can then be removed safely without causing a serious loss of accuracy. Such rough screening of channels simplifies the subsequent pruning.

In addition, the success of AutoML [16,32] in the field of network pruning demonstrates that the well-designed pruning rates are beneficial for further reducing the performance drop while causing little influence on the size of pruned models. Thus, it is necessary to carefully set the pruning rates for each layer in the DCNN models. Manually setting these pruning rates is challenging and inefficient. So an automatic pruning strategy, which assigns suitable pruning rates for each layer, is required as well.

3.2. Framework of the proposed method

As analyzed in Section 3.1, the sparsity of DCNN models and the well-designed pruning rates are the keys to reducing performance drop for pruned models. Aiming at these two keys, we design the framework shown in Fig. 2.

In our framework, we firstly introduce the scaling factor γ as a "gate" to scale the output of each channel. If the scaling factor of a channel is zero, no information will be transmitted to the next layer through this channel. So it is feasible to achieve channel-level-sparsity by restricting the scaling factors. Then, the l_0 norm of these scaling factors is added to the loss function. Considering that l_0 norm is not differentiable, we propose optimizing the scaling factors with the Alternating Direction

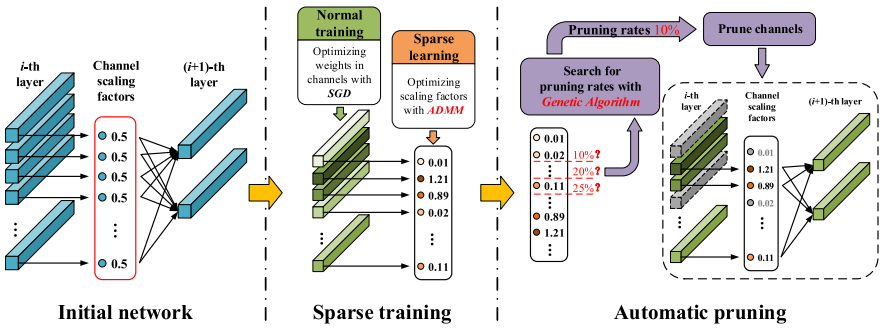


Fig. 2. The framework of *RFPruning*.

Method of Multipliers (ADMM), while the other parameters in the network are optimized by Stochastic Gradient Descent (SGD) algorithm. Finally, we design an automatic pruning scheme that searches the suitable channel pruning rates for each layer by genetic algorithm.

3.3. Training based on sparse-learning

3.3.1. Problem formulation

Considering a N-layer deep convolutional neural network, where the collection of operations (convolution and normalization) in a layer is denoted by $f(\cdot)$ and the input of the lth layer is x_l. So the output of a channel can be represented as

$$y_{l,i} = f\left(x_l, w_{l,i}, b_{l,i}\right), \quad l = 1, 2, \ldots, N; i = 1, 2, \ldots, n, \tag{1}$$

where l is the index of layer and i is the index of channel, $y_{l,i}$ is the output of the ith channel in the lth layer, $w_{l,i}$ is the collection of weights in the channel and $b_{l,i}$ is the bias, n is the total number of channels in the lth layer.

By introducing the scaling factor to each channel in the network, Eq. (1) is transformed as Eq. (2),

$$y_{l,i} = \gamma_{l,i} \cdot f\left(x_l, w_{l,i}, b_{l,i}\right), \quad l = 1, 2, \ldots, N; i = 1, 2, \ldots, n, \tag{2}$$

where $\gamma_{l,i}$ is the scaling factor of the ith channel in the lth layer. Obviously, if $\gamma_{l,i}$ is zero, the output $y_{l,i}$ is zero. The corresponding channel is thus invalid and can be pruned safely. So the sparsity of channels can now be represented by the sparsity of the scaling factors. The problem of achieving channel-level-sparsity mentioned in Section 3.1 is transformed into an optimization problem for scaling factors.

To achieve sparse scaling factors, the number of nonzero factors should be less than the total number of channels. By denoting the collection of $\gamma_{l,i}$ with a vector Γ, the number of nonzero factors can be represented by $|\Gamma|_0$, where $|\cdot|_0$ is the l_0 norm. The problem of training a model of channel-level sparsity can then be regarded as a constrained optimization problem as below,

$$\begin{aligned} \underset{\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma}{\arg\min} \quad & E\left(\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma\right), \\ s.t. \quad & |\Gamma|_0 < C, \end{aligned} \tag{3}$$

where $E(\cdot)$ is the loss function of the network, W_l and B_l are the collection of weights and biases in the lth layer respectively, C is the total number of channels. Through the application of Lagrange Multiplier, Eq. (3) can be further transformed into the following unconstrained optimization problem,

$$\underset{\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma}{\arg\min} \quad E\left(\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma\right) + \lambda|\Gamma|_0, \quad \lambda > 0, \tag{4}$$

where λ is a hyperparameter that can be set. $E\left(\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma\right) + \lambda|\Gamma|_0$ is now the new loss function of the network.

3.3.2. ADMM-based sparse-learning

The optimization problem modeled by Eq. (4) is difficult to be solved by the gradient-based methods (e.g., ADAM [45]) because l_0 norm is not differentiable. Fortunately, the ADMM algorithm is efficient in addressing optimization problems with non-differentiable constraints.

To apply ADMM, we firstly incorporate auxiliary variables and rewrite Eq. (4) as

$$\begin{aligned} \underset{\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma}{\arg\min} \quad & E\left(\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma\right) + \lambda|Z|_0, \\ s.t. \quad & \Gamma - Z = 0, \end{aligned} \tag{5}$$

where vector Z is the collection of auxiliary variables. Then, by applying the augmented Lagrangian [46] and completing the square, Eq. (5) is converted to the following form,

$$\begin{aligned} \underset{\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma}{\arg\min} \quad & E\left(\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma\right) + \lambda|Z|_0 \\ & + \frac{\rho}{2}\left[(\Gamma - Z + U)^2 - U^2\right] \end{aligned} \tag{6}$$

where ρ is a constant and vector U is the collection of dual variables used for completing the square. Finally, Eq. (6) is further decomposed into three subproblems and these subproblems will be solved iteratively.

The first subproblem:

$$\begin{aligned} \underset{\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma}{\arg\min} \quad & E\left(\{W_l\}_{l=1}^N, \{B_l\}_{l=1}^N, \Gamma\right) \\ & + \tfrac{\rho}{2}\left[\left(\Gamma - Z^k + U^k\right)^2 - \left(U^k\right)^2\right] \\ & k = 0, 1, 2, \ldots, T, \end{aligned} \tag{7}$$

where k is the index of iteration, Z^k and U^k are the collections of auxiliary variables and dual variables in the kth iteration. In the first subproblem, Z^k and U^k can be regarded as constant. The first term of this subproblem is the original loss function of DCNN and the second term is convex and differentiable. Thus, the first subproblem can be solved by SGD and the process is the same as the original training of DCNNs.

The second subproblem:

$$\underset{Z}{\arg\min}\, \lambda|Z|_0 + \tfrac{\rho}{2}\left(\Gamma^{k+1} - Z + U^k\right)^2, \quad k = 0, 1, 2, \ldots, T, \tag{8}$$

where Γ^{k+1} is the solution of the first subproblem in the $(k+1)$th iteration, U^k is the dual variable in the kth iteration, Γ^{k+1} and U^k are constant in this subproblem. Because Z, Γ^{k+1}, and U^k are all vectors, solving the second subproblem is equivalent to solving the following problem,

$$\begin{aligned} & \underset{z_j}{\arg\min}\, \lambda\left|z_j\right|_0 + \tfrac{\rho}{2}\left(\gamma_j^{k+1} - z_j + u_j^k\right)^2, \\ & j = 1, 2, \ldots, C; \quad k = 0, 1, 2, \ldots, T, \end{aligned} \tag{9}$$

where z_j, γ_j^{k+1}, and u_j^k are elements in the vectors Z, Γ^{k+1}, and U^k respectively, C is the total number of channels which equals the length of these vectors.

For each auxiliary variable z_j, if z_j is not zero, the value of $|z_j|_0$ is 1, and Eq. (9) can then be rewritten as

$$\arg\min_{z_j} \lambda + \frac{\rho}{2}\left(\gamma_j^{k+1} - z_j + u_j^k\right)^2, \quad j = 1, 2, \ldots, C; \quad z_j \neq 0. \tag{10}$$

Obviously, the solution of Eq. (10) is $z_j^{k+1} = \gamma_j^{k+1} + u_j^k$ and the minimum value is λ. If z_j is zero, the value of $|z_j|_0$ is 0. The minimum value of Eq. (9) is $\frac{\rho}{2}\left(\gamma_j^{k+1} + u_j^k\right)^2$, which has nothing to do with z_j. Hence, if $\frac{\rho}{2}\left(\gamma_j^{k+1} + u_j^k\right)^2 > \lambda$, the solution of Eq. (9) is $z_j^{k+1} = \gamma_j^{k+1} + u_j^k$. If $\frac{\rho}{2}\left(\gamma_j^{k+1} + u_j^k\right)^2 \leq \lambda$, the solution of Eq. (9) is $z_j^{k+1} = 0$. As a result, the solution of the second subproblem can be represented as $Z^{k+1} = \left(z_1^{k+1}, z_2^{k+1}, \ldots, z_C^{k+1}\right)$, where the value of z_j^{k+1} is determined with Eq. (11),

$$z_j^{k+1} = \begin{cases} \gamma_j^{k+1} + u_j^k, & \frac{\rho}{2}\left(\gamma_j^{k+1} + u_j^k\right)^2 > \lambda; \\ 0, & \frac{\rho}{2}\left(\gamma_j^{k+1} + u_j^k\right)^2 \leq \lambda; \end{cases} \quad j = 1, 2, \ldots, C. \tag{11}$$

The third subproblem:

$$\arg\min_{U} \frac{\rho}{2}\left(\Gamma^{k+1} - Z^{k+1} + U^k\right)^2 - U^2, \quad k = 0, 1, 2, \ldots, T. \tag{12}$$

Because this subproblem is a convex quadratic optimization problem, the solution of this subproblem is the extremum of it. So the solution U^{k+1} can be achieved via Eq. (13),

$$U^{k+1} = \Gamma^{k+1} - Z^{k+1} + U^k, \tag{13}$$

where Γ^{k+1} and Z^{k+1} are the solution of the other two subproblems in the $(k+1)$th iteration, U^k is the dual variable in the kth iteration.

The solution U^{k+1} of the third subproblem will be fed into the first subproblem in the next iteration. By solving the three subproblems iteratively, we will get the solution of Eq. (6) and finally obtain a DCNN model with channel-level sparsity.

3.4. Automatic pruning based on genetic algorithm

The pruning strategy is the soul of channel pruning methods because the saved channels will affect the model size and performance of pruned models. Therefore, the pruning strategy should strike the tradeoff between the compression ratio and the accuracy, which is a multiobjective optimization problem. The key of solving this problem is to assign suitable pruning rates for layers. The optimality of the manual setting of pruning rates is questionable [47] because some channels may be pruned by mistake. The methods based on meta-learning or reinforcement learning (e.g., MetaPruning [16] and AMC [32]) generate the pruning rates by the auxiliary networks whose optimization is also challenging. Thus, an adaptive pruning strategy without desire of the extra network is conceptually desirable.

As an automatic optimization method, the genetic algorithm [48], which has no requirement of auxiliary networks, is effective in solving multiobjective optimization problems. So in this paper, we design a pruning strategy to automatically search for the optimal pruning rates of each layer with the genetic algorithm as shown in Fig. 3.

To implement the proposed pruning strategy, we need to initialize some individuals as the initial population of the genetic algorithm. Each individual is a combination of the channel pruning rates and represented as

$$s_i = \left\{p_{i,j} | 0 \leq p_{i,j} < 1, j = 1, 2, \ldots, N\right\}, \quad i = 1, 2, \ldots, M, \tag{14}$$

where s_i is the ith individual, $p_{i,j}$ represents the channel pruning rate of jth layer in s_i, and M is the total number of individuals. In order to improve the efficiency of searching, we design an initialization method based on the distribution of scaling factors instead of random initialization. In our initialization, all scaling factors are grouped into several bins. The factors in the first bin are the smallest ones of all. Assuming that the number of factors in the first bin is r and the total number of factors is C, the ratio r/C is regarded as the target channel pruning rate. Then, we set the pruning rates of each layer with some manual setting strategies proposed by previous pruning methods (e.g., [25,27,31,49]) and make the channel pruning rate of the whole network equals r/C .

After initialization, the proposed strategy searches from the initial population for the optimal individual by iterating the four steps shown in Fig. 3.

Step1: codes generation.

Each individual is encoded into a binary code by the following formula,

$$c_i = \sum_{j=1}^{N} \left\lfloor p_{i,j} \times (2^m - 1) \right\rfloor \times 2^{m(N-j)}, \quad i = 1, 2, \ldots, M, \tag{15}$$

where c_i is the ith code, and m is a hyperparameter that determines the length of the binary code. Several new codes will be generated from these initial codes through crossover and mutation. The crossover is to randomly swap some bits between two codes and the mutation is to change some bits (0 to 1 or 1 to 0) in the codes.

Step2: decoding.

All codes (codes from encoding, crossover, and mutation) will be decoded to individuals through the inverse process of Eq. (15). The pruning rate of jth layer in the ith individual is decoded as below,

$$p_{i,j} = \left\lfloor c_j \div 2^{m(N-j)} \right\rfloor \div \left(2^m - 1\right). \tag{16}$$

Step3: fitness evaluation.

According to the pruning rates in an individual, we sample some channels associated with large scaling factors from each layer to generate a subnetwork as that zoomed in Fig. 3. The accuracy, model size (e.g., number of parameters), and computational cost (e.g., number of float operations) of each subnetwork are utilized to evaluate the fitness of the corresponding individual.

To achieve the individuals that strike tradeoff between accuracy and compression ratio, the fitness f_i of the ith individual is designed as Eq. (17),

$$f_i = \begin{cases} -a_i \log\left(\theta \cdot \frac{b_i}{B_0} + (1-\theta) \cdot \frac{d_i}{D_0}\right), & a_i \geq A_0 - \varepsilon; \\ -\frac{a_i}{2} \log\left(\theta \cdot \frac{b_i}{B_0} + (1-\theta) \cdot \frac{d_i}{D_0}\right), & a_i < A_0 - \varepsilon; \end{cases} \quad 0 \leq \theta \leq 1, \tag{17}$$

where a_i, b_i, and d_i denote the accuracy, the number of parameters, and FLOPs (float operations) of the subnetwork corresponding to the ith individual, A_0, B_0, and D_0 are those of the original network, θ is a hyperparameter that balances the different parts of Eq. (17), ε represents the permissible deviation of accuracy. Obviously, the individuals that have higher accuracy, fewer parameters, and lower computational cost will get better fitness.

Step4: selection.

After getting the fitness, we calculate a selection probability for each individual by this formula,

$$q_i = \frac{f_i}{sum(f)}, \tag{18}$$

where q_i is the selection probability of the ith individual, $sum\,(f)$ is the summary of all fitness. M individuals are selected by

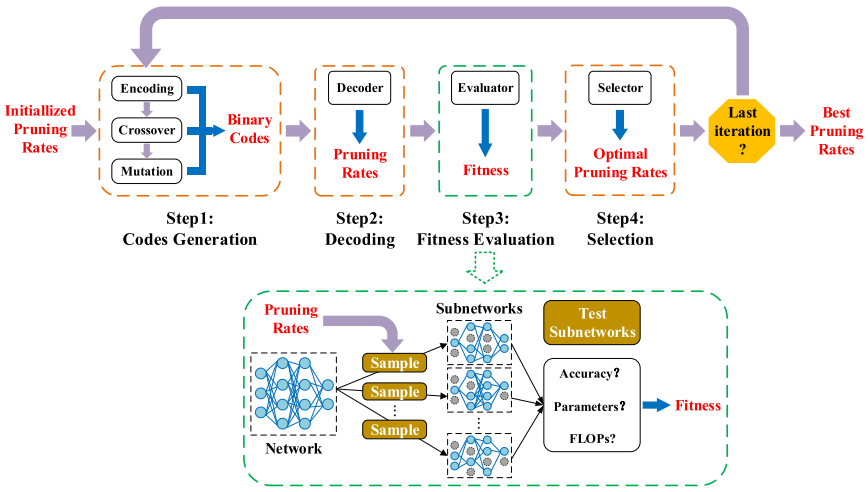


Fig. 3. Searching pruning rates with genetic algorithm. The process of fitness evaluation is zoomed.

the roulette-wheel selection method [50] based on the selection probability. The selected individuals will be fed into Step1 for the next iteration.

After several iterations, the individual with the best fitness is chosen as the final combination of channel pruning rates. By pruning channels with small scaling factors based on these pruning rates, the pruned model that strikes the optimal tradeoff between performance and cost is achieved.

3.5. Adjustment for the cross-layer connections

Although the pruning process introduced in Section 3.4 can be directly applied to various DCNN architectures (e.g., VGGNet [17], GoogLeNet [19]), some special adjustments are required when the proposed pruning strategy is applied to the network with cross-layer connections (e.g., ResNet [18]). Such a network fuses the outputs of multiple layers, so the dimensions of these outputs have to be uniform. However, the different pruning rates of layers will lead to the damage of the dimensional uniformity as shown in Fig. 4. To solve this problem, previous methods force the pruning rates of the fused layers to be the same or do not prune the fused layers. Such solutions limit the freedom of pruning and decrease the reduction.

In this paper, two special operations illustrated in Fig. 4 are introduced for effective pruning of the fused layers. The channel-dimension-increasing (CDI) inserts zero-filled feature maps to the outputs so that the dimension of different outputs can be uniform. The channel-dimension-decreasing (CDD) removes the zero-filled feature maps from the fused outputs to get the effective inputs of the next layer. By adopting the two special operations, the network with cross-layer connections can be freely pruned without the limitation of dimension.

4. Experiments

In this section, the performance of the proposed method is demonstrated and analyzed through a series of experiments. The experimental settings are introduced in Section 4.1. The practicality of the proposed *RFPruning* is evaluated in Section 4.2. In addition, the ablation experiments are conducted in Section 4.3 to analyze the effects of the proposed sparse-learning scheme and automatic pruning strategy in more detail.

4.1. Experimental settings

Datasets and baselines. To evaluate the efficiency of our method in compressing network models, we conduct experiments on the CIFAR-10 [51] and ImageNet [52] datasets. CIFAR-10 is a small dataset in the field of image classification, and consists of 32 $\times$ 32 natural images, in 10 classes. ImageNet is a large classification dataset that contains 1.2 million training images and 50,000 validation images, in 1000 classes. The mainstream DCNN models, including VGGNet [17], ResNet [18], GoogLeNet [19], and DenseNet [20], are selected as the pruning subjects. VGGNet is a typical network with plain structures. ResNet is a representative model with branches called "shortcut". GoogLeNet consists of inception modules and DenseNet contains several dense blocks.

Evaluation protocols. To evaluate the proposed method synthetically, we analyze the accuracy, model size, and computational cost of the pruned models. The number of parameters and FLOPs are chosen as the evaluation protocols of the model size and computational cost. To provide an intuitive comparison with other methods, we also illustrate the compression ratios (CR) of parameters and FLOPs for the pruned models.

Configurations. We implement all experiments with PyTorch [53]. The Stochastic Gradient Descent (SGD) optimizer is chosen to update the weights of the networks with a Nesterov momentum [54] of 0.9 and a weight decay of 10^{-4}. The initial learning rate is 0.1 and the scaling factors are initialized to 0.5. The hyperparameter λ is 10^{-3} and m is 10. For CIFAR-10, all networks are trained for 240 epochs on an NVIDIA GTX Titan-XP GPU, with a batch size of 64. The learning rate is divided by 10 at 120, 160, and 200 epochs. For ImageNet, the batch size is 256. All models are trained for 120 epochs on 4 GPUs. The learning rate is multiplied by 0.1 when the epoch is 60, 80, or 100.

4.2. Results and analysis on classification networks

The proposed method is evaluated on several popular DCNN models. The results are summarized in Table 1, including the accuracy, parameters, FLOPs, and compression ratio of the pruned models. For different models, *RFPruning* prunes 41.4% $\sim$ 81.6% channels respectively. For CIFAR-10, the proposed *RFPruning* achieves 87.8%/65.6%/88.4%/77.0% reduction in parameters and 56.0%/69.8%/83.6%/72.4% reduction in FLOPs on VGG-16/ResNet-110/GoogLeNet/DenseNet-40. For ImageNet, our method also obtains 45.8%/47.9% reduction in parameters/FLOPs on ResNet-50. More detailed analysis and comparative experiments are shown in Sections 4.2.1 and 4.2.2.

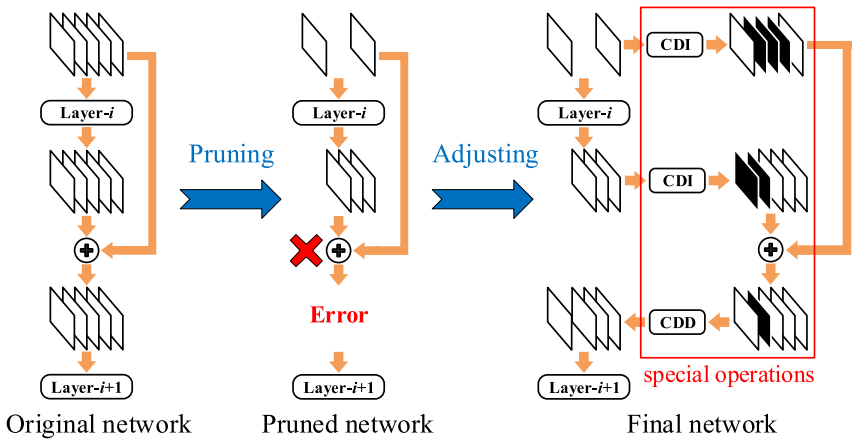


Fig. 4. Pruning for the network with cross-layer connections.

Table 1
The pruning results of *RFPruning*. "M" represents million (10^6) and "B" represents billion (10^9). "CRP" denotes "Compression Ratio of Parameters", "CRP" denotes "Compression Ratio of FLOPs", and "CRC" represents "Compression Ratio of Channels".

Dataset	Network	Accuracy	Parameters	CRP	FLOPs	CRF	CRC
CIFAR-10	VGG-16	94.14%	1.79M	87.8%	0.138B	56.0%	65.6%
	ResNet-56	93.41%	0.49M	43.0%	0.062B	51.2%	41.4%
	ResNet-110	93.75%	0.60M	65.6%	0.077B	69.8%	62.5%
	GoogLeNet	94.84%	0.72M	88.4%	0.253B	83.6%	68.1%
	DenseNet-40	94.20%	0.25M	77.0%	0.079B	72.4%	81.6%
ImageNet	ResNet-50	73.29%	13.87M	45.8%	2.14B	47.9%	60.2%

Table 2
Pruning results of VGG-16 on CIFAR-10.

Model	Retraining?	Accuracy	Parameters(CR)	FLOPs(CR)
VGG-16 Base	–	94.09%	14.73M(0.0%)	0.314B(0.0%)
PFEC [7]	Yes	93.40%	5.40M(63.3%)	0.206B(34.4%)
SSS [55]	Yes	93.02%	3.93M(73.3%)	0.183B(41.7%)
HRank [28]	Yes	93.43%	2.51M(83.0%)	0.146B(53.5%)
GAL-0.05 [14]	No	92.03%	3.36M(77.2%)	0.189B(39.8%)
VCNNP [15]	No	93.18%	3.92M(73.4%)	0.190B(39.5%)
RFPruning	**No**	**94.14%**	**1.79M(87.8%)**	**0.138B(56.0%)**

4.2.1. Experiments on CIFAR

To analyze the performance on CIFAR-10, we apply the proposed method on several popular DCNNs, including VGGNet, ResNet, GoogLeNet, and DenseNet. All networks are adjusted to fit the class number of CIFAR-10 as that in HRank [28].

VGGNet. Table 2 shows the pruning result of the proposed method on VGG-16 network. The result is compared to several state-of-the-art pruning methods, including PFEC [7], GAL [14], VCNNP [15], Hrank [28], and SSS [55]. Compared with the base model, *RFPruning* effectively compresses the model (1.79M vs. 14.73M parameters, and 0.138B vs. 0.314B FLOPs) and maintains the accuracy (94.14% vs. 94.09%). Compared to the other retraining-free methods (e.g., GAL and VCNNP), the proposed method significantly increases the reduction in both parameters and FLOPs. In addition, *RFPruning* can even achieve better accuracy and more reduction than the methods that retrain the pruned models (e.g., PFEC, SSS, and HRank).

ResNet. We apply the proposed method on two popular ResNet models, ResNet-56 and ResNet-110. PFEC [7], GAL [14], FPGM [26], HRank [28], SFP [31], and NISP [56] are selected as the competitors. All results are displayed in Table 3. For ResNet-56, *RFPruning* reduces 43.0% parameters and 51.2% FLOPs from the base model. The proposed method achieves the highest compression ratio in FLOPs and the best accuracy among all methods. For ResNet-110, *RFPruning* gets 65.6% reduction in parameters and 69.8% reduction FLOPs. Compared with the other competitors, *RFPruning* yields better accuracy and more reduction (both in parameters and FLOPs).

Table 3
Pruning results of ResNet-56/110 on CIFAR-10.

Model	Retraining?	Accuracy	Parameters(CR)	FLOPs(CR)
ResNet-56 Base	–	93.42%	0.86M(0.0%)	0.127B(0.0%)
PFEC [7]	Yes	93.06%	0.73M(15.1%)	0.091B(28.3%)
NISP [56]	Yes	93.01%	0.49M(43.0%)	0.081B(36.2%)
HRank [28]	Yes	93.17%	0.49M(43.0%)	0.063B(50.4%)
GAL-0.6 [14]	No	92.98%	0.75M(12.8%)	0.078B(38.6%)
SFP [31]	No	93.10%	–	0.074B(41.7%)
RFPruning	**No**	**93.41%**	**0.49M(43.0%)**	**0.062B(51.2%)**
ResNet-110 Base	–	93.72%	1.73M(0.0%)	0.255B(0.0%)
PFEC [7]	Yes	93.30%	1.16M(32.9%)	0.155B(39.2%)
FPGM [26]	Yes	93.74%	–	0.121B(52.5%)
HRank [28]	Yes	93.36%	0.70M(59.5%)	0.106B(58.4%)
GAL-0.5 [14]	No	92.55%	0.95M(45.1%)	0.130B(49.0%)
SFP [31]	No	93.38%	–	0.150B(41.2%)
RFPruning	**No**	**93.75%**	**0.60M(65.6%)**	**0.077B(69.8%)**

Table 4
Pruning results of GoogLeNet on CIFAR-10.

Model	Retraining?	Accuracy	Parameters(CR)	FLOPs(CR)
GoogLeNet Base	–	94.81%	6.17M(0.0%)	1.542B(0.0%)
PFEC [7]	Yes	94.54%	3.51M(43.1%)	1.020B(33.8%)
HRank [28]	Yes	94.53%	2.74M(55.6%)	0.690B(55.3%)
GAL-0.05 [14]	No	93.93%	3.12M(49.4%)	0.939B(39.1%)
GAL-APoZ [14]	No	92.11%	2.85M(53.8%)	0.760B(50.7%)
RFPruning	**No**	**94.84%**	**0.72M(88.4%)**	**0.253B(83.6%)**

Table 5
Pruning results of DenseNet-40 on CIFAR-10.

Model	Retraining?	Accuracy	Parameters(CR)	FLOPs(CR)
DenseNet-40 Base	–	94.22%	1.07M(0.0%)	0.288B(0.0%)
Slimming [49]	Yes	94.35%	0.35M(67.2%)	0.120B(58.3%)
HRank [28]	Yes	93.68%	0.48M(55.1%)	0.110B(61.8%)
GAL-0.05 [14]	No	93.53%	0.45M(57.9%)	0.128B(55.6%)
VCNNP [15]	No	93.16%	0.42M(60.7%)	0.156B(45.8%)
RFPruning	**No**	**94.20%**	**0.25M(77.0%)**	**0.079B(72.4%)**

GoogLeNet. The pruning results of GoogLeNet are illustrated in Table 4. The proposed method demonstrates its ability by obtaining 88.4% reduction in parameters, 83.6% reduction in FLOPs, and better accuracy than the base model (94.84% vs. 94.81%). The comparison results with other methods (e.g., PFEC [7], GAL [14], HRank [28]) show that *RFPruning* improves the compression ratios significantly.

Table 6
Pruning results of ResNet-50 on ImageNet.

Model	Retraining?	Accuracy	Parameters(CR)	FLOPs(CR)
ResNet-50 Base	–	73.31%	25.58M(0.0%)	4.11B(0.0%)
SSS [55]	Yes	71.82%	15.60M(39.0%)	2.33B(43.3%)
CPNN [57]	Yes	72.30%	–	2.73B(33.6%)
ThiNet-70 [25]	Yes	72.04%	16.94M(33.8%)	2.44B(40.6%)
HRank [28]	Yes	74.98%	16.15M(36.9%)	2.30B(44.0%)
GAL-0.5 [14]	No	71.95%	21.20M(17.1%)	2.33B(43.3%)
RFPruning	**No**	**73.29%**	**13.87M(45.8%)**	**2.14B(47.9%)**

DenseNet. For DenseNet, we conduct experiments on a 40-layer DenseNet (DenseNet-40) with a growth rate 12 [20]. The results are summarized in Table 5. From Table 5, it can be found that *RFPruning* obtains 77.0% reduction in parameters and 72.4% reduction in FLOPs. It is worth noticing that the proposed method leads to only 0.02% loss of accuracy without retraining the pruned model, which is even better than the method with retraining (e.g., HRank [28]). Compared with GAL [14] and VCNNP [15], our method achieves significant improvement in reduction and performance. Although the precision of Slimming [49] is higher, the pruned model of *RFPruning* is more compact.

All the experimental results demonstrate that *RFPruning* is practical for various DCNN models on the CIFAR dataset. Additionally, the proposed method frees the network pruning from the time-consuming retraining and achieves similar or better results than the methods with retraining.

4.2.2. Experiments on ImageNet

ImageNet is another notable image classification dataset. We conduct the experiment on this dataset as well. The ResNet-50 is selected as the pruning object. The proposed method is compared with several state-of-the-art methods (e.g., GAL [14], ThiNet [25], HRank [28], SSS [55], and CPNN [57]), whose results are shown in Table 6.

The result demonstrates that *RFPruning* reduces 45.8% parameters and 47.9% FLOPs from the base model with only 0.02% loss of accuracy. Compared to SSS, CPNN, and ThiNet, the proposed method achieves both better performance and a more compact model. Specially, the process of retraining is very time-consuming on a large dataset like ImageNet, so the proposed retraining-free method is more efficient. Although HRank obtains the best accuracy, its reduction is lower than *RFPruning*. Compared with another retraining-free pruning method GAL, *RFPruning* significantly improves the reduction of parameters. Therefore, the proposed method also works well on the large and complex dataset.

4.3. Ablation study

To analyze the proposed *RFPruning* in more detail, we conduct ablation studies on the two main processes, sparse-learning and automatic pruning in Sections 4.3.1 and 4.3.2. For brevity, we report the results of VGG-16 on the CIFAR-10 dataset. The same results can be achieved in other networks and datasets. In addition, the computational cost of different stages in the proposed method is analyzed in Section 4.3.3 and the filter selection is discussed in Section 4.3.4, respectively.

4.3.1. Analysis of ADMM based sparse-learning

To analyze the effect of the ADMM based sparse-learning scheme, we demonstrate the sparsity of the models by visualizing the distribution of scaling factors in Fig. 5. In this figure, it is obvious that the zero factors of the sparse model are more than those in the normal model. Such a result illustrates that the proposed sparse-learning scheme is effective.

Table 7
Training results of different sparse-learning schemes.

Network	Sparse-learning method	Accuracy
VGG-16	Slimming (SGD+l_1)	93.45%
	SSS (APG+l_1)	91.41%
	NPSG (SGD+$l_{1/2}$)	93.02%
	RFPruning (ADMM+l_1)	93.50%
	RFPruning (ADMM+l_0)	94.09%

Table 8
Results of different pruning strategies.

Network	Pruning scheme	Accuracy	Parameters(CR)	FLOPs(CR)
VGG-16	No pruning (Base)	94.09%	14.73M(0.0%)	0.314B(0.0%)
	L_2 norm (SFP)	93.21%	1.82M(87.6%)	0.135B(57.1%)
	L_2 **norm (Ours)**	93.97%	1.70M(88.4%)	0.136B(56.6%)
	BN weight (Slimming)	92.66%	1.81M(87.7%)	0.137B(56.4%)
	BN weight (Ours)	93.80%	1.72M(88.3%)	0.137B(56.4%)
	Scaling factor (SSS)	94.02%	1.78M(87.9%)	0.136B(56.8%)
	Scaling factor (Ours)	94.14%	1.79M(87.8%)	0.138B(56.0%)

We also visualize some feature maps to give a more straightforward exhibition of the effect of our scheme. As shown in Fig. 6, the number of invalid features obtained from the sparse model is more than that achieved from the normal model. The phenomenon is consistent with what was observed from the distribution of scaling factors in Fig. 5. This means that the sparsity of scaling factors can represent the sparsity of channels.

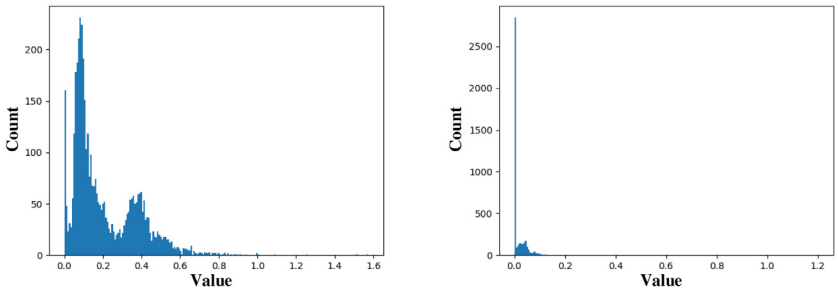
In addition, we compare our ADMM based sparse-learning scheme with some other sparse schemes, Slimming [49], SSS [55], and NPSG [58]. SSS utilizes l_1 norm and Accelerated Proximal Gradient (APG) method to get sparsity. Slimming constraints the network with l_1 norm and optimizes it with SGD method. NPSG optimizes the network with $l_{1/2}$ regularization and SGD method. Two regularizations, l_1 norm and l_0 norm, are tested when we implement the proposed sparse scheme. For a fair comparison, we adopt all sparse schemes to the scaling factors and the hyperparameter λ for all methods is 10^{-3}. The training results are shown in Table 7. Compared with the other three schemes, *RFPruning* achieves the best accuracy. For the two kinds of regularizations, l_1 norm and l_0 norm, the results demonstrate that the model with l_0 norm constraint can achieve better performance by the proposed ADMM based optimization algorithm.

All the visualization and the compared results prove that the proposed ADMM based sparse-learning method is effective both in achieving sparsity and maintaining the performance of the network.

4.3.2. Analysis of automatic pruning based on genetic algorithm

The genetic algorithm-based pruning strategy is also analyzed in detail. During the experiment, the proposed pruning strategy is compared with three popular pruning strategies, SFP, Slimming, and SSS. SFP selects channels according to the l_2 norm of weights in convolutional layers. Slimming prunes channels according to the absolute value of weight in batch normalization (BN) layers. SSS chooses channels through the scaling factors that are similar to ours. For a fair comparison, we utilize the same evaluation criteria as the competitors only with the channel pruning rates changed. All pruning methods are applied on the same base model and all the pruned models are not retrained. The results are shown in Table 8.

From Table 8, it can be found that the accuracy of the models pruned by the proposed pruning strategy is better than the competitors. The results illustrate that the channel pruning rates searched by the proposed strategy are more conducive to maintaining the precision of the network. Compared with SFP, the



(a) Normal model (b) Sparse model

Fig. 5. Distribution of the scaling factors in two VGG-16 models. The normal model is trained in the traditional training way while the sparse model is trained with the ADMM based sparse-learning scheme.

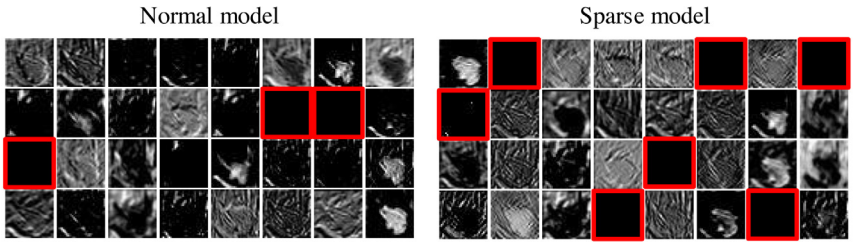


Fig. 6. Sparsity of feature maps from two kinds of VGG-16 models, with the red boxes being the invalid feature maps.

Table 9
Time consumption on CIFAR-10. "Selection Time" is the time of filter selection which is based on the optimal pruning rates. "h" represents "hour" and "s" represents "second".

Network	Method	Training time	Pruning time	Selection time
VGG-16	Ours	**2.7** h	**0.2** h	**0.033** s
GoogLeNet	Ours	**10.2** h	**0.5** h	**0.112** s
DenseNet-40	Ours	**5.6** h	**0.5** h	**0.058** s
ResNet-56	SFP [31]	3.0 h	1.1 h	–
	FPGM [26]	1.7 h	8.1 h	–
	Ours	**2.3** h	**0.3** h	**0.091** s
ResNet-110	SFP [31]	10.3 h	5.3 h	–
	FPGM [26]	6.1 h	4.3 h	–
	Ours	**4.8** h	**0.6** h	**0.184** s

proposed strategy improves not only the accuracy but also the reduction of parameters. The results compared to Slimming demonstrate that our strategy can achieve better and more compact pruned models. Although the compression ratios of SSS are a little higher than ours, the proposed strategy achieves better accuracy than the base model. Thus, the genetic algorithm-based pruning scheme can effectively improve the performance of pruned models.

The number of channels in each layer of the original model and the pruned model is also displayed in Fig. 7 to give a straightforward exhibition of the pruning results. Different from the original model, the number of channels in the pruned model does not keep increasing with the increasing of the layer index. Besides, the reduction of channels in the last six layers is more than that of the previous layers. Inspired by such phenomena, it may be unnecessary to set too many channels in the deep layers of the network.

4.3.3. Analysis of computational cost

To evaluate the computational cost of each stage in the proposed method, we count the training time and pruning time respectively. The time of filter selection, based on the optimal pruning rates, is also presented. The time consumption of the proposed method and compared methods (e.g., SFP [31] and FPGM [26]) is shown in Table 9. The pruning time for other methods includes their retraining time. Specifically, the proposed method consumes less time in pruning than the other methods, which demonstrates the low computational cost of the proposed pruning strategy. The time of filter selection for all networks is less than 0.2 s, which is very fast and efficient. Compared with SFP, the proposed method achieves 3× speed up on ResNet-56 and 9× speed up on ResNet-110 in pruning time. Compared to FPGM, the overall time consumption of *RFPruning* is lower. The main reason for the fast deployment of the proposed method is that we abandon the idea of maintaining the performance with retraining, while focus on making the models prunable and adjusting the pruning structure to avoid accuracy drop.

4.3.4. Analysis of filter selection

In order to present which filters are selected and analyze the properties of selected filters, we visualize the feature maps corresponding to filters. Fig. 8 shows the feature maps in a layer of the original VGGNet. The feature maps with red boxes are from pruned filters and the others are from selected filters. Compared with the unmarked ones, the feature maps with red boxes are almost dark, which means that the pruned filters provide little information. On the contrary, the feature maps from selected filters contain obvious characteristic information. Such selection maintains the feature extraction ability of the network to the maximum extent. This is why the proposed method maintains the performance of models after pruning.

To give an intuitive presentation for the structure of the pruned model, we list the weight size of each layer of the original and pruned VGGNets in Table 10. The weights for convolutional layers are 4D tensors, whose size is denoted by "*kernel size* × *kernel size* × *input channels* × *output channels*". The weights for fully-connected layers are 2D tensors, whose size is denoted by "*input channels* × *output channels*".

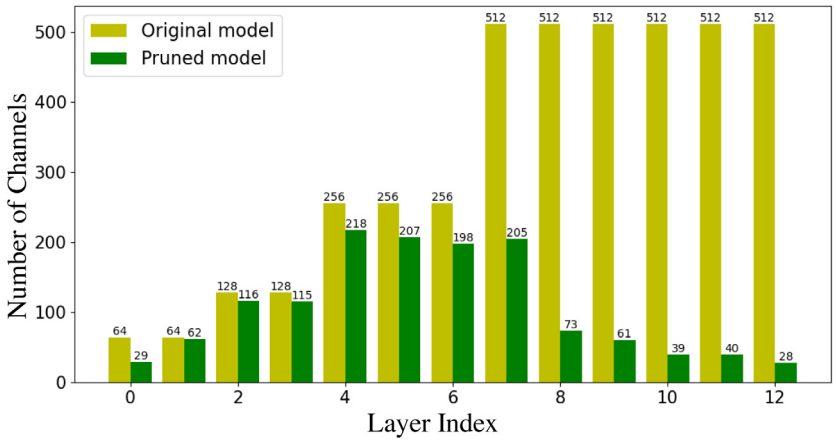


Fig. 7. The number of each layer channel of the original and pruned models.

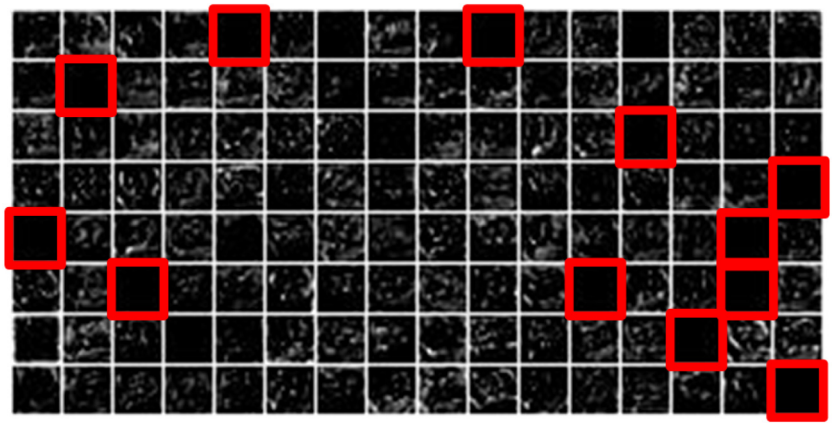


Fig. 8. Visualization of feature maps. The red boxes are the outputs of the pruned filters and the others are the outputs of selected filters.

Table 10
Structure of the original and pruned VGGNets for CIFAR-10. "Conv" denotes "Convolutional layer" and "FC" denotes "Fully-Connected layer".

Layers	Original VGGNet	Pruned VGGNet
layer-1	Conv $3 \times 3 \times 3 \times 64$	Conv $3 \times 3 \times 3 \times 29$
layer-2	Conv $3 \times 3 \times 64 \times 64$	Conv $3 \times 3 \times 29 \times 62$
layer-3	Conv $3 \times 3 \times 64 \times 128$	Conv $3 \times 3 \times 62 \times 116$
layer-4	Conv $3 \times 3 \times 128 \times 128$	Conv $3 \times 3 \times 116 \times 115$
layer-5	Conv $3 \times 3 \times 128 \times 256$	Conv $3 \times 3 \times 115 \times 218$
layer-6	Conv $3 \times 3 \times 256 \times 256$	Conv $3 \times 3 \times 218 \times 207$
layer-7	Conv $3 \times 3 \times 256 \times 256$	Conv $3 \times 3 \times 207 \times 198$
layer-8	Conv $3 \times 3 \times 256 \times 512$	Conv $3 \times 3 \times 198 \times 205$
layer-9	Conv $3 \times 3 \times 512 \times 512$	Conv $3 \times 3 \times 205 \times 73$
layer-10	Conv $3 \times 3 \times 512 \times 512$	Conv $3 \times 3 \times 73 \times 61$
layer-11	Conv $3 \times 3 \times 512 \times 512$	Conv $3 \times 3 \times 61 \times 39$
layer-12	Conv $3 \times 3 \times 512 \times 512$	Conv $3 \times 3 \times 39 \times 40$
layer-13	Conv $3 \times 3 \times 512 \times 512$	Conv $3 \times 3 \times 40 \times 28$
layer-14	FC 512×10	FC 28×10

5. Conclusion

In this paper, we propose a retraining-free pruning method, named *RFPruning*, to compress the DCNNs in a fast way and maintain the performance simultaneously. The network pruning task is formulated as two optimization problems, which are solved by the ADMM based sparse-learning and genetic algorithm respectively. Experimental results showed that the ADMM based sparse-learning is beneficial for making the DCNN models prunable and the application of the genetic algorithm is effective in improving the performance of pruned models. Extensive comparison results on CIFAR-10 and ImageNet datasets demonstrated that the proposed retraining-free pruning method works as well as the methods with retraining while achieves $3\times$ speed up in pruning. In the future, combined with neural architecture search (NAS), we would like to explore more practical and compact network models.

CRediT authorship contribution statement

Zhenyu Wang: Conceptualization, Methodology, Software, Investigation, Writing – original draft, Writing – review & editing. **Xuemei Xie:** Writing – review & editing, Resources, Project administration, Funding acquisition. **Guangming Shi:** Supervision, Resources.

Declaration of competing interest

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.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant (No. 61632019, and 61836008) and the National Key R&D Program of China under Grant (No. 2020AAA0109301).

References

[1] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, N. de Freitas, Predicting parameters in deep learning, in: Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2, 2013, pp. 2148–2156.

[2] M. Jaderberg, A. Vedaldi, A. Zisserman, Speeding up convolutional neural networks with low rank expansions, in: Proceedings of the British Machine Vision Conference, BMVA Press, 2014.

[3] H. Yang, Y. Zhu, J. Liu, Energy-constrained compression for deep neural networks via weighted sparse projection and layer input masking, in: International Conference on Learning Representations, 2018.

[4] H. Yang, Y. Zhu, J. Liu, Ecc: Platform-independent energy-constrained deep neural network compression via a bilinear regression model, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019.

[5] N. Lee, T. Ajanthan, P.H. Torr, Snip: Single-shot network pruning based on connection sensitivity, in: International Conference on Learning Representations, ICLR, 2019.

[6] H.M. S. Han, W. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, in: International Conference on Learning Representations, ICLR, 2016.

[7] A. H. Li, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, in: International Conference on Learning Representations, ICLR, 2017.

[8] S. Anwar, K. Hwang, W. Sung, Structured pruning of deep convolutional neural networks, ACM J. Emerg. Technol. Comput. Syst. (JETC) 13 (3) (2017) 32.

[9] P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz, Pruning convolutional neural networks for resource efficient inference, in: 5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings, 2019.

[10] J. Ye, X. Lu, Z. Lin, J.Z. Wang, Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers, in: 6th International Conference on Learning Representations, ICLR 2018, 2018.

[11] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li, Learning structured sparsity in deep neural networks, in: Advances in neural information processing systems, 2016, pp. 2074–2082.

[12] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, in: International Conference on Learning Representations, 2018.

[13] X. Ding, T. Hao, J. Liu, J. Han, Y. Guo, G. Ding, Lossless CNN channel pruning via decoupling remembering and forgetting, 2020, arXiv preprint arXiv:2007.03260.

[14] S. Lin, R. Ji, C. Yan, B. Zhang, L. Cao, Q. Ye, F. Huang, D. Doermann, Towards optimal structured cnn pruning via generative adversarial learning, in: Computer Vision and Pattern Recognition, CVPR, 2019.

[15] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, Q. Tian, Variational convolutional neural network pruning, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020.

[16] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, J. Sun, Metapruning: Meta learning for automatic neural network channel pruning, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3296–3305.

[17] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, Comput. Sci. (2014).

[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[20] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[21] Y. LeCun, J.S. Denker, S.A. Solla, Optimal brain damage, in: Advances in neural information processing systems, 1990, pp. 598–605.

[22] B. Hassibi, D.G. Stork, Second order derivatives for network pruning: Optimal brain surgeon, in: Advances in neural information processing systems, 1993, pp. 164–171.

[23] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, in: Advances in neural information processing systems, 2015, pp. 1135–1143.

[24] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, J. Kautz, Importance estimation for neural network pruning, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020.

[25] J.-H. Luo, J. Wu, W. Lin, Thinet: A filter level pruning method for deep neural network compression, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 5058–5066.

[26] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020.

[27] H. Hu, R. Peng, Y.-W. Tai, C.-K. Tang, Network trimming: A data-driven neuron pruning approach towards efficient deep architectures, 2016, arXiv preprint arXiv:1607.03250.

[28] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, L. Shao, HRank: Filter Pruning using High-Rank Feature Map, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 1529–1538.

[29] J.-H. Luo, J. Wu, An entropy-based pruning method for cnn compression, 2017, arXiv preprint arXiv:1706.05791.

[30] L. Theis, I. Korshunova, A. Tejani, F. Huszár, Faster gaze prediction with dense networks and fisher pruning, 2018, arXiv preprint arXiv:1801.05787.

[31] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, in: Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18, 2018.

[32] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, Amc: Automl for model compression and acceleration on mobile devices, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 784–800.

[33] R.S. Sutton, A.G. Barto, Reinforcement learning, Bradford Book 15 (7) (1998) 665–685.

[34] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Adv. Neural Inf. Process. Syst. 3 (2014) 2672–2680.

[35] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, J. Ye, AutoCompress: An automatic DNN structured pruning framework for ultra-high compression rates, in: Vol. 34, Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 4876–4883.

[36] W. Chen, J. Wilson, S. Tyree, K. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, in: International Conference on Machine Learning, 2015, pp. 2285–2294.

[37] S. Anwar, K. Hwang, W. Sung, Fixed point optimization of deep convolutional neural networks for object recognition, 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015.

[38] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 4114–4122.

[39] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: European Conference on Computer Vision, 2016, pp. 525–542.

[40] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2016, arXiv preprint arXiv:1606.06160.

[41] A. Zhou, A. Yao, Y. Guo, L. Xu, Y. Chen, Incremental network quantization: Towards lossless cnns with low-precision weights, 2017, arXiv preprint arXiv:1702.03044.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.

[43] X. Zhang, X. Zhou, M. Lin, J. Sun, ShuffleNet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.

[44] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, C. Xu, GhostNet: More features from cheap operations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1580–1589.

[45] D. Kingma, J. Ba, Adam: A method for stochastic optimization, Comput. Sci. (2014).

[46] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Found Trends Mach. Learn. 3 (1) (2011) 1–122.

[47] X. Lu, H. Huang, W. Dong, X. Li, G. Shi, Beyond network pruning: a joint search-and-training approach, in: Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence IJCAI-PRICAI-20, 2020.

[48] A.P. French, A.C. Robinson, J.M. Wilson, Using a hybrid genetic-algorithm/branch and bound approach to solve feasibility and optimization integer programming problems, J. Heuristics 7 (6) (2001) 551–564.

[49] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2736–2744.

[50] A. Lipowski, D. Lipowska, Roulette-wheel selection via stochastic acceptance, Physica A 391 (6) (2012) 2193–2196.

[51] G. Hinton, A. Krizhevsky, Learning multiple layers of features from tiny images, Technical report, Citeseer, 2009.

[52] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[53] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, Int. J. Comput. Vis. (2017).
[54] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: International conference on machine learning, 2013, pp. 1139–1147.
[55] Z. Huang, N. Wang, Data-driven sparse structure selection for deep neural networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 304–320.
[56] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V.I. Morariu, X. Han, M. Gao, C.-Y. Lin, L.S. Davis, Nisp: Pruning networks using neuron importance score propagation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 9194–9203.
[57] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1389–1397.
[58] Z. Wang, F. Li, G. Shi, X. Xie, F. Wang, Network pruning using sparse learning and genetic algorithm, Neurocomputing 404 (2020) 247–256.