Learning-based Tensor Decomposition with Adaptive Rank Penalty for CNNs Compression

Deli Yu[1,2], Peipei Yang[2,1], Cheng-Lin Liu[2,1]
[1]School of Artifical Intelligence, University of Chinese Academy of Sciences, Beijing, P.R. China
[2]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, P.R. China
yudeli2018@ia.ac.cn, ppyang@nlpr.ia.ac.cn, liucl@nlpr.ia.ac.cn

***Abstract*—Low-rank tensor decomposition is a widely-used strategy to compress convolutional neural networks (CNNs). Existing learning-based decomposition methods encourage low-rank filter weights via regularizer of filters' pair-wise force or nuclear norm during training. However, these methods can not obtain the satisfactory low-rank structure. We propose a new method with an adaptive rank penalty to learn more compact CNNs. Specifically, we transform rank constraint into a differentiable one and impose its adaptive violation-aware penalty on filters. Moreover, this paper is the first work to integrate the learning-based decomposition and group decomposition to make a better trade-off, especially for the tough task of compression of 1×1 convolution.**

The obtained low-rank model can be easily decomposed while nearly keeping the full accuracy without additional fine-tuning process. The effectiveness is verified by compression experiments of VGG and ResNet on CIFAR-10 and ILSVRC-2012. Our method can reduce about 65% parameters of ResNet-110 with 0.04% Top-1 accuracy drop on CIFAR-10, and reduce about 60% parameters of ResNet-50 with 0.57% Top-1 accuracy drop on ILSVRC-2012.

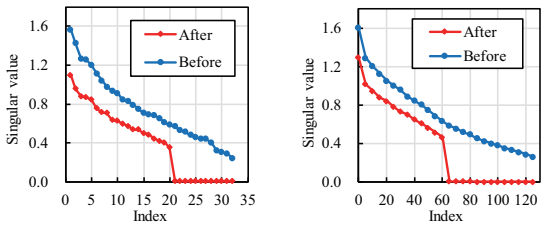
***Index Terms*—low-rank decomposition, network compression, learning-based decomposition, adaptive rank penalty.**

I. Introduction

Deep convolutional neural networks (CNNs) perform remarkably in many vision-based applications [1], [2]. However, it is difficult to deploy CNNs on edge devices, such as mobile phone and embedded devices, due to their huge model size, heavy run-time memory and computational latency. Compression of CNNs aims at reducing parameters and computation with little accuracy drop, and attracts much attention in community.

Low-rank tensor decomposition has shown much promise for CNN compression. Many researchers replace convolutional layers with their low-rank approximations by separable filter [3], [4], SVD [5], [6], Tucker-2 decomposition [7], [8] or Canonical Polyadic (CP) decomposition [9], [10]. In this way, the linearity redundancy of filters is exploited and eliminated.

According to the way of solution, the compression of low-rank decomposition can be divided into two categories: *direct decomposition* and *learning-based decomposition*. *Direct decomposition* is the most frequently-used and simple scheme, and it directly decomposes a full-size model to obtain its low-rank compressed version [3], [5], [7]–[10]. However, this manner reduces the rank of filters too sharply, and thus may lead to performance degradation and the following fine-tuning is needed. Unfortunately, the compressed model tends to be stuck at local minimum [5], [11] during fine-tuning. Thus, there is usually a relatively large accuracy gap between the fine-tuned compressed model and the original model.



(a) ResNet-56 (b) ResNet-50

Fig. 1: Singular values distribution of filter weights before and after applying our learning-based decomposition method. The number of non-zero singular values reflects the rank of filter weights. The last singular values are so close to zero as to be ignored, which indicates the low-rank structure, after applying our method.

Different from direct decomposition, *learning-based decomposition* methods encourage filter weights to be low-rank or correlated during training process, and can be seen as training-based rank reduction (soft reduction). Thus, the post low-rank decomposition and fine-tuning becomes much easier, and this scheme can produce better compressed models. In spite of its soundness, this scheme attracts little study except for several works [12]–[14]. Wen et al. [12] coordinates filters to spin around the origin closer together by a regularizer of pair-wise attractive forces. The regularization on pair-wise distance of filters is heuristic and has no theoretical connections with the rank. Alvarez et al. [13] uses a regularizer of nuclear norm, and optimizes it by soft-thresholding singular values. The nuclear norm enjoys easy optimization, but this convex relaxation hardly satisfies the demand of presetting the target rank. Idelbayev et al. [14] makes rank-constrained optimization on filters to reduce the rank, which is similar to ours except that the penalty strength of [14] increases exponentially as scheduled. That may impose too large strength on rank constraint and then affect the optimization of the network loss due to its non-convexity. Thus, the decomposed model still needs fine-tuning to further optimize the network loss. Moreover, the pre-defined schedule does not consider the actual constraint violation. Above all, these methods can not obtain the satisfactory low-rank weights.

There are two types of decomposition from another perspective. *Single group* is adopted in most existing methods [3],

[5], [7]–[10], [12]–[14]. SVD and Tucker-2 decomposition with single group are shown in Fig.2 (a) and (b), respectively. Compared with coefficient matrices, the basis filters are main carriers of parameter and computation load in the classical networks with a large kernel size, e.g. 3×3 or 5×5. Nowadays, efficient ResNet frequently uses 1×1 convolutions, and the cost of coefficient matrices becomes relatively prominent after compression. In this way, acceptable compression ratio demands severe rank ratio (the ratio of reduced rank to the original rank), and thus it may cause serious accuracy drop for the compression of 1×1 convolution, which makes it a tough compression task. We will analyse this difference in the Sec. III-D. *Multi-Group* decomposition is an intuitive idea to achieve better trade-off by introducing group structures. Peng et al. [11] proposes filter group decomposition, as shown in Fig.2 (c). However, [11] does not utilize the learning-based decomposition for compression, and thus may obtain sub-optimal solution. Besides, the method can neither compress 1×1 convolution well.

In this paper, our method falls into the category of learning-based decomposition, and we formulate it as a rank-constrained optimization problem. Inspire by quadratic penalty function in constrained optimization [15], [16], we propose a novel method called adaptive rank penalty. Specifically, we make rank constraint on filters, transform the rank constraint into equivalent differentiable constraint, and then use an adaptive violation-aware penalty term for the constraint. The rank-constrained optimization reduces the full rank to the target low rank, and this direct reduction of rank can be more effective than the relaxation method of Alvarez et al. [13]. The strength will grow fast if the violation is large, and grow slowly otherwise in our adaptive schedule. Moreover, the adaptive schedule will lead to mild convergence of strength. Compared with the exponentially increased schedule of Idelbayev et al. [14], our method can achieve better balance between optimization of network loss and optimization of rank constraint violation. The obtained full-size model can have both relatively high accuracy and low-rank structure, which is shown in Fig. 1, and thus it can be decomposed while nearly keeping the full accuracy without additional fine-tuning process.

Both learning-based decomposition and group decomposition have virtues, but existing methods only adopt one of them. We are the first to integrate the both. Based on the advanced and frequently-used Tucker-2 decomposition, we make multi-group Tucker-2 decomposition by introducing group coefficient matrices, as shown in Fig.2 (d). More importantly, the group decomposition can compress 1×1 convolution, which is a hard job for the existing methods.

The contributions of this paper can be summarized in three folds:

- We propose a novel learning-based tensor decomposition method called adaptive rank penalty to generate low-rank weights to compress CNNs.
- The paper is the first work to integrate the strategy of group decomposition with learning-based decomposition, and the integrated method can handle the tough task of compression of 1×1 convolution very well.
- The obtained model can be decomposed while nearly keeping the full accuracy without additional fine-tuning process. We conduct extensive experiments of compression of VGG and ResNet on CIFAR-10 and ILSVRC-2012 to validate the effectiveness.

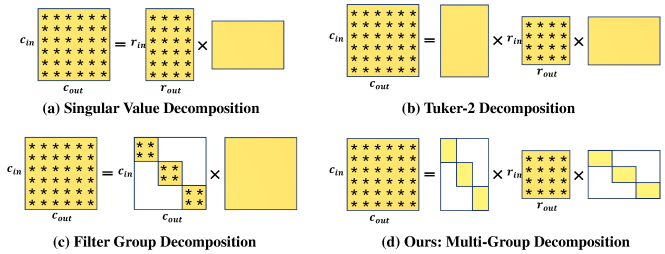


Fig. 2: Different decomposition of filter weight. Filter weight $\mathcal{F} \in \mathbb{R}^{c_{out} \times c_{in} \times w \times h}$ is decomposed into basis filters, which are denoted by symbol "*", whose dim is $\mathbb{R}^{w \times h}$, and reconstruction coefficient matrices, which are denoted by solid color filling. (a), (b) and (d) refer to SVD, Tucker-2 and our multi-group Tucker-2, respectively. (c) refers to filter group decomposition [11], which is different from ours. (a), (b) and (c) cannot deal with compression of 1×1 convolution very well.

II. Related Works

Low-rank decomposition based compression is summarized into two categories: direct decomposition and learning-based decomposition.

Direct Decomposition directly decomposes the full-size model to obtain a low-rank one, and it is conducted mostly in a data-independent manner. (1) Some researchers start from a pre-trained model. [3] decomposes the 2D $d \times d$ kernel into combinations of separable filter $1 \times d$ and $d \times 1$. [4] also uses the same method, but designs an algorithm to get its closed-form solution. [9] and [10] decompose the filter weight (4D tensor) into rank-1 2D tensors by CP decomposition. [5] uses linear combination of basis filters to reconstruct the original filters. [7] and [8] consider the linearity redundancy at both $mode$-1 and $mode$-2 dimension, and use Tucker-2 decomposition. [17] adopts flexible graphical notation to enumerate a series of decomposition classes. [18] decomposes the original weight matrices into their low-rank and sparse components. [19] projects the output and input filter channels of successive layers to a unified low dimensional space to slim the channel. (2) Other researchers replace original convolutions with their low-rank parameterized expansion from scratch. [4] also uses separable filters to replace the original filters, and trains its low-rank model from scratch. [6] learns basis filters to represent the original filters, and jointly minimizes filter reconstruction error and network classification loss. [20] uses tensor-train decomposition to represent original filters, and studies the effect of initialization.

Learning-based decomposition encourages the weights of the full-size model to be correlated or low-rank in the training process, and can learn compact filter weights. Thus, the post decomposition and fine-tuning become easier. Wen et al. [12] coordinates filters closer together to spin around the origin by pair-wise attractive forces, and obtains more correlated

filters. The filters can be reconstructed with fewer basis, but still needs fine-tuning. Alvarez et al. [13] uses a regularizer of nuclear norm to induce low-rank structure of weights, and solves the optimization problem by the method of soft-thresholding singular values, and this method is proposed by Cai et al. [21] in the area of matrix completion. The nuclear norm method can work to some degree, but hardly satisfies the demand of presetting target rank, which is related with whole parameter or FLOPs budget. Idelbayev et al. [14] adopts a dynamic exponential increased schedule of penalty strength with iterations. However, the pre-defined schedule may not be suitable for the actual constraint violation at each iteration.

Different from other methods, we propose an improved method with an adaptive rank penalty to solve the learning-based decomposition. Moreover, we also integrate group decomposition with it. By contrast, existing methods do not integrate the two points.

III. Methods

We first review Tucker-2 decomposition based compression and introduce its multi-group version. Then, we formulate learning-based decomposition as a rank-constrained problem, and then we introduce an adaptive rank penalty method to get its solution. Finally, we analyse the theoretical compression and acceleration ratio of our method.

Preliminaries. Let $\mathcal{F} \in \mathbb{R}^{c_{out} \times c_{in} \times w \times h}$ denote filter weights in a convolutional layer, where c_{in} and c_{out} denote the number of in-channels and out-channels, and kernel size is $w \times h$. Let $\mathcal{F}_{(n)}$ denote $mode$-n flattening of $\mathcal{F}$, where $\mathcal{F}_{(1)} \in \mathbb{R}^{c_{in}wh \times c_{out}}$ and $\mathcal{F}_{(2)} \in \mathbb{R}^{c_{out}wh \times c_{in}}$. Filter $f_j^{out} \in \mathbb{R}^{c_{in}wh}$, $j \in [1, c_{out}]$, and filter $f_i^{in} \in \mathbb{R}^{c_{out}wh}$, $i \in [1, c_{in}]$, are column vectors of $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$, respectively. We refer to the rank of filter set $\{f_j^{out} | \forall j \in [1, c_{out}]\}$ and $\{f_i^{in} | \forall i \in [1, c_{in}]\}$ as rank$(\mathcal{F}_{(1)}) = \min(c_{out}, c_{in}wh)$ and rank$(\mathcal{F}_{(2)}) = \min(c_{in}, c_{out}wh)$ respectively.

A. Review of Tucker-2 Decomposition Based Network Compression

The linearity redundancy of filters is reflected by the rank of filter set. Given rank ratio p (compression hyperparameter), Tucker-2 decomposition reduces linearity redundancy by approximating original filter weights $\mathcal{F}$ by tensors with the reduced rank $r_{out} = p$ rank$(\mathcal{F}_{(1)})$ and $r_{in} = p$ rank$(\mathcal{F}_{(2)})$. Some researchers [7], [8] use Tucker-2 decomposition to decompose filter weight $\mathcal{F}$ into slimmed basis filters (core tensor) $\mathcal{K} \in \mathbb{R}^{r_{out} \times r_{in} \times w \times h}$ and coefficient matrices $A \in \mathbb{R}^{c_{in} \times r_{in}}$ and $Z \in \mathbb{R}^{r_{out} \times c_{out}}$ [1] under reduced rank r_{out} and r_{in} according to rank ratio p, as follows:

$$\begin{cases} \mathcal{F}_{j,i,w,h} = \sum_{m=1}^{r_{in}} a_{i,m} \sum_{n=1}^{r_{out}} \mathcal{K}_{n,m,w',h'} z_{n,j} \\ \mathcal{F} = \mathcal{K} \times_1 Z \times_2 A, \end{cases} \tag{1}$$

Compact high order SVD can directly solve the decomposition of Eq.1 in a data-independent manner (without training data). $\times_1$ and $\times_2$ denote $mode$-1 and $mode$-2 multiply operator in high order SVD. Let columns of $A \in \mathbb{R}^{c_{in} \times r_{in}}$ consist of r_{in} normal orthogonal right singular vectors of $\mathcal{F}_{(2)}$, and rows of $Z \in \mathbb{R}^{r_{out} \times c_{out}}$ consist of the r_{out} ones of $\mathcal{F}_{(1)}$.

For input feature maps $\mathcal{X} \in \mathbb{R}^{c_{in} \times W \times H}$, output feature maps are $\mathcal{Y} = \mathcal{X} * \mathcal{F}$, $\mathcal{Y} \in \mathbb{R}^{c_{out} \times W \times H}$, after the convolution operation. The detailed convolution operation is as follows:

$$\mathcal{Y}_{j,u,v} = \sum_{i=1}^{c_{in}} \sum_{w'=1}^{w} \sum_{h'=1}^{h} \mathcal{X}_{i,u-w',v-h'} \mathcal{F}_{j,i,w',h'}, \tag{2}$$

The accelerated forward computation is $\mathcal{Y} = \mathcal{X} * (\mathcal{K} \times_1 Z \times_2 A)$ after $\mathcal{F}$ is substituted by its element-wise decomposition form of Eq. 1. And its detail is as follows:

$$\begin{aligned} \mathcal{Y}_{j,u,v} &= \sum_{i=1}^{c_{in}} \sum_{w'=1}^{w} \sum_{h'=1}^{h} \mathcal{X}_{i,u-w',v-h'} \left[\sum_{m=1}^{r_{in}} a_{i,m} \sum_{n=1}^{r_{out}} \mathcal{K}_{n,m,w',h'} z_{n,j} \right] \\ &= \sum_{n=1}^{r_{out}} \left[\sum_{m=1}^{r_{in}} \sum_{w'=1}^{w} \sum_{h'=1}^{h} (\sum_{i=1}^{c_{in}} a_{i,m} \mathcal{X}_{i,u-w',v-h'}) \mathcal{K}_{n,m,w',h'} \right] z_{n,j}. \end{aligned} \tag{3}$$

Feature map $\mathcal{X}$ is projected to a low dimensional space (r_{in} channels) at first, and then is convoluted by a few basis filters $\mathcal{K}$, and finally is re-projected to the out-channel space to generate $\mathcal{Y}$. The main parameter and computation load comes from $\mathcal{K}$ and its operation, and thus the network gets accelerated and compressed.

The multi-group version of decomposition is illustrated as follows. Assume we use N groups. Let $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$ be divided into N submatrices $\mathcal{F}_{(1)}^k \in \mathbb{R}^{c_{in}wh \times \frac{c_{out}}{N}}$ and $\mathcal{F}_{(2)}^k \in \mathbb{R}^{c_{out}wh \times \frac{c_{in}}{N}}$ along the column dimension, respectively. Let $A_k \in \mathbb{R}^{\frac{c_{in}}{N} \times \frac{r_{in}}{N}}$ contain the $\frac{r_{in}}{N}$ right singular vectors of $\mathcal{F}_{(2)}^k$ and $Z_k \in \mathbb{R}^{\frac{r_{out}}{N} \times \frac{c_{out}}{N}}$ contain the $\frac{r_{out}}{N}$ right singular vectors of $\mathcal{F}_{(1)}^k$. A or Z is formed by stack of A_k or B_k along the diagonal, respectively. The N-group decomposition can be formulated as:

$$\mathcal{F} = \mathcal{K} \times_1 \text{Diag}(Z_1, Z_2, \cdots, Z_N) \times_2 \text{Diag}(A_1, A_2, \cdots, A_N). \tag{4}$$

B. Problem Formulation of Learning-based Decomposition

Since Tucker-2 decomposition aims to reduce $mode$-1 rank and $mode$-2 rank of $\mathcal{F}$, we want to learn a full-size model $\mathcal{F}$ with low-rank constraint of rank$(\mathcal{F}_{(1)})$=r_{out} and rank$(\mathcal{F}_{(2)})$=r_{in} in a data-dependent manner. The learning-based decomposition can be formulated as follows:

$$\begin{aligned} \min_{\mathcal{F}} \quad & L_{cls}(\mathcal{F}; \{X, Y\}) \\ \text{s.t.} \quad & \text{rank}(\mathcal{F}_{(1)}) = r_{out} \\ & \text{rank}(\mathcal{F}_{(2)}) = r_{in}, \end{aligned} \tag{5}$$

Where, L_{cls} is the task loss (network loss), e.g. cross entropy loss in classification task, and $\{X,Y\}$ is training data. Since low-rank constraint is not convex and differentiable, we transform it to the equivalent equality constraints $B_1 {B_1}^T \mathcal{F}_{(1)} = \mathcal{F}_{(1)}$ and $B_2 {B_2}^T \mathcal{F}_{(2)} = \mathcal{F}_{(2)}$, where $B_1 =$

[1] We don't use $Z \in \mathbb{R}^{c_{out} \times r_{out}}$ in Tucker-2, but its transposed form for easy illustration in this paper.

$\{b_1, b_2, \cdots b_{r_{out}}\} \in \mathbb{R}^{d \times r_{out}}$. B_1 contains r_{out} normal orthogonal basis vectors and spans the out-channel subspace. So does $B_2 \in \mathbb{R}^{d \times r_{in}}$ for the in-channel subspace. The original problem of Eq. 5 can be converted as follows:

$$\begin{aligned} \min_{\mathcal{F}, B_1, B_2} \quad & L_{cls}(\mathcal{F}; \{X, Y\}) \\ \text{s.t.} \quad & B_1 {B_1}^T \mathcal{F}_{(1)} = \mathcal{F}_{(1)} \\ & B_2 {B_2}^T \mathcal{F}_{(2)} = \mathcal{F}_{(2)}. \end{aligned} \tag{6}$$

The meaning of the equality constraint can be interpreted as: as long as the projection of each vector in $\mathcal{F}_{(1)}$ into subspace B_1 equals itself, the rank of $\mathcal{F}_{(1)}$ will equal the dimension of B_1, which is equivalent to rank($\mathcal{F}_{(1)}$)=r_{out}. Both the subspace and filter weights are optimized to get the optimal solution in Eq. 6.

Under the setting of multi-group decomposition, we let B_1^k span a target subspace which $\mathcal{F}_{(1)}^k$ is projected to, $k \in [1, N]$. For simplicity, we assume the dimension of each subspace share the same bound, so we let dim(B_1^k) = $\frac{r_{out}}{N}$. In the same way, we have dim(B_2^k) = $\frac{r_{in}}{N}$. The optimization problem of decomposition with N group subject to constraints of $B_1^k {B_1^k}^T \mathcal{F}_{(1)}^k = \mathcal{F}_{(1)}^k$ and $B_2^k {B_2^k}^T \mathcal{F}_{(2)}^k = \mathcal{F}_{(2)}^k$ can be given as $\min_{\mathcal{F}, B_1^k, B_2^k} L_{cls}(\mathcal{F}; \{X, Y\})$.

C. Optimization Algorithm

Quadratic Penalty [15] is the most common method of solving constrained optimization problem. It adds penalty term to the task loss L_{cls} to transform the original problem into a sequence of unconstrained optimization. When the strength on penalty term approaches infinity, the transformed problem is theoretically equivalent to the origin.

In this section, we omit the superscript k in all formulations for easy illustration, and only give the algorithm for single group decomposition. The same principle is shared by multi-group version. The transformed optimization problem of Eq. 6 can be formulated as follows: ($\|.\|$ denotes Frobenius norm)

$$\begin{aligned} \min_{\mathcal{F}, B_1, B_2} \quad & L_{cls}(\mathcal{F}; \{X, Y\}) + \frac{\lambda_1}{2} \|B_1 {B_1}^T \mathcal{F}_{(1)} - \mathcal{F}_{(1)}\|^2 \\ & + \frac{\lambda_2}{2} \|B_2 {B_2}^T \mathcal{F}_{(2)} - \mathcal{F}_{(2)}\|^2, \end{aligned} \tag{7}$$

By penalizing the reconstruction error, we push filters gradually into a low dimensional subspace.

Inspired by the concept of adaptive penalty strength [16], we consider that the penalty strength should be related with the constraint violation. For simplicity, we update penalty strength λ in a violation-aware manner, as follows:

$$\begin{cases} \lambda_1 := \lambda_1 + \eta \|B_1 {B_1}^T \mathcal{F}_{(1)} - \mathcal{F}_{(1)}\|^2 \\ \lambda_2 := \lambda_2 + \eta \|B_2 {B_2}^T \mathcal{F}_{(2)} - \mathcal{F}_{(2)}\|^2. \end{cases} \tag{8}$$

When the constraint violation does not equal zero, λ will increase. That will penalize the violation more severely and force the weights closer to the feasible region (low-rank structure). The advantage of this adaptive penalty is that increasing of the penalty strength will be determined by the current actual constraint violation. In the experiments, we find it beneficial to limit the maximum value of λ by λ = max(λ, M).

Algorithm 1 Adaptive rank penalty method

Input: Original model $\mathcal{F}$
Output: Low-rank model $\mathcal{F}^*$
for m = 1 $\rightarrow$ Max **do**
  $B_1 := \arg\min_{B_1} \|B_1 {B_1}^T \mathcal{F}_{(1)} - \mathcal{F}_{(1)}\|^2$
  $B_2 := \arg\min_{B_2} \|B_2 {B_2}^T \mathcal{F}_{(2)} - \mathcal{F}_{(2)}\|^2$
  $\mathcal{F} := \mathcal{F} - lr \nabla_{\mathcal{F}} L(\mathcal{F}, B_1, B_2; \{X, Y\})$
  adaptively update λ by Eq. 8
  if $\|B_1 {B_1}^T \mathcal{F}_{(1)} - \mathcal{F}_{(1)}\|^2 < \epsilon$ and $\|B_2 {B_2}^T \mathcal{F}_{(2)} - \mathcal{F}_{(2)}\|^2 < \epsilon$ **then**
    break
  end if
end for

This unconstrained optimization of Eq. 7 with adaptive penalty term can be solved by alternating optimization on $\mathcal{F}$ and (B_1, B_2). The penalty strength is adaptively scheduled by Eq. 8 with iteration. The alternating algorithm is illustrated as follows:

Update B_1 and B_2 when fixing $\mathcal{F}$. For current filters, B_1 and B_2 are optimized by minimizing the violation of constraint, which is formulated as $B_1 = \arg\min_{B_1} \|B_1 {B_1}^T \mathcal{F}_{(1)} - \mathcal{F}_{(1)}\|^2$ and $B_2 = \arg\min_{B_2} \|B_2 {B_2}^T \mathcal{F}_{(2)} - \mathcal{F}_{(2)}\|^2$. The closed-form solution of the problems can be given by SVD. B_1 is taken from the first r_{out} eigenvectors of SVD of $\mathcal{F}_{(1)}$ while B_2 is taken from the first r_{in} ones of $\mathcal{F}_{(2)}$.

Update $\mathcal{F}$ when fixing B_1 and B_2. For current B_1 and B_2, optimal $\mathcal{F}$ is obtained by solving $\arg\min_{\mathcal{F}} L_{cls}(\mathcal{F}; \{X, Y\}) + \frac{\lambda_1}{2} \|B_1 {B_1}^T \mathcal{F}_{(1)} - \mathcal{F}_{(1)}\|^2 + \frac{\lambda_2}{2} \|B_2 {B_2}^T \mathcal{F}_{(2)} - \mathcal{F}_{(2)}\|^2$. Due to the complexity of L_{cls}, the problem is usually solved by gradient method, such as SGD, instead. The gradient of filter weight $\nabla_{\mathcal{F}} L(\mathcal{F}, B_1, B_2; \{X, Y\})$ comes from the sum of two parts: gradient of task loss $\nabla_{\mathcal{F}} L_{cls}(\mathcal{F}; \{X, Y\})$ and gradient of constraint violation $\lambda_1(\mathcal{F}_{(1)} - B_1 {B_1}^T \mathcal{F}_{(1)}) + \lambda_2(\mathcal{F}_{(2)} - B_2 {B_2}^T \mathcal{F}_{(2)})$. The latter part should be reshaped as 4D tensor before adding.

Above all, the algorithm can be summarized as Alg.1. After convergence, the ranks of $\mathcal{F}_{(1)}$ and $\mathcal{F}_{(2)}$ are reduced to r_{out} and r_{in} respectively, which indicates the low-rank structure of the full-size model. Then, the post decomposition of $\mathcal{F}$ in Eq.1 leads to negligible reconstruction error.

D. Theoretical Compression and Acceleration

The original weight $\mathcal{F}$ is decomposed into a slimmed weight $\mathcal{K}$ with N block matrices A_k and Z_k. There is sparse matrix factorization $Z = [Z^{'} | Z^{''}] = Z^{'}[I | Z^{'''}] = Z^{'} Z^s$ to reduce parameters further, where $Z^s \in \mathbb{R}^{r_{out} \times c_{out}}$, $Z^{'} \in \mathbb{R}^{r_{out} \times r_{out}}$ and $Z^{''}, Z^{'''} \in \mathbb{R}^{r_{out} \times (c_{out} - r_{out})}$. $Z^{'}$ can be merged into $\mathcal{K}$. Since the identity submatrix Z^s needs no storage, the number of parameters is reduced from $r_{out} c_{out}$ to $r_{out}(c_{out} - r_{out})$ after sparse matrix factorization of Z. In the same way, we can factorize A_k and Z_k to get A_k^s and Z_k^s, respectively. Thus, the reduction ratio of parameters is:

$$R = 1 - \frac{\text{Param}_{compressed}}{\text{Param}_{original}} = 1 - \frac{r_{in}r_{out}}{c_{in}c_{out}} - \frac{r_{in}(c_{in} - r_{in}) + r_{out}(c_{out} - r_{out})}{c_{in}c_{out}whN}. \quad (9)$$

Given output feature maps $W \times H \times c_{out}$, the computation of convolution operation on slimmed filter $\mathcal{K}$ is $WHc_{out}c_{in}wh$, and the computation of reconstruction operation is $N(\frac{r_{out}}{N}(\frac{c_{out}}{N} - \frac{r_{out}}{N})HW)$ and $N(\frac{r_{in}}{N}(\frac{c_{in}}{N} - \frac{r_{in}}{N})HW)$ at the dimension of out-channel and in-channel, respectively. Thus, the reduction of computation (FLOPs) equals R.

3×3 convolution and 1×1 convolution are frequently-used in most CNNs. Thus, we give a discussion on the R for the two kinds. Assume the numbers of in-channel and out-channel are equal. 3×3 convolution has relatively large kernel-size, e.g. $wh = 9$, and R can be simplified as $1 - p^2 - \frac{2p}{9N}$. For 1×1 convolution, R is simplified as $1 - p^2 - \frac{2p}{N}$. The third term $\frac{2p}{9N}$ or $\frac{2p}{N}$ refers to the part of parameters of coefficient matrices. Given the same rank ratio p, R of 1×1 convolution is smaller than that of 3×3 convolution due to the larger third term, which is caused by small kernel-size. The comparasion shows that the compression of 1×1 convolution needs more severe rank ratio p to achieve comparable R, and that may cause worse accuracy. It suggests that we should apply large group number N to hedge the small kernel size for compression of 1×1 convolution. We also experimentally demonstrate the necessity of large N in Sec. IV-C.

IV. Experiments

We conduct compression experiments for VGG [23] and ResNet [24] to make comparisons with previous methods in terms of metrics, such as the reduction of parameters (or FLOPs) (in Eq.9) and absolute accuracy drop, and also make some ablation study experiments.

Datasets. CIFAR-10 [25] contains 50000 training color images and 10000 testing colorful images, whose size is 32×32, in 10 different classes. ILSVRC-2012 [26] is a large-scale classification dataset, which contains 1.28 million training color images and 50k validation colorful images, whose size is 224×224, in 1000 classes.

Baseline training settings. Models are trained on CIFAR-10 with the common optimizer, data-augmentation and scheduling of learning rate as [24], [27]. The training is scheduled for about 300 epochs with learning rate initialized to 0.1 and divided by 10 at 150, 250 epoch. The optimizer uses SGD with 0.9 momentum and $1e^{-4}$ weight decay. We adopt the common data-augmentation [24], including random 32×32 cropping from a zero-padded 40×40 image or its flipping. The models are trained with a mini-batch size of 64 on a single GPU. On ILSVRC-2012, the training is scheduled for about 90 epochs with the learning rate initialized to 0.1 and divided by 10 at 30 and 60 epoch, which is the same as [24]. We adopt the same optimizer as CIFAR-10. We use the similar data-augmentation in torch example [28], [29]. The models are trained with a mini-batch size of 256 on 4 GPUs.

Training settings of adaptive rank penalty. These are the same as baseline training settings, except for the scheduling of learning rate. On CIFAR-10, we use the same scheduling as [27]. The learning rate is scheduled for about 450 epochs, initialized to $3e^{-3}$ and divided by 10 at 150, 350 epoch. η and M are set to $1e^{-6}$ and 0.1, respectively. On ILSVRC-2012, learning rate is scheduled for about 70 epochs, initialized to $3e^{-3}$ and divided by 10 at 20, 40, 60 epoch. η and M are set to $1e^{-5}$ and 1.0, respectively. There is no additional fine-tuning process after decomposition in ALL experiments.

A. Results on CIFAR-10

We compare our method with previous methods for compression of ResNet-32, 56 and 110. There are almost 3×3 large convolutions in these networks, and we set group number $N = 1$. Small rank ratio will produce more reduction of parameters and FLOPs.

The results on CIFAR-10 are shown in Tab.I. For ResNet-32, our method ($p = \frac{6}{16}$) outperforms previous methods with less accuracy drop while making more or comparable reduction of parameters. When more relaxed rank ratio $p = \frac{10}{16}$ is taken, the compressed model is a little better than baseline model, which we attribute to the effect of regularizer. For ResNet-56, our method ($p = \frac{8}{16}$) reduces much parameters and FLOPs while keeping nearly lossless accuracy. Learning Basis [6] makes more parameter reduction, but the accuracy drop is worsen. For ResNet-110, our method ($p = \frac{8}{16}$) produces more reduction of FLOPs than C-SGD [27], which is a channel pruning method, with a large margin. That may result from that our method can exploit linearity redundancy in very deep network, and doesn't suffer from constraint pruning problem [27]. Our method also outperforms Minnehan et al. [19] in terms of reduction of FLOPs, which is another learning-based decomposition, due to more optimal solution.

B. Results on ILSVRC-2012

Compressions of ResNet-50 and VGG-16 are made on ILSVRC-2012. VGG-16 is compressed with group number $N = 1$. We compress 3×3 convolution with group number $N = 1$ and rank ratio $p = \frac{8}{16}$ and compress 1×1 convolution with group number $N = 4$ and rank ratio $p = \frac{12}{16}$ in ResNet-50.

The results of ILSVRC-2012 are shown in Tab.II. For VGG-16, our method ($p = \frac{6}{16}$,$N = 1$) outperforms Asymmetric [5], Tucker-2 [7], Separable Filter [4] and Group [11] in terms of accuracy drop and FLOPs reduction. Our method also produces more reduction of parameters than Yu et al. [18] under acceptable accuracy drop. There are lots of 1×1 convolutions in ResNet-50, and there is little previous evaluation of tensor decomposition methods on ResNet-50, except for Alvarez et al [13]. That may due to the difficulty of low-rank compression of 1×1 convolutions. Our method ($p = \frac{12}{16}$,$N = 4$) reduces more FLOPs than not only Alvarez et al [13]. but also previous filter pruning methods, like C-SGD [27], CCP [32], and GBN [31], by a large margin. It can verify *the effectiveness on compression of* 1×1 *convolution*.

TABLE I: Compression results on CIFAR-10. ResNet-32, 56 and 110 have 464.15K, 853.02K and 1.73M parameters respectively. Our method is implemented with group number $N = 1$. Param↓ and FLOPs↓ mean reduction of parameters and computation (the more the better), and Acc↓ means accuracy drop (the less the better). The result of TNN is cited form TRN [20].

Model	Method	Baseline Top1(%)	Compressed Top1(%)	Acc↓(%)	Param↓(%)	FLOPs↓(%)
ResNet-32	TNN [30]	92.50	88.30	4.20	79.13	-
	TRN [20]	92.50	90.60	1.90	80.43	-
	Our($p = \frac{6}{16}$)	92.30	91.32	**0.98**	80.00	80.00
	Our($p = \frac{10}{16}$)	92.30	92.36	-0.06	56.22	56.22
ResNet-56	Minnehan et al. [19]	93.60	93.70	-0.10	-	50.20
	GBN-40 [31]	93.10	93.43	-0.33	53.50	60.1
	Learning Basis [6]	93.72	93.40	0.32	78.10	-
	C-SGD-5/8 [27]	93.39	93.44	-0.05	-	60.85
	Our($p = \frac{8}{16}$)	93.53	**93.52**	**0.01**	64.00	**64.00**
ResNet-110	Minnehan et al. [19]	94.29	94.14	0.15	-	50.10
	C-SGD [27]	94.38	94.27	0.11	-	60.89
	Our($p = \frac{8}{16}$)	94.38	**94.34**	**0.04**	**64.98**	**64.98**

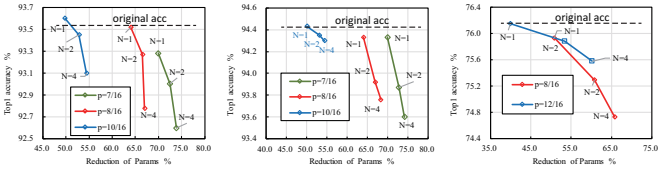
TABLE II: Compression results on ILSVRC-2012. FLOPs↓ means reduction of computation (the more the better), and Acc↓ means accuracy drop (the less the better). FLOPs↓ only involves convolutional layers in VGG-16, same practice as [4], [5], [7], [11]. We calculate the compression ratio of convolutional layers from the statistics of Yu et al. [18] by ourselves.

Model	Method	Baseline(%)		Compressed(%)		Acc↓(%)		FLOPs↓(%)
		Top1	Top5	Top1	Top5	Top1	Top5	
VGG-16	Separable Filter [4]	-	90.60	-	90.31	-	0.29	63.63
	Yu et al. [18]	68.50	88.68	68.75	89.06	-0.15	-0.38	72.71
	Asymmetric(SVD) [5]	-	89.91	-	89.61	-	0.30	75.00
	Group [11]	-	-	-	-	0.28	0.07	77.86
	Tucker-2 [7]	-	89.90	-	89.40	-	0.50	79.72
	Our($p = \frac{6}{16}$,$N = 1$)	71.59	90.38	**71.42**	**90.48**	**0.17**	-0.10	**80.60**
ResNet-50	Alvarez et al. [13]	-	-	75.2	-	-	-	48.59
	CCP [32]	76.15	92.87	75.32	92.54	0.83	0.33	54.10
	GBN-50 [31]	75.85	92.67	75.18	92.41	0.67	0.26	55.06
	C-SGD-50 [27]	75.33	92.56	74.54	92.09	0.79	0.47	55.76
	Our($p = \frac{12}{16}$,$N = 4$)	76.15	92.87	**75.58**	**92.68**	**0.57**	**0.19**	**60.23**

C. Ablation Study on Group Number N

We will empirically explore the effect of group number on parameter (FLOPs) reduction and accuracy. ResNet-56 and 110 are representatives of 3×3 convolution. For 1×1 convolution, we focus on the ablation study of 1×1 convolution in ResNet-50.

There are consistent trendies of accuracy and parameter reduction with group number N incrementally set to 1, 2 and 4, in Fig.3 (a), (b) and (c). The metric of accuracy will decrease, which indicates the drop of network capacity, while the metric of parameter reduction will increase. We are interested in how to choose suitable group number to achieve better tradeoff. The inconsistent conclusion is that there are different suitable group number for 3×3 convolution and 1×1 convolution, respectively. Lower rank ratio p with *smaller group number N is better for* 3×3 *convolution* in Fig.3 (a) and (b). For expample, the result of $p = \frac{8}{16}$ with $N = 1$ can produce more reduction of parameters and achieve more accuracy than the result of $p = \frac{10}{16}$ with $N = 4$ in Fig.3 (a) and (b). By contrast, higher rank ratio p with *larger group number N is better for* 1×1 *convolution* in Fig.3 (c). Thus, We choose $N = 1$ for 3×3 convolution and $N = 4$ for 1×1 convolution in aforementioned compression experiments.



(a) ResNet-56 (b) ResNet-110 (c) ResNet-50

Fig. 3: Effect of N-group decomposition on tradeoff between accuracy and reduction of parameters. The number tagged around data points denotes group number N. (a) and (b) focus on compression of 3×3 convolution, while (c) focuses on compression of 1×1 convolution. (a) and (b) is made on CIFAR-10, while (c) is made on ILSVRC-2012.

Conclusion

In this paper, we propose a novel method called adaptive rank penalty for low-rank compression of CNNs. The obtained low-rank model can be easily decomposed while nearly keeping full performance without additional fine-tuning process. We also lead the first to integrate learning-based decomposition with the strategy of group decomposition, which can handle the tough task of compression of 1×1 convolution. Extensive compression experiments for VGG and ResNet on CIFAR-10 and ILSVRC-2012 prove that our method is more effective than previous methods. In the future, we will explore adaptive configuration of rank ratio p across different layers to make the method more automated.

Acknowledgements

This work has been supported by the Major Project for New Generation of AI under Grant No. 2018AAA0100400, the National Natural Science Foundation of China (NSFC) grants U20A20223 and 61721004.

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, 2015.

[3] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," *arXiv preprint arXiv:1405.3866*, 2014.

[4] C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, "Convolutional neural networks with low-rank regularization," 2015.

[5] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 1943–1955, 2015.

[6] Y. Li, S. Gu, L. V. Gool, and R. Timofte, "Learning filter basis for convolutional neural network compression," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5623–5632.

[7] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," 2016.

[8] H. Ding, K. Chen, and Q. Huo, "Compressing cnn-dblstm models for ocr with teacher-student learning and tucker decomposition," *Pattern Recognition*, vol. 96, p. 106957, 2019.

[9] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[10] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," 2015.

[11] B. Peng, W. Tan, Z. Li, S. Zhang, D. Xie, and S. Pu, "Extreme network compression via filter group approximation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 300–316.

[12] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li, "Coordinating filters for faster deep neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

[13] J. M. Alvarez and M. Salzmann, "Compression-aware training of deep networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 856–867.

[14] Y. Idelbayev and M. A. Carreira-Perpinán, "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*, 2020, pp. 8049–8059.

[15] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[16] R. Vilaa and A. M. A. C. Rocha, "An adaptive penalty method for direct algorithm in engineering optimization," in *International Conference of Numerical Analysis and Applied Mathematics*, 2012.

[17] K. Hayashi, T. Yamaguchi, Y. Sugawara, and S.-i. Maeda, "Exploring unexplored tensor network decompositions for convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 5553–5563.

[18] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7370–7379.

[19] B. Minnehan and A. Savakis, "Cascaded projection: End-to-end network compression and acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 715–10 724.

[20] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal, "Wide compression: Tensor ring nets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9329–9338.

[21] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM Journal on optimization*, vol. 20, no. 4, pp. 1956–1982, 2010.

[22] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with ga's," in *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*. IEEE, 1994, pp. 579–584.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

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

[25] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[26] J. Deng, W. Dong, R. Socher, L. J. Li, and F. F. Li, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 2009.

[27] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal sgd for pruning very deep convolutional networks with complicated structure," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4943–4953.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[29] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4340–4349.

[30] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in neural information processing systems*, 2015, pp. 442–450.

[31] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 2130–2141.

[32] H. Peng, J. Wu, S. Chen, and J. Huang, "Collaborative channel pruning for deep networks," in *International Conference on Machine Learning*, 2019, pp. 5113–5122.