Signal, Image and Video Processing (2021) 15:1555–1563
https://doi.org/10.1007/s11760-021-01888-4

ORIGINAL PAPER

Filter pruning-based two-step feature map reconstruction

Yongsheng Liang[1] · Wei Liu[2] · Shuangyan Yi[3] · Huoxiang Yang[4] · Zhenyu He[2]

Received: 5 December 2019 / Accepted: 4 March 2021 / Published online: 31 March 2021

Abstract
In deep neural network compression, channel/filter pruning is widely used for compressing the pre-trained network by judging the redundant channels/filters. In this paper, we propose a two-step filter pruning method to judge the redundant channels/filters layer by layer. The first step is to design a filter selection scheme based on $\ell_{2,1}$-norm by reconstructing the feature map of current layer. More specifically, the filter selection scheme aims to solve a joint $\ell_{2,1}$-norm minimization problem, i.e., both the regularization term and feature map reconstruction error term are constrained by $\ell_{2,1}$-norm. The $\ell_{2,1}$-norm regularization plays a role in the channel/filter selection, while the $\ell_{2,1}$-norm feature map reconstruction error term plays a role in the robust reconstruction. In this way, the proposed filter selection scheme can learn a column-sparse coefficient representation matrix that can indicate the redundancy of filters. Since pruning the redundant filters in current layer might dramatically influence the output feature map of the following layer, the second step needs to update the filters of the following layer to assure output of feature map approximates to that of baseline. Experimental results demonstrate the effectiveness of this proposed method. For example, our pruned VGG-16 on ImageNet achieves 4× speedup with 0.95% top-5 accuracy drop. Our pruned ResNet-50 on ImageNet achieves 2× speedup with 1.56% top-5 accuracy drop. Our pruned MobileNet on ImageNet achieves 2× speedup with 1.20% top-5 accuracy drop.

Keywords Filter pruning · Channel pruning · Feature map reconstruction · $\ell_{2,1}$-norm

1 Introduction

In the past few years, we have witnessed a rapid development of convolutional neural networks [1–7]. In order to achieve higher accuracy, the general strategy is to make deeper and more complicated networks [8–12]. However, these strategies to improve accuracy are not efficient with respect to model size and speed. In many mobile terminal devices such as robotics, self-driving car and augmented reality, the recognition tasks need to be carried out in a timely fashion on a computationally limited platform [13–16].

There has been rising interest in building small and efficient neural networks in the recent literature [17–28]. Many different approaches can be generally categorized as two groups: (1) training small networks directly [22–27,29]; (2) compressing pre-trained networks [17–21,28,30].

The former aims to train a small network structure [22–27,29], where the popular method is MobileNets including three versions. More specifically, MobileNet-V1 adopts the depthwise separable convolution to greatly reduce the amount of computation and the number of parameters, thereby improving the computation efficiency. Based on MobileNet-V1, MobilieNet-V2 introduces the inverted residual structure with a linear bottleneck. Based on MobileNet-V1 and MobileNet-V2, MobileNet-V3 is proposed recently. However, the above MobileNets do not consider the case of

✉ Wei Liu
liuwei@sziit.edu.cn

Yongsheng Liang
liangys@hit.edu.cn

Shuangyan Yi
shuangyanshuangfei@163.com

Huoxiang Yang
yhuoxiang@gmail.com

Zhenyu He
zyhe@hitsz.edu.cn

1 School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, China

2 Research Center of Networks and Communications, Peng Cheng Laboratory, Shenzhen, China

3 School of Software Engineering, Shenzhen Institute of Information Technology, Shenzhen, China

4 School of Electronics and Information Engineering, Shenzhen University, Shenzhen, China

redundant filters. In fact, the redundancy in the filters takes up the computation in the process of forward and back propagation. Generally speaking, convolutional neural network has the high redundancy of filters [8,31]. Therefore, it would reduce the running time of neural networks by removing the redundant filters.

The latter aims to compress the pre-trained convolutional neural network (CNN), where the popular method is pruning. Pruning includes parameter pruning and channel/filter pruning. For most CNNs, convolutional layers are the most time-consuming part, while fully connected layers involve massive network parameters. Therefore, the parameter pruning aims to reduce the storage, while the channel/filter pruning aims to reduce the computation cost. Generally, parameter pruning may suffer from the irregular memory acquisition and eliminates the possibility of improving efficiency. Therefore, special hardware or software is needed to assist with the calculation, which may increase computation time [19,32–34]. To avoid the limitations of parameter pruning mentioned above, this paper focuses on studying the channel/filter pruning by removing the entire channels/filters [12,18–21,35,36], whose benefits of removing the redundant channels/filters can be seen from [12,35,36]. Lebedev and Lempitsky [18], Wen et al. [19] employ the group sparsity to select the redundant filters, but the bad convergence speed and structured filter generating speed will heavily influence the pruning efficiency. Max response [20] uses the ℓ_1-norm to calculate the sum of its absolute weights of a filter, and the high absolute weight sum means that the filter is important. Since max response measures the importance of filter one by one, it may ignore the correlations between different filters. To this end, channel pruning [21] aims to use ℓ_1-norm to indirectly select the redundant filters by using the feature map of current layer and the filter of the next layer to reconstruct the feature map of the next layer, which needs to solve a lasso problem and thus has a high computation complexity in terms of optimal solution.

In this paper, we propose a two-step feature map reconstruction method to prune the redundant filters and channels. In the proposed method, both the reconstruction term and the regularization term employ the $\ell_{2,1}$-norm to implement the learning task of filter pruning under the robust reconstruction. To the best of our knowledge, we are the first one to propose a filter pruning method based on two-step feature map reconstruction, where robust reconstruction and filter selection are simultaneously performed. Unlike most of filter pruning methods, our method is able to select the representative filters by two-step feature map reconstruction, so that the removed filters would not influence the following layers.

The remainder of this paper is organized as follows: In Sect. 2, we present the background. In Sect. 3, we present the proposed method and its optimal solution. In Sect. 4, we give the theoretical analysis of our method. In Sect. 5, we perform the experiments to demonstrate the effectiveness and efficiency of our method. Finally, a conclusion is drawn in Sect. 6.

2 Background

To prune a feature map with n_i channels, $n_{i+1} \times n_i \times k_h \times k_w$ convolutional filters $\boldsymbol{W}$ are often applied on $N \times n_i \times k_h \times k_w$ input volumes $\boldsymbol{X}$ sampled from this feature map of i-th layer, which produces $N \times n_{i+1}$ output matrix $\boldsymbol{Y}_{i+1}$. Here, N is the number of samples, n_{i+1} is the number of output channels, and k_h, k_w are the kernel size. For simple representation, bias term is ignored in the filter pruning methods. To prune the input channels from n_i to desired $n_i^{'}$ $(0 \le n_i^{'} \le n_i)$, while minimizing reconstruction error, the channel pruning method [21] is proposed as follows:

$$\begin{aligned} &\min_{\boldsymbol{\beta},\boldsymbol{W}} \left\| \boldsymbol{Y}_{i+1} - \sum_{c=1}^{n_i} \beta_c \boldsymbol{X}_c \boldsymbol{W}_c^T \right\|_F^2 + \lambda \|\boldsymbol{\beta}\|_1, \\ &s.t., \|\boldsymbol{\beta}\|_0 \le n_i^{'}, \forall i \|\boldsymbol{W}_c\|_F = 1. \end{aligned} \tag{1}$$

$\|\cdot\|_F$ is Frobenius norm. $\boldsymbol{X}_c$ is $N \times k_h \times k_w$ matrix sliced from c-th channel of input volumes $\boldsymbol{X}$, $c = 1, 2, \ldots, n_i$. $\boldsymbol{W}_c$ is $n_i \times k_h \times k_w$ filter weights sliced from c-th channel of $\boldsymbol{W}$. $\boldsymbol{\beta}$ is coefficient vector of length n_i for channel selection, and β_c (c-th entry of $\boldsymbol{\beta}$) is a scalar mask to c-th channel (i.e., to drop the whole channel or not).

Similar to the above channel pruning method [21], some other filter-level pruning methods [12,20,30,35] also have been explored. The core of the filter pruning is to measure the importance of each filter. The major difference of filter pruning is the selection strategy: Max response [20] calculates the absolute weight sum of each filter (i.e., $\sum \boldsymbol{W}(i, :, :, :)$, where i means the i-th filter, $i \in \{1, 2, \ldots, n_{i+1}\}$) as its importance score. ThiNet [12,35] first uses a greedy strategy to search a subset of feature map such that the output by some channels is almost same with that by all the channels. More specifically, ThiNet aims to search a subset of feature map by minimizing the following reconstruction error.

$$\begin{aligned} &\min_{S} \sum_{d=1}^{N} \left(\boldsymbol{Y}_{i+1}^d - \sum_{c \in S} \boldsymbol{X}_c^d \boldsymbol{W}_c^{dT} \right)^2, \\ &s.t., |S| = n_i \times r, S \subset \{1, 2, \ldots, n_i\} \end{aligned} \tag{2}$$

where d is the sampling number, r is the compression ratio, S is the subset of feature map-based channels, and $|S|$ is the number of elements in a subset S.

After obtaining the subset S, the redundant channels of feature map $\boldsymbol{X}_c^d$ and filter $\boldsymbol{W}_c^d$ are removed. For simplicity, we call the feature map and filter without redundancy as $\hat{\boldsymbol{X}}_c^d$

and $\hat{\boldsymbol{W}}_c^d$. ThiNet further minimizes the reconstruction error by assigning weights $\boldsymbol{q}$ for $\hat{\boldsymbol{W}}_c^d$.

$$\min_{\boldsymbol{w}} \sum_{d=1}^{N} (\boldsymbol{Y}_{i+1}^d - \sum_{c \in S} \hat{\boldsymbol{X}}_c^d \hat{\boldsymbol{W}}_c^{dT} \boldsymbol{q})^2, \quad s.t., |S| = n_i \times r, S \subset \{1, 2, \ldots, n_i\}. \tag{3}$$

It is worth noting that both channel pruning method and ThiNet method are driven by data to demonstrate the effectiveness of filter selection strategy, and first k and max response are non-data-driven methods. Besides, HRank [30], as a data-driven method, is proposed as follows:

$$\min_{\delta_{ij}} \sum_{i=1}^{K} \sum_{j=1}^{n_i} \delta_{ij}(\boldsymbol{w}_j^i) \sum_{t=1}^{g} Rank(o_j^i(t, :, :)), s.t., \sum_{j=1}^{n_i} \delta_{ij} = n_{i2}. \tag{4}$$

where K means the number of convolutional layers, n_i represents the number of filters in the i-th convolutional layer, δ_{ij} is an indicator which is 1 if the j-th filter in the i-th layer (i.e., $\boldsymbol{w}_j^i$) is unimportant or 0 if $\boldsymbol{w}_j^i$ is important, g means the number of input images, $o_j^i(t, :, :)$ means the feature map generated by $\boldsymbol{w}_j^i$, and n_{i2} means the number of least important filters in the i-th layer.

3 Building model of filter pruning

Formally, for one input image, let n_i denote the number of input channels for the i-th convolutional layer and h_i, w_i be the height and width of the input feature maps. The convolutional layer transforms the input feature map $\boldsymbol{y}_i \in \mathbb{R}^{n_i \times h_i \times w_i}$ into the output feature map $\boldsymbol{y}_{i+1} \in \mathbb{R}^{n_{i+1} \times h_{i+1} \times w_{i+1}}$, which are used as input feature maps for the next convolutional layer. This is achieved by applying n_{i+1} 3D filters $\boldsymbol{F}_{i,j} \in \mathbb{R}^{n_i \times k \times k}$ (All the filters, together, constitute the filter matrix $\boldsymbol{F}_{i+1} \in \mathbb{R}^{n_{i+1} \times n_i \times k \times k}$) on the n_i input channels, in which one filter generates one feature map channel. The number of operations of the convolutional layer is $n_{i+1} n_i k^2 h_{i+1} w_{i+1}$. If a filter $\boldsymbol{F}_{i,j}$ is pruned, its corresponding feature map $x_{i+1,j}$ is removed, which reduces $n_i k^2 h_{i+1} w_{i+1}$ operations. The filters that apply on the removed feature map channels from the filters of the next convolutional layer are also removed, which saves an additional $n_{i+2} k^2 h_{i+2} w_{i+2}$ operations.

Furthermore, if there are m input images, they will produce the feature map, such as the i-th feature map $\boldsymbol{y}_i \in \mathbb{R}^{m \times n_i \times h_i \times w_i}$, and the $i+1$-th feature map $\boldsymbol{y}_{i+1} \in \mathbb{R}^{m \times n_{i+1} \times h_{i+1} \times w_{i+1}}$. For simplicity, we sample from $\boldsymbol{y}_i$ and generate $\boldsymbol{Y}_i \in \mathbb{R}^{N_i \times n_i}$. The detailed sampling way can refer [21]. Here, N_i is the number of samples of i-th layer, and n_i is the channel number of feature map of i-th layer. To prune the output channels from n_i to desired $n_i^{'}$, while minimizing reconstruction error, we formulate the proposed objective function as follows:

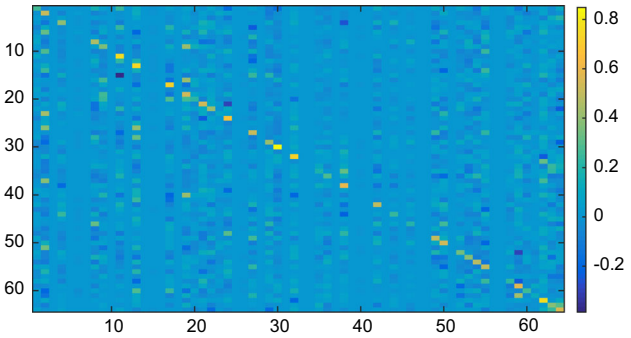

Fig. 1 Illustration of redundant filters learned automatically by our method

$$\min_{\boldsymbol{b},\boldsymbol{A}} \left\| (\boldsymbol{Y}_i^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}_i^T - \boldsymbol{b}\boldsymbol{1}^T) \right\|_{2,1} + \lambda \|\boldsymbol{A}\|_{2,1}. \tag{5}$$

where $\boldsymbol{Y}_i$ is $N_i \times n_i$ matrix. $\boldsymbol{A} \in \mathbb{R}^{n_i \times n_i}$ is a coefficient representation matrix. The designed objective function can make $\boldsymbol{A}$ be column-sparse, and thus, it can indicate the redundancy of channels of feature map and filters in current layer (see Fig. 1).

Without loss of generality, we remove the layer index i, and thus, our objective function can be rewritten as follows:

$$\min_{\boldsymbol{b},\boldsymbol{A}} \left\| (\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) \right\|_{2,1} + \lambda \|\boldsymbol{A}\|_{2,1}. \tag{6}$$

Using some mathematical techniques, problem (6) can be rewritten as

$$\min_{\boldsymbol{b},\boldsymbol{A}} \left\| \left(\left(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T \right) - \boldsymbol{A} \left(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T \right) \right) \sqrt{\boldsymbol{W}_1} \right\|_F^2 + \lambda \left\| \boldsymbol{A} \sqrt{\boldsymbol{W}_2} \right\|_F^2, \tag{7}$$

where $\boldsymbol{W}_2 \in \mathbb{R}^{n \times n}$ and $\boldsymbol{W}_1 \in \mathbb{R}^{N \times N}$ are two diagonal matrices, whose diagonal elements are $\boldsymbol{W}_2^{cc} = \frac{1}{2\|(\boldsymbol{A})^c\|_2}$ and $\boldsymbol{W}_1^{cc} = \frac{1}{2\left\|((\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T))^c\right\|_2}$, respectively. $(\boldsymbol{A})^c$ means the c-th column of matrix $\boldsymbol{A}$. When $\|(\boldsymbol{A})^c\|_2 = 0$, we let $\boldsymbol{W}_2^{cc} = \frac{1}{2\|(\boldsymbol{A})^c\|_2 + \zeta}$. ($\zeta$ is a very small constant.) Similarly, when $\left\|((\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T))^c\right\|_2 = 0$, we let $\boldsymbol{W}_1^{cc} = \frac{1}{2\left\|((\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T))^c\right\|_2 + \zeta}$. In this way, the smaller $\boldsymbol{W}_1^{cc}$ is, the higher possibility to be outliers the c-th response has. The smaller $\boldsymbol{W}_2^{cc}$ is, the more important the c-th filter is. Here, $\sqrt{\boldsymbol{W}_1}$ gives the weights of the responses. The clean responses are weighted more heavily, while the responses that are outliers are weighted less heavily. This leads to the robustness of our method to outliers. On the other hand, the regularization term $\boldsymbol{A}\sqrt{\boldsymbol{W}_2}$ can guide the selection

of filters. Through adjusting the parameter λ, our method can select the effective filters under the robust reconstruction criterion. Moreover, it can be seen that the minimization of $2tr((\boldsymbol{Y}^T - (\boldsymbol{A}\boldsymbol{Y}^T + \boldsymbol{b}\boldsymbol{1}^T))\boldsymbol{W}_1(\boldsymbol{Y}^T - (\boldsymbol{A}\boldsymbol{Y}^T + \boldsymbol{b}\boldsymbol{1}^T))^T + 2\lambda tr(\boldsymbol{A}$ forces $\left\|((\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T))^c\right\|_2$ and $\|(\boldsymbol{A})^c\|_2$ to be very small when $\boldsymbol{W}_2$ and $\boldsymbol{W}_1$ are large. Finally, some columns of $(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T)$ and $\boldsymbol{A}$ may be close to zero, and thus, a column-sparse $(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y}^T - \boldsymbol{b}$ and $\boldsymbol{A}$ can be obtained.

Our goal is to remove some redundant output channels without the loss of the performance. After we design an algorithm to judge the redundant channels and filters and then prune them, we should assure that the feature map of next layer is almost kept so that the removed channels does not influence the final classification result. Therefore, we need to reconstruct the filters in next layer with current remaining channels by linear least squares, whose objective function is shown as follows:

$$\min_{\boldsymbol{F}'_{i+1}} \left\| \boldsymbol{Y}_{i+1} - (\boldsymbol{Y}'_i)(\boldsymbol{F}'_{i+1})^T \right\|_F^2. \tag{8}$$

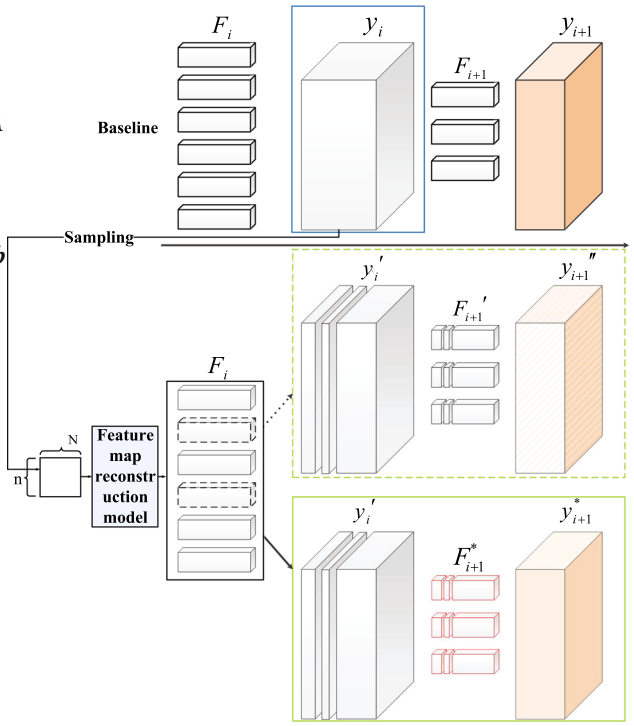

Fig. 2 Flowchart of the proposed neural network compression method

where $\boldsymbol{Y}_{i+1}$ means the feature map of $i+1$-th layer, $\boldsymbol{Y}'_i$ means the feature map of i-th layer after the removal of redundant channels, and $\boldsymbol{F}'_{i+1}$ means the filters of $i+1$-th layer after the removal of redundant channels. Here, $\boldsymbol{F}'_{i+1}$ is $n_{i+1} \times n_i kk$ reshaped $\boldsymbol{F}_{i+1}$. It is worth noting that if r channels are redundant in $\boldsymbol{Y}_i$, $\boldsymbol{Y}'_i \in \mathbb{R}^{N\times(n_i-r)}$, $\boldsymbol{F}'_{i+1} \in \mathbb{R}^{n_{i+1}\times(n_i-r)}$.

To sum up, the flowchart is given in Fig. 2, which mainly includes two steps: One is to judge the redundant filters by reconstructing the feature map of the current layer, and the second step is to learn the new filters by reconstructing the feature map of the next layer. Our method is proceeded layer by layer. For one layer such as $i+1$-th layer, the original computation cost is $n_{i+1}n_i k^2 h_{i+1}w_{i+1}$ flops, while the remained computation cost is $(n_{i+1} - r_f)(n_i - r_c)k^2 h_{i+1}w_{i+1}$ flops.

Discussion: Some recent works [20,21] also introduce the sparse norm, such as ℓ_1-norm [20] or Lasso [21]. However, we must emphasize that we use different formulations and different ideas. Lasso [21] uses the current filters and the previous feature map to reconstruct the feature map of current layer and add the sparse constraint on each channel, but the computation complexity of their model is very high. Moreover, both of them [20,21] need to give the value of sparsity n'_i. Different from Lasso, we perform robust reconstruction for the feature map of current layer. If the feature map has the redundancy, our model can automatically conclude the redundant filters of its previous layer. Furthermore, we need to assure that the remaining filters can recover the feature map of next layer. Besides, they [20,21] use ℓ_1-norm to select the redundant channel, while we use $\ell_{2,1}$-norm to select the redundant channel from the perspective of feature map of current layer.

3.1 The optimal solution of problem (6)

The global optimal solution of problem (7) can be easily obtained by using an iterative re-weighting method, which includes the following two steps.

Step 1: Given $\boldsymbol{A}$, we compute $\boldsymbol{b}$. The optimization problem (6) becomes,

$$\min_{\boldsymbol{b}} \left\| (\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T) - \boldsymbol{A}(\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T) \right\|_{2,1}. \tag{9}$$

Setting the derivative of (9) with respect to $\boldsymbol{b}$ to be zero, we get $\boldsymbol{v} = \frac{(\boldsymbol{X}\boldsymbol{W}_1 - \boldsymbol{A}\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{1}}{\boldsymbol{1}^T\boldsymbol{W}_1\boldsymbol{1}}$.

Step 2: Given $\boldsymbol{b}$, we compute $\boldsymbol{A}$. The optimization problem (7) becomes,

$$\min_{\boldsymbol{A}} \left\| \left(\left(\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T \right) - \boldsymbol{A}(\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T) \right) \sqrt{\boldsymbol{W}_1} \right\|_F^2 + \lambda \left\| \boldsymbol{A}\sqrt{\boldsymbol{W}_2} \right\|_F^2. \tag{10}$$

Setting the derivative of (10) with $\boldsymbol{A}$ to be zero, we get $\boldsymbol{A} = (\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T)\boldsymbol{W}_1(\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T)^T(\lambda\boldsymbol{W}_2 + (\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T)\boldsymbol{W}_1 (\boldsymbol{Y} - \boldsymbol{b}\boldsymbol{1}^T)^T)^{-1}$.

Iterating the above two steps will reach the global optimal solution. Algorithm 1 gives more details.

Algorithm 1. Optimization Algorithm of Problem (6)

Input: Feature map $\boldsymbol{Y}$, parameter λ;
1: Initialize $\boldsymbol{W}_1 = \boldsymbol{I}$, $\boldsymbol{W}_2 = \boldsymbol{I}$ and $\boldsymbol{b} = \boldsymbol{0}$;
2: **while** not converge **do**
2.1: Compute $\boldsymbol{A} = (\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T)\boldsymbol{W}_1(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T)^T (\lambda \boldsymbol{W}_2 + (\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T)\boldsymbol{W}_1(\boldsymbol{Y}^T - \boldsymbol{b}\boldsymbol{1}^T)^T)^{-1}$;
2.2: Compute $\boldsymbol{b} = \frac{\boldsymbol{Y}^T\boldsymbol{W}_1\boldsymbol{1}}{\boldsymbol{1}^T\boldsymbol{W}_1\boldsymbol{1}}$;
2.3: Compute $\boldsymbol{W}_1$
2.4: Compute $\boldsymbol{W}_2$
end while
Output: Representation matrix $\boldsymbol{A}$, optimal mean vector $\boldsymbol{b}$.

4 Theoretical analysis

4.1 Convergence analysis

Before giving the convergence proof of the optimization algorithm, we need to first give Lemma 1 [37].

Lemma 1 *For any nonzero vectors* $\boldsymbol{U}, \boldsymbol{q} \in \mathbb{R}^d$,

$$\|\boldsymbol{U}\|_2 - \frac{\|\boldsymbol{U}\|_2^2}{2\|\boldsymbol{q}\|_2} \leq \|\boldsymbol{q}\|_2 - \frac{\|\boldsymbol{q}\|_2^2}{2\|\boldsymbol{q}\|_2}. \tag{11}$$

Based on Lemma 1, we prove Theorem 1.

Theorem 1 *Algorithm 1 will monotonically decrease the value of the objective function of the optimization problem (7) in each iteration and converge to a local optimal solution.*

Proof For simplicity, we denote the updated $\boldsymbol{b}$ and $\boldsymbol{A}$ by $\widetilde{\boldsymbol{b}}$ and $\widetilde{\boldsymbol{A}}$. Since the updated $\widetilde{\boldsymbol{b}}$ and $\widetilde{\boldsymbol{A}}$ are the optimal solution of problem (5), according to the definition of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, we have

$$\begin{aligned} & tr\left(\sum_{i=1}^{N} \frac{\left\|\boldsymbol{x}_i - \widetilde{\boldsymbol{A}}\boldsymbol{x}_i - (\boldsymbol{I} - \widetilde{\boldsymbol{A}})\widetilde{\boldsymbol{b}}\right\|_2^2}{2\left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\right\|_2}\right) + \lambda tr\left(\sum_{i=1}^{n} \frac{\|\widetilde{\boldsymbol{a}}_i\|_2^2}{2\|\boldsymbol{a}_i\|_2}\right) \\ \leq \; & tr\left(\sum_{i=1}^{N} \frac{\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\|_2^2}{2\left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\right\|_2}\right) + \lambda tr\left(\sum_{i=1}^{n} \frac{\|\boldsymbol{a}_i\|_2^2}{2\|\boldsymbol{a}_i\|_2}\right). \end{aligned} \tag{12}$$

On the one hand, according to Lemma 1, we have

$$\begin{aligned} & \left\|\boldsymbol{x}_i - \widetilde{\boldsymbol{A}}\boldsymbol{x}_i - (\boldsymbol{I} - \widetilde{\boldsymbol{A}})\widetilde{\boldsymbol{b}}\right\|_2 - \frac{\left\|\boldsymbol{x}_i - \widetilde{\boldsymbol{A}}\boldsymbol{x}_i - (\boldsymbol{I} - \widetilde{\boldsymbol{A}})\widetilde{\boldsymbol{b}}\right\|_2^2}{2\left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\right\|_2} \\ \leq \; & \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\|_2 - \frac{\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\|_2^2}{2\left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\right\|_2}. \end{aligned} \tag{13}$$

Using matrix calculus for problem (13), we have the following formulation:

$$\begin{aligned} & \sum_{i=1}^{N} \left\|\boldsymbol{x}_i - \widetilde{\boldsymbol{A}}\boldsymbol{x}_i - (\boldsymbol{I} - \widetilde{\boldsymbol{A}})\widetilde{\boldsymbol{b}}\right\|_2 - \sum_{i=1}^{N} \frac{\left\|\boldsymbol{x}_i - \widetilde{\boldsymbol{A}}\boldsymbol{x}_i - (\boldsymbol{I} - \widetilde{\boldsymbol{A}})\widetilde{\boldsymbol{b}}\right\|_2^2}{2\left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\right\|_2} \\ \leq \; & \sum_{i=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\|_2 - \sum_{i=1}^{N} \frac{\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\|_2^2}{2\left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{b}\right\|_2}. \end{aligned} \tag{14}$$

On the other hand, according to Lemma 1, we have

$$\|\widetilde{\boldsymbol{a}}_i\|_2 - \frac{\|\widetilde{\boldsymbol{a}}_i\|_2^2}{\|\boldsymbol{a}_i\|_2} \leq \|\boldsymbol{a}_i\|_2 - \frac{\|\boldsymbol{a}_i\|_2^2}{\|\boldsymbol{a}_i\|_2}. \tag{15}$$

Similarly, using matrix calculus for problem (15), we have the following formulation:

$$\begin{aligned} & \sum_{i=1}^{n} \left(\|\widetilde{\boldsymbol{a}}_i\|_2 - \frac{\|\widetilde{\boldsymbol{a}}_i\|_2^2}{2\|\boldsymbol{a}_i\|_2}\right) \\ \leq \; & \sum_{i=1}^{n} \left(\|\boldsymbol{a}_i\|_2 - \frac{\|\boldsymbol{a}_i\|_2^2}{2\|\boldsymbol{a}_i\|_2}\right). \end{aligned} \tag{16}$$

By combining problem (12) and problem (14) with problem (16), we have

$$\begin{aligned} & \left\|\boldsymbol{x}_i - \widetilde{\boldsymbol{A}}\boldsymbol{x}_i - (\boldsymbol{I} - \widetilde{\boldsymbol{A}})\widetilde{\boldsymbol{b}}\right\|_{2,1} + \lambda\left\|\widetilde{\boldsymbol{A}}\right\|_{2,1} \\ \leq \; & \left\|\boldsymbol{x}_i - \boldsymbol{A}\boldsymbol{x}_i - (\boldsymbol{I} - \boldsymbol{A})\widetilde{\boldsymbol{b}}\right\|_{2,1} + \lambda\|\boldsymbol{A}\|_{2,1}. \end{aligned} \tag{17}$$

Since problem (5) has an obvious lower bound 0, the optimization problem (5) converges to the global optimal solution.

4.2 Computational complexity analysis

The main computational complexity of Problem (6) has two steps in each iteration: The first step is to compute $\boldsymbol{b}$, whose computational complexity is $O(n^3)$; The second step is to compute $\boldsymbol{A}$, whose computational complexity is also $O(n^3)$ at most. Therefore, the computational complexity of one iteration will be up to $O(n^3)$. If Algorithm 1 needs t iterations, the total computational complexity is on the order of $O(tn^3)$.

5 Experiments

We prune the filters of three types of networks, i.e., VGG-16 [6], ResNet-50 [38] and MobileNet [22], which is implemented on ImageNet [39], CIFAR-10 [40] and CIFAR-100 [40]. ImageNet comprises 1.28 million training images and 50000 validation images from 1000 classes. We fine-tune networks on the training set and report the accuracy on the

validation set with the shorter side of images resized to 256. For data augmentation, we follow the standard practice [21] and perform the random size cropping to 224×224 and random horizontal flipping, and more experimental details can refer [21]. CIFAR-10 consists of 10 classes images, and each class consists of 6000 images, where 50000 images are for training and 10000 for validation. Similarly, CIFAR-100 consists of 100 classes images and each class consists of 600 images, where 50000 images are for training and 10000 for validation. On CIFAR-10 and CIFAR-100 datasets, we fine-tune networks with the size of training images which are resized to 32x32, and with the per-pixel mean subtracted on the training and validation set. For data augmentation, we adopt random horizontal flipping.

Our method is compared to the classical first-k and max response [20], the state-of-the-art channel pruning [21], Thin Net [35] and HRank [30] that are similar to our method to some extent.

Implementation: Our method is performed on the network layer by layer. In our method, there is a parameter λ, and thus, our method involves in the process of parameter selection. More specifically, when our method performs parameter selection on a layer, the other layers are fixed as baseline. One common way of parameter determination is the grid searching method. We vary this parameter within a certain range $\{10^{-6}, 10^{-5}, \ldots, 10^{6}\}$, and the value of parameter with the highest classification result is considered as the most optimal parameter. After the parameters of our method on all the layers of a network are determined, the compressed network is obtained. Theoretically, if the redundant filters of i-th layer are removed, the updated filter of the $i+1$-th layer almost can recover the feature map of the $i+1$-th layer, and thus, the final classification performance can be preserved. For a fair comparison, all the methods adopt the same speedup ratio. For example, all the methods use the 2 times of speedup ratio (i.e., 2×) on the ResNet-50. More specifically, based on the 2 times of speedup ratio on the ResNet-50, the effective filters number is first acquired by channel pruning method [21], and then, the same effective filters number acquired by channel pruning method [21] is adopted by all the other methods including ours.

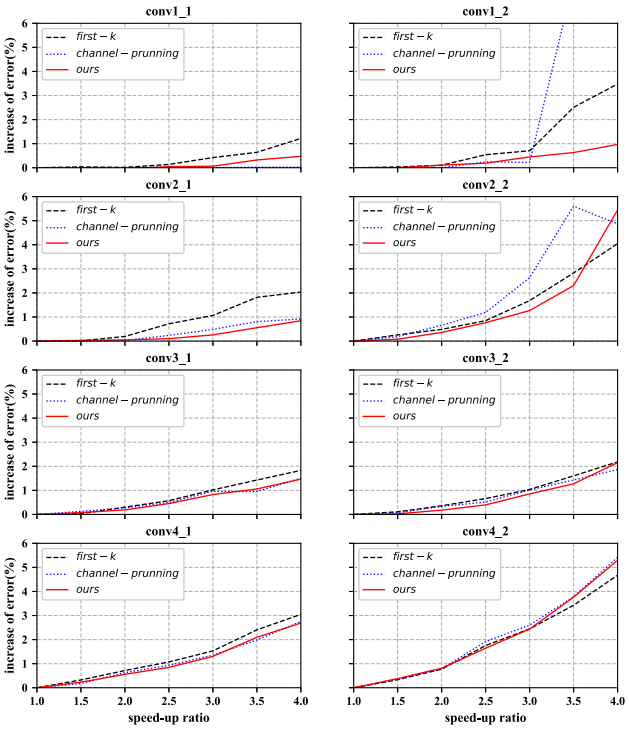

Fig. 3 Result of single layer under different speedup ratios (without fine-tuning)

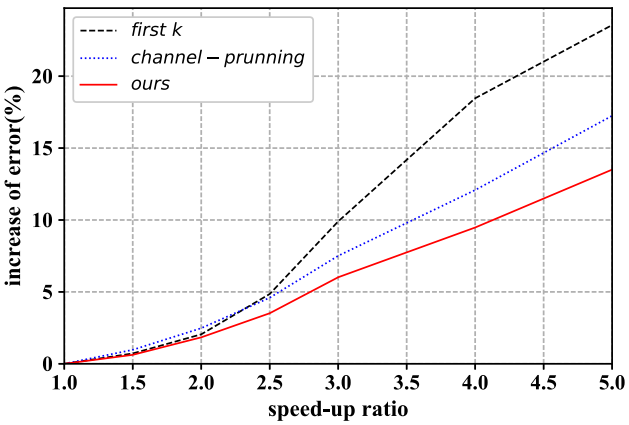

Fig. 4 Result of whole model under different speedup ratios on VGG-16(without fine-tuning)

5.1 VGG-16 pruning

5.1.1 Experimental results of single layer

We implement three methods to compress VGG-16 network, and the experimental results are shown in Fig. 3. It can be seen from Conv2_1 layer that, with the increase in speedup ratio, the classification accuracy of three methods drops dramatically. However, our method outperforms the other methods when the speedup ratio is from 2× to 4×, where 2× means that the running time of compressed network is 0.5 times of that of baseline network. At this moment, the classification accuracy drops about from 0.1 to 0.84% compared to the classification accuracy of baseline. More specifically, compared to baseline, when the classification accuracy drops 0.84%, our method has a 4× speedup ratio (i.e., our flops are 25% of baseline).

5.1.2 Experimental results across all layers

Guided by the experimental results of single layer, we observe that there is a big redundancy on the first sev-

Table 1 Fine-tuning results (accuracy drops) of VGG-16 on ImageNet with 4× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	3.9B	131.7M	− 2.23	− 1.27
Max response	3.9B	131.7M	− 1.91	− 1.26
Channel pruning	3.9B	131.7M	− 2.04	− 1.21
Thin Net	3.9B	131.7M	− 2.04	− 1.22
HRank	3.9B	131.7M	− 2.15	− 1.09
Ours	3.9B	134.1M	− **1.86**	− **0.95**

The bold fonts indicate the best result

Table 2 Fine-tuning results (accuracy drops) of VGG-16 on CIFAR-10 with 4× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	79.5M	4.2M	− 1.23	− 0.16
Max response	79.5M	4.2M	− 0.25	+ **0.1**
Channel pruning	79.5M	4.2M	− 0.09	+ 0.06
ThiNet	79.5M	4.2M	− 0.34	− 0.11
HRank	79.5M	4.2M	− 0.37	− 0.06
Ours	79.5M	4.2M	− **0.07**	− 0.04

The bold fonts indicate the best result

Table 3 Fine-tuning results (accuracy drops) of VGG-16 on CIFAR-100 with 4× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	79.5M	4.4M	− 6.49	− 3.71
Max response	79.5M	4.4M	− 5.25	− 2.61
Channel pruning	79.5M	4.4M	− 4.55	− 2.30
ThiNet	79.5M	4.4M	− 4.42	− 1.77
HRank	79.5M	4.4M	− 5.73	− 2.59
Ours	79.5M	4.4M	− **3.56**	− **1.47**

The bold fonts indicate the best result

eral layers of VGG-16, while its last layers are not very redundant. To this end, we prune more filters on shallow layers while remaining the origin filters on conv5_x layers, and the detailed filter pruning case can refer channel pruning [21]. The experimental results without fine-tuning are shown in Fig. 4, which shows that three methods obtain the similar results when the speedup ratio is small. With the increase in speedup ratio, the advantage of our method is highlighted. The experimental results with fine-tuning are shown in Tables 1, 2 and 3. It can be seen that, with the same speedup ratio (thus the flops (i.e., flops) and parameters (i.e., #Param) of all the methods are same), our method outperforms the other methods with respect to top-1 classification accuracy.

Table 4 Fine-tuning results (accuracy drops) of ResNet-50 on ImageNet with 2× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	1.7B	12.3M	− 6.71	− 3.63
Max response	1.7B	12.3M	− 5.14	− 2.26
Channel pruning	1.7B	12.3M	− 4.83	− 2.23
ThiNet	1.7B	12.3M	− **3.48**	− 1.58
HRank	1.7B	12.3M	− 3.86	− 2.46
Ours	1.7B	12.3M	− 3.54	− **1.56**

The bold fonts indicate the best result

Table 5 Fine-tuning results (accuracy drops) of ResNet-50 on CIFAR-10 with 2× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	65.0B	448.6M	− 1.09	+ 0.08
Max response	65.0B	448.6M	− 0.51	+ 0.06
Channel pruning	65.0B	448.6M	− 0.55	+ 0.06
ThiNet	65.0B	448.6M	− 0.53	− 0.21
HRank	65.0B	448.6M	− 0.39	+ **0.12**
Ours	65.0B	448.6M	− **0.27**	− 0.12

The bold fonts indicate the best result

5.2 ResNet pruning

We also apply the pruning methods on ResNet that is a multi-path network. The structure is more complex than VGG-16. Through the experiments of single layers, we observe that there is a big redundancy on the shallow layers. To this end, we prune the branch2a and branch2b layers but remain the branch2c layer in this network. The detailed filter pruning case on ResNet-50 can refer channel pruning [21]. We compared the four pruning methods (i.e., first k, max response [20], channel pruning [21], ThiNet [35] and HRank [30]) with 2× speedup ratio, and the experimental results are shown in Tables 4, 5 and 6, where the negative value means the accuracy is higher than baseline. It can be seen that with the same speedup ratio, our method is comparable with channel pruning, ThiNet and HRank, but outperforms the classical first k and max response methods. For example, in Table 4, our method obtains the better top-1 classification accuracy than first k, max response, channel pruning and HRank, but worse than ThiNet. However, we obtain the better top-5 classification accuracy than ThiNet. Therefore, we say our method is comparable with ThiNet. In Table 6, our method obtains the worse top-1 classification accuracy than channel pruning and HRank, while we obtain the better top-5 classification accuracy than HRank. Therefore, we say our method is comparable with HRank.

Table 6 Fine-tuning results (accuracy drops) of ResNet-50 on CIFAR-100 with 2× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	64.4M	429.8K	− 4.45	− 0.05
Max response	64.4M	429.8K	− 2.90	− 1.94
Channel pruning	64.4M	429.8K	− **1.34**	+ **0.24**
ThiNet	64.4M	429.8K	− 2.22	− 1.32
HRank	64.4M	429.8K	− 1.57	− 0.20
Ours	64.4M	429.8K	− 2.08	− 0.08

Table 7 Fine-tuning results (accuracy drops) of MobileNet on ImageNet with 1.5× speedup ratio

	Flops	#Param	Top-1 (%)	Top-5 (%)
First k	0.36B	3.4M	− 2.14	− 1.46
Max response	0.36B	3.4M	− 2.05	− 1.27
Channel pruning	0.36B	3.4M	− 2.15	− 1.29
ThiNet	0.36B	3.4M	− 2.14	− 1.32
HRank	0.36B	3.4M	− 1.94	− 1.33
Ours	0.36B	3.4M	− **1.79**	− **1.20**

The bold fonts indicate the best result

5.3 MobileNet pruning

As a lightweight network, MobileNet does not have a high degree of redundancy. The detailed filter pruning case on MobileNet can refer the strategy used in channel pruning [21]. We compared the four pruning methods (i.e., first k, max response [20], channel pruning [21], ThiNet [35] and HRank [30]) with 1.5× speedup ratio. The experimental results in Table 7 show that our method outperforms the other methods with the same speedup ratios.

6 Conclusion

In this paper, we propose a two-step feature map reconstruction method to prune the redundant filters and channels, which is used to compress the CNN networks, such as VGG-16, ResNet-50 and MobileNet. The experimental results on different networks with different datasets show the effectiveness of our method.

Acknowledgements This research was supported by the Shenzhen Research Council (KJYY20170724152625446), by the National Natural Science Foundation of China (Grant Nos. 61871154, 61906124, 61906103, 62031013), by the Chinese Postdoctoral Science Foundation (Grant No. 2018M630158) and by the Scientific Research Platform Cultivation Project of SZIIT (PT201704).

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS, pp. 1097–1105 (2012)
2. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Comput. **1**(2), 270–280 (1989)
3. Jia, Y., Shelhamer, E., Donahue, J.: Caffe: Convolutional architecture for fast feature embedding. In: ACM MM, pp. 675–678 (2014)
4. Ciresan, D.C., Meier, U., Masci, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI, p. 1237 (2011)
5. Szegedy, C., Liu, W., Jia, Y.: Going deeper with convolutions. In: CVPR, pp. 1–9 (2015)
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
7. Taigman, Y., Yang, M., Ranzato, M.A.: Deepface: Closing the gap to human-level performance in face verification. In: CVPR, pp. 1701–1708 (2014)
8. Shang, W., Sohn, K., Almeida, D.: Understanding and improving convolutional neural networks via concatenated rectified linear units. In: ICML, pp. 2217–2225 (2016)
9. Chatfield, K., Simonyan, K., Vedaldi, A.: Return of the devil in the details: delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
10. Lin, M., Chen, Q., Yan, S.: Network in network. arXiv preprint arXiv:1312.4400 (2013)
11. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, pp. 2278–2324 (1998)
12. Luo, J.H., Wu, J., Lin, W.: Thinet: A filter level pruning method for deep neural network compression. In: ICCV, pp. 5058–5066 (2017)
13. Liu, X., Zhou, Y., Zhao, J., Yao, R., Liu, B., Ma, D., Zheng, Y.: Multiobjective ResNet pruning by means of EMOAs for remote sensing scene classification. Neurocomputing **381**, 298–305 (2020)
14. Zou, J., Rui, T., Zhou, Y., Yang, C., Zhang, S.: Convolutional neural network simplification via feature map pruning. Comput. Electr. Eng.. **70**, 950–958 (2018)
15. Singh, P., Kadi, V.S.R., Namboodiri, V.P.: Convolutional neural network simplification via feature map pruning. Image Vis. Comput. **93**, 1–14 (2019)
16. Mao, Y., He, Z., Ma, Z., Tang, X., Wang, Z.: Efficient convolution neural networks for object tracking using separable convolution and filter pruning. IEEE Access. **7**, 106466–106474 (2019)
17. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. In: CVPR, pp. 1–12 (2014)
18. Lebedev, V., Lempitsky, V.: Fast convnets using group-wise brain damage. In: CVPR, pp. 2554–2564 (2016)

19. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: NIPS, pp. 2074–2082 (2016)
20. Li, H., Kadav, A., Durdanovic, I., Samet, H.: Pruning filters for efficient convnets. In: ICLR, pp. 1–13 (2016)
21. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: ICCV, pp. 1–10 (2017)
22. Howard, A.G., Zhu, M., Chen, B.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: CVPR, pp. 1–9 (2017)
23. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C.: Mobilenetv 2: Inverted residuals and linear bottlenecks. In: CVPR, pp. 1–14 (2018)
24. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. In: CVPR, pp. 1–5 (2016)
25. Jin, J., Dundar, A., Culurciello, E.: Flattened convolutional neural networks for feedforward acceleration. In: ICLR, pp. 1–11 (2015)
26. Rastegari, M., ordonez, V., Redmon, J., Farhadi, A.: Xnornet: Imagenet classification using binary convolutional neural networks. In: CVPR, pp. 1–17 (2016)
27. Wang, M., Liu, B., Foroosh, H.: Factorized convolutional neural networks. In: CVPR, pp. 1–10 (2016)
28. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: CVPR, pp. 4820–4828 (2016)
29. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: CVPR, pp. 6848–6856 (2018)
30. Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., Shao, L.: HRank: Filter pruning using high-rank feature map. In: CVPR (2020)
31. Misha, D., Babak, S., Laurent, D.: Predicting parameters in deep learning. In: NIPS, pp. 2148–2156 (2013)
32. Anwar, S., Hwang, K., Sung, W.: Structured pruning of deep convolutional neural networks. ACM J. Emerg. Technol. Comput. Syst. **13**(3), 1–18 (2017)
33. Yang, C., Yang, Z., Khattak, A.M., Yang, L., Zhang, W., Gao, W., Wang, M.: Structured pruning of convolutional neural networks via L1 regularization. IEEE Access. **7**, 106385–106394 (2019)
34. Garg, I., Panda, P., Roy, K.: A low effort approach to structured CNN design using PCA. IEEE Access. **8**, 1347–1360 (2020)
35. Luo, J., Zhang, H., Zhou, H., Xie, C., Wu, J., Lin, W.: Thinet: pruning CNN filters for a thinner net. IEEE Trans. Pattern Anal. Mach. Intell. **41**(10), 2525–2538 (2018)
36. Yang, W., Jin, L., Wang, S., Cu, Z., Chen, X., Chen, L.: Thinning of convolutional neural network with mixed pruning. IET Image Proc. **13**(5), 779–784 (2019)
37. Zheng, Z.: Sparse locality preserving embedding. In: CISP, pp. 1–5 (2009)
38. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR, pp. 770–778 (2016)
39. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Li, F.: Imagenet: A large-scale hierarchical image database. In: CVPR, pp. 248–255 (2009)
40. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)