Machine Learning (2022) 111:831–851
https://doi.org/10.1007/s10994-021-06077-5

CMD: controllable matrix decomposition with global optimization for deep neural network compression

Haonan Zhang[1] · Longjun Liu[1] · Hengyi Zhou[1] · Hongbin Sun[1] · Nanning Zheng[1]

Received: 14 May 2021 / Revised: 25 July 2021 / Accepted: 22 September 2021 /
Published online: 6 January 2022

Abstract
The compression and acceleration of Deep neural networks (DNNs) are necessary steps to deploy sophisticated networks into resource-constrained hardware systems. Due to the weight matrix tends to be low-rank and sparse, several low-rank and sparse compression schemes are leveraged to reduce the overwhelmed weight parameters of DNNs. In these previous schemes, how to make the most of the low-rank and sparse components of weight matrices and how to globally decompose the weight matrix of different layers for efficient compression need to be further investigated. In this paper, in order to effectively utilize the low-rank and sparse characteristics of the weight matrix, we first introduce a sparse coefficient to dynamically control the allocation between the low-rank and sparse components, and an efficient reconstructed network is designed to reduce the inference time. Secondly, since the results of low-rank decomposition can affect the compression ratio and accuracy of DNNs, we establish an optimization problem to automatically select the optimal hyper-parameters of the compressed network and achieve global compression for all the layers of network synchronously. Finally, to solve the optimization problem, we present a decomposition-searching algorithm to search the optimal solution. The algorithm can dynamically keep the balance between the compression ratio and accuracy. Extensive experiments of AlexNet, VGG-16 and ResNet-18 on CIFAR-10 and ImageNet are employed to evaluate the effectiveness of the proposed approach. After slight fine-tuning, compressed networks have gained 1.2× to 11.3× speedup and our method reduces the size of different networks by 1.4× to 14.6×.

Keywords Deep neural network · Low-rank decomposition · Global optimization · Model compression

Editors: Yu-Feng Li, Mehmet Gönen, Kee-Eung Kim.

✉ Longjun Liu
liulongjun@xjtu.edu.cn

[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Shannxi, China

1 Introduction

The increasing number of parameters and model complexity of DNNs have raised critical challenges in deploying large DNNs into mobile devices and embedded systems due to their limited storage, and computational resources. In order to deploy advanced networks in the resource-limited devices, such as cellphones and wearable devices, model compression has been widely adopted to reduce the memory and computational demands from sophisticated networks. There are mainly five categories of approaches for network compression: (1) matrix decomposition (Kim et al. 2015; Denton et al. 2014; Tai et al. 2015; Lebedev et al. 2014; Yu et al. 2017), which compresses the DNN by decomposing each large-sized weight matrix into multiple small-sized matrices; (2) network pruning (Zhao et al. 2019; Han et al. 2015; Tung and Mori 2018; Hu et al. 2016), which saves resources by removing the unimportant structure (connections, channels, filters, etc.) of the network; (3) quantization (Chen et al. 2015; Han et al. 2015; Tung and Mori 2018; Rastegari et al. 2016), which speeds up inference by reducing the bit of weight parameters and feature-map values; (4) knowledge distillation (Bai et al. 2020; Hinton et al. 2015; Cheng et al. 2020), which compresses the DNN by transferring effective knowledge from the large-sized network to the corresponding small-sized network and (5) compact network design (Howard et al. 2017; Gholami et al. 2018; Ma et al. 2018), which implements the network architecture by designing inference-friendly blocks. As for the current research field of pretrained DNN compression, pruning and low-rank matrix decomposition are the most popular approaches. This article focuses on the low-rank matrix decomposition for DNN compression.

Low-rank decomposition can compress and accelerate the network by approximating the weight matrix to the small-sized matrix. Moreover, the matrix decomposition can be quickly inferred on the common chips, especially GPUs (Idelbayev and Carreira-Perpinan 2020). Common matrix decomposition methods include QR (Deb et al. 2018), CUR (Kishore Kumar and Schneider 2017) and singular value decomposition (SVD) (Yuan et al. 2019). Among these methods, SVD uses singular values to describe the importance of features in the matrix, and compresses the matrix by removing unimportant singular values and corresponding eigenvectors. SVD is widely used in DNN compression. For example, a 15-layer convolutional neural network trained on ImageNet is compressed through SVD (Denton et al. 2014). Moreover, SVD is used to compress the fully connected layer (FC), which reduces 25% parameters of the compressed network (Girshick 2015). Besides, Masana et al. (2017) leveraged SVD to compress FC6 and FC7 of VGG-19. Although SVD can effectively compress the networks, it may affect the accuracy of the compressed network when the weight matrix does not have low-rank characteristics. As shown in Fig. 1, we test the variations of Top-1 accuracy with different ranks. Since there are only a small number of similarity elements in the weight matrix, a small variation of rank will cause a obvious decrease in the accuracy. To solve this problem, low-rank and sparse scheme is introduced to compress DNNs for the compact network with the higher accuracy.

Low-rank and sparse schemes (LSSs) (Bouwmans et al. 2016; Wen et al. 2017) decompose the weight matrix into low-rank and sparse matrices and leverage different strategies to compress them. A greedy algorithm (Yu et al. 2017) is used to approximatively decompose the weight matrix into low-rank and sparse matrices, and then QR, SVD and Huffman coding methods (Han et al. 2015) are used to compress the network. Alvarez and Salzmann (2017) also introduces the concept of sparsity and low-rank into network compression. In this article, a preliminary compressed network is obtained through SVD, and the final

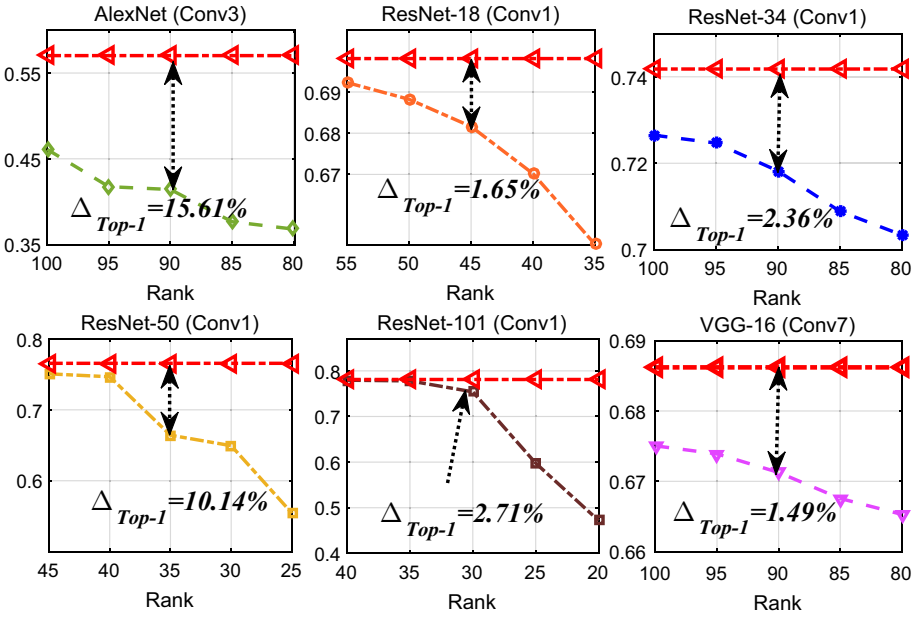


Fig. 1 Variations of Top-1 with ranks. The initialization is the rank with almost 1-compression ratio. The red dotted line represents the accuracy of original network, $\triangle_{Top-1}$ represents the difference of accuracy (colour figure online)

compressed network is obtained by kernel-wise pruning (a group-sparse method). One of the critical challenges for LSS-based network compression is how to find a reasonable allocation of low-rank and sparse components. In addition, since low-rank matrix decomposition can only decompose weight matrices with the same size and similar information, this method is impossible to further investigate the relationship between different layers for fast global compression.

In this article, we propose a controllable matrix decomposition (CMD) with global optimization to produce the network with less weight parameters and higher accuracy. We first introduce a special sparse coefficient in the LSS for flexible and configurable DNN compression. Based on the result of LSS, we propose a reconstructed network to reduce the inference time of the DNN. In addition, to automatically select the optimal parameters and compress each layer of the network in a parallel scenario, we investigate the accuracy and compression ratio to establish an optimization problem. Moreover, to find the optimal compressed network, a two-phase algorithm, which contains decomposition and searching phases, is proposed to solve the optimal problem. In brief, our contributions are summarized as follows:

(1) We first introduce a generalized controllable LSS for DNN compression. In the case of SVD-based low-rank component compression, our scheme degenerates into traditional LSS with a sparse coefficient. Furthermore, we reconstruct the topology of the network to save the inference time for the LSS-based compression method.

(2) We establish an optimization problem to globally compress the network, and automatically select optimal ranks and sparse coefficients. The optimization problem comprehensively considers the accuracy and compression ratio of the compressed network.

(3) We propose a decomposition-searching algorithm to solve the optimization problem. In decomposition phase, weight matrices are decomposed into different components. In searching phase, Bayesian Optimization is used to select the optimal hyperparameters by expectation and uncertainty. In the experiment, we implement various types of DNNs on different benchmarks, and our method can dynamically obtain the balance between the compression ratio and accuracy. Notably, CMD saves 14.6× of memory of AlexNet on ImageNet with negligible accuracy drop.

The rest of this paper is organized as follows: In Sect. 2, we introduce the related work and compare our work with other LSS-based compression method. Then, we present the overall compression framework in Sect. 3, and we further illustrate details from the controllable and global compression, respectively. In Sect. 4, we propose a decomposition-searching algorithm to find the optimal hyperparameter in the compressed model. After that, we leverage two benchmarks (CIFAR-10 and ImageNet) on various types of DNNs to demonstrate the compression results in Sect. 5, and finally conclude in Sect. 6.

2 Related work

Recently, it has attracted intense attention on the research of network compression, which is an feasible approach to reduce neural network parameters and to deploy DNNs into edge-computing devices. Currently, in the field of network compression, matrix decomposition is a promising way to compress DNNs because matrix (tensor) calculation accounts for a significant portion in the DNN. Moreover, matrix decomposition is also conducive to the inference of hardware system due to a parallelizable memory access pattern has applied for low-rank approximation (Idelbayev and Carreira-Perpinan 2020).

Singular value decomposition (SVD) SVD is an effective low-rank decomposition method, which can measure the effective information in the matrix through the singular values, and finally retain the vectors with larger singular values to reduce the storage of the original matrix (Zhou et al. 2016; Li and Lin 2020; Mizutani and Tanaka 2018; Jung and Sael 2020). SVD is widely used in DNN compression. Girshick (2015) leverages SVD to compress the fully connected layer in the Fast-RCNN, and the size of compressed network becomes 25% of that of the original network. In order to find the redundant information in the weight matrix, bi-clustering is used to pre-process the matrix and then SVD is used to compress the network (Denton et al. 2014). In order to save the memory, the layers (FC6 and FC7) of VGG-19 are compressed through SVD without accuracy drop (Masana et al. 2017). Although SVD can effectively compress the network, the compression ratio and accuracy of the compressed network is actually affected by the relevant information in the weight matrix. In addition, matrix decomposition can only decompose matrices with the same size and similar information. Therefore, matrix decomposition method can only compress the network layer by layer, which may lead to low-efficient compression for the network with numerous of layers.

Low-rank and sparse scheme (LSS) LSS-based method has been used in many scenarios (e.g., background segmentation (Zhou et al. 2020; Masi et al. 2020), outlier detection (Wang et al. 2020; Liao et al. 2020; Zhou and Feng 2017), compressed sensing (Shi et al. 2019; Zhang and Ghanem 2018), etc.) (Bouwmans et al. 2016; Hirayama et al. 2016). In the field of network compression, LSS is used to decompose the weight matrix, and a greedy algorithm is used to select the rank of low-rank component (Yu et al. 2017). In addition, several frameworks combining low-rank decomposition with pruning is proposed

to compress the DNN (Alvarez and Salzmann 2017). The method first reduces parameters through the low-rank decomposition, and then implements kernel-wise pruning through group-sparse to achieve network acceleration. However, we notice that one of the critical challenges for LSS-based method is the reasonable allocation between low-rank and sparse components. Hence, it is important to introduce the appropriate hyperparameter to control the allocation between the two different components, and it is better to automatically select the optimal hyperparameter for the allocation.

Bayesian optimization (BO) BO provides an approach to maximize (if minimized, the objective function is preceded by a minus) black-box objective functions, which are non-convex, expensive to evaluate, hard to differential, and may not be easily expressed as closed forms (Tung and Mori 2018; Wang et al. 2013; Kim et al. 2021). Since the DNN can be considered as a black-box function, BO is optimization-friendly to DNN. BO usually assumes that parameters obey a certain distribution, and it contains exploitation and exploration phases (Snoek et al. 2012): (1) In the exploitation phase, a new sample is used to update the probability model of the objective function. (2) In the exploration phase, an acquisition function, which combines information about the expectation and uncertainty of the objective function, is used to select new sample. These two phases are alternated to obtain the optimal value. Since Gaussian process (Rasmussen 2003; Brochu et al. 2010) has favorable statistical and computational characteristics, it is usually set as the distribution of the objective function. Besides, there are several acquisition functions, such as, the probability of improvement (PI), expected improvement (EI), upper confidence bounds (UCB) (Snoek et al. 2012; Brochu et al. 2010). In this paper, we use UCB as the acquisition function because it is easy for calculation and can achieve good results. It can be expressed as: $UCB(x) = \mu(x) + \kappa\sigma(x)$, where $\mu(x)$ and $\sigma(x)$ represent mean value and variance, respectively. They are used to illustrate the expectation and uncertainty of the sample space. κ is used to balance exploitation against exploration.

Discussion To the best of our knowledge, Yu et al. (2017) also leveraged LSS scheme to compress the DNN. However, we need to emphasize that our method is orthogonal to it. In the CMD, we introduce a sparse coefficient for the sparse matrix (in Sect. 3.1), and it can control the sparse component in the decomposed matrix. Moreover, we establish an optimization problem (in Sect. 3.2) to select the optimal rank and sparse coefficient of the global network. Compared with Yu et al. (2017), CMD can automatically select the rank instead of adopting greedy algorithm, which makes the compression more reasonable. In addition, this paper clearly reveals the structure of the reconstructed network. And we record both parameters and inference speed of the compressed network in the experiment.

3 Problem formulation

In this section, we first introduce a controllable LSS with the sparse coefficient for DNN compression. Furthermore, to save the inference time, we reconstruct the topology of the network. In addition, to globally compress DNNs and select the optimal compressed network, we establish an optimization problem based on the hyperparameter of the compressed network.

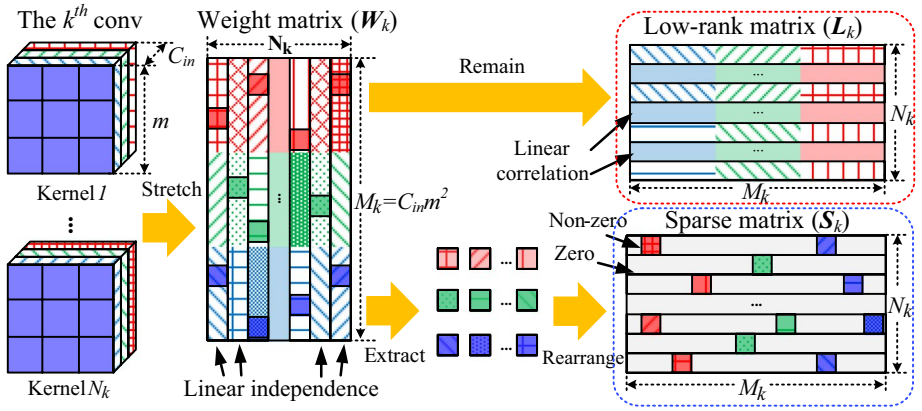


Fig. 2 The decomposition process of LLS. Firstly, the 4-D Convs of the same layer are stretched into a 2-D matrix $\mathbf{W}_k \in \mathbb{R}^{M_k \times N_k}$, where $M_k = m \times m \times C_{in}$, $m \times m$ represents the size of each channel, C_{in} represents the number of channels, and N_k represents the number of outputs. It should be noted that M_k and N_k are the dimensions related to the input and output, respectively. Secondly, the low-rank matrix and sparse matrix are leveraged to approximate $\mathbf{W}_k$ (colour figure online)

3.1 Strategy for a controllable compression

As shown in Fig. 2, since there are a large number of linearly independent Convs in the weight matrix, we cannot represent all the Convs only through several important Convs. In this case, the accuracy and compression ratio of the compressed network cannot be guaranteed simultaneously. In order to find the low-rank characteristics of the weight matrix, we extract a small number of elements from it, and ensure that there are more relevant kernels in the remaining matrix (marked by the red square in Fig. 2). We rearrange the extracted elements in the corresponding locations, and set other locations to zero to form a sparse matrix $\mathbf{S}_k$. The remaining matrix is represented as $\mathbf{L}_k$ (marked by the blue square in Fig. 2), where k represents the index of layer, then we can build an optimization problem:

$$\begin{aligned} \min_{\mathbf{L}_k, \mathbf{S}_k} \quad & \psi(\mathbf{L}_k) + \lambda_k \left\| \mathbf{S}_k \right\|_0 \\ s.t. \quad & \mathbf{W}_k \approx \mathbf{L}_k + \mathbf{S}_k \end{aligned} \tag{1}$$

where $\psi(\mathbf{L}_k)$ represents the dissimilarity measure of matrix elements. For example, $\psi(\mathbf{L}_k)$ can be used to count the number of linearly independent kernels in $\mathbf{L}_k$. $||\mathbf{S}_k||_0$ represents the number of non-zero elements in $\mathbf{S}_k$. Since the main information of the original weight matrix cannot be greatly changed, it is necessary to ensure that there are only a few non-zero elements in $\mathbf{S}_k$, i.e., $\mathbf{S}_k$ is a sparse matrix. λ_k represents the sparse coefficient, which is used to control the number of non-zero elements in $\mathbf{S}_k$. In this paper, SVD is used to compress the weight matrix, and singular values is used to measure $\mathbf{L}_k$. Thus $\psi(\mathbf{L}_k) = \sum_i \chi(\mathbf{\Sigma}_{ii}(\mathbf{L}_k))$, where $\mathbf{\Sigma}_{ii}(\mathbf{L}_k)$ represents the i^{th} value on the diagonal of the singular value matrix. When $\mathbf{\Sigma}_{ii}(L_k) \neq 0$, then $\chi(\mathbf{\Sigma}_{ii}(\mathbf{L}_k)) = 1$, otherwise $\chi(\mathbf{\Sigma}_{ii}(\mathbf{L}_k)) = 0$. Then, the optimization problem (1) can be transformed into the following controllable low-rank and sparse combination scheme:

$$\begin{aligned} \min_{\mathbf{L}_k,\mathbf{S}_k} \quad & \sum_i \chi(\boldsymbol{\Sigma}_{ii}(\mathbf{L}_k)) + \lambda_k \|\mathbf{S}_k\|_0 \\ s.t. \quad & \mathbf{W}_k \approx \mathbf{L}_k + \mathbf{S}_k \end{aligned} \tag{2}$$

In the optimization problem (2), low-rank matrix $\mathbf{L}_k$ ($\mathbf{L}_k \in \mathbb{R}^{M_k \times N_k}$) can be represented as $\mathbf{L}_k \approx \hat{\mathbf{U}}_k \boldsymbol{\Sigma}_k (\hat{\mathbf{V}}_k)^T = (\hat{\mathbf{U}}_k \boldsymbol{\Sigma}_k^{1/2})(\boldsymbol{\Sigma}_k^{1/2} \hat{\mathbf{V}}_k)^T = \mathbf{U}_k \mathbf{V}_k^T$, where $\hat{\mathbf{U}}_k$ ($\hat{\mathbf{U}}_k \in \mathbb{R}^{M_k \times r_k}$) and $\hat{\mathbf{V}}_k$ ($\hat{\mathbf{V}}_k \in \mathbb{R}^{N_k \times r_k}$) represent left and right eigenvector matrices, respectively, $\boldsymbol{\Sigma}_k$ ($\boldsymbol{\Sigma}_k \in \mathbb{R}^{r_k \times r_k}$) represents a diagonal matrix, where $r_k = \sum_i \chi(\boldsymbol{\Sigma}_{ii}(L_k))$. $\mathbf{S}_k$ can be compressed by different storage formats (e.g., compressed sparse column format, compressed sparse row format, etc.). Although the decomposed matrix can save the hardware memory resources, the inference time is also important in many edge computing applications. In order to reduce the inference time, we apply the small-sized matrices to reconstruct the network to speed up the computation. As shown in Fig. 3d, each weight matrix can be reconstructed into two layers. For the first layer, each column of $\mathbf{U}_k$ is reconstructed into a Conv (as shown in Fig. 3a). For the second layer, each row of $\mathbf{V}_k$ can be considered as a set of weight factors (as shown in Fig. 3b). Besides, each column of $\mathbf{S}_k$ is also reconstructed into a Conv (as shown in Fig. 3c). To obtain the feature maps of the reconstructed network, the inputs are first calculated with Convs reconstructed by $\mathbf{U}_k$ to obtain r_k hidden feature maps. After that, the hidden feature maps are calculated with the weight factors reconstructed by $\mathbf{V}_k$ and then added to the feature maps obtained by $\mathbf{S}_k$-reconstructed Convs to get the practical feature maps. In the reconstructed network, the FLOPs become $\mathcal{O}((M_k + N_k)r_k + F(\mathbf{S}_k))$, where $F(\mathbf{S}_k) = \|\mathbf{S}_k\|_0$.

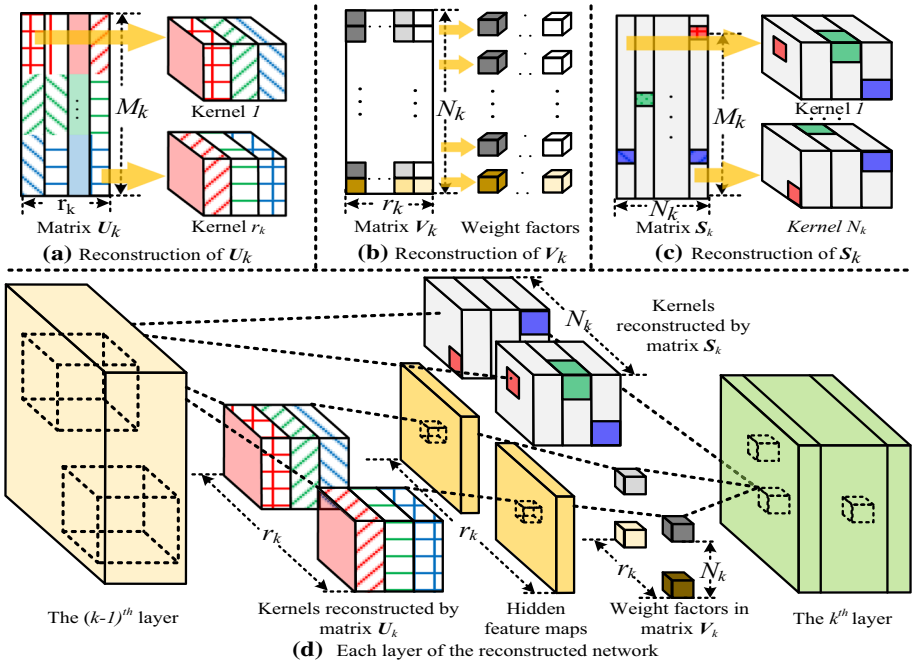
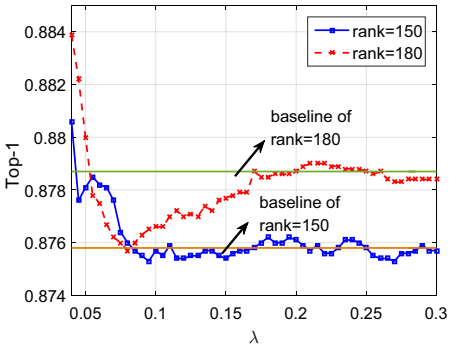
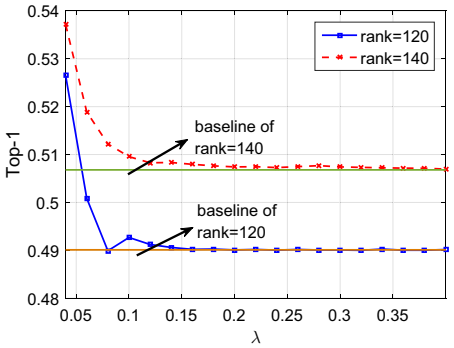


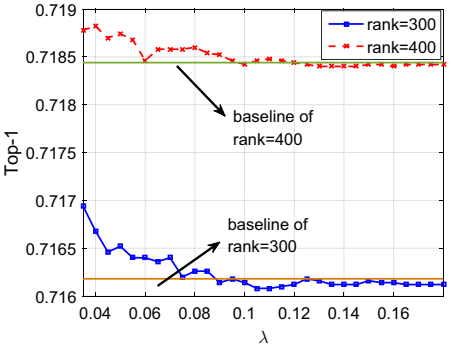
Fig. 3 Structure of the reconstructed network. Each layer of the restructured network consists of three main parts, where two serial layers are reconstructed by $\mathbf{U}_k$ and $\mathbf{V}_k$, and a parallel branch reconstructed by sparse matrix $\mathbf{S}_k$ (colour figure online)



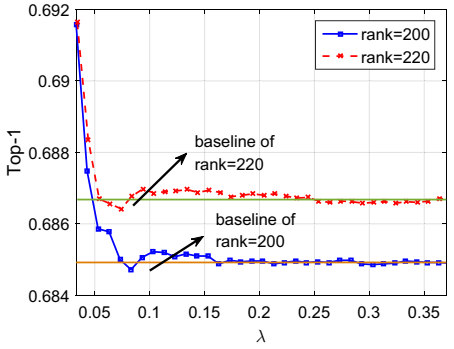
(a) Conv3 of AlexNet on CIFAR-10.



(b) Conv3 of AlexNet on ImageNet.



(c) Conv10 of VGG-16 on ImageNet.



(d) Conv14 of ResNet-18 on ImageNet.

Fig. 4 Variation of Top-1 accuracy with different λ_k. We take different networks as examples to show the results. The blue and red curves represent the performance changes with the lower and higher rank, respectively. Besides, the green and orange lines represent the Top-1 of the SVD-based compressed DNN (baseline). When λ_k is smaller, $\mathbf{S_k}$ contains a large number of non-zero elements, thus the decomposed weight matrix is more similar to the original weight matrix. Therefore, the compressed network has higher Top-1 accuracy. when λ_k is large enough, all elements in $\mathbf{S_k}$ are of 0-value. At this time, the LSS is approximately same as SVD. Hence, the accuracy will eventually approach its baseline (colour figure online)

In summary, we decompose the weight matrix into low-rank and sparse matrices, and reconstruct the network architecture to reduce the parameters and inference time of the network. In the LSS, r_k and λ_k have a great impact on the network performance. As shown in Fig. 4, we use different λ_k to compress the DNNs and record the the variation of Top-1 accuracy, the increase of λ_k reduces the accuracy of the compressed network in the figure, thus λ_k is important in network compression. Besides, Fig. 1 shows that r_k has ability to impact the network accuracy. Therefore, we should choose the appropriate r_k and λ_k for the DNN compression. Moreover, the interaction between different layers should also be considered in the compression. Hence, we will establish an optimization problem to select the optimal hyperparameters, and achieve the global compression in the next section.

3.2 Strategy for global compression

In order to select the optimal compressed network and globally compress the DNN, we use the rank and sparse coefficient to establish an optimization problem. The DNN can be represented as a black-box function f, then the optimization problem can be expressed as follows:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \alpha L(f_{\boldsymbol{\theta}}) + \rho^{-1}(f_{\boldsymbol{\theta}}, f) \\ s.t. \quad & \boldsymbol{\theta} = (r_1, ..., r_K, \lambda_1, ..., \lambda_K),\ k = 1, ..., K, \\ & 1 \le r_k \le r_k^{\max},\ \lambda_k^{\min} \le \lambda_k \le \lambda_k^{\max} \end{aligned} \tag{3}$$

where $r_k^{\max} = INT[M_k \cdot N_k/(M_k + N_k)]$, $INT[\bullet]$ represents the round down. $\lambda_k^{\min}$ and $\lambda_k^{\max}$ represent the lower and upper bound of λ_k. The change of λ_k from small to large makes the number of non-zero elements in the matrix $\mathbf{S}_k$ from more to less, hence when $\mathbf{S}_k$ is full of non-zero elements and zero elements, the corresponding λ_k are selected as $\lambda_k^{\min}$ and $\lambda_k^{\max}$, respectively. $L(f)$ represents the loss function (e.g., cross entropy). $f_{\boldsymbol{\theta}}$ represents the mapping of the compressed network. $\boldsymbol{\theta} = (r_1, ..., r_K, \lambda_1, ..., \lambda_K)$ represents the vector of hyperparameters. α is the trade-off between the accuracy and compression ratio of the compressed network. $\rho(f_{\boldsymbol{\theta}}, f)$ represents the compression ratio.

$$\rho(f_{\theta}, f) = \frac{F(f)}{F(f_{\theta})} = \frac{\sum_{k=1}^{K} M_k N_k}{\sum_{k=1}^{K} ((M_k + N_k) \cdot r_k + F(\mathbf{S}_k))} \tag{4}$$

In Eq. (3), $L(f_{\boldsymbol{\theta}})$ and $\rho(f_{\boldsymbol{\theta}}, f)$ are used to constrain the accuracy and compression ratio of the compressed network, respectively. Since $\mathbf{r}$ is discontinuous in Eq. (3), the optimal hyperparameters cannot be obtained by stochastic gradient descent (SGD). Thus, we proposed a decomposition-searching (DS) algorithm to compress the network.

4 Optimization algorithm

In order to prevent distortion caused by the difference between the compressed and original network, we use the quadratic-penalty method (Nocedal and Wright 2006) to establish the connection between the compressed and original network. Then, Eq. (3) can be rewritten as follows:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \alpha L(f_{\boldsymbol{\theta}}) + \rho^{-1}(f_{\boldsymbol{\theta}}, f) + \frac{\delta}{2} \left\| f_{\boldsymbol{\theta}} - f \right\|_F^2 \\ s.t. \quad & \boldsymbol{\theta} = (r_1, ..., r_K, \lambda_1, ..., \lambda_K),\ k = 1, ..., K, \\ & 1 \le r_k \le r_k^{\max},\ \lambda_k^{\min} \le \lambda_k \le \lambda_k^{\max} \end{aligned} \tag{5}$$

where δ represents the effect of the quadratic-penalty term, $f_{\boldsymbol{\theta}}$ and f represent the mapping of the compressed network and original network, respectively.

To obtain the optimal ranks and sparse coefficients, we propose a decomposition-searching (DS) algorithm to solve Eq. (5). The algorithm contains two phases, which are matrix decomposition phase and hyperparameter searching phase. The details are described as follows:

Decomposition phase In this phase, we solve the optimization problem (2) to obtain the low-rank matrix and sparse matrix. We turn the Eq. (2) into a convex optimization problem, where $\sum_i \chi(\boldsymbol{\Sigma}_{ii}(\mathbf{L}_k))$ and $\|\mathbf{S}_k\|_0$ are relaxed to $\sum_i \boldsymbol{\Sigma}_{ii}(\mathbf{L}_k)$ and $\|\mathbf{S}_k\|_1$, respectively. Then, the augmented Lagrangian function (Lin 2010) is expressed as follows:

$$\begin{aligned}\mathcal{L}(\mathbf{L}_k, \mathbf{S}_k, \mathbf{Y}_k, \mu) = &\|\mathbf{L}_k\|_* + \lambda\|\mathbf{S}_k\|_1 \\ &+ \langle \mathbf{Y}_k, (\mathbf{W}_k - \mathbf{L}_k - \mathbf{S}_k)\rangle + \frac{\mu}{2}\|\mathbf{W}_k - \mathbf{L}_k - \mathbf{S}_k\|_F^2\end{aligned} \tag{6}$$

where $\langle\rangle$ is the inner product, $\mathbf{Y}_k$ is the Lagrange multiplier matrix. μ represents the effect of regularization terms. Then, $\mathbf{S}_k$ and $\mathbf{L}_k$ can be obtained by the soft thresholding function (Wright et al. 2009), where $\mathbf{S}_k = \mathcal{ST}_{\lambda_k/\mu}(\mathbf{W}_k - \mathbf{L}_k + \mathbf{Y}_k/\mu)$, $\mathbf{L}_k = \mathbf{U}_k \cdot [\mathcal{ST}_{\mu^{-1}}(\mathbf{W}_k - \mathbf{S}_k + \mathbf{Y}_k/\mu)] \cdot \mathbf{V}_k^T$, $\mathcal{ST}(\bullet)$ represents the soft thresholding function. The steps of decomposition phase are summarized in Algorithm 1.

Algorithm 1: Decomposition of weight matrix.

Input: Weight matrix $\mathbf{W}_k$.
Output: $\mathbf{L}_k$, $\mathbf{S}_k$, λ_k.
Initialize λ_k, $\mathbf{L}_k$, $\mathbf{S}_k$, $\mathbf{Y}_k$, $\mu_k > 0$, $\eta > 1$, ε.
while $\|\boldsymbol{W}_k - \boldsymbol{L}_k - \boldsymbol{S}_k\|_F / \|\boldsymbol{W}_k\|_F \geq \varepsilon$ **do**
  $\mathbf{U}_k, \boldsymbol{\Sigma}_k, \mathbf{V}_k = SVD(\mathbf{W}_k - \mathbf{S}_k + \mathbf{Y}_k/\mu)$.
  $\mathbf{L}_k = \mathbf{U}_k \left[\mathcal{ST}_{1/\mu}(\mathbf{W}_k - \mathbf{S}_k + \mathbf{Y}_k/\mu)\right] (\mathbf{V})_k^T$.
  $\mathbf{S}_k = \mathcal{ST}_{\lambda_k/\mu}(\mathbf{W}_k - \mathbf{L}_k + \mathbf{Y}_k/\mu)$.
  $\mathbf{Y}_k = \mathbf{Y}_k + \mu(\mathbf{W}_k - \mathbf{L}_k - \mathbf{S}_k)$.
  $\mu_k = \eta\mu_k$.
end
return $\mathbf{L}_k$, $\mathbf{S}_k$, λ_k.

As shown in Algorithm 1, λ_k can control the decomposition result of $\mathbf{W}_k$. When the element in $\mathbf{W}_k - \mathbf{L}_k + \mathbf{Y}_k/\mu$ is greater than λ_k/μ , it is extracted as a sparse component. Then, we introduce the searching phase to search for the optimal ranks and sparse components.

Searching phase Since $\mathbf{r}$ is discontinuous and the DNN can be considered as a black-box function, we choose BO to search hyperparameters. BO has two phases, for the first phase, the set of ranks and sparse coefficients is sampled by the upper confidence bound in the search space. In the second phase, the distribution of the objective function is updated by sampled hyperparameters and the corresponding objective function value. We assume that the objective function $J(\boldsymbol{\theta})$ obeys a Gaussian process with a mean function of $\mu_{\mathcal{GP}}(\boldsymbol{\theta}) = E[J(\boldsymbol{\theta})]$ and a variance function of $K_{\mathcal{GP}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}) = E[(J(\boldsymbol{\theta}) - \mu_{\mathcal{GP}}(\boldsymbol{\theta}))(J(\hat{\boldsymbol{\theta}}) - \mu_{\mathcal{GP}}(\hat{\boldsymbol{\theta}}))]$, i.e., $J(\boldsymbol{\theta}) \sim \mathcal{GP}(\mu_{\mathcal{GP}}(\boldsymbol{\theta}), K_{\mathcal{GP}}(\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}))$. Then, updating the distribution of objective function is transformed into updating the mean and variance. According to the property of the Gaussian process (Rasmussen 2003), the updated objective function can be expressed as follows: $\tilde{J}(\boldsymbol{\theta}^*) \sim \mathcal{N}(\tilde{\mu}_{\mathcal{GP}}(\boldsymbol{\theta}^*), \tilde{\Sigma}^2_{\mathcal{GP}}(\boldsymbol{\theta}^*))$, where $\boldsymbol{\theta}^*$ represents the set of current hyperparameters sampled by the upper confidence bound function. $\tilde{\mu}_{\mathcal{GP}}$ and $\tilde{\Sigma}^2_{\mathcal{GP}}$ represent the updated mean and variance, respectively. Their update formulas are as follows:

$$\begin{aligned}\tilde{\mu}_{\mathcal{GP}}(\boldsymbol{\theta}^*) = &\mu_{\mathcal{GP}}(\boldsymbol{\theta}^*) + \\ &K_{\mathcal{GP}}(\boldsymbol{\theta}^*, \boldsymbol{\Theta}) K_{\mathcal{GP}}(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} (J(\boldsymbol{\Theta}) - \mu_{\mathcal{GP}}(\boldsymbol{\Theta}))\end{aligned} \tag{7}$$

$$\tilde{\Sigma}^2_{\mathcal{GP}}(\boldsymbol{\theta}^*) = K_{\mathcal{GP}}(\boldsymbol{\theta}^*, \boldsymbol{\theta}^*) + K_{\mathcal{GP}}(\boldsymbol{\theta}^*, \boldsymbol{\Theta}) K_{\mathcal{GP}}(\boldsymbol{\Theta}, \boldsymbol{\Theta})^{-1} K_{\mathcal{GP}}(\boldsymbol{\Theta}, \boldsymbol{\theta}^*) \tag{8}$$

where $\boldsymbol{\Theta}$ is the set of $\boldsymbol{\theta}$. We summarize the DS algorithm in Algorithm 2.

In Algorithm 2, several initialization parameters have been introduced in Algorithm 1. Besides, the initialization of the boundary in (5) is not listed. In BO, objective function is $J(\boldsymbol{\theta}) = -[\alpha L(f_{\boldsymbol{\theta}}) + \rho^{-1}(f_{\boldsymbol{\theta}}, f) + \delta ||f_{\boldsymbol{\theta}} - f||_F^2/2]$. To sum up, in this section, we design a DS algorithm to search for the optimal ranks and sparse coefficients of the compressed network. In the next section, we will verify the effectiveness of the method on different datasets.

Algorithm 2: Global compression (DS algorithm).

Input: Well-trained model f ($\mathbf{W} = \{\mathbf{W}_1, ..., \mathbf{W}_K\} \leftarrow f$).
Output: $\hat{\boldsymbol{\theta}}$.

Initialize $\boldsymbol{\lambda}$, $\mathbf{r}$, $\eta \geq 1$, $i_{BO} = 1$, $I_{BO}^{\max}$.
while $i_{BO} \leq I_{BO}^{\max}$ **do**
  for $k = 1, ..., K$ **do**
    Algorithm 1. *Decomposition phase*
    $\mathbf{U}\{k\}, \mathbf{V}\{k\} \leftarrow SVD_{\mathbf{L}_k}(r_k)$ (i.e., $\mathbf{r}(k) \leftarrow r_k$).
    $\mathbf{S}\{k\} \leftarrow \mathbf{S}_k$ (i.e., $\boldsymbol{\lambda}_k(k) \leftarrow \lambda_k$).
  end
  $f_{\boldsymbol{\theta}} \leftarrow \mathbf{U}, \mathbf{V}, \mathbf{S}$ (i.e., $\boldsymbol{\theta} \leftarrow \mathbf{r}, \boldsymbol{\lambda}$).
  Update $J(\boldsymbol{\theta})$ by Eq. (7) and (8). *Searching phase*
  Sample $\boldsymbol{\theta}^*$ by acquisition function.
  $\mathbf{r}, \boldsymbol{\lambda} \leftarrow \boldsymbol{\theta}^*$, $\delta = \eta\delta$, $i_{BO} = i_{BO} + 1$.
end
$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$.
return $\hat{\boldsymbol{\theta}}$.

5 Experiments

In this work, AlexNet (Krizhevsky and Hinton 2009), VGG-16 (Simonyan and Zisserman 2014) and ResNet-18 (He et al. 2016) on CIFAR-10 (Krizhevsky et al. 2012) and ImageNet (Russakovsky et al. 2015) are performed to prove the effectiveness of our method. Experiments are initialized from models in Pytorch.

We set up the experiment as follows with minor changes in different networks. In the matrix decomposition phase, we set $\eta = 1.5$, $\mu_k = 1.25/||\mathbf{W}_k||_F$, $\varepsilon = 10^{-3}$. In the searching phase, we prefer the network without distortion, thus set $\alpha = 1$. Besides, set $\delta = 0.01$, $\eta = 1$ and $I_{BO}^{\max} = 15$. The lower bounds of ranks are 1, the upper bounds are shown as green dotted lines in Figs. 5b, 6b, 7b and 8b. The calculated upper bounds of fully connected layers are too high, which reduce the search efficiency, hence we lower some of them after testing. Moreover, when $\mathbf{S}_k$ is full of non-zero elements and zero elements, we select the corresponding λ_k as the lower and upper bounds of λ_k, respectively. The search bounds of the sparse coefficients are shown in Figs. 5c, 6c, 7c and 8c (green and yellow dotted lines).

For each compressed network, we report its compression ratio (sparse component is stored by the compressed sparse column format (Han et al. 2015)), Top-1 and Top-5

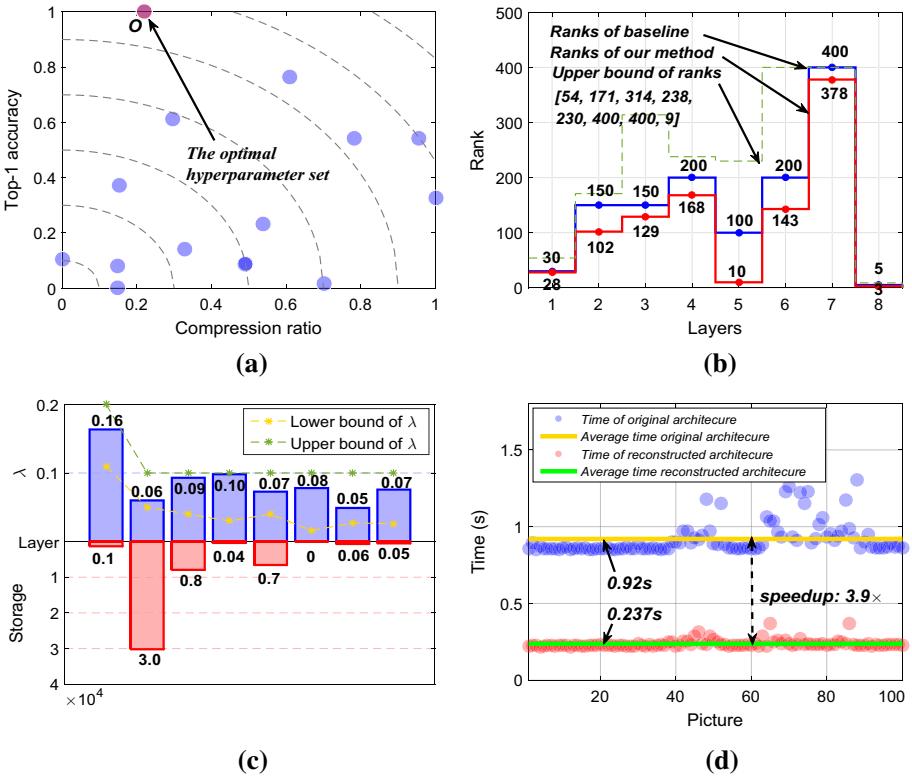


Fig. 5 Compression of AlexNet on CIFAR-10. **a** The result of DS algorithm. **b** Ranks of different layers. **c** Sparse coefficients and sparse components of different layers. **d** Inference time of reconstructed network (colour figure online)

accuracies, and inference time. It is worth mentioning that a fine-tuning phase is adopted to restore the accuracy of the network after reconstruction. In the fine-tuning phase, the learning rate of AlexNet on CIFAR is set to 3×10^{-4}, and the weight decay is 10^{-4}. The leaning rates of AlexNet, VGG and ResNet on ImageNet are set to 10^{-3}, 10^{-4} and 10^{-3}, respectively, and the weight decays are 5×10^{-4}, 10^{-4} and 5×10^{-4}, respectively. To test the inference time, we deployed each network in a hardware system with NVIDIA GeForce GTX 1080Ti and an Intel i7-7700 CPU. Then, we randomly select 100 pictures to test the inference time of the reconstructed network structure and original network structure, in which the former is calculated by pipeline technique of memory data load and the latter contains the merging time for decomposed matrices. We take the ratio of the average inference time to illustrate the acceleration of the compressed network. In addition, a heuristic SVD is used as the baseline. In this method, the decomposed matrix satisfies the following formula, $||\mathbf{W}_k - \mathbf{U}_k \mathbf{V}_k^T||_F \leq (1-p)||\mathbf{W}_k||_F$ (Wen et al. 2017; Xu et al. 2018).

5.1 Experiment on CIFAR-10

The Top-1 and Top-5 accuracies of original AlexNet on CIFAR-10 are 88.79% and 99.75%, respectively. We use our DS algorithm to search the optimal ranks and sparse coefficients,

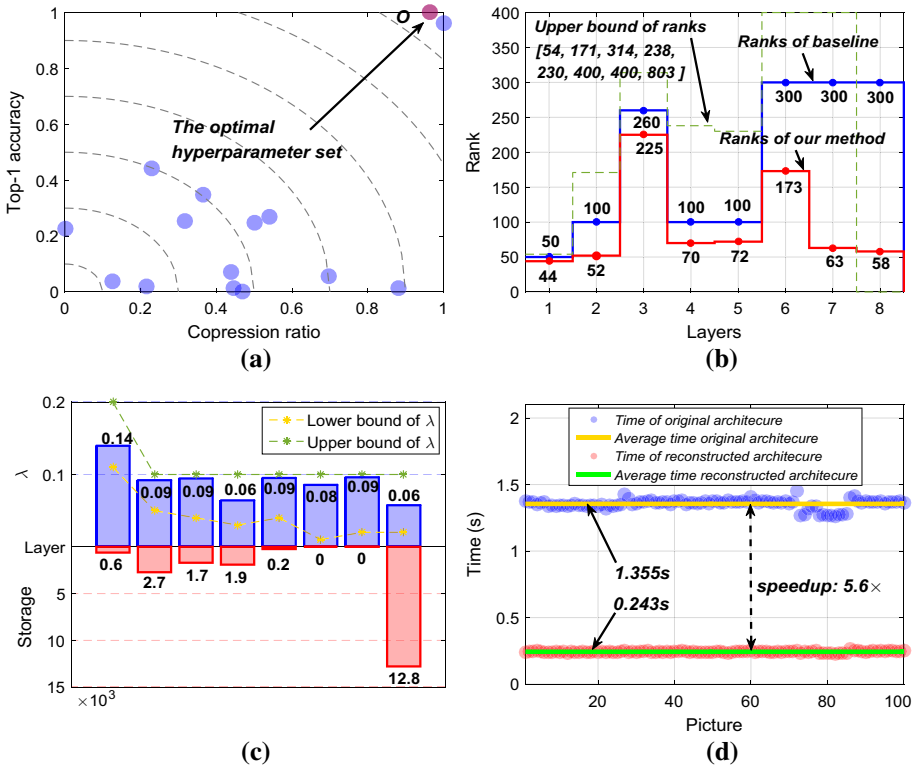


Fig. 6 Compression of AlexNet on ImageNet. **a** The result of DS algorithm. **b** Ranks of different layers. **c** Sparse coefficients and sparse components of different layers. **d** Inference time of reconstructed network (colour figure online)

the performance evaluation metrics of compressed networks corresponding to the search results are shown in Fig. 5a. Each point represents a network compressed by a set of hyperparameter, the abscissa and ordinate represent the normalized compression ratio and Top-1 accuracy, respectively. The dotted quarter circle represents the equipotential surface of the network performance. The larger the radius, the better the performance. The point closer to the ordinate on the same equipotential surface has higher accuracy, and the point closer to the horizontal axis with the same equipotential surface has higher compression ratio. The red point O, which is on the outermost circle and close to the ordinate, represents the optimal compressed model. This is because $\alpha = 1$ makes the compressed model pay more attention to the accuracy. The rank corresponding to the point O is shown as the red broken line in Fig. 5b. Since the image features extracted by the first Conv can affect subsequent layers, the rank of the first layer is close to the upper bound. The 6th layer (FC) have a large amount of redundant information, thus it has a small rank (143) without extracting sparse components. The sparse coefficients and the corresponding storages are shown as the blue and red histogram in Fig. 5c. The sparse components extracted by Convs are more than that of FCs, which indicates that there are more relevant elements in the Convs, and more elements need to be extracted to ensure the low-rank characteristics of the remaining part. In the 2th, 3th and 5th layers (Convs), our method extracts 3×10^4, 0.8×10^4 and 0.7×10^4 of

Table 1 Comparison of CMD with other model compression methods on AlexNet Δ_{Top-1} (Δ_{Top-5}) represents the Top-1 (Top-5) accuracy of the compressed network minus that of the original network

Method	Δ_{Top-1}	Δ_{Top-5}	ρ_{Memory}	Speedup
BN (Tai et al. 2015)	–	+0.53%	4.9×	1.1×
NISP (Yu et al. 2018)	0.00%	–	1.9×	1.7×
CP (Lebedev et al. 2014)	–	−0.37%	5.0×	1.8×
Tucker (Kim et al. 2015)	–	−1.70%	5.5×	1.8×
LRA (Wen et al. 2017)	–	−0.17%	–	–
MUSCO (Gusak et al. 2019)	–	−0.81%	4.9×	–
LC (Idelbayev and Carreira-Perpinan 2020)	−0.34%	−0.39%	4.4×	–
GreBdec (Yu et al. 2017)	−0.04%	−0.04%	10.0×	–
Baseline	−1.98%	−1.56%	6.6×	–
CMD	−0.85%	−0.93%	14.6×	5.6×

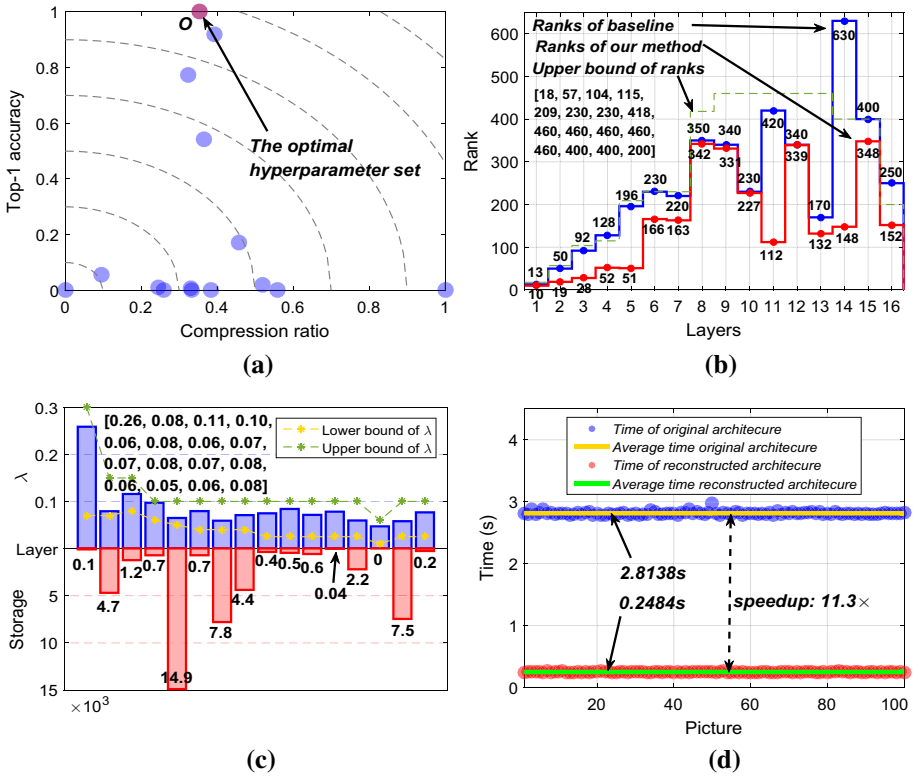


Fig. 7 Compression of VGG on ImageNet. **a** The result of DS algorithm. **b** Ranks of different layers. **c** Sparse coefficients and sparse components of different layers. **d** Inference time of reconstructed network (colour figure online)

sparse components, respectively, thus it can choose a smaller rank (102, 129, and 10) compared with the baseline (blue line).

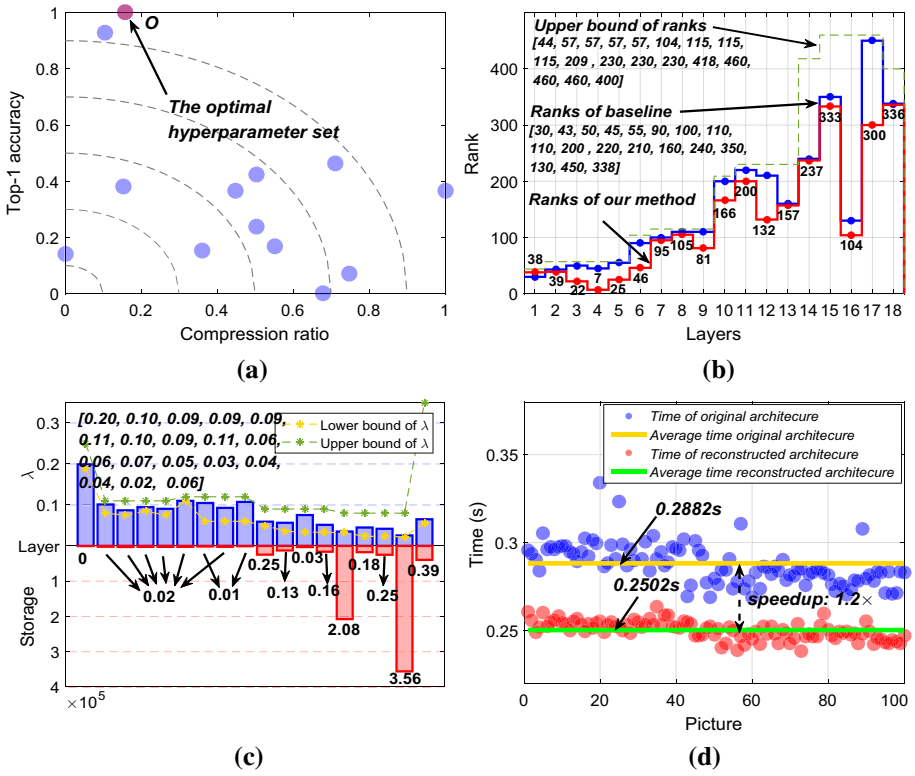


Fig. 8 Compression of ResNet on ImageNet. **a** The result of DS algorithm. **b** Ranks of different layers. **c** Sparse coefficients and sparse components of different layers. **d** Inference time of reconstructed network (colour figure online)

Considering both the storage of low-rank and sparse components, the compression ratio of the compressed network is 9.15. After 30 times of fine-tuning, Top-1 and Top-5 accuracies of the compressed network reach 89.20% and 99.70%, respectively. We set $p = 0.75$ to compute the baseline, the ranks of baseline are shown as blue broken line in Fig. 5b. The compression ratio of baseline is 7.55, the Top-1 and Top-5 accuracies are 88.49% and 99.78% after fine-tuning. Moreover, we use 100 pictures to test the inference time, the results are shown in Fig. 5d. The yellow and green lines indicate the average inference time of the unreconstructed and reconstructed networks, respectively, and CMD provides 3.9× speedup.

5.2 Experiments on imageNet

We use AlexNet, VGG-16 and ResNet-18 on ImageNet to further illustrate the effectiveness of our method. The accuracies of the original AlexNet are 57.02% (Top-1) and 80.08% (Top-5). The metrics of compressed networks searched by DS algorithm are shown in Fig. 6a, the hyperparameter set corresponding to the red point is the optimal. The ranks, sparse coefficients and the storage of sparse components corresponding to this point are shown in Fig. 6b and c, respectively. The blue broken line in Fig. 6b

represents the ranks of baseline when $p = 0.75$. Since a small number of elements in the weight matrix are extracted in the 1st to 5th layers (Convs), the rank selected by our method is slightly lower than that of baseline. In the FCs (the 6th and 7th layers), even if no sparse components are extracted, we can choose the small ranks for low-rank matrix (173 and 63, respectively). Although the last layer is a FC, our method extracts 12.8×10^3 of sparse components. We owe its situation to that our method selects the small ranks in the 6th to 8th layers (173, 63 and 58, respectively), which leads to a large number of sparse components need to be extracted to reduce the accuracy drop under global compression. After 7 epochs of fine-tuning, the accuracies of the compressed network are restored to 56.17% (Top-1) and 79.15% (Top-5), and the compression ratio is 14.6. The Top-1 and Top-5 accuracies of the baseline reach 55.04% and 78.52% after 18 epochs of fine-tuning, and the compression ratio is 6.52. Compared with the unreconstructed model, the reconstructed network has a speedup of 5.6 times (Fig. 6d). In Table 1, we compare the result of our method with that of other model compression methods, and the performance of the compressed network obtained by our method is better than that of most methods.

The Top-1 and Top-5 accuracies of VGG are 68.62% and 89.12%, respectively. The results of DS algorithm are shown in Fig. 7a, and the red point is the optimal. The ranks, sparse coefficients and the storage of sparse components of the optimal model are shown in Fig. 7b and c, respectively. The blue broken line (Fig. 7b) represents the ranks of baseline when $p = 0.85$. The ranks of Convs are close to the upper bound, hence they can guarantee the accuracy of the compressed network. The ranks of FCs are much smaller than the upper bound, hence they can guarantee the compression ratio. Our method extracts a large number of sparse components in the 5th Conv (14.9×10^3), thus the rank selected by our method is smaller than that of the baseline (145 smaller than the baseline). The 1st FC (the 14th layer) has a large amount of redundant information, thus we can greatly compress it without extracting the sparse components. The compression ratio of the compressed network is 8.75, the Top-1 and Top-5 accuracies are 68.53% and 88.85% after 3 times of fine-tuning. In addition, the compression ratio of the baseline is 4.16, and the accuracies are 66.78% (Top-1) and 87.88% (Top-5) after 3 times of fine-tuning. Finally, as shown in Fig. 7d, our method achieves an 11.3× speedup. As

Table 2 Comparison of CMD with other model compression methods on VGG

Method	Δ_{Top-1}	Δ_{Top-5}	ρ_{Memory}	Speedup
SLIM (Liu et al. 2017)	+0.03%	–	5.7×	1.4×
BN (Tai et al. 2015)	–	−0.13%	2.7×	1.5×
Reborn Filters (Tang et al. 2020)	–	−1.47%	–	2.0×
CC-GAP (Li et al. 2021)	-2.87%	−1.68%	16.5×	2.1×
CP (Lebedev et al. 2014)	–	−0.29%	2.8×	2.1×
Tucker (Kim et al. 2015)	–	−0.50%	1.1×	2.3×
COBLA (Li and Shi 2018)	–	−0.9%	1.4×	–
Dynamic Pruning (Wang et al. 2020)	–	−0.15%	1.5×	2.4×
MUSCO (Gusak et al. 2019)	–	−1.33%	–	5.3×
GreBdec (Yu et al. 2017)	+0.25%	−0.38%	15.0×	–
Baseline	−1.84%	−1.24%	4.2×	–
CMD	−0.09%	−0.27%	8.8×	11.3×

Table 3 Comparison of CMD with other model compression methods on ResNet

Method	Δ_{Top-1}	Δ_{Top-5}	ρ_{Memory}	Speedup
FPGM (He et al. 2019)	−1.87%	−1.15%	–	1.7×
SFP (He et al. 2018)	−3.18%	−1.85%	–	1.7×
TRP1 (Xu et al. 2018)	–	−2.06%	–	1.8×
Baseline	−1.53%	−1.54%	1.4×	–
CMD	0.00%	+0.27%	1.4×	1.2×

shown in Table 2, since CMD better balances the accuracy and compression ratio, our method can obtain a better compressed network compared with other methods.

Greedy bilateral decomposition (Yu et al. 2017) reported 10× to 15× compression ratio without accuracy drop. However, greedy algorithm cannot control the reasonable allocation of different components, and layer-by-layer compression requires layer-by-layer fine-tuning to restore accuracy. Besides, Yu et al. (2017) did not report the inference time for LSS-based compression. We emphasize that our method is orthogonal to Yu et al. (2017), because we compress the networks by a controllable global LSS. In Tables 1 and 2, the different sparse processing method and experimental setup between CMD and GreBdec may cause the differences in DNN performance. The higher compression ratio in Table 1 and Top-5 accuracy in Table 2 demonstrate that CMD has ability to compress the network.

The accuracies of original ResNet-18 are 69.78% (Top-1) and 89.04% (Top-5). The results of DS algorithm are shown in Fig. 8a and the red point represents the optimal model selected by our method. The ranks of the optimal compressed model are shown as red broken line in Fig. 8b (the Convs with *stride* = 2 are not compressed), and the ranks of baseline ($p = 0.7$) are shown as blue broken line in this figure. The sparse coefficients (blue histogram) and the storage of sparse components (red histogram) are shown in Fig. 8c. In the 17^{th} layer (Conv), the rank chosen by our method is 150 smaller than the baseline, this is because a large number of sparse components (3.56×10^5) are extracted in this layer. Even though a lot of sparse components are extracted in the 14th layer (Conv), the rank chosen by our method is similar to that of the baseline (237 and 240, respectively). The reason is that our method investigates the interactions between different layers to restore the accuracy by extracting sparse components and maintaining rank at this layer when compressing the previous layers (the 3th to 6th layers) with the small ranks and few sparse components causing the accuracy drop. After 4 epochs of fine-tuning, the compression ratio of the compressed network is 1.40. The Top-1 and Top-5 accuracies are 69.78% and 89.31%, respectively. The accuracies and compression ratio of baseline are 68.25% (Top-1), 88.50% (Top-5) and 1.38, respectively (4 epoch of fine-tuning). Moreover, as shown in Fig. 8d, the reconstructed network compressed by our method has 1.2× speedup. In Table 3, CMD can provides the compressed DNN without accuracy drop, and the results show that the sparse components may lower the calculation speed, but it can slightly improve the accuracy.

6 Conclusion

In this paper, we propose a controllable matrix decomposition with global optimization for DNN compression. Our works include the following key improvements: (i) a sparse coefficient is introduced to control the allocation between low-rank and sparse components,

and the network is reconstructed to reduce the inference time. (ii) an optimization problem is established to globally compress all the layers of the network synchronously. (iii) we design a decomposition-searching algorithm to search the optimal hyperparameters for network compression. Experimental results show that compressed networks can effectively save the memory and reduce the inference time. For example, CMD saves 14.6× of memory of AlexNet on ImageNet with negligible accuracy loss.

Author Contributions H.N.Z and L.J.L conceived of the study, performed the research, analysed data, and wrote the paper. The remaining authors contributed to refining the ideas, carrying out additional analyses and discussed the results and revised the manuscript.

Funding This research was supported by National Key R & D Program of China (NO. 2017YFA0700800), and National Natural Science Foundation of China (NFSC) (NO. 61774125, 61790563).

Data availability Not Applicable.

Code availability Not Applicable.

Declarations

Conflict of interest The authors declare there is no conflicts of interest regarding the publication of this paper.

Ethical approval Not Applicable.

Consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Alvarez, J. M., & Salzmann, M. (2017). Compression-aware training of deep networks. arXiv preprint arXiv:1711.02638.

Bai, H., Wu, J., King, I., & Lyu, M. (2020). Few shot network compression via cross distillation. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 04, pp. 3203–3210).

Bouwmans, T., Aybat, N. S., & Zahzah, E. H. (Eds.). (2016). Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing. CRC Press.

Brochu, E., Cora, V. M., & De Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv preprint arXiv:1012.2599.

Chen, W., Wilson, J., Tyree, S., Weinberger, K., & Chen, Y. (2015). Compressing neural networks with the hashing trick. *In International conference on machine learning* (pp. 2285–2294). PMLR.

Cheng, X., Rao, Z., Chen, Y., & Zhang, Q. (2020). Explaining knowledge distillation by quantifying the knowledge. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12925–12935).

Deb, T., Ghosh, A. K., & Mukherjee, A. (2018). Singular value decomposition applied to associative memory of Hopfield neural network. *Materials Today: Proceedings, 5*(1), 2222–2228.

Denton, E., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. arXiv preprint arXiv:1404.0736.

Gholami, A., Kwon, K., Wu, B., Tai, Z., Yue, X., Jin, P., & Keutzer, K. (2018). Squeezenext: Hardware-aware neural network design. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 1638–1647).

Girshick, R. (2015). Fast r-cnn. *In Proceedings of the IEEE international conference on computer vision* (pp. 1440–1448).

Gusak, J., Kholiavchenko, M., Ponomarev, E., Markeeva, L., Blagoveschensky, P., Cichocki, A., & Oseledets, I. (2019). Automated multi-stage compression of neural networks. *In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4340–4349).

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866.

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Hirayama, J. I., Hyvarinen, A., & Ishii, S. (2016). Sparse and low-rank matrix regularization for learning time-varying Markov networks. *Machine Learning, 105*(3), 335–366.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861

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

Idelbayev, Y., & Carreira-Perpinan, M. A. (2020). Low-rank compression of neural nets: Learning the rank of each layer. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8049–8059).

Jung, J., & Sael, L. (2020). Fast and accurate pseudoinverse with sparse matrix reordering and incremental approach. *Machine Learning, 109*(12), 2333–2347.

Kim, Y. D., Park, E., Yoo, S., Choi, T., Yang, L., & Shin, D. (2015). Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv preprint arXiv:1511.06530.

Kim, J., McCourt, M., You, T., Kim, S., & Choi, S. (2021). Bayesian optimization with approximate set kernels. Machine Learning, 1–23.

Kishore Kumar, N., & Schneider, J. (2017). Literature survey on low rank approximation of matrices. *Linear and Multilinear Algebra, 65*(11), 2212–2244.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances Neural Information Processing Systems, 25,* 1097–1105.

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., & Lempitsky, V. (2014). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553.

Li, H., & Lin, Z. (2020). Provable accelerated gradient method for nonconvex low rank optimization. *Machine Learning, 109*(1), 103–134.

Li, C., & Shi, C. J. (2018). Constrained optimization based low-rank approximation of deep neural networks. *In Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 732–747).

Li, Y., Lin, S., Liu, J., Ye, Q., Wang, M., Chao, F., & Ji, R. (2021). Towards Compact CNNs via Collaborative Compression. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6438–6447).

Liao, Y., Liu, S., Wang, F., Chen, Y., Qian, C., & Feng, J. (2020). Ppdm: Parallel point detection and matching for real-time human-object interaction detection. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 482–490).

Lin, Z., Chen, M., & Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. arXiv preprint arXiv:1009.5055.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017). Learning efficient convolutional networks through network slimming. *In Proceedings of the IEEE international conference on computer vision* (pp. 2736–2744).

Ma, N., Zhang, X., Zheng, H. T., & Sun, J. (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. *In Proceedings of the European conference on computer vision (ECCV)* (pp. 116–131).

Masana, M., van de Weijer, J., Herranz, L., Bagdanov, A. D., & Alvarez, J. M. (2017). Domain-adaptive deep network compression. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 4289–4297).

Masi, I., Mathai, J., & AbdAlmageed, W. (2020). Towards Learning Structure via Consensus for Face Segmentation and Parsing. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 5508–5518).

Mizutani, T., & Tanaka, M. (2018). Efficient preconditioning for noisy separable nonnegative matrix factorization problems by successive projection based low-rank approximations. *Machine Learning, 107*(4), 643–673.

Nocedal, J., & Wright, S. (2006). Numerical optimization. Springer Science & Business Media.

Rasmussen, C. E. (2003). Gaussian processes in machine learning. In Summer school on machine learning (pp. 63–71). Springer, Berlin, Heidelberg.

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. *In European conference on computer vision* (pp. 525–542). Springer, Cham.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal Computer Vision, 115*(3), 211–252.

Shi, W., Jiang, F., Liu, S., & Zhao, D. (2019). Scalable convolutional neural network for image compressed sensing. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 12290–12299).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. arXiv preprint arXiv:1206.2944.

Tai, C., Xiao, T., Zhang, Y., & Wang, X. (2015). Convolutional neural networks with low-rank regularization. arXiv preprint arXiv:1511.06067.

Tang, Y., You, S., Xu, C., Han, J., Qian, C., Shi, B., & Zhang, C. (2020, April). Reborn filters: Pruning convolutional neural networks with limited data. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 04, pp. 5972–5980).

Tung, F., & Mori, G. (2018). Clip-q: Deep network compression learning by in-parallel pruning-quantization. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7873–7882).

Wang, Z., Zoghi, M., Hutter, F., Matheson, D., & De Freitas, N. (2013, August). Bayesian Optimization in High Dimensions via Random Embeddings. In IJCAI (pp. 1778–1784).

Wang, F., Xue, N., Yu, J. G., & Xia, G. S. (2020). Zero-assignment constraint for graph matching with outliers. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 3033–3042).

Wang, Y., Zhang, X., Hu, X., Zhang, B., & Su, H. (2020, April). Dynamic network pruning with interpretable layerwise channel selection. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 34, No. 04, pp. 6299–6306).

Wen, W., Xu, C., Wu, C., Wang, Y., Chen, Y., & Li, H. (2017). Coordinating filters for faster deep neural networks. *In Proceedings of the IEEE International Conference on Computer Vision* (pp. 658–666).

Wright, S. J., Nowak, R. D., & Figueiredo, M. A. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on signal processing, 57*(7), 2479–2493.

Xu, Y., Li, Y., Zhang, S., Wen, W., Wang, B., Qi, Y., ... & Xiong, H. (2018). Trained rank pruning for efficient deep neural networks. arXiv preprint arXiv:1812.02402.

Yu, X., Liu, T., Wang, X., & Tao, D. (2017). On compressing deep models by low rank and sparse decomposition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7370–7379).

Yu, R., Li, A., Chen, C. F., Lai, J. H., Morariu, V. I., Han, X., & Davis, L. S. (2018). Nisp: Pruning networks using neuron importance score propagation. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9194–9203).

Yuan, L., Li, C., Mandic, D., Cao, J., & Zhao, Q. (2019). Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion. *In Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 9151–9158).

Zhang, J., & Ghanem, B. (2018). ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1828–1837).

Zhao, C., Ni, B., Zhang, J., Zhao, Q., Zhang, W., & Tian, Q. (2019). Variational convolutional neural network pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2780–2789).

Zhou, H., Alvarez, J. M., & Porikli, F. (2016). Less is more: Towards compact cnns. *In European Conference on Computer Vision* (pp. 662–677). Springer, Cham.

Zhou, D., Fang, J., Song, X., Liu, L., Yin, J., Dai, Y., & Yang, R. (2020). Joint 3D Instance Segmentation and Object Detection for Autonomous Driving. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1839–1849).

Zhou, P., & Feng, J. (2017). Outlier-robust tensor PCA. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2263–2271).