Stable Low-Rank Tensor Decomposition for Compression of Convolutional Neural Network

Anh-Huy Phan[1(✉)], Konstantin Sobolev[1], Konstantin Sozykin[1], Dmitry Ermilov[1], Julia Gusak[1], Petr Tichavský[2], Valeriy Glukhov[3], Ivan Oseledets[1], and Andrzej Cichocki[1]

[1] Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia
{a.phan,konstantin.sobolev,konstantin.sozykin,dmitrii.ermilov,y.gusak,
i.oseledets,a.cichocki}@skoltech.ru
[2] The Czech Academy of Sciences, Institute of Information Theory and Automation, Prague, Czech Republic
tichavsk@utia.cas.cz
[3] Noah's Ark Lab, Huawei Technologies, Shenzhen, China
glukhov.valery@huawei.com

Abstract. Most state-of-the-art deep neural networks are overparameterized and exhibit a high computational cost. A straightforward approach to this problem is to replace convolutional kernels with its low-rank tensor approximations, whereas the Canonical Polyadic tensor Decomposition is one of the most suited models. However, fitting the convolutional tensors by numerical optimization algorithms often encounters diverging components, i.e., extremely large rank-one tensors but canceling each other. Such degeneracy often causes the non-interpretable result and numerical instability for the neural network ne-tuning. This paper is the first study on degeneracy in the tensor decomposition of convolutional kernels. We present a novel method, which can stabilize the low-rank approximation of convolutional kernels and ensure efficient compression while preserving the high-quality performance of the neural networks. We evaluate our approach on popular CNN architectures for image classification and show that our method results in much lower accuracy degradation and provides consistent performance.

Keywords: Convolutional neural network acceleration · Low-rank tensor decomposition · Sensitivity · Degeneracy correction

1 Introduction

Convolutional neural networks (CNNs) and their recent extensions have significantly increased their ability to solve complex computer vision tasks, such as image classification, object detection, instance segmentation, image generation, etc. Together with big data and fast development of the internet of things, CNNs bring new tools for solving computer science problems, which are intractable using classical approaches.

Despite the great successes and rapid development of CNNs, most modern neural network architectures contain a huge number of parameters in the convolutional

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12374, pp. 522–539, 2020.
https://doi.org/10.1007/978-3-030-58526-6_31

and fully connected layers, therefore, demand extremely high computational costs [46], which makes them difficult to deploy on devices with limited computing resources, like PC or mobile devices. Common approaches to reduce redundancy of the neural network parameters are: structural pruning [13,20,21,59], sparsification [12,15,36], quantization [2,44] and low-rank approximation [4,10,14,26,28,33].

The weights of convolutional and fully connected layers are usually overparameterized and known to lie on a low-rank subspace [9]. Hence, it is possible to represent them in low-rank tensor/tensor network formats using e.g., Canonical Polyadic decomposition (CPD) [1,10,33], Tucker decomposition [14,26], Tensor Train decomposition [37,55]. The decomposed layers are represented by a sequence of new layers with much smaller kernel sizes, therefore, reducing the number of parameters and computational cost in the original model.

Various low-rank tensor/matrix decompositions can be straightforwardly applied to compress the kernels. This article intends to promote the simplest tensor decomposition model, the Canonical Polyadic decomposition (CPD).

1.1 Why CPD

In neural network models working with images, the convolutional kernels are usually tensors of order 4 with severely unbalanced dimensions, e.g., $D \times D \times S \times T$, where $D \times D$ represents the filter sizes, S and T denote the number of input and output channels, respectively. The typical convolutional filters are often of relatively small sizes, e.g., 3×3, 7×7, compared to the input (S) and output (T) dimensions, which in total may have hundred of thousands of filters. This leads to excessive redundancy among the kernel filters, which are particularly suited for tensor decomposition methods. Among low-rank tensor decomposition and tensor networks, the Canonical Polyadic tensor decomposition [17,22] is the simplest and elegant model, which represents a tensor by sum of rank-1 tensors[1] or equivalently by factor matrices interconnected through a diagonal tensor (Fig. 1a). The number of parameters for a CP model of rank-R is $R(2D + S + T)$ or $R(D^2 + S + T)$ when we consider kernels as order-4 tensors or their reshaped order-3 versions, respectively. Usually, CPD gains a relatively high compression ratio since the decomposition rank is not very large [14,33].

Representation of the high order convolutional kernels in the form of the CP model is equivalent to the use of separable convolutions. In [28], the authors modeled the high order kernels in the generalized multiway convolution by the CP model.

The Tucker tensor decomposition (TKD) [52] is an alternative tensor decomposition method for convolutional kernel compression [26]. The TKD provides more flexible interaction between the factor matrices through a core tensor, which is often dense in practice (Fig. 1b). Kim et al. [26] investigated low-rank models at the most suited noise level for different unfoldings[2] of the kernel tensor. This heuristic method does not consider a common noise level for multi modes and is not optimal to attain the approximation error bound.

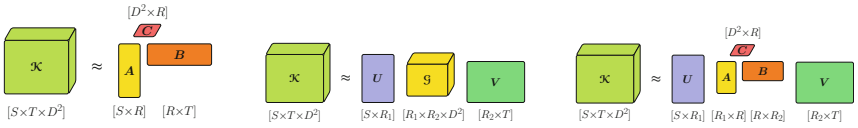
[1] Rank-1 tensor of size $n_1 \times n_2 \times \cdots \times n_d$ is an outer product of d vectors with dimensions $n_1, n_1, \ldots, n_d$.

[2] The mode-j unfolding of an order-d tensor of size $n_1 \times n_2 \times \cdots \times n_d$ reorders the elements of the tensor into a matrix with n_j rows and $n_1 \ldots n_{j-1} n_{j+1} \ldots n_d$ columns.

Block tensor decomposition [6] is an extension of the TKD, which models data as the sum of several Tucker or Kruskal terms, i.e., a TKD with block-diagonal core tensor. For the same multilinear rank as in TKD, BTD exhibits a smaller number of parameters; however, there are no available proper criteria for block size selection (rank of BTD).

In addition, the other tensor networks, e.g., Tensor Train [38] or Tensor Chain (TC) [11,25], are not applicable unless the kernel filters are tensorized to higher orders. Besides, the Tensor Chain contains a loop, is not closed and leads to severe numerical instability to nd the best approximation, see Theorem 14.1.2.2 [16,31].

We later show that CPD can achieve much better performance with an even higher compression ratio by further compression the Tucker core tensors by solving a suitably formulated optimization problem.



(a) CPD (b) TKD (c) TKD-CPD

Fig. 1. Approximation of a third-order tensor using Canonical Polyadic tensor decomposition (CPD), Tucker-2 tensor decomposition (TKD), and their combination (TKD-CPD). CPD and TKD are common methods applied for CNN compression.

1.2 Why Not Standard CPD

In one of the first works applying CPD to convolutional kernels, Denton et al. [10] computed the CPD by sequentially extracting the best rank-1 approximation in a greedy way. This type of deflation procedure is not a proper way to compute CPD unless decomposition of orthogonally decomposable tensors [57] or with a strong assumption, e.g., at least two factor matrices are linearly independent, and the tensor rank must not exceed any dimension of the tensor [41]. The reason is that subtracting the best rank-1 tensor does not guarantee to decrease the rank of the tensor [49].

In [33], the authors approximated the convolution kernel tensors using the Nonlinear Least Squares (NLS) algorithm [54], one of the best existing algorithms for CPD. However, as mentioned in the Ph.D. thesis [32], it is not trivial to optimize a neural network even when weights from a single layer are factorized, and the authors "failed to find a good SGD learning rate" with fine-tuning a classification model on the ILSVRC-12 dataset.

Diverging Component - Degeneracy. Common phenomena when using numerical optimization algorithms to approximate a tensor of relatively high rank by a low-rank model or a tensor, which has nonunique CPD, is that there should exist at least two rank-one tensors such that their Frobenius norms or intensities are relatively high but cancel each other [47], $\|\mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(d)}\|_F \to \infty$.

The degeneracy of CPD is reported in the literature, e.g., in [5,18,29,35,39,45]. Some efforts which impose additional constraints on the factor matrices can improve stability and accelerate convergence, such as, column-wise orthogonality [29,45], positivity or nonnegativity [34]. However, the constraints are not always applicable in some data, and thus prevent the estimator from getting lower approximation error, yielding to the trade-off between estimation stability and good approximation error.[3]

We have applied CPD approximations for various CNNs and confirm that the diverging component occurs for most cases when we used either Alternating Least Squares (ALS) or NLS [54] algorithm. As an example, we approximated one of the last convolutional layers from ResNet-18 with rank-500 CPD and plotted in Fig. 2(left) intensities of CPD components, i.e., Frobenius norm of rank-1 tensors. The ratio between the largest and smallest intensities of rank-1 tensors was greater than 30. Figure 2(right) shows that the sum of squares of intensities for CPD components is (exponentially) higher when the decomposition is with a higher number of components. Another criterion, sensitivity (Definition 1), shows that the standard CPD algorithms are not robust to small perturbations of factor matrices, and sensitivity increases with higher CP rank.

Such degeneracy causes the instability issue when training a CNN with decomposed layers in the CP (or Kruskal) format. More specifically, it causes difficulty for a neural network to perform fine-tuning, selecting a good set of parameters, and maintaining stability in the entire network. This problem has not been investigated thoroughly. To the best of our knowledge, there is no method for handling this problem.

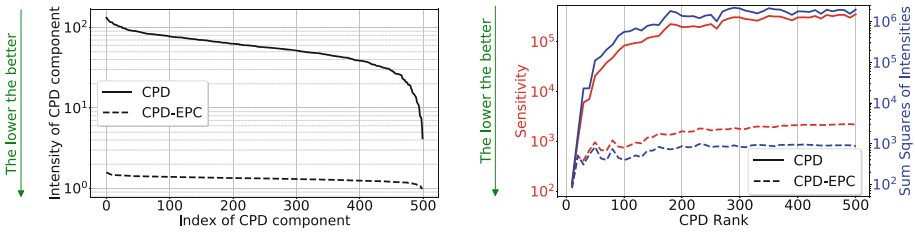


Fig. 2. (Left) Intensity (Frobenius norm) of rank-1 tensors in CPDs of the kernel in the 4^{th} layer of ResNet-18. (Right) Sum of squares of the intensity and Sensitivity vs Rank of CPD. EPC-CPD demonstrates much lower intensity and sensitivity as compared to CPD.

1.3 Contributions

In this paper, we address the problem of CNN stability compressed by CPD. The key advantages and major contributions of our paper are the following:

– We propose a new stable and efficient method to perform neural network compression based on low-rank tensor decompositions.

[3] As shown in [53], RMS error is not the only one minimization criterion for a particular computer vision task.

– We demonstrate how to deal with the degeneracy, the most severe problem when approximating convolutional kernels with CPD. Our approach allows finding CPD a reliable representation with minimal sensitivity and intensity.
– We show that the combination of Tucker-2 (TKD) and the proposed stable CPD (Fig. 1c) outperforms CPD in terms of accuracy/compression trade-off.

We provide results of extensive experiments to confirm the efficiency of the proposed algorithms. Particularly, we empirically show that the neural network with weights in factorized CP format obtained using our algorithms is more stable during fine-tuning and recovers faster (close) to initial accuracy.

2 Stable Tensor Decomposition Method

2.1 CP Decomposition of Convolutional Kernel

In CNNs, the convolutional layer performs mapping of an input (source) tensor $\mathcal{X}$ of size $H \times W \times S$ into output (target) tensor $\mathcal{Y}$ of size $H' \times W' \times T$ following the relation

$$\mathcal{Y}_{h',w',t} = \sum_{i=1}^{D} \sum_{j=1}^{D} \sum_{s=1}^{S} \tilde{\mathcal{K}}_{i,j,s,t} \mathcal{X}_{h_i,w_j,s}, \tag{1}$$

where $h_i = (h' - 1)\,\Delta + i - P$, and $w_j = (w' - 1)\,\Delta + j - P$, and $\tilde{\mathcal{K}}$ is an order-4 kernel tensor of size $D \times D \times S \times T$, Δ is stride, and P is zero-padding size.

Our aim is to decompose the kernel tensor $\tilde{\mathcal{K}}$ by the CPD or the TKD. As it was mentioned earlier, we treat the kernel $\tilde{\mathcal{K}}$ as order-3 tensor $\mathcal{K}$ of the size $D^2 \times S \times T$, and represent the kernel $\mathcal{K}$ by sum of R rank-1 tensors

$$\mathcal{K} \simeq \hat{\mathcal{K}} = \sum_{r=1}^{R} \boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r, \tag{2}$$

where $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_R]$, $\mathbf{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_R]$ and $\mathbf{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_R]$ are factor matrices of size $D^2 \times R$, $S \times R$ and $T \times R$, respectively. See an illustration of the model in Fig. 1a. The tensor $\hat{\mathcal{K}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ in the Kruskal format uses $(D^2 + S + T) \times R$ parameters.

2.2 Degeneracy and Its Effect to CNN Stability

Degeneracy occurs in most CPD of the convolutional kernels. The Error Preserving Correction (EPC) method [42] suggests a correction to the decomposition results in order to get a more stable decomposition with lower sensitivity. There are two possible measures for assessment of the degeneracy degree of the CPD: sum of Frobenius norms of the rank-1 tensors [42]

$$\mathrm{sn}([\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) = \sum_{r=1}^{R} \|\boldsymbol{a}_r \circ \boldsymbol{b}_r \circ \boldsymbol{c}_r\|_F^2 \tag{3}$$

and sensitivity, defined as follows.

Definition 1 (Sensitivity [51]). *Given a tensor* $\mathcal{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$*, define the sensitivity as*

$$\mathtt{ss}([\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) = \lim_{\sigma^2 \to 0} \frac{1}{R\sigma^2} E\{||\mathcal{T} - [\![\mathbf{A} + \delta\mathbf{A}, \mathbf{B} + \delta\mathbf{B}, \mathbf{C} + \delta\mathbf{C}]\!]||_F^2\} \tag{4}$$

where $\delta\mathbf{A}$*,* $\delta\mathbf{B}$*,* $\delta\mathbf{C}$ *have random i.i.d. elements from* $N(0, \sigma^2)$*.*

The sensitivity of the decomposition can be measured by the expectation ($E\{\cdot\}$) of the normalized squared Frobenius norm of the difference. In other words, sensitivity of the tensor $\mathcal{T} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is a measure with respect to perturbations in individual factor matrices. CPDs with high sensitivity are usually useless.

Lemma 1.

$$\mathtt{ss}([\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]) = \mathrm{tr}\{(\mathbf{A}^T\mathbf{A}) \circledast (\mathbf{B}^T\mathbf{B}) + (\mathbf{B}^T\mathbf{B}) \circledast (\mathbf{C}^T\mathbf{C}) + (\mathbf{A}^T\mathbf{A}) \circledast (\mathbf{C}^T\mathbf{C})\}. \tag{5}$$

where $\circledast$ *denotes the Hadamard element-wise product.*

Proof. First, the perturbed tensor in (4) can be expressed as sum of 8 Kruskal terms

$$\begin{aligned}[\![\mathbf{A} + \delta\mathbf{A}, \mathbf{B} + \delta\mathbf{B}, \mathbf{C} + \delta\mathbf{C}]\!] &= [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] + [\![\delta\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] + [\![\mathbf{A}, \delta\mathbf{B}, \mathbf{C}]\!] + [\![\mathbf{A}, \mathbf{B}, \delta\mathbf{C}]\!] \\ &+ [\![\delta\mathbf{A}, \delta\mathbf{B}, \mathbf{C}]\!] + [\![\delta\mathbf{A}, \mathbf{B}, \delta\mathbf{C}]\!] + [\![\mathbf{A}, \delta\mathbf{B}, \delta\mathbf{C}]\!] + [\![\delta\mathbf{A}, \delta\mathbf{B}, \delta\mathbf{C}]\!].\end{aligned}$$

Since these Kruskal terms are uncorrelated and expectation of the terms composed by two or three factor matrices $\delta\mathbf{A}$, $\delta\mathbf{B}$ and $\delta\mathbf{C}$ are negligible, the expectation in (4) can be expressed in the form

$$\begin{aligned}E\{||\mathcal{T} - [\![\mathbf{A} + \delta\mathbf{A}, \mathbf{B} + \delta\mathbf{B}, \mathbf{C} + \delta\mathbf{C}]\!]||_F^2\} &= E\{||[\![\delta\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]||_F^2\} \\ &+ E\{||[\![\mathbf{A}, \delta\mathbf{B}, \mathbf{C}]\!]||_F^2\} + E\{||[\![\mathbf{A}, \mathbf{B}, \delta\mathbf{C}]\!]||_F^2\}.\end{aligned} \tag{6}$$

Next we expand the Frobenius norm of the three Kruskal tensors

$$\begin{aligned}E\{||[\![\delta\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]||_F^2\} &= E\{||\,((\mathbf{C} \odot \mathbf{B}) \otimes \mathbf{I})\,\mathrm{vec}(\delta\mathbf{A})\,||^2\} \\ &= E\{\mathrm{tr}((\mathbf{C} \odot \mathbf{B}) \otimes \mathbf{I})^T\,((\mathbf{C} \odot \mathbf{B}) \otimes \mathbf{I})\,\mathrm{vec}(\delta\mathbf{A})\,\mathrm{vec}(\delta\mathbf{A})^T)\} \\ &= \sigma^2\,\mathrm{tr}((\mathbf{C} \odot \mathbf{B})^T(\mathbf{C} \odot \mathbf{B}) \otimes \mathbf{I}) \\ &= R\sigma^2\,\mathrm{tr}((\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{B}^T\mathbf{B}))\end{aligned} \tag{7}$$

$$E\{||[\![\mathbf{A}, \delta\mathbf{B}, \mathbf{C}]\!]||_F^2\} = R\sigma^2\,\mathrm{tr}((\mathbf{C}^T\mathbf{C}) \circledast (\mathbf{A}^T\mathbf{A})) \tag{8}$$

$$E\{||[\![\mathbf{A}, \mathbf{B}, \delta\mathbf{C}]\!]||_F^2\} = R\sigma^2\,\mathrm{tr}((\mathbf{B}^T\mathbf{B}) \circledast (\mathbf{A}^T\mathbf{A})) \tag{9}$$

where $\odot$ and $\otimes$ are Khatri-Rao and Kronecker products, respectively.

Finally, we replace these above expressions into (6) to obtain the compact expression of sensitivity.

2.3 Stabilization Method

Sensitivity Minimization. The first method to correct CPD with diverging components proposed in [42] minimizes the sum of Frobenius norms of rank-1 tensors while the approximation error is bounded. In [51]. the Krylov Levenberg-Marquardt algorithm was proposed for the CPD with bounded sensitivity constraint.

In this paper, we propose a variant of the EPC method which minimizes the sensitivity of the decomposition while preserving the approximation error, i.e.,

$$\begin{aligned} \min_{\{\mathbf{A},\mathbf{B},\mathbf{C}\}} \quad & \mathtt{ss}([\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]) \\ \text{s.t.} \quad & \|\mathcal{K} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 \leq \delta^2 \,. \end{aligned} \tag{10}$$

The bound, δ^2, can represent the approximation error of the decomposition with diverging components. Continuing the CPD using a new tensor $\hat{\mathcal{K}} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]$ with a lower sensitivity can improve its convergence.

Update Rules. We derive alternating update formulas for the above optimization problem. While $\mathbf{B}$ and $\mathbf{C}$ are kept fixed, the objective function is rewritten to update $\mathbf{A}$ as

$$\begin{aligned} \min_{\mathbf{A}} \quad & \mathrm{tr}\{(\mathbf{A}^T\mathbf{A}) \circledast \mathbf{W}\} = \|\mathbf{A}\,\mathrm{diag}(\boldsymbol{w})\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{K}_{(1)} - \mathbf{A}\mathbf{Z}^T\|_F^2 \leq \delta^2, \end{aligned} \tag{11}$$

where $\mathbf{K}_{(1)}$ is mode-1 unfolding of the kernel tensor $\mathcal{K}$, $\mathbf{Z} = \mathbf{C} \odot \mathbf{B}$ and $\mathbf{W} = \mathbf{B}^T\mathbf{B} + \mathbf{C}^T\mathbf{C}$ is a symmetric matrix of size $R \times R$, $\boldsymbol{w} = [\sqrt{w_{1,1}}, \ldots, \sqrt{w_{R,R}}]$ is a vector of length R taken from the diagonal of $\mathbf{W}$.

Remark 1. The problem (11) can be reformulated as a regression problem with bound constraint

$$\begin{aligned} \min_{\widetilde{\mathbf{A}}} \quad & \|\widetilde{\mathbf{A}}\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{K}_{(1)} - \widetilde{\mathbf{A}}\widetilde{\mathbf{Z}}^T\|_F^2 \leq \delta^2, \end{aligned} \tag{12}$$

where $\widetilde{\mathbf{A}} = \mathbf{A}\,\mathrm{diag}(\boldsymbol{w})$ and $\widetilde{\mathbf{Z}} = \mathbf{Z}\,\mathrm{diag}(\boldsymbol{w}^{-1})$. This problem can be solved in closed form solution through the quadratic programming over a sphere [43]. We skip the algorithm details and refer to the solver in [43].

Remark 2. If factor matrices $\mathbf{B}$ and $\mathbf{C}$ are normalized to unit length columns, i.e., $\|\boldsymbol{b}_r\|_2 = \|\boldsymbol{c}_r\|_2 = 1$, $r = 1, \ldots, R$, then all entries of the diagonal of $\mathbf{W}$ are identical. The optimization problem in (11) becomes seeking a weight matrix, $\mathbf{A}$, with minimal norm

$$\begin{aligned} \min_{\mathbf{A}} \quad & \|\mathbf{A}\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{K}_{(1)} - \mathbf{A}\mathbf{Z}^T\|_F^2 \leq \delta^2. \end{aligned} \tag{13}$$

This sub-optimization problem is similar to that in the EPC approach [42].

2.4 Tucker Decomposition with Bound Constraint

Another well-known representation of multi-way data is the Tucker Decomposition [52], which decomposes a given tensor into a core tensor and a set of factor matrices (see Fig. 1b for illustration). The Tucker decomposition is particularly suited as prior-compression for CPD. In this case, we compute CPD of the core tensor in TKD, which is of smaller dimensions than the original kernels.

For our problem, we are interested in the Tucker-2 model (see Fig. 1b)

$$\mathcal{K} \simeq \mathcal{G} \times_2 \mathbf{U} \times_3 \mathbf{V}, \tag{14}$$

where $\mathcal{G}$ is the core tensor of size $D^2 \times R_1 \times R_2$, $\mathbf{U}$ and $\mathbf{V}$ are matrices of size $S \times R_1$ and $T \times R_2$, respectively. Because of rotational ambiguity, without loss in generality, the matrices $\mathbf{U}$ and $\mathbf{V}$ can be assumed to have orthonormal columns.

Different from the ordinary TK-2, we seek the smallest TK-2 model which holds the approximation error bound δ^2 [40], i.e.,

$$\begin{aligned} \min_{\{\mathcal{G},\mathbf{U},\mathbf{V}\}} \quad & R_1 S + R_2 T + R_1 R_2 D^2 \\ \text{s.t.} \quad & \|\mathcal{K} - \mathcal{G} \times_2 \mathbf{U} \times_3 \mathbf{V}\|_F^2 \le \delta^2 \\ & \mathbf{U}^T\mathbf{U} = \mathbf{I}_{R_1}, \mathbf{V}^T\mathbf{V} = \mathbf{I}_{R_2}\,. \end{aligned} \tag{15}$$

We will show that the core tensor $\mathcal{G}$ has closed-form expression as in the HOOI algorithm for the orthogonal Tucker decomposition [7], and the two-factor matrices, $\mathbf{U}$ and $\mathbf{V}$, can be *sequentially estimated* through Eigenvalue decomposition (EVD).

Lemma 2. *The core tensor* $\mathcal{G}$ *has closed-form expression* $\mathcal{G}^\star = \mathcal{K} \times_2 \mathbf{U}^T \times_3 \mathbf{V}^T$.

Proof. From the error bound condition, we can derive

$$\delta^2 \ge \|\mathcal{K} - \mathcal{G} \times_2 \mathbf{U} \times_3 \mathbf{V}\|_F^2 = \|\mathcal{K}\|_F^2 - \|\mathcal{G}^\star\|_F^2 + \|\mathcal{G} - \mathcal{G}^\star\|_F^2,$$

which indicates that the core tensor can be expressed as $\mathcal{G} = \mathcal{G}^\star + \mathcal{E}$, where $\mathcal{E}$ is an error tensor such that its norm $\gamma^2 = \|\mathcal{E}\|_F^2 \le \delta^2 + \|\mathcal{G}^\star\|_F^2 - \|\mathcal{K}\|_F^2$.

Next define a matrix $\mathbf{Q}_1$ of size $S \times S$

$$\mathbf{Q}_1(i,j) = \sum_{r=1}^{R_2} \mathbf{V}(:,r)^T\, \mathbf{K}(:,i,:)\mathbf{K}(:,j,:)^T\, \mathbf{V}(:,r)\,. \tag{16}$$

Assume that $\mathbf{V}^\star$ is the optimal factor matrix with the minimal rank $R_2^\star$. The optimization in (15) becomes the rank minimization problem for $\mathbf{U}$

$$\begin{aligned} \min_{\mathbf{U}} \quad & \text{rank}(\mathbf{U}) \\ \text{s.t.} \quad & \text{tr}(\mathbf{U}^T\mathbf{Q}_1\mathbf{U}) \ge \|\mathcal{K}\|_F^2 + \gamma^2 - \delta^2\,, \\ & \mathbf{U}^T\mathbf{U} = \mathbf{I}_{R_1}. \end{aligned} \tag{17}$$

The optimal factor matrix $\mathbf{U}^\star$ comprises R_1 principal eigenvectors of $\mathbf{Q}_1$, where R_1 is the smallest number of eigenvalues, $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_{R_1}$ such that their norm exceeds

the bound $\|\mathcal{Y}\|_F^2 - \delta^2 + \gamma^2$, that is, $\sum_{r=1}^{R_1} \lambda_r \geq \|\mathcal{K}\|_F^2 - \delta^2 + \gamma^2 > \sum_{r=1}^{R_1-1} \lambda_r$. It is obvious that the minimal number of columns R_1 is achieved, when the bound $\|\mathcal{K}\|_F^2 + \gamma^2 - \delta^2$ is smallest, i.e., $\gamma = 0$. Implying that the optimal $\mathcal{G}$ is $\mathcal{G}^\star$. This completes the proof.

Similar to the update of $\mathbf{U}$, the matrix $\mathbf{V}$ comprises R_2 principal eigenvectors of the matrix $\mathbf{Q}_2$ of size $T \times T$

$$\mathbf{Q}_2(i, j) = \sum_{r=1}^{R_1} \mathbf{U}(:, r)^T \, \mathbf{K}(:, :, i)^T \mathbf{K}(:, :, k) \, \mathbf{U}(:, r), \tag{18}$$

where R_2 is either given or determined based on the bound $\|\mathcal{Y}\|_F^2 - \delta^2$. The algorithm for TKD sequentially updates $\mathbf{U}$ and $\mathbf{V}$.

3 Implementation

Our method for neural network compression includes the following main steps (see Fig. 3):

1. Each convolutional kernel is approximated by a tensor decomposition (CPD/TKD-CPD in case of ordinary convolutions and SVD in case of 1×1 convolution) with given rank R.
2. The CP decomposition with diverging components is corrected using the error preserving method. The result is a new CP model with minimal sensitivity.
3. An initial convolutional kernel is replaced with a tensor in CPD/TKD-CPD or SVD format, which is equivalent to replacing one convolutional layer with a sequence of convolutional layers with a smaller total number of parameters.
4. The entire network is then fine-tuned using backpropagation.

CPD Block results in three convolutional layers with shapes ($C_{in} \times R \times 1 \times 1$), depthwise ($R \times R \times D \times D$) and ($R \times C_{out} \times 1 \times 1$), respectively (see Fig. 3a). In obtained structure, all spatial convolutions are performed by central $D \times D$ group convolution with R channels. 1×1 convolutions allow the transfer of input data to a more compact channel space (with R channels) and then return data to initial channel space.

TKD-CPD Block is similar to the CPD block, but has 4 (1×1) convolutional layers with the condition that the CP rank must exceed the multilinear ranks, R_1 and R_2 (see Fig. 3c). This structure allows additionally to reduce the number of parameters and floating operations in a factorized layer. Otherwise, when $R < R_1$ and $R < R_2$, sequential 1×1 convolutions can be merged into one 1×1 convolution, converting the TKD-CPD layer format to CPD block.

SVD Block is a variant of CPD Block but comprises only two-factor layers, computed using SVD. Degeneracy is not considered in this block, and no correction is applied (see Fig. 3b).

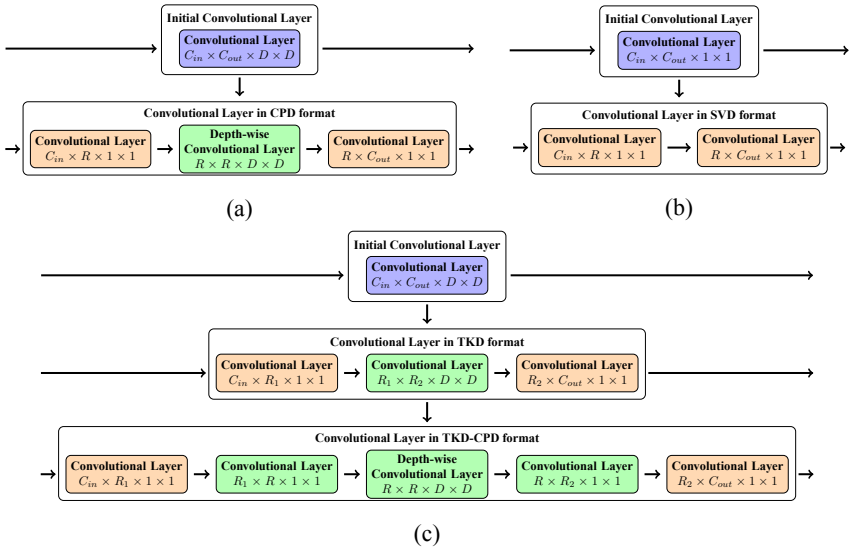


Fig. 3. Graphical illustration to the proposed layer formats that show how decomposed factors are used as new weights of the compressed layer. C_{in}, C_{out} are the number of input of and output channels and D is a kernel size. (a) CPD layer format, R is a CPD rank. (b) SVD layer format, R is a SVD rank. (c) TKD-CPD layer format, R is a CPD rank, R_1 and R_2 are TKD ranks.

Rank Search Procedure. Determination of CP rank is an NP-hard problem [22]. We observe that the drop in accuracy by a factorized layer influences accuracy with fine-tuning of the whole network. In our experiments, we apply a heuristic binary search to find the smallest rank such that drop after single layer fine-tuning does not exceed a predefined accuracy drop threshold EPS.

4 Experiments

We test our algorithms on three representative convolutional neural network architectures for image classification: *VGG-16* [48], *ResNet-18*, *ResNet-50* [19]. We compressed 7×7 and 3×3 convolutional kernels with CPD, CPD with sensitivity correction (CPD-EPC), and Tucker-CPD with the correction (TKD-CPD-EPC). The networks after fine-tuning are evaluated through *top 1* and *top 5* accuracy on *ILSVRC-12* [8] and *CIFAR-100* [30].

We conducted a series of layer-wise compression experiments and measured accuracy recovery and whole model compression of the decomposed architectures. Most of our experiments were devoted to the approximation of single layers when other layers remained intact. In addition, we performed compression of entire networks.

The experiments were conducted with the popular neural networks framework *Pytorch* on GPU server with NVIDIA V-100 GPUs. As a baseline for ILSVRC-12 we used a pre-trained model shipped with *Torchvision*. Baseline CIFAR-100 model was trained using the Cutout method. The fine-tuning process consists of two parts: local or

single layer fine-tuning, and entire network fine-tuning. The model was trained with an SGD optimizer with an initial learning rate of 10^{-3} and learning decay of 0.1 at each loss saturation stage, weight decay was set as 10^{-4}.

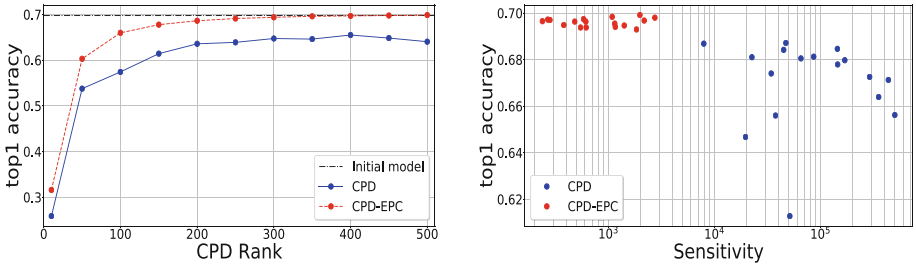


Fig. 4. (Left) Performance evaluation of ResNet-18 on ILSVRC-12 dataset after replacing `layer4.1.conv1` by its approximation using CPD and CPD-EPC with various ranks. The networks are fine-tuned after compression. (Right) Top-1 accuracy and sensitivity of the models estimated using CPD (blue) and CPD-EPC (red). Each model has a single decomposed layer with the best CP rank and was fine-tuned after compression. CPD-EPC outperforms CPD in terms of accuracy/sensitivity trade-off. `layer4.1.conv1` – layer 4, residual block 2 (indexing starts with 0), convolutional layer 1 (Color figure online)

4.1 Layer-Wise Study

CPD-EPC Vs CPD. For this study, we decomposed the kernel filters in 17 convolutional layers of ResNet-18 with different CP ranks, R, ranging from small (10) to relatively high rank (500). The CPDs were run with a sufficiently large number of iterations so that all models converged or there was no significant improvement in approximation errors.

Experiment results show that for all decomposition ranks, the CPD-EPC regularly results in considerably higher *top 1* and *top 5* model accuracy than the standard CPD algorithm. Figure 4 (left) demonstrates an illustrative example for `layer4.1.conv1`. An important observation is that the compressed network using CPD even with the rank of 500 (and fine-tuning) does not achieve the original network's accuracy. However, with EPC, the performances are much better and attain the original accuracy with the rank of 450. Even a much smaller model with the rank of 250 yields a relatively good result, with less than 1% loss of accuracy.

Next, each convolutional layer in ResNet-18 was approximated with different CP ranks and fine-tuned. The best model in terms of top-1 accuracy was then selected. Figure 4 (right) shows relation between the sensitivity and accuracy of the best models. It is straightforward to see that the models estimated using CPD exhibit high sensitivity, and are hard to train. The CPD-EPC suppressed sensitivities of the estimated models and improved the performance of the compressed networks. The CPD-EPC gained the most remarkable accuracy recovery on deeper layers of CNNs.

The effect is significant for some deep convolutional layers of the network with $\sim 2\%$ top-1 accuracy difference.

CPD-EPC Vs TKD-EPC. Next, we investigated the proposed compression approach based on the hybrid TKD-CPD model with sensitivity control. Similar experiments were conducted for the CIFAR-100 dataset. The TK multi-linear ranks (R_1, R_2) were kept fixed, while the CP rank varied in a wide range.

In Fig. 5, we compare accuracy of the two considered compressed approaches applied to the layer `4.0.conv1` in ResNet-18. For this case, CPD-EPC still demonstrated a good performance. The obtained accuracy is very consistent, implying that the layer exhibits a low-rank structure. The hybrid TKD-CPD yielded a rather low accuracy for small models, i.e., with small ranks, which are much worse than the CPD-based model with less or approximately the same number of parameters. However, the method quickly attained the original top-1 accuracy and even exceeded the top-5 accuracy when the $R_{CP} \geq 110$.

Comparison of accuracy vs. the number of FLOPs and parameters for the other layers is provided in Fig. 6. Each dot in the figure represents (accuracy, no. FLOPs) for each model. The dots for the same layers are connected by dashed lines. Once again, TKD-EPC achieved higher *top 1* and *top 5* accuracy with a smaller number of parameters and FLOPs, compared to CPD-EPC.

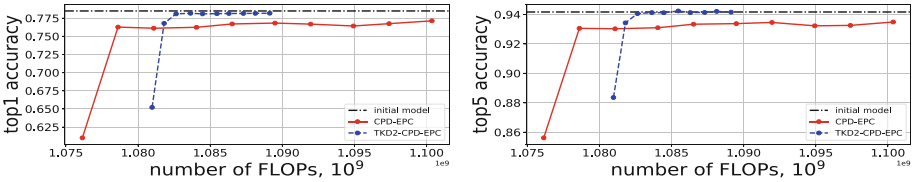


Fig. 5. Performance comparison (top1 accuracy – left, top5 accuracy – right) of CPD-EPC and TKD-CPD-EPC in compression of the layer `4.0.conv1` in the pre-trained ResNet-18 on ILSVRC-12 dataset. TKD-CPD-EPC shows better accuracy recovery with a relatively low number of FLOPs. Initial model has $\approx 1.11 \times 10^9$ FLOPs.

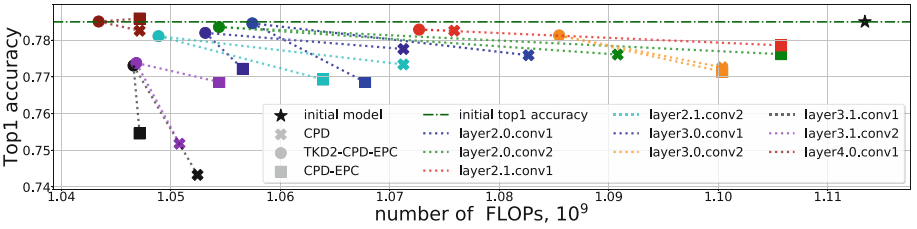


Fig. 6. Accuracy vs FLOPs for models obtained from ResNet-18 (CIFAR-100) via compression of one layer using standard CPD (cross), CPD-EPC (square), or TKD-CPD-EPC (circle) decomposition. Each color corresponds to one layer, which has been compressed using three different methods. For each layer, TKD-CPD-EPC outperforms other decompositions in terms of FLOPs, or accuracy, or both.

4.2 Full Model Compression

In this section, we demonstrate the efficiency of our proposed method in a full model compression of three well-known CNNs *VGG-16* [48], *ResNet-18*, *ResNet-50* [19] for the ILSVRC-12. We compressed all convolutional layers remaining fully-connected layers intact. The proposed scheme gives ($\times 1.10, \times 5.26$) for VGG-16, ($\times 3.82, \times 3.09$) for ResNet-18 and ($\times 2.51, \times 2.64$) for ResNet-50 reduction in the number of weights and FLOPs respectively. Table 1 shows that our approach yields a high compression ratio while having a moderate accuracy drop.

VGG [48]. We compared our method with other low-rank compression approaches on *VGG-16*. The Asym method [58] is one of the first successful methods on the whole VGG-16 network compression. This method exploits matrix decomposition, which is based on SVD and is able to reduce the number of flops by a factor of 5. Kim et al. [26] applied TKD with ranks selected by VBMF, and achieved a comparable compression ratio but with a smaller accuracy drop. As can be seen from the Table 1, our approach outperformed both Asym and TKD in terms of compression ratio and accuracy drop.

Table 1. Comparison of different model compression methods on ILSVRC-12 validation dataset. The baseline models are taken from Torchvision.

Model	Method	↓ FLOPs	Δ top-1	Δ top-5
VGG-16	Asym. [58]	≈ 5.00	–	–1.00
	TKD + VBMF [26]	4.93	–	–0.50
	Our (EPS[a] = 0.005)	**5.26**	**–0.92**	**–0.34**
ResNet-18	Channel gating NN [24]	1.61	–1.62	–1.03
	Discrimination-aware channel Pruning [59]	1.89	–2.29	–1.38
	FBS [13]	1.98	–2.54	–1.46
	MUSCO [14]	2.42	–0.47	–0.30
	Our (EPS[a] = 0.00325)	**3.09**	**–0.69**	**–0.15**
ResNet-50	**Our** (EPS[1] = 0.0028)	**2.64**	**–1.47**	**–0.71**

[a] EPS: accuracy drop threshold. Rank of the decomposition is chosen to maintain the drop in accuracy lower than EPS.

ResNet-18 [19]. This architecture is one of the lightest in the ResNet family, which gives relatively high accuracy. Most convolutional layers in ResNet-18 are with kernel size 3×3, making it a perfect candidate for the low-rank based methods for compression. We have compared our results with channel pruning methods [13,24,59] and iterative low-rank approximation method [14]. Among all the considered results, our approach has shown the best performance in terms of compression - accuracy drop trade-off.

ResNet-50 [19]. Compared to *ResNet-18*, *ResNet-50* is a deeper and heavier neural network, which is used as backbone in various modern applications, such as object detection and segmentation. A large number of 1×1 convolutions deteriorate performance of low-rank decomposition-based methods. There is not much relevant literature available

for compression of this type of ResNet. To the best of our knowledge, the results we obtained can be considered the first attempt to compress the entire *ResNet-50*.

Inference Time for Resnet-50. We briefly compare the inference time of Resnet-50 for the image classification task in Table 2. The measures were taken on 3 platforms: CPU server with Intel® Xeon® Silver 4114 CPU 2.20 GHz, NVIDIA GPU server with ® Tesla® V100 and Qualcomm mobile CPU ® Snapdragon™ 845. The batch size was choosen to yield small variance in inference measurements, e.g., 16 for the measures on CPU server, 128 for the GPU server and 1 for the mobile CPU.

Table 2. Inference time and acceleration for ResNet-50 on different platforms.

Platform	Model inference time	
	Original	Compressed
Intel® Xeon®Silver 4114 CPU 2.20 GHz	3.92 ± 0.02 s	2.84 ± 0.02 s
NVIDIA®Tesla®V100	102.3 ± 0.5 ms	89.5 ± 0.2 ms
Qualcomm®Snapdragon™845	221 ± 4 ms	171 ± 4 ms

5 Discussion and Conclusions

Replacing a large dense kernel in a convolutional or fully-connected layer by its low-rank approximation is equivalent to substituting the initial layer with multiple ones, which in total have fewer parameters. However, as far as we concerned, the sensitivity of the tensor-based models has never been considered before. The closest method proposes to add regularizer on the Frobenius norm of each weight to prevent over-fitting.

In this paper, we have shown a more direct way to control the tensor-based network's sensitivity. Through all the experiments for both ILSVRC-12 and CIFAR-100 dataset, we have demonstrated the validity and reliability of our proposed method for compression of CNNs, which includes a stable decomposition method with minimal sensitivity for both CPD and the hybrid TKD-CPD.

As we can see from recent deep learning literature [23,28,50], modern state-of-the-art architectures exploit the CP format when constructing blocks of consecutive layers, which consist of 1×1 convolution followed by depth-wise separable convolution. The intuition that stays behind the effectiveness of such representation is that first 1×1 convolution maps data to a higher-dimensional subspace, where the features are more separable, so we can apply separate convolutional kernels to preprocess them. Thus, representing weights in CP format using stable and efficient algorithms is the simplest and efficient way of constructing reduced convolutional kernels.

To the best of our knowledge, our paper is the first work solving a problem of building weights in the CP format that is stable and consistent with the fine-tuning procedure.

The ability to control sensitivity and stability of factorized weights might be crucial when approaching incremental learning task [3] or multi-modal tasks, where information fusion across different modalities is performed through shared weight factors.

Our proposed CPD-EPC method can allow more stable fine-tuning of architectures containing higher-order CP convolutional layers [27,28] that are potentially very promising due to the ability to propagate the input structure through the whole network. We leave the mentioned directions for further research.

Acknowledgements. The work of A.-H. Phan, A. Cichocki, I. Oseledets, J. Gusak, K. Sobolev, K. Sozykin and D. Ermilov was supported by the Ministry of Education and Science of the Russian Federation under Grant 14.756.31.0001. The results of this work were achieved during the cooperation project with Noah's Ark Lab, Huawei Technologies. The authors sincerely thank the Referees for very constructive comments which helped to improve the quality and presentation of the paper. The computing for this project was performed on the Zhores CDISE HPC cluster at Skoltech [56].

References

1. Astrid, M., Lee, S.: CP-decomposition with tensor power method for convolutional neural networks compression. In: 2017 IEEE International Conference on Big Data and Smart Computing, BigComp 2017, Jeju Island, South Korea, 13–16 February 2017, pp. 115–118. IEEE (2017). https://doi.org/10.1109/BIGCOMP.2017.7881725
2. Bulat, A., Kossaifi, J., Tzimiropoulos, G., Pantic, M.: Matrix and tensor decompositions for training binary neural networks. arXiv preprint arXiv:1904.07852 (2019)
3. Bulat, A., Kossaifi, J., Tzimiropoulos, G., Pantic, M.: Incremental multi-domain learning with network latent tensor factorization. In: AAAI (2020)
4. Chen, T., Lin, J., Lin, T., Han, S., Wang, C., Zhou, D.: Adaptive mixture of low-rank factorizations for compact neural modeling. In: CDNNRIA Workshop, NIPS (2018)
5. Cichocki, A., Lee, N., Oseledets, I., Phan, A.H., Zhao, Q., Mandic, D.P.: Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. Found. Trends® Mach. Learn. **9**(4–5), 249–429 (2016)
6. De Lathauwer, L.: Decompositions of a higher-order tensor in block terms – Part I and II. SIAM J. Matrix Anal. Appl. **30**(3), 1022–1066 (2008). http://publi-etis.ensea.fr/2008/De08e. special Issue on Tensor Decompositions and Applications
7. De Lathauwer, L., De Moor, B., Vandewalle, J.: On the best rank-1 and rank-(R1, R2,., RN) approximation of higher-order tensors. SIAM J. Matrix Anal. Appl. **21**, 1324–1342 (2000)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255 (2009)
9. Denil, M., Shakibi, B., Dinh, L., Ranzato, M., de Freitas, N.: Predicting parameters in deep learning. In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS 2013, pp. 2148–2156. Curran Associates Inc. (2013)
10. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in Neural Information Processing Systems, vol. 27, pp. 1269–1277. Curran Associates, Inc. (2014)
11. Espig, M., Hackbusch, W., Handschuh, S., Schneider, R.: Optimization problems in contracted tensor networks. Comput. Vis. Sci. **14**(6), 271–285 (2011)
12. Figurnov, M., Ibraimova, A., Vetrov, D.P., Kohli, P.: PerforatedCNNs: acceleration through elimination of redundant convolutions. In: Advances in Neural Information Processing Systems, pp. 947–955 (2016)
13. Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., Xu, C.Z.: Dynamic channel pruning: feature boosting and suppression. In: International Conference on Learning Representations (2019)

14. Gusak, J., et al.: Automated multi-stage compression of neural networks. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 2501–2508 (2019)
15. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 1135–1143 (2015)
16. Handschuh, S.: Numerical Methods in Tensor Networks. Ph.D. thesis, Faculty of Mathematics and Informatics, University Leipzig, Germany, Leipzig, Germany (2015)
17. Harshman, R.A.: Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis. In: UCLA Working Papers in Phonetics, vol. 16 pp. 1–84 (1970)
18. Harshman, R.A.: The problem and nature of degenerate solutions or decompositions of 3-way arrays. In: Tensor Decomposition Workshop, Palo Alto, CA (2004)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
20. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. In: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, pp. 2234–2240 (7 2018)
21. He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., Han, S.: AMC: AutoML for model compression and acceleration on mobile devices. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 815–832. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_48
22. Hillar, C.J., Lim, L.H.: Most tensor problems are NP-hard. J. ACM (JACM) **60**(6), 45 (2013)
23. Howard, A., et al.: Searching for MobileNetv3. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1314–1324 (2019)
24. Hua, W., Zhou, Y., De Sa, C.M., Zhang, Z., Suh, G.E.: Channel gating neural networks. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 32, pp. 1886–1896 (2019)
25. Khoromskij, B.: $O(d \log N)$-quantics approximation of N-d tensors in high-dimensional numerical modeling. Constr. Approximation **34**(2), 257–280 (2011)
26. Kim, Y., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. In: 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016, Conference Track Proceedings (2016). http://arxiv.org/abs/1511.06530
27. Kossaifi, J., Bulat, A., Tzimiropoulos, G., Pantic, M.: T-net: parametrizing fully convolutional nets with a single high-order tensor. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 7822–7831 (2019)
28. Kossaifi, J., Toisoul, A., Bulat, A., Panagakis, Y., Hospedales, T.M., Pantic, M.: Factorized higher-order CNNs with an application to spatio-temporal emotion estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6060–6069 (2020)
29. Krijnen, W., Dijkstra, T., Stegeman, A.: On the non-existence of optimal solutions and the occurrence of "degeneracy" in the CANDECOMP/PARAFAC model. Psychometrika **73**, 431–439 (2008)
30. Krizhevsky, A.: Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, Toronto (2009)
31. Landsberg, J.M.: Tensors: Geometry and Applications, vol. 128. American Mathematical Society, Providence (2012)
32. Lebedev, V.: Algorithms for speeding up convolutional neural networks. Ph.D. thesis, Skoltech, Russia (2018). https://www.skoltech.ru/app/data/uploads/2018/10/Thesis-Final.pdf

33. Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., Lempitsky, V.: Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In: International Conference on Learning Representations (2015)
34. Lim, L.H., Comon, P.: Nonnegative approximations of nonnegative tensors. J. Chemom. **23**(7–8), 432–441 (2009)
35. Mitchell, B.C., Burdick, D.S.: Slowly converging PARAFAC sequences: Swamps and two-factor degeneracies. J. Chemom. **8**, 155–168 (1994)
36. Molchanov, D., Ashukha, A., Vetrov, D.: Variational dropout sparsifies deep neural networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70, pp. 2498–2507 (2017). JMLR.org
37. Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.: Tensorizing neural networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS 2015, pp. 442–450. MIT Press, Cambridge (2015)
38. Oseledets, I., Tyrtyshnikov, E.: Breaking the curse of dimensionality, or how to use SVD in many dimensions. SIAM J. Sci. Comput. **31**(5), 3744–3759 (2009)
39. Paatero, P.: Construction and analysis of degenerate PARAFAC models. J. Chemometrics **14**(3), 285–299 (2000)
40. Phan, A.H., Cichocki, A., Uschmajew, A., Tichavský, P., Luta, G., Mandic, D.: Tensor networks for latent variable analysis: novel algorithms for tensor train approximation. IEEE Trans. Neural Network Learn. Syst. (2020). https://doi.org/10.1109/TNNLS.2019.2956926
41. Phan, A.H., Tichavský, P., Cichocki, A.: Tensor deflation for CANDECOMP/PARAFAC. Part 1: alternating subspace update algorithm. IEEE Trans. Sig. Process. **63**(12), 5924–5938 (2015)
42. Phan, A.H., Tichavský, P., Cichocki, A.: Error preserving correction: a method for CP decomposition at a target error bound. IEEE Trans. Signal Process. **67**(5), 1175–1190 (2019)
43. Phan, A.H., Yamagishi, M., Mandic, D., Cichocki, A.: Quadratic programming over ellipsoids with applications to constrained linear regression and tensor decomposition. Neural Comput. Appl. (2020). https://doi.org/10.1007/s00521-019-04191-z
44. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 525–542. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46493-0_32
45. Rayens, W., Mitchell, B.: Two-factor degeneracies and a stabilization of PARAFAC. Chemometr. Intell. Lab. Syst. **38**(2), 173–181 (1997)
46. Rigamonti, R., Sironi, A., Lepetit, V., Fua, P.: Learning separable filters. In: Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2754–2761. CVPR '13, IEEE Computer Society, Washington, DC, USA (2013)
47. de Silva, V., Lim, L.H.: Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM J. Matrix Anal. Appl. **30**, 1084–1127 (2008)
48. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: 3rd International Conference on Learning Representations, ICLR (2015)
49. Stegeman, A., Comon, P.: Subtracting a best rank-1 approximation may increase tensor rank. Linear Algebra Appl. **433**(7), 1276–1300 (2010)
50. Tan, M., Le, Q.V.: EfficientNet: rethinking model scaling for convolutional neural networks. In: ICML (2019)
51. Tichavský, P., Phan, A.H., Cichocki, A.: Sensitivity in tensor decomposition. IEEE Signal Process. Lett. **26**(11), 1653–1657 (2019)
52. Tucker, L.R.: Implications of factor analysis of three-way matrices for measurement of change. Probl. Measuring Change **15**, 122–137 (1963)

53. Vasilescu, M.A.O., Terzopoulos, D.: Multilinear subspace analysis of image ensembles. In: 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), Madison, WI, USA, 16–22 June 2003, pp. 93–99. IEEE Computer Society (2003). https://doi.org/10.1109/CVPR.2003.1211457
54. Vervliet, N., Debals, O., Sorber, L., Barel, M.V., Lathauwer, L.D.: Tensorlab 3.0, March 2016. http://www.tensorlab.net
55. Wang, D., Zhao, G., Li, G., Deng, L., Wu, Y.: Lossless compression for 3DCNNs based on tensor train decomposition. CoRR abs/1912.03647 (2019). http://arxiv.org/abs/1912.03647
56. Zacharov, I., et al.: Zhores – petaflops supercomputer for data-driven modeling, machine learning and artificial intelligence installed in Skolkovo Institute of Science and Technology. Open Eng. **9**(1) (2019)
57. Zhang, T., Golub, G.H.: Rank-one approximation to high order tensors. SIAM J. Matrix Anal. Appl. **23**(2), 534–550 (2001). https://doi.org/10.1137/S0895479899352045
58. Zhang, X., Zou, J., He, K., Sun, J.: Accelerating very deep convolutional networks for classification and detection. IEEE Trans. Pattern Anal. Mach. Intell. **38**(10), 1943–1955 (2016)
59. Zhuang, Z., et al.: Discrimination-aware channel pruning for deep neural networks. In: Advances in Neural Information Processing Systems, pp. 883–894 (2018)