HODEC: Towards Efficient High-Order DEcomposed Convolutional Neural Networks

Miao Yin, Yang Sui, Wanzhao Yang, Xiao Zang, Yu Gong and Bo Yuan
Department of Electrical and Computer Engineering, Rutgers University
{miao.yin, yang.sui, wanzhao.yang, xiao.zang, yu.gong}@rutgers.edu, bo.yuan@soe.rutgers.edu

Abstract

High-order decomposition is a widely used model compression approach towards compact convolutional neural networks (CNNs). However, many of the existing solutions, though can efficiently reduce CNN model sizes, are very difficult to bring considerable saving for computational costs, especially when the compression ratio is not huge, thereby causing the severe computation inefficiency problem. To overcome this challenge, in this paper we propose efficient High-Order DEcomposed Convolution (HODEC). By performing systematic explorations on the underlying reason and mitigation strategy for the computation inefficiency, we develop a new decomposition and computation-efficient execution scheme, enabling simultaneous reductions in computational and storage costs.

To demonstrate the effectiveness of HODEC, we perform empirical evaluations for various CNN models on different datasets. HODEC shows consistently outstanding compression and acceleration performance. For ResNet-56 on CIFAR-10 dataset, HODEC brings 67% fewer model parameters and 62% fewer FLOPs with 1.17% accuracy increase than the baseline. For ResNet-50 on ImageNet dataset, HODEC achieves 63% FLOPs reduction with 0.31% accuracy increase than the uncompressed model.

1. Introduction

Deep neural networks (DNNs) have been widely adopted in various fundamental and downstream computer vision tasks, such as image classification [11], object detection [30], action recognition [44], super-resolution [15], seismic signal analysis [28] and channel decoding [18]. Considering the inherent high computational and storage costs of modern large-scale neural networks, in recent years many *model compression* approaches have been proposed and developed. For instance, by exploring and removing the model redundancy at the neuron level and bit level, respectively, pruning [1, 9, 10, 13, 16, 21, 22, 35, 43] and quantization [4,14,23,29,33] can efficiently reduce DNN sizes while preserving high task accuracy.

DNN Compression via Tensor Decomposition. Alternatively, the redundancy of a DNN model can also exhibit and be further reduced at the network topology level. Motivated by this philosophy and observation, *high-order tensor decomposition*, a technique that explores multi-dimensional low-rankness of DNN models, has been proposed and studied in the recent years. By decomposing the original large-scale weight matrices and/or weight tensors to a set of small tensor cores, tensor decomposition-based compression can bring significant reduction in neural network sizes. Notably, as reported in many prior works [2, 3, 24, 26, 37–42], the state-of-the-art tensor decomposition approaches, e.g., tensor train (TT) and its variant tensor ring (TR) [25, 45], can achieve ultra-high compression ratio (more than 1,000×) for various recurrent neural networks (RNNs). Evidently, such outstanding compression performance is very attractive for designing and producing compact DNN models.

Limitations on Computation Efficiency. Despite their very promising potentials in model size reduction, the advanced TT and TR decomposition approaches suffer severe *computation inefficiency* problem, especially when aiming to accelerate the computation-intensive convolutional neural networks (CNNs). To be specific, when a CNN model is decomposed to the TT or TR format, though its total number of weight parameters is indeed reduced significantly, the corresponding computational cost (a.k.a., floating point operations per second (FLOPs)) is not saved accordingly, and sometimes it even increases if only moderate compression ratio is allowed to preserve accuracy. For instance, when using the classical TT decomposition [7,24] to factorize the convolutional layer, with 2.02× compression ratio setting a TT-format CNN model will even suffer around five times higher computational cost than the original uncompressed network. Consequently, consider 1) CNNs are very fundamental neural network models that have been very popularly deployed in practice; and 2) tremendous computer vision tasks, e.g., object detection and motion prediction, are very time-sensitive applications and highly demand real-

time processing and acceleration; such computation inefficiency problem of the tensor decomposition approaches, if cannot be addressed properly, will severely hinder their further widespread deployment in many practical applications.

Technical Preview and Contributions. To overcome this severe computation inefficiency challenge for the advanced tensor decomposed CNNs, in this paper we propose to study and develop efficient High-Order tensor DEcomposed Convolution (HODEC). By performing systematic analysis and exploration on the underlying reason and mitigation strategy for the inefficiency of the classical TT-format convolution, we further propose and develop the new decomposition and execution scheme towards computation-efficient TT-format convolutional layer [1]. As a type of plug-in component, this new tensor decomposed layer can be directly used in the CNN models and bring simultaneous reduction in computational and storage costs. Overall, the contributions of this paper are summarized as follows:

- We systematically study and analyze the underlying reason for the computation inefficiency of the conventional TT-format convolution. We then identify and develop a series of strategies that can further mitigate this problem and reduce the computational costs.
- Based upon the obtained understanding from our analysis, we further propose and develop the new decomposition and execution scheme for the computation-efficient TT-format convolutional layer, which enjoys very significant reduction in computational costs and memory consumption.
- We perform empirical evaluations for various CNN models on different datasets, and the experimental results show that HODEC can consistently outperform the existing pruning and low-rank matrix/tensor decomposition approaches. For ResNet-56 on CIFAR-10 dataset, using HODEC can bring 67% fewer model parameters and 62% fewer FLOPs with 1.17% accuracy increase than the baseline. For ResNet-50 on ImageNet dataset, HODEC can achieve 63% FLOPs reduction with 0.31% accuracy increase than the uncompressed model.

2. Background and Motivation

2.1. Preliminaries

Notation. Throughout this paper the vectors, matrices and tensors are denoted by boldface lower-case letters, boldface capital letters and boldface calligraphic script letters, respectively, e.g., $\boldsymbol{a}$, $\boldsymbol{A}$ and $\boldsymbol{\mathcal{A}}$. In addition, we use non-boldface letters with indices in parentheses to represent the entry. For instance, $\mathcal{A}(i_1, \cdots, i_d)$ denotes the $(i_1, \cdots, i_d)$-th entry of a d-order tensor $\boldsymbol{\mathcal{A}}$.

Tensor Contraction. Consider two tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1 \times N_2 \times M}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{M \times N_3 \times N_4}$. When the third dimension of $\boldsymbol{\mathcal{A}}$ matches the first dimension of $\boldsymbol{\mathcal{B}}$, the contraction $\boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{C}} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times N_4}$ can be calculated as

$$\mathcal{C}(n_1, n_2, n_3, n_4) = \sum_{m}^{M} \mathcal{A}(n_1, n_2, m)\mathcal{B}(m, n_3, n_4). \quad (1)$$

Tensor Train Decomposition. Given a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N_1 \times \cdots \times N_d}$, it can be represented in tensor train (TT) [25] format if each element is computed as

$$\mathcal{A}(i_1, i_d, \cdots, i_d) = \sum_{r_0, \cdots, r_d}^{R_0, \cdots, R_d} \mathcal{G}_1(r_0, i_1, r_1)\mathcal{G}_2(r_1, i_2, r_2) \cdots \mathcal{G}_d(r_{d-1}, i_d, r_d), \quad (2)$$

where $\{\boldsymbol{\mathcal{G}}_j \in \mathbb{R}^{R_{j-1} \times N_j \times R_j}\}_{j=1}^{d}$ are called *TT-cores*, and $R_0, R_1, \cdots, R_d$ are *TT-ranks*. Notice that here R_0 and R_d are always equal to 1. Notably, with such TT-format the original explosive storage complexity is reduced to linear complexity determined by TT-ranks.

TT-format Matrix-Vector Multiplication. Consider a matrix $\boldsymbol{W} \in \mathbb{R}^{I \times O}$ and input vector $\boldsymbol{x} \in \mathbb{R}^{I}$. When decomposing this matrix and transforming the corresponding matrix-vector multiplication to TT format, the original $\boldsymbol{W}$ and $\boldsymbol{x}$ are first reshaped and transposed as a d-order weight tensor $\bar{\boldsymbol{\mathcal{W}}} \in \mathbb{R}^{I_1 O_1 \times \cdots \times I_d O_d}$ and a d-order input tensor $\bar{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{I_1 \times \cdots \times I_d}$, respectively, where $\prod_{j=1}^{d} I_j = I, \prod_{j=1}^{d} O_j = O$. Then the tensorized $\bar{\boldsymbol{\mathcal{W}}}$ can be decomposed to TT format with Eq. 2, and the corresponding TT-format matrix-vector multiplication is performed as [24]:

$$\bar{\mathcal{Y}}(o_1, \cdots, o_d) = \sum_{r_0, \cdots, r_d} \sum_{i_1, \cdots, i_d} \mathcal{G}_1(r_0, i_1, o_1, r_1) \cdots \mathcal{G}_d(r_{d-1}, i_d, o_d, r_d)\bar{\mathcal{X}}(i_1, \cdots, i_d), \quad (3)$$

where $\bar{\boldsymbol{\mathcal{Y}}} \in \mathbb{R}^{O_1 \times \cdots \times O_d}$ is the output tensor, which can be further reshaped to form the desired output vector $\boldsymbol{y} \in \mathbb{R}^{O}$. Notice $\{\boldsymbol{\mathcal{G}}_j \in \mathbb{R}^{R_{j-1} \times I_j \times O_j \times R_j}\}_{j=1}^{d}$ are 4-order TT-cores.

TT-format Convolutional Layer (TT-CONV). As indicated in [7], the execution on a classical TT-CONV layer is essentially built on TT-format matrix-vector multiplication described in Eq. 3. To be specific, consider a convolutional layer with a 4-order weight tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{K \times K \times I \times O}$; when it is convolved with a 3-order input tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{M \times N \times I}$, the original output tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{M' \times N' \times O} = \boldsymbol{\mathcal{X}} * \boldsymbol{\mathcal{W}}$ is explicitly calculated as:

$$\mathcal{Y}(m', n', o) = \sum_{k_1=1}^{K} \sum_{k_1=1}^{K} \sum_{i=1}^{I} \mathcal{X}(m, n, i)\mathcal{W}(k_1, k_2, i, o),$$
$$\text{with} \quad m' = m - k_1 + 1, n' = n - k_2 + 1,$$
$$\text{and} \quad M' = M - K + 1, N' = N - K + 1, \quad (4)$$

[1] This paper only presents the efficient TT-format convolution because of page limit. Other variants of TT, such as computation-efficient TR-format convolution, can be derived in a similar way.

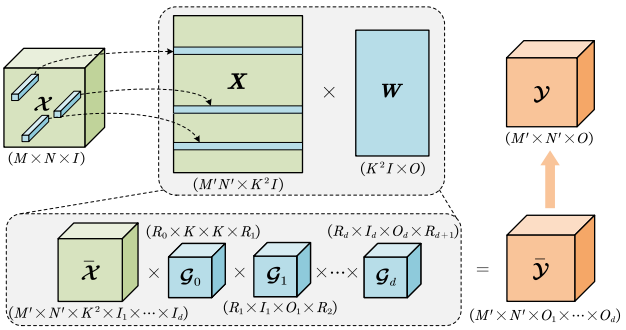

Figure 1. Computing scheme of the classical TT-CONV layer. Here rectangles represent matrices and cubes denote arbitrary high-order tensor (not limited to 3-order tensor).

where $*$ denotes the convolution operation.

When compressing the convolutional layer via TT decomposition, a new matrix $\boldsymbol{X} \in \mathbb{R}^{M'N' \times K^2 I}$ is first constructed by concatenating all the flattened patches in the original input tensor $\boldsymbol{\mathcal{X}}$ as:

$$\begin{aligned} \mathcal{X}(m, n, i) = X(m - k_1 + 1 + M'(n - k_2), \\ k_1 + K(k_2 - 1) + K^2(i - 1)), \end{aligned} \tag{5}$$

and meanwhile a new weight matrix $\boldsymbol{W} \in \mathbb{R}^{K^2 I \times O}$ is obtained by reshaping the original weight tensor $\boldsymbol{\mathcal{W}}$ as:

$$\mathcal{W}(k_1, k_2, i, o) = W(k_1 + K(k_2 - 1) + K^2(i - 1), o). \tag{6}$$

Then, the original convolution operation described in Eq. 4 is equivalent to the matrix multiplication $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}$. With further reshaping and transposing $\boldsymbol{X}$ and $\boldsymbol{W}$ to tensor $\bar{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{M' \times N' \times K^2 \times I_1 \times \cdots \times I_d}$ and $\bar{\boldsymbol{\mathcal{W}}} \in \mathbb{R}^{K^2 \times I_1 O_1 \times \cdots \times I_d O_d}$, the computation on the TT-CONV layer can be transformed to the stack of TT-format matrix-vector multiplications as:

$$\begin{aligned} &\bar{\mathcal{Y}}(m', n', o_1, \cdots, o_d) = \\ &\sum_{k_1=1}^{K} \sum_{k_2=1}^{K} \sum_{r_0, \cdots, r_{d+1}} \sum_{i_1, \cdots, i_d} \mathcal{G}_0(r_0, k_1, k_2, r_1) \\ &\quad \mathcal{G}_1(r_1, i_1, o_1, r_2) \cdots \mathcal{G}_d(r_d, i_d, o_d, r_{d+1}) \\ &\quad \bar{\mathcal{X}}(m, n, k_1, k_2, i_1, \cdots, i_d). \end{aligned} \tag{7}$$

Furthermore, according to Eq. 1, the above element-wise format computation can be simplified to tensor contraction:

$$\bar{\boldsymbol{\mathcal{Y}}} = \bar{\boldsymbol{\mathcal{X}}} \times \boldsymbol{\mathcal{G}}_0 \times \boldsymbol{\mathcal{G}}_1 \times \cdots \times \boldsymbol{\mathcal{G}}_d. \tag{8}$$

2.2. Challenge on Computation Inefficiency

To date, all of the existing TT decomposed CNNs adopt the computation scheme described in Eq. 8 to perform model inference. Unfortunately, though the model size can be indeed reduced via tensor decomposition, as we will analyze later, the computational cost is not correspondingly reduced. To simplify the notation during the analysis, we first assume I and O are evenly factorized, i.e., $I_1 = I_2 = \cdots = I_d = I_m$ and $O_1 = O_2 = \cdots = O_d = O_m$.

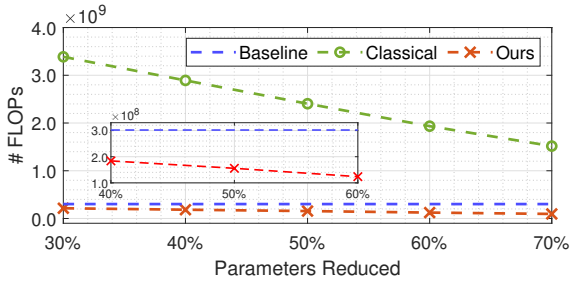

Figure 2. FLOPs vs Parameter reduction for `layer3.0.conv1` in ResNet-18 when using conventional TT-CONV and our proposed HODEC. It is seen that conventional TT-CONV ("Classical") causes even higher FLOPs consumption than the uncompressed one ("Baseline"); while the proposed HODEC ("Ours") can bring considerable computational saving (**see zoom-in box**).

Meanwhile without loss of generality, we also assume all the elements of TT-ranks are equal and denote them as $R_1 = R_2 = \cdots = R_{d-1} = R$.

As described in Eq. 8, the first step of the computation scheme of a TT-CONV layer is the contraction between $\bar{\boldsymbol{\mathcal{X}}}$ and $\boldsymbol{\mathcal{G}}_0$, where the consumed number of FLOPs is $I_m^d K^2 R M' N'$. Starting from the 2nd computation step, i.e., the contraction between the intermediate result and $\boldsymbol{\mathcal{G}}_j$ where $j > 0$, the FLOPs count grows by $I_m^{d-j+1} O_m^j R^2 M' N'$ after each step since the TT-ranks for $\boldsymbol{\mathcal{G}}_j$ are not 1 any more. Consequently, the overall computational complexity of a TT-CONV layer is $\mathcal{O}(dR \max(RI_m, K^2) \max(I, O) M' N')$. Compared with the uncompressed CONV layer with $\mathcal{O}(IOK^2 M' N')$ computational complexity, the FLOPs consumption after using TT decomposition can be even higher. For instance, consider a TT-format convolutional layer with $I = 16 = 4 \times 4, O = 32 = 8 \times 4, K = 3, R = 8$. Although such decomposition setting can bring $2\times$ reduction in weight parameters, the number of the required FLOPs increases to 8M; while the corresponding original uncompressed layer only consumes 3.6M FLOPs. In other words, this example decomposed TT-CONV layer exhibit $2\times$ reduction in model size with penalty of more than $2\times$ increase in FLOPs, thereby causing severe computation inefficiency.

3. The Proposed Efficient TT-CONV Solution

Aiming to address and overcome this computation inefficiency challenge of the existing TT-format convolutional layer, in this section we propose to perform systematic analysis on the underlying reason of such inefficiency, and then explore the important design knobs to further develop the efficient TT-CONV solution. To be specific, we will analyze and answer the following five important questions.

Question #1: *Why does the existing TT-CONV execution scheme cause this computation inefficiency?*

Analysis. As illustrated in Fig. 1, the existing computation on a TT-CONV layer is performed as the consecu-

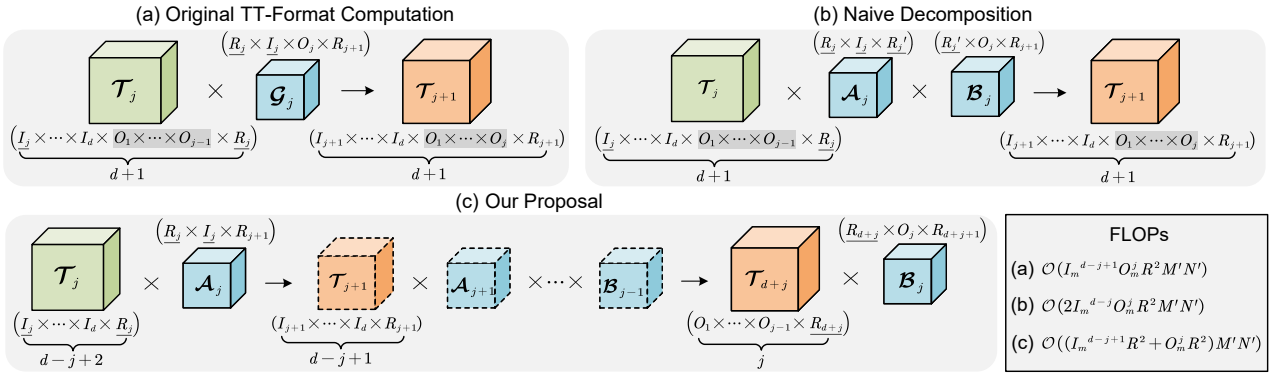

Figure 3. Different TT-CONV schemes. (a) Original TT-CONV. (b) Decompose each $\mathcal{G}_j$ to $\mathcal{A}_j$ and $\mathcal{B}_j$. (c) Split $\{\mathcal{A}_j\}$ and $\{\mathcal{B}_j\}$. M and N are omit for simple notation. Notice here the FLOPs is counted for the computation involved with one $\mathcal{G}_j$ or one pair of $\mathcal{A}_j$ and $\mathcal{B}_j$.

tive tensor contractions between a $(d+3)$-order input tensor $\bar{\mathcal{X}} \in \mathbb{R}^{M'\times N'\times K^2\times I_1\times\cdots\times I_d}$ with $(d+1)$ 4-order tensor cores $\mathcal{G}_j \in \mathbb{R}^{R_j\times I_j\times O_j\times R_{j+1}}$. Notice that for most $\mathcal{G}_j$ with $j > 0$, they always contain two component dimensions as I_j and O_j. Therefore, according to the computation scheme described in Eq. 1 and Eq. 8, after each step of tensor contraction, two component dimensions of the input intermediate result, as I_j and R_{j-1}, are contracted and eliminated, and two new component dimensions, as O_j and R_j, will appear in the shape of output intermediate result. Consequently, the intermediate results that are involved with each step of the TT-CONV computation are always $(d+1)$-order tensors, thereby causing high computational costs. To be specific, as shown in Fig. 3 (a), at the j-th step, the input intermediate result $\mathcal{T}_j \in \mathbb{R}^{M'\times N'\times I_j\times\cdots\times I_d\times O_1\times\cdots\times O_{j-1}\times R_j}$ is a $(d+1)$-order tensor that will be contracted with tensor core $\mathcal{G}_j$; while the corresponding output intermediate result is still a $(d+1)$-order tensor $\mathcal{T}_{j+1} \in \mathbb{R}^{M'\times N'\times I_{j+1}\times\cdots\times I_d\times O_1\times\cdots\times O_j\times R_{j+1}}$. Consider $I = \prod_{j=1}^{d} I_j$ and $O = \prod_{j=1}^{d} O_j$, this *order-remaining* phenomenon means that each step of TT-CONV computation is always involved with an input with $\mathcal{O}(I_m^{d-j+1}O_m^{j-1}RM'N')$ complexity and and output with $\mathcal{O}(I_m^{d-j}O_m^{j}RM'N')$ complexity. Evidently, the corresponding incurred computational cost is very huge.

Question #2: *How should we improve the computation efficiency of TT-CONV layer?*

Analysis. As previously analyzed, the computation inefficiency of TT-CONV layer is caused by the phenomenon that the intermediate are always kept as $(d+1)$-order large-scale tensors. From the perspective of low-rank analysis, the underlying reason for this order-remaining phenomenon is that the tensorized $\bar{\mathcal{W}}$ is not thoroughly decomposed. To be specific, as illustrated in Fig. 1, the conventional TT decomposition for $\bar{\mathcal{W}} \in \mathbb{R}^{K^2\times I_1O_1\times\cdots\times I_dO_d}$ only captures the correlation between I_jO_j dimension and $I_{j+1}O_{j+1}$ dimension; while the potential correlation between I_j dimension and O_j dimension is not explored yet but only simply assumed to exhibit full-rankness. Consequently, for each $\mathcal{G}_j$ with $j > 0$ its two component dimensions I_j and O_j are always bundled, and thereby causing the order-remaining problem of the intermediate results $\mathcal{T}_j$.

Our Proposal. Based on the above analysis, we propose to further decompose the TT-cores $\mathcal{G}_j$ to explore the potential low-rankness correlation between the I_j and O_j dimensions. More specifically, as illustrated in Fig. 3(b), each TT-core $\mathcal{G}_j \in \mathbb{R}^{R_j\times I_j\times O_j\times R_{j+1}}$ can be further factorized to two new tensor cores $\mathcal{A}_j \in \mathbb{R}^{R_j\times I_j\times R'_j}$ and $\mathcal{B}_j \in \mathbb{R}^{R'_j\times O_j\times R_{j+1}}$. As revealed by its explicit decomposition format, this proposed additional factorization can be intuitively interpreted as a matrix decomposition-like operation, which is essentially designed to explore the low-rankness correlation between I_j and O_j dimensions and decouple the computations involved with I_j and O_j dimensions during the consecutive tensor contractions.

Question #3: *What is the suitable tensor contraction scheme for $\{\mathcal{A}_j\}$ and $\{\mathcal{B}_j\}$?*

Analysis. As discussed above, the motivation of decomposing the original TT-cores $\{\mathcal{G}_j\}_{j=1}^{d}$ to two new tensor core sets $\{\mathcal{A}_j\}$ and $\{\mathcal{B}_j\}$ is to avoid the order-remaining problem and improve computation efficiency. However, only simply replacing each $\mathcal{G}_j$ by its corresponding pair of $\mathcal{A}_j$ and $\mathcal{B}_j$ cannot bring the expected computational reduction. To be specific, as illustrated in Fig. 3(b), because such direct replacement will make the tensor contraction involved with $\mathcal{A}_j \in \mathbb{R}^{R_j\times I_j\times R'_j}$ and the one involved with $\mathcal{B}_j \in \mathbb{R}^{R'_j\times O_j\times R_{j+1}}$ are performed consecutively, the intermediate result after such two contractions will drop the I_j and R_j dimensions but obtain the O_j and R_{j+1} dimensions, and hence it will still suffer the order-remaining problem and cannot enjoy the computational saving.

Our Proposal. Based on such observation and aiming to truly enable the reduction in computational cost, we propose to re-arrange the sequence of the tensor contraction for $\{\mathcal{A}_j\}$ and $\{\mathcal{B}_j\}$ and split them to two separate groups. Fig. 3(c) illustrates the key idea of our proposed scheme.

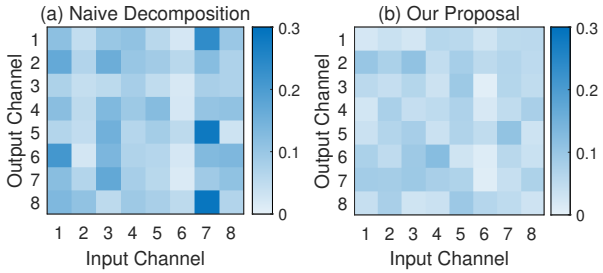

Figure 4. Comparison on approximation error between using two-stage decomposition and one-stage decomposition for a convolutional layer in ResNet-32. Here the approximation error for each component kernel filter of this layer is visualized. The same compression ratio of $2.1\times$ is set for both cases.

Here the overall execution scheme of the proposed TT-CONV layer consists of three phase, namely *Contract-in*, *Core Convolution* and *Contract-out*. To be specific, all the tensor cores $\{\boldsymbol{\mathcal{A}}_j\}$ are only involved with the consecutive tensor contraction in the Contract-in phase as:

$$\boldsymbol{\mathcal{Z}}_1 = \bar{\boldsymbol{\mathcal{X}}} \times \boldsymbol{\mathcal{A}}_1 \times \cdots \times \boldsymbol{\mathcal{A}}_d, \tag{9}$$

where $\boldsymbol{\mathcal{Z}}_1 \in \mathbb{R}^{M' \times N' \times R_d}$ is the output of this phase. And meanwhile, all the tensor cores $\{\boldsymbol{\mathcal{B}}_j\}$ only participate in the tensor contraction operation in the Contract-out phase as below:

$$\bar{\boldsymbol{\mathcal{Y}}} = \boldsymbol{\mathcal{Z}}_2 \times \boldsymbol{\mathcal{B}}_1 \times \cdots \times \boldsymbol{\mathcal{B}}_d, \tag{10}$$

where $\bar{\boldsymbol{\mathcal{Y}}} \in \mathbb{R}^{M' \times N' \times O_1 \times \cdots \times O_d}$ is the reshaped version of the final desired output tensor $\boldsymbol{\mathcal{Y}}$, and $\boldsymbol{\mathcal{Z}}_2$ is the output of the Core Convolution phase that will be discussed later.

As illustrated in Fig. 3, this proposed re-scheduled contraction scheme can bring practical computational cost reduction. This is because by splitting $\{\boldsymbol{\mathcal{A}}_j\}$ and $\{\boldsymbol{\mathcal{B}}_j\}$ to two different phases, I_j dimension will be continuously contracted and eliminated during the Contract-in phase, and hence the tensor orders of the intermediate results $\boldsymbol{\mathcal{T}}_j$ in this phase are gradually reduced. Meanwhile, it will also lead to very low-tensor order for the first input intermediate results in the Contract-out phase since no I_j dimension will appear in its shape. Consequently, though during the Contract-out phase the tensor order of the intermediate results will gradually increase, most of them are still very small. Overall, we now only have two intermediate results (the one before the contraction with $\boldsymbol{\mathcal{A}}_1$ and the one after the contraction with $\boldsymbol{\mathcal{B}}_d$) that exhibit the maximum tensor order $(d+1)$; while all the other intermediate results are ensured to have lower tensor order than their counterparts calculated in the classical TT-CONV scheme, thereby leading to significant saving in computational costs.

Question #4: *What is the proper way to realize the proposed further decomposition from $\boldsymbol{\mathcal{G}}_j$ to $\boldsymbol{\mathcal{A}}_j$ and $\boldsymbol{\mathcal{B}}_j$?*

Analysis. As mentioned above, each 4-order tensor core $\boldsymbol{\mathcal{G}}_j$ with $j > 0$ can be further decomposed into two 3-order tensor components $\boldsymbol{\mathcal{A}}_j$ and $\boldsymbol{\mathcal{B}}_j$ to reduce the computational costs of TT-CONV layer. However, from the perspective of model compression, such two-stage decomposition is not an ideal solution. This is because as a type of low-rank approximation approach, decomposing $\bar{\boldsymbol{\mathcal{W}}}$ into $\{\boldsymbol{\mathcal{G}}_j\}$ already introduces inevitable approximation error for the original weight tensor $\bar{\boldsymbol{\mathcal{W}}}$. Simply performing additional decomposition on the tensor cores $\{\boldsymbol{\mathcal{G}}_j\}$ will further aggregate the approximation effect, cause considerable information loss and finally affect the accuracy performance of the entire compressed model. In other words, even though these two decomposition stages may achieve the optimal approximation performance for their individual stage, the combinations of the two locally optimal approaches do not necessarily bring the globally optimal solution. In particular, considering the complicated relationship and correlations among the multi-dimensional information of the original weight tensor $\widetilde{\boldsymbol{\mathcal{W}}}$, it is very challenging for the straightforward multi-stage decomposition to identify and capture the multi-dimensional low-rankness with satisfied performance.

Our Proposal. In order to maximally preserve the important information in the original uncompressed weight tensor, we propose to perform one-stage decomposition. To be specific, we aim to directly decompose $\bar{\boldsymbol{\mathcal{W}}}$ to TT-cores $\{\boldsymbol{\mathcal{A}}_j \in \mathbb{R}^{R_{j-1} \times I_j \times R_j}\}_{j=1}^d$ and $\{\boldsymbol{\mathcal{B}}_j \in \mathbb{R}^{R_{d+j} \times O_j \times R_{d+j+1}}\}_{j=1}^d$ without the involvement of $\{\boldsymbol{\mathcal{G}}_j\}_{j=1}^d$. We believe such single-stage end-to-end decomposition strategy is more suitable, since it can leverage the complete and global information of the original weight tensor, thereby minimizing the information loss and reducing approximation error. Fig. 4 verifies our hypothesis on an example weight tensor of one layer of ResNet-32. It is seen that to obtain the same size tensor cores $\{\boldsymbol{\mathcal{A}}_j\}$ and $\{\boldsymbol{\mathcal{B}}_j\}$, single-stage decomposition brings much lower approximation error than its two-stage counterpart.

Question #5: *How should $\boldsymbol{\mathcal{G}}_0$ be properly handled in the new contraction scheme?*

Analysis. As described in the analysis for **Question #3**, our proposed new decomposition and contraction scheme replaces the original $\{\boldsymbol{\mathcal{G}}_j\}_{j=1}^d$ by two new groups of new tensor cores $\{\boldsymbol{\mathcal{A}}_j\}_{j=1}^d$ and $\{\boldsymbol{\mathcal{B}}_j\}_{j=1}^d$ with different contraction phases. In such scenario, the rest tensor core $\boldsymbol{\mathcal{G}}_0 \in \mathbb{R}^{R_0 \times K \times K \times R_1}$ can still be kept as the first tensor core to be contracted with the input tensor $\widetilde{\boldsymbol{\mathcal{X}}}$. However, such strategy suffers a same drawback that the classical TT-CONV execution scheme already has – the original input tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{M \times N \times I}$ has to be enlarged to $\bar{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{M' \times N' \times K^2 \times I_1 \times \cdots \times I_d}$ via data-repeating operation (see Eq. 5). Evidently, this operation causes huge memory overhead since it approximately causes K^2 times increase in input tensor size; however, such cost is inevitable for the classical TT-CONV computation because of the need of ensuring mathematical equivalence for convolution function.

Our Proposal. Fortunately, when $\{\boldsymbol{\mathcal{A}}_j\}_{j=1}^d$ and $\{\boldsymbol{\mathcal{B}}_j\}_{j=1}^d$

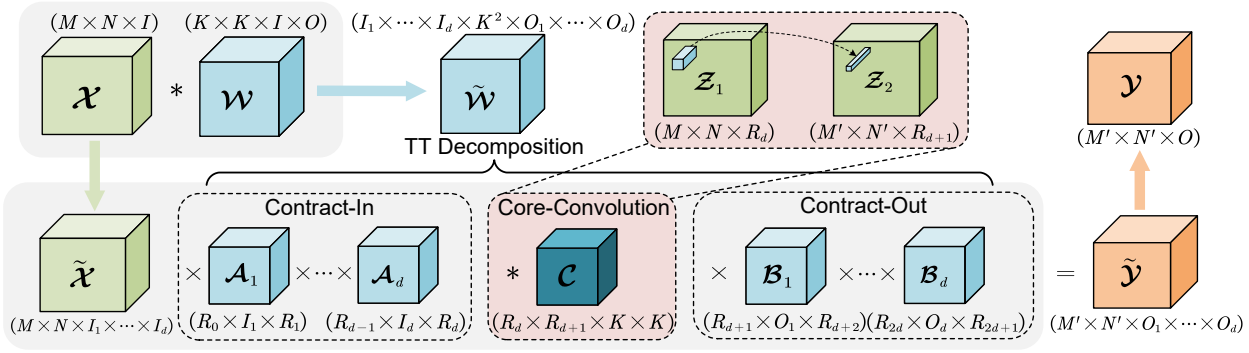

Figure 5. The overall decomposition format and execution scheme of the proposed computation-efficient TT-CONV (HODEC).

Algorithm 1 The overall computation scheme of HODEC

Input: TT-cores $\{\mathcal{A}_j\}_{j=1}^d$, $\{\mathcal{B}_j\}_{j=1}^d$ and $\mathcal{C}$, input tensor $\mathcal{X}$, factorized input channels $[I_1, \cdots, I_d]$;
Output: Output tensor $\mathcal{Y}$;

$\tilde{\mathcal{X}} \leftarrow$ RESHAPE($\mathcal{X}$, $[M, N, I_1, \cdots, I_d]$);
$\mathcal{Z}_1 \leftarrow \tilde{\mathcal{X}}$;
for $j = 1$ to d **do** ▷ *Contract-In*
  $\mathcal{Z}_1 \leftarrow$ TENSORCONTRACT($\mathcal{Z}_1$, $\mathcal{A}_j$);
end for
$\mathcal{Z}_2 \leftarrow$ CONV2D($\mathcal{Z}_1$, $\mathcal{C}$); ▷ *Core-Convolution*
for $j = 1$ to d **do** ▷ *Contract-Out*
  $\mathcal{Z}_2 \leftarrow$ TENSORCONTRACT($\mathcal{Z}_2$, $\mathcal{B}_j$);
end for
$\mathcal{Y} \leftarrow$ RESHAPE($\mathcal{Z}_2$, $[M', N', O]$).

are split in our proposed tensor contraction scheme for efficient TT-CONV, the original data repeating-based enlargement for input tensor $\mathcal{X}$ can be avoided. As illustrated in Fig. 5, we can place $\mathcal{C} \in \mathbb{R}^{R_d \times R_{d+1} \times K \times K}$, as the new notation of original $\mathcal{G}_0$ in our proposed scheme, between $\{\mathcal{A}_j\}_{j=1}^d$ and $\{\mathcal{B}_j\}_{j=1}^d$ in the execution scheme for the TT-CONV layer. With such arrangement, the first tensor contraction of the entire computation is involved with $\mathcal{X}$ instead of $\bar{\mathcal{X}}$, thereby significantly reducing memory consumption. Notice that due to the requirement for mathematical equivalence, the operation between $\mathcal{C}$ and its input $\mathcal{Z}_1$ is not tensor contraction any more but an explicit standard convolution as:

$$\mathcal{Z}_2 = \mathcal{Z}_1 * \mathcal{C}, \tag{11}$$

which we name it as *Core Convolution* phase in our proposed TT-CONV computation scheme.

The Overall Decomposition and Execution Scheme. Based on the above five analytic outcomes, we now can summarize and formalize the decomposition and execution scheme for the computation-efficient TT-format convolutional layer. As illustrated in Fig. 5 and described in Algorithm 1, the original 4-D weight tensor is decomposed to two sets of 3-D tensor cores plus a 4-D convolution core. Different from the case for conventional TT-CONV layer, the new computation scheme consists of three individual phases, i.e., contract-in, core convolution and contract-out, whose details are outlined in Algorithm 1. As illustrated in Fig. 2, such new computation scheme can bring significant saving for FLOPs counts. In general, the overall computational complexity of our TT-format convolution can now be reduced from $\mathcal{O}(IOK^2M'N')$ to $\mathcal{O}((IR+OR+K^2R^2)MN)$, thereby ensuring the simultaneous saving in both model sizes and computational costs.

Training for High-accuracy. To obtain the proposed efficient TT-CONV-based CNN model, a straightforward way is to simply perform this new TT-CONV decomposition on the uncompressed model and then fine-tune it. However, such direct decomposition and fine-tune strategy may suffer from significant performance degradation due to the potentially high decomposition error and increased depth of the model. In order to fully reap the benefits of this computation-efficient TT-format convolution and improve the performance, motivated by the idea of gradually imposing the constraint in [43], we propose and develop a similar training framework that is customized for our efficient TT-CONV solution. To be specific, we first train the original uncompressed model in the full-rank format with gradually imposing TT-ranks onto the weight tensors. Consider the following training objective:

$$\begin{aligned} &\min\ \ell(\mathcal{W}), \\ &\text{s.t. tt-rank}(\widetilde{\mathcal{W}}) \leqslant \widehat{\mathbf{R}}, \end{aligned} \tag{12}$$

where ℓ is loss function, $\widetilde{\mathcal{W}} \in \mathbb{R}^{I_1 \times \cdots \times I_d \times K^2 \times O_1 \times \cdots \times O_d}$ is the reordered tensor of $\mathcal{W}$ by our proposal (see Fig. 5), and $\widehat{\mathbf{R}} = [\widehat{R}_0, \widehat{R}_1, \cdots, \widehat{R}_{2d+1}]$ are the desired TT-ranks for each TT-cores. We leverage alternating direction of augmented Lagrangian (ADAL) algorithm [5, 8] to efficiently solve this problem. After that, we decompose the well trained uncompressed model into the proposed efficient TT-format, and then fine-tune it with a few epochs to further minimize $\ell(\{\mathcal{A}_j\}, \{\mathcal{B}_j\}, \mathcal{C})$. The overall training procedure is summarized in Algorithm 2.

Algorithm 2 Pseudo-code for high-accuracy training

```
def training(w, tt_shapes, tt_ranks, tau,
dense_epochs, tt_epochs):
  # Train original model with ADAL
  train_dense(w, tau, dense_epochs)
  # Decompose to TT-format
  tt_cores = dense_to_tt(w)
  # Retrain compressed TT-format model
  train_tt(tt_cores, tt_epochs)
def train_dense(w, tau, epochs):
  u, v = zeros(w.shape), Tensor(w)
  for e in range(epochs):
    x, y = sample_data()
    y_ = model_predict(w, x)
    loss = cross_entropy(y, y_)
    v = truncate_tt_ranks(w + u)
    loss += tau * norm(w - v + u, p=2)
    loss.backward()
    u += w - v
```

4. Experiments

To evaluate the performance of the proposed efficient TT-format convolution, we conduct compression experiments for different CNN models on CIFAR-10 and ImageNet datasets. On CIFAR-10 dataset, the evaluated models are ResNet-32 and ResNet-56. On ImageNet dataset, ResNet-18 and ResNet-50 are selected for evaluation.

Hyper-Parameter Settings. In all experiments the optimizer is set as momentum SGD. For ResNet-32 and ResNet-56 models, the learning rate is set as 0.01 and is multiplied by 0.1 every 45 epochs. For ResNet-18 and ResNet-50 models, the learning rate is set as 0.001.

4.1. Performance on CIFAR-10

ResNet-32. As shown in Table 1, when compressing ResNet-32 with 65% fewer model parameters and 60% fewer FLOPs, our proposed HODEC can provide 93.05% top-1 accuracy, which is even 0.56% higher than the baseline uncompressed model model; while the existing TT-format convolution [7] suffers from significant computational overhead (500% FLOPs increase) even it can achieve 50% model size reduction. Meanwhile, compared with the existing pruning approaches, HODEC enjoys higher FLOPs reduction and higher accuracy simultaneously.

ResNet-56. Our approach also shows promising performance when compressing ResNet-56. Specifically, HODEC enjoys 1.17% top-1 accuracy improvement over the uncompressed baseline with even high FLOPs reduction of 62%. Compared to the state-of-the-art pruning [31] and low-rank matrix decomposition [17, 36], our solution enables higher FLOPs reduction with even higher accuracy.

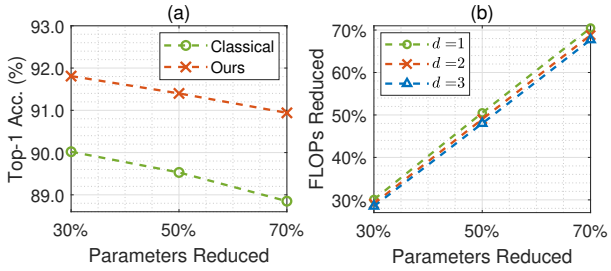

Figure 6. (a) Comparison between our HODEC and classical TT-CONV for training a compressed ResNet-32 model from scratch on CIFAR-10 dataset. (b) FLOPs reduction curves w.r.t. parameters reduction rate with multiple order d's for `layer3.0.conv1` in ResNet-18.

4.2. Performance on ImageNet

ResNet-18. Table 2 summarizes the compression performance on ImageNet dataset. It is seen that among different approaches for compressing ResNet-18, our HODEC enjoys the smallest accuracy drop on both top-1 and top-5 accuracy while achieving the highest FLOPs reduction.

ResNet-50. When compressing ResNet-50, with 63% FLOPs reduction the proposed HODEC can bring 0.31% and 0.29% top-1 and top-5 accuracy increase, respectively, over the baseline uncompressed model. Compared to the state-of-the-art pruning and low-rank matrix decomposition methods, our solution enjoys nearly 1% more top-1 accuracy and 10% more FLOPs reduction. Besides, HODEC also outperforms the state-of-the-art tensor decomposition solution [27] via providing almost 2% higher top-1 accuracy with similar FLOPs reduction rate.

4.3. Ablation Study

Accuracy Benefit of using Core Convolution. As shown in Fig. 5, our proposed HODEC performs the core convolution between contract-in and contract-out phase instead of at the beginning as classical TT-CONV does (see Fig. 1). In addition to the computational cost saving, we hypothesize such arrangement can also bring benefit of accuracy improvement. This is because now the core convolution with the kernel information-contained $\mathcal{C} \in \mathbb{R}^{R_d \times R_{d+1} \times K \times K}$ is not performed till all the input channel information-involved tensor cores, i.e., $\mathcal{A}_j \in \mathbb{R}^{R_{j-1} \times I_j \times R_j}$ containing I_j channel, are contracted with $\tilde{\mathcal{X}} \in \mathbb{R}^{M \times N \times I_1 \times \cdots \times I_d}$. Compared with the conventional TT-CONV that directly performs convolution with $\mathcal{G}_0 \in \mathbb{R}^{R_0 \times K \times K \times R_1}$ at the beginning without fully leveraging input channel information, the computing scheme of HODEC is likely able to better preserve and utilize important spatial information, thereby providing better performance.

To verify this hypothesis, we compare the training-from-scratch performance between using HODEC and conventional TT-CONV. Here no pre-trained model or training op-

Model	Compression Method	Top-1 Acc. (%)			FLOPs↓	Params.↓
		Baseline	Compressed	Δ		
ResNet-32						
Rethinking [20]	Pruning	N/A	92.56	N/A	30%	30%
FPGM [12]	Pruning	92.63	92.82	+0.19	53%	N/A
SCOP	Pruning	92.66	92.13	-0.53	56%	56%
Wide [34]	Tensor Ring	92.49	90.30	-2.19	N/A	80%
Ultimate [7, 24]	Classical TT	92.49	88.30	-4.19	×	80%
HODEC (Ours)	Proposed TT	92.49	91.28	-1.21	**72%**	**80%**
HODEC (Ours)	Proposed TT	92.49	**93.05**	**+0.56**	**60%**	65%
ResNet-56						
HRank [19]	Pruning	93.26	93.17	-0.09	50%	42%
SCOP [32]	Pruning	93.70	93.64	-0.06	56%	56%
NPPM [6]	Pruning	93.04	93.40	+0.36	50%	N/A
CHIP [31]	Pruning	93.26	94.16	+0.75	47%	43%
TRP [36]	Low-rank Matrix	93.14	92.63	-0.51	60%	N/A
CC [17]	Low-rank Matrix	93.33	93.64	+0.31	52%	48%
Ultimate [7, 24]	Classical TT	93.04	91.14	-1.90	×	50%
HODEC (Ours)	Proposed TT	93.04	**94.20**	**+1.16**	**62%**	**67%**

Table 1. Performance comparison for compressing CNN models on CIFAR-10 dataset.

Model	Compression Method	Top-1 Acc. (%)			Top-5 Acc. (%)			FLOPs↓
		Baseline	Compr.	Δ	Baseline	Compr.	Δ	
ResNet-18								
FPGM [12]	Pruning	70.28	68.41	-1.87	89.63	88.48	-1.15	42%
SCOP [32]	Pruning	69.76	68.62	-1.14	89.08	88.45	-0.63	45%
TRP [36]	Low-rank Matrix	69.10	65.51	-3.59	88.94	86.74	-2.20	60%
Stable [27]	Tucker-CP	69.76	69.07	-0.69	89.08	88.93	-0.15	67%
HODEC (Ours)	Proposed TT	69.76	**69.15**	**-0.61**	89.08	**88.99**	**-0.09**	**68%**
ResNet-50								
FPGM [12]	Pruning	76.15	75.59	-0.56	92.87	92.63	-0.24	42%
HRank [19]	Pruning	76.15	74.98	-1.17	92.87	92.33	-0.54	44%
SCOP [32]	Pruning	76.15	75.26	-0.89	92.87	92.53	-0.34	55%
NPPM [6]	Pruning	76.15	75.96	-0.19	92.87	92.75	-0.12	56%
CHIP [31]	Pruning	76.15	76.15	0.00	92.87	92.91	+0.04	49%
TRP [36]	Low-rank Matrix	75.90	74.06	-1.84	92.70	92.07	-0.63	45%
CC [17]	Low-rank Matrix	76.15	75.59	-0.56	92.87	92.64	-0.23	53%
Stable [27]	Tucker-CP	76.13	74.66	-1.47	92.87	92.16	-0.71	62%
HODEC (Ours)	Proposed TT	76.13	**76.44**	**+0.31**	92.87	**93.16**	**+0.29**	**63%**

Table 2. Performance comparison for compressing CNN models on ImageNet dataset.

timization technique is used for fair comparison. Fig. 6 (a) shows the curves of accuracy-vs-compression ratio. It is seen that HODEC provides nearly 2% higher accuracy over the classical TT-CONV while training from the same randomly initialized ResNet-32 models on CIFAR-10 dataset, thereby demonstrating the benefit of using core convolution.

Influence of Order d. In the proposed HODEC, the decomposition of the reshaped $\widetilde{\mathcal{W}}$ can vary via using different d's (the number of orders). Fig. 6 (b) shows the curve of FLOPs reduction-vs-parameters reduction with different d's for an example layer (`layer3.0.conv1` in ResNet-18). It is seen that larger d can slightly lower the FLOPs reduction but the change is not significant.

5. Conclusion

In this paper, we propose HODEC, an efficient high-order decomposed convolution solution based on TT-format, which can simultaneously provide high FLOPs and parameters reduction. Experimental results show HODEC exhibits state-of-the-art performance for compressing and accelerating various CNN models on multiple datasets.

Acknowledgements

This work was partially supported by National Science Foundation under Grant CCF-1955909.

References

[1] Chunhua Deng, Yang Sui, Siyu Liao, Xuehai Qian, and Bo Yuan. Gospa: an energy-efficient high-performance globally optimized sparse convolutional neural network accelerator. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 1110–1123. IEEE, 2021. 1

[2] Chunhua Deng, Fangxuan Sun, Xuehai Qian, Jun Lin, Zhongfeng Wang, and Bo Yuan. Tie: energy-efficient tensor train-based inference engine for deep neural network. In *Proceedings of the 46th International Symposium on Computer Architecture*, pages 264–278, 2019. 1

[3] Chunhua Deng, Miao Yin, Xiao-Yang Liu, Xiaodong Wang, and Bo Yuan. High-performance hardware architecture for tensor singular value decomposition. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2019. 1

[4] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 293–302, 2019. 1

[5] Silvia Gandy, Benjamin Recht, and Isao Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011. 6

[6] Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9270–9280, 2021. 8

[7] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*, 2016. 1, 2, 7, 8

[8] Donald Goldfarb and Zhiwei Qin. Robust low-rank tensor recovery: Models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1):225–253, 2014. 6

[9] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[10] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in Neural Information Processing Systems*, 28:1135–1143, 2015. 1

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1

[12] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 8

[13] Shaoyi Huang, Dongkuan Xu, Ian EH Yen, Sung-en Chang, Bingbing Li, Shiyang Chen, Mimi Xie, Hang Liu, and Caiwen Ding. Sparse progressive distillation: Resolving overfitting under pretrain-and-finetune paradigm. *arXiv preprint arXiv:2110.08190*, 2021. 1

[14] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018. 1

[15] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017. 1

[16] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016. 1

[17] Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6438–6447, 2021. 7, 8

[18] Siyu Liao, Chunhua Deng, Miao Yin, and Bo Yuan. Doubly residual neural decoder: Towards low-complexity high-performance channel decoding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8574–8582, 2021. 1

[19] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020. 8

[20] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 8

[21] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5058–5066, 2017. 1

[22] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11264–11272, 2019. 1

[23] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019. 1

[24] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in Neural Information Processing Systems*, 28:442–450, 2015. 1, 2, 8

[25] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. 1, 2

[26] Yu Pan, Jing Xu, Maolin Wang, Jinmian Ye, Fei Wang, Kun Bai, and Zenglin Xu. Compressing recurrent neural networks with tensor ring for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4683–4690, 2019. 1

[27] Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *European Conference on Computer Vision*, pages 522–539. Springer, 2020. 7, 8

[28] Feng Qian, Miao Yin, Xiao-Yang Liu, Yao-Jun Wang, Cai Lu, and Guang-Min Hu. Unsupervised seismic facies analysis via deep convolutional autoencoders. *Geophysics*, 83(3):A39–A43, 2018. 1

[29] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 1

[30] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1

[31] Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. In *Advances in Neural Information Processing Systems*, 2021. 7, 8

[32] Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing XU, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. In *Advances in Neural Information Processing Systems*, volume 33, pages 10936–10947, 2020. 8

[33] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019. 1

[34] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018. 8

[35] Dongkuan Xu, Ian EH Yen, Jinxi Zhao, and Zhibin Xiao. Rethinking network pruning–under the pre-train and fine-tune paradigm. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2376–2382, 2021. 1

[36] Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. Trp: Trained rank pruning for efficient deep neural networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 977–983, 2020. 7, 8

[37] Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. In *International Conference on Machine Learning*, pages 3891–3900, 2017. 1

[38] Jinmian Ye, Linnan Wang, Guangxi Li, Di Chen, Shandian Zhe, Xinqi Chu, and Zenglin Xu. Learning compact recurrent neural networks with block-term tensor decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9378–9387, 2018. 1

[39] Miao Yin, Siyu Liao, Xiao-Yang Liu, Xiaodong Wang, and Bo Yuan. Compressing recurrent neural networks using hierarchical tucker tensor decomposition. *arXiv preprint arXiv:2005.04366*, 2020. 1

[40] Miao Yin, Siyu Liao, Xiao-Yang Liu, Xiaodong Wang, and Bo Yuan. Towards extremely compact rnns for video recognition with fully decomposed hierarchical tucker structure. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12085–12094, June 2021. 1

[41] Miao Yin, Huy Phan, Xiao Zang, Siyu Liao, and Bo Yuan. Batude: Budget-aware neural network compression based on tucker decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022. 1

[42] Miao Yin, Yang Sui, Siyu Liao, and Bo Yuan. Towards efficient tensor decomposition-based dnn model compression with optimization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10683, June 2021. 1

[43] Tianyun Zhang, Shaokai Ye, Kaiqi Zhang, Jian Tang, Wujie Wen, Makan Fardad, and Yanzhi Wang. A systematic dnn weight pruning framework using alternating direction method of multipliers. In *European Conference on Computer Vision*, pages 184–199, 2018. 1, 6

[44] Yanyi Zhang, Xinyu Li, Chunhui Liu, Bing Shuai, Yi Zhu, Biagio Brattoli, Hao Chen, Ivan Marsic, and Joseph Tighe. Vidtr: Video transformer without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13577–13587, 2021. 1

[45] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016. 1