

# Towards Efficient Tensor Decomposition-Based DNN Model Compression with Optimization Framework

Miao Yin<sup>1</sup>, Yang Sui<sup>1</sup>, Siyu Liao<sup>2†</sup> and Bo Yuan<sup>1</sup>

<sup>1</sup>Department of ECE, Rutgers University, <sup>2</sup>Amazon

{miao.yin, yang.sui}@rutgers.edu, liasiyu@amazon.com, bo.yuan@soe.rutgers.edu

## Abstract

Advanced tensor decomposition, such as tensor train (TT) and tensor ring (TR), has been widely studied for deep neural network (DNN) model compression, especially for recurrent neural networks (RNNs). However, compressing convolutional neural networks (CNNs) using TT/TR always suffers significant accuracy loss. In this paper, we propose a systematic framework for tensor decomposition-based model compression using Alternating Direction Method of Multipliers (ADMM). By formulating TT decomposition-based model compression to an optimization problem with constraints on tensor ranks, we leverage ADMM technique to systemically solve this optimization problem in an iterative way. During this procedure, the entire DNN model is trained in the original structure instead of TT format, but gradually enjoys the desired low tensor rank characteristics. We then decompose this uncompressed model to TT format, and fine-tune it to finally obtain a high-accuracy TT-format DNN model. Our framework is very general, and it works for both CNNs and RNNs, and can be easily modified to fit other tensor decomposition approaches. We evaluate our proposed framework on different DNN models for image classification and video recognition tasks. Experimental results show that our ADMM-based TT-format models demonstrate very high compression performance with high accuracy. Notably, on CIFAR-100, with  $2.3\times$  and  $2.4\times$  compression ratios, our models have 1.96% and 2.21% higher top-1 accuracy than the original ResNet-20 and ResNet-32, respectively. For compressing ResNet-18 on ImageNet, our model achieves  $2.47\times$  FLOPs reduction without accuracy loss.

## 1. Introduction

Deep Neural Network (DNNs) have already obtained widespread applications in many computer vision tasks, such as image classification [19, 28], video recognition

[10, 2], objective detection [13, 41], and image caption [45, 9]. Despite these unprecedented success and popularity, executing DNNs on the edge devices is still very challenging. For most embedded and Internet-of-Things (IoT) systems, the sizes of many state-of-the-art DNN models are too large, thereby causing high storage and computational demands and severely hindering the practical deployment of DNNs. To mitigate this problem, to date many *model compression* approaches, such as pruning [16, 17, 35, 50] and quantization [17, 47, 40], have been proposed to reduce the sizes of DNN models with limited impact on accuracy.

**Tensor Decomposition for Model Compression.** Recently, *tensor decomposition*, as a mathematical tool that explores the low tensor rank characteristics of the large-scale tensor data, have become a very attractive DNN model compression technique. Different from other model compression methods, tensor decomposition, uniquely, can provide ultra-high compression ratio, especially for recurrent neural network (RNN) models. As reported in [48, 38], the advanced tensor decomposition approaches, such as tensor train (TT) and tensor ring (TR), can bring more than  $1,000\times$  parameter reduction to the input-to-hidden layers of RNN models, and meanwhile the corresponding classification accuracy in the video recognition task can be even significantly improved. Motivated by such strong compression performance, many prior research works have been conducted on tensor decomposition-based DNN models [12, 37, 43]. In addition, to fully utilize the benefits provided by those models, several TT-format DNN hardware accelerators have been developed and implemented in different chip formats, such as digital CMOS ASIC [7], memristor ASIC [23] and IoT board [4].

**Limitations of the State of the Art.** Despite its promising potentials, the performance of tensor decomposition is not satisfied enough as a mature model compression approach. Currently all the reported success of tensor decomposition are narrowly limited to compressing RNN models in video recognition tasks. For compressing convolutional neural network (CNN) in the image classification task, which are the most commonly used and representa-

<sup>†</sup>This work was done when the author was with Rutgers University.

tive setting for evaluating model compression performance, all the state-of-the-art tensor decomposition approaches, including TT and TR, suffer very significant accuracy loss. For instance, even the very recent progress [32] using TR still has 1.0% accuracy loss when the compression ratio is only  $2.7\times$  for ResNet-32 model on CIFAR-10 dataset. For the larger compression ratio as  $5.8\times$ , the accuracy loss further increases to 1.9%.

**Why Limited Performance?** The above limitation of tensor decomposition is mainly due to the unique challenges involved in training the tensor decomposed DNN models. In general, there are two ways to use tensor decomposition to obtain a compressed model: 1) Train from scratch in the decomposed format; and 2) Decompose a pre-trained uncompressed model and then retrain. In the former case, when the required tensor decomposition-based, e.g. TT-format model, is directly trained from scratch, because the structure of the models are already pre-set to low tensor rank format before the training, the corresponding model capacity is typically limited as compared to the full-rank structure, thereby causing the training process being very sensitive to initialization and more challenging to achieve high accuracy. In the later scenario, though the pre-trained uncompressed model provides good initialization position, the straightforwardly decomposing full-rank uncompressed model into low tensor rank format causes inevitable and non-negligible approximation error, which is still very difficult to be recovered even after long-time re-training period. Besides, no matter which training strategy is adopted, tensor decomposition always brings linear increase in network depth, which implies training the tensor decomposition-format DNNs are typically more prone to gradient vanishing problem and hence being difficult to be trained well.

**Technical Preview and Contributions.** To overcome the current limitations of tensor decomposition and fully unlock its potentials for model compression, in this paper we propose a systematic framework for tensor decomposition-based model compression using alternating direction method of multipliers (ADMM). By formulating TT decomposition-based model compression to an optimization problem with constraints on tensor ranks, we leverage ADMM technique [1] to systemically solve this optimization problem in an iterative way. During this procedure the entire DNN model is trained in the original structure instead of TT format, but gradually enjoys the desired low tensor rank characteristics. We then decompose this uncompressed model to TT format, and fine-tune it to finally obtain a high-accuracy TT-format DNN model. In overall, the contributions of this paper are summarized as follows:

- We propose a systematic framework to formulate and solve the tensor decomposition-based model compression problem. With formulating this problem to a constrained non-convex optimization problem, our frame-

work gradually restricts the DNN model to the target tensor ranks without explicitly training on the TT format, thereby maintaining the model capacity as well as avoiding huge approximation error and increased network depth.

- We propose to use ADMM to efficiently solve this reformulated optimization problem via separately solving two sub-problems: one is to directly optimize the loss function with a regularization of the DNN by stochastic gradient descent, and the other is to use the introduced projection to constraint the tensor ranks analytically.
- We evaluate our proposed framework on different DNN models for image classification and video recognition tasks. Experimental results show that our ADMM-based TT-format models demonstrate very high compression performance with high accuracy. Notably, on CIFAR-100, with  $2.3\times$  and  $2.4\times$  compression ratios, our models have 1.96% and 2.21% higher top-1 accuracy than the original ResNet-20 and ResNet-32, respectively. For compressing ResNet-18 on ImageNet, our model achieves  $2.47\times$  FLOPs reduction with no accuracy loss.

## 2. Related Work on DNN Model Compression

**Sparsification.** Sparsification is the most popular DNN compression approach. Different levels of network structure can be sparse, such as weight [16, 17], filter [35, 20] and channel [22, 52]. To obtain the sparsity, a DNN model can be either pruned [16, 35] or trained with sparsity-aware regularization [33, 51]. Also, the introduced sparsity can be either structured or unstructured. Unstructured sparse models [16, 50] enjoy high accuracy and compression ratio, but brings irregular memory access and imbalanced workload problems [15] to the underlying hardware platform. Structured sparse models [44] are more hardware friendly; however, their compression ratio and accuracy are typically inferior to the unstructured counterparts.

**Quantization.** Quantization [40, 16, 47] is another widely adopt model compression approach. By reducing the number of bits for weight representation, quantization enables immediate reduction in DNN model size. The most aggressive quantization scheme brings binary networks [40, 6], which only use 1-bit weight parameters. Quantization is inherently hardware friendly, and have become a standard adopted model compression method for most DNN hardware accelerators [15, 3, 25]. However, quantization is limited by the maximum compression ratio that can be offered (up to  $32\times$ ).

**Tensor Decomposition.** Rooted in tensor theory, tensor decomposition approach factorizes weight tensors into

smaller tensors to reduce model sizes. In [24], matrix-oriented singular value decomposition (SVD), as the low-dimensional instance of tensor decomposition, is used to perform model compression. However, using this method, or other classical high-dimensional tensor decomposition methods, such as Tucker [42] and CP decomposition [18], causes significant accuracy loss ( $> 0.5\%$ ) with limited compression ratios [26, 30, 14, 39]. Starting from [12], advanced tensor decomposition approaches, such as tensor train (TT) and tensor ring (TR) decomposition, have become the more popular options. These methods have very attractive advantages – the compression ratio can be very high (e.g.  $> 1,000\times$ ) because of their unique mathematical property. Such benefits have been demonstrated on RNN compression in video recognition tasks. As reported in [48, 38],  $17,560\times$  to  $34,203\times$  compression ratios can be achieved by using TT or TR decomposition on the input-to-hidden layer of RNN models for video recognition. However, TT and TR approaches do not perform well on CNN models. For instance, even the very recent progress [43, 32] still suffers 1.0% accuracy loss with  $2.7\times$  compression ratio, or even 1.9% accuracy loss with  $5.8\times$  compression ratio, both for ResNet-32 model on CIFAR-10 dataset. From the perspective of practical deployment, such non-negligible accuracy degradation severely hinders the widespread adoption of tensor decomposition for many CNN-involved model compression scenarios.

### 3. Background and Preliminaries

#### 3.1. Notation

$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ ,  $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2}$ , and  $\mathbf{x} \in \mathbb{R}^{n_1}$  represent  $d$ -order tensor, matrix and vector, respectively. Also,  $\mathcal{X}_{(i_1, \dots, i_d)}$  and  $\mathbf{X}_{(i,j)}$  denote the single entry of tensor  $\mathcal{X}$  and matrix  $\mathbf{X}$ , respectively.

#### 3.2. Tensor Train (TT) Decomposition

Given a tensor  $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_d}$ , it can be decomposed to a sort of 3-order tensors via Tensor Train Decomposition (TTD) as follows:

$$\begin{aligned} \mathcal{A}_{(i_1, i_2, \dots, i_d)} &= \mathcal{G}_1(:, i_1, :) \mathcal{G}_2(:, i_2, :) \dots \mathcal{G}_d(:, i_d, :) \\ &= \sum_{\alpha_0, \alpha_1, \dots, \alpha_d} \mathcal{G}_1(\alpha_0, i_1, \alpha_1) \mathcal{G}_2(\alpha_1, i_2, \alpha_2) \dots \mathcal{G}_d(\alpha_{d-1}, i_d, \alpha_d), \end{aligned} \quad (1)$$

where  $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$  are called *TT-cores* for  $k = 1, 2, \dots, d$ , and  $\mathbf{r} = [r_0, r_1, \dots, r_d]$ ,  $r_0 = r_d = 1$  are called *TT-ranks*, which determine the storage complexity of TT-format tensor. An example is demonstrated in Figure 1.

#### 3.3. Tensor Train (TT)-format DNN

**TT Fully-Connected Layer.** Consider a simple fully-connected layer with weight matrix  $\mathbf{W} \in \mathbb{R}^{M \times N}$  and in-

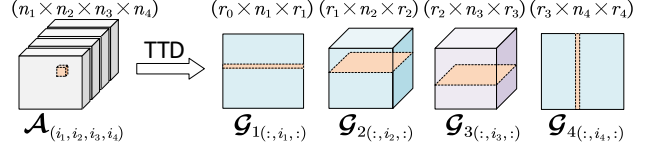


Figure 1: Illustration of Tensor Train Decomposition (TTD) for a 4-order tensor.  $r_0$  and  $r_4$  are always equal to 1.

put  $\mathbf{x} \in \mathbb{R}^N$ , where  $M = \prod_{k=1}^d m_k$  and  $N = \prod_{k=1}^d n_k$ , the output  $\mathbf{y} \in \mathbb{R}^M$  is obtained by  $\mathbf{y} = \mathbf{W}\mathbf{x}$ . In order to transform this standard layer to TT fully-connected (TT-FC) layer, we first tensorize the weight matrix  $\mathbf{W}$  to a weight tensor  $\mathcal{W} \in \mathbb{R}^{(m_1 \times n_1) \times \dots \times (m_d \times n_d)}$  by reshaping and order transposing. Then  $\mathcal{W}$  can be decomposed to TT-format:

$$\mathcal{W}_{((i_1, j_1), \dots, (i_d, j_d))} = \mathcal{G}_1(:, i_1, j_1, :) \dots \mathcal{G}_d(:, i_d, j_d, :). \quad (2)$$

Here, each TT-core  $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times m_k \times n_k \times r_k}$  is a 4-order tensor, which is one dimension more than the standard one since the output and input dimensions of  $\mathbf{W}$  are divided separately. Hence, the forward propagation on the TT-FC layer can be expressed in tensor format as follows:

$$\mathcal{Y}_{(i_1, \dots, i_d)} = \sum_{j_1, \dots, j_d} \mathcal{G}_1(:, i_1, j_1, :) \dots \mathcal{G}_d(:, i_d, j_d, :) \mathcal{X}_{(j_1, \dots, j_d)}, \quad (3)$$

where  $\mathcal{X} \in \mathbb{R}^{m_1 \times \dots \times m_d}$  and  $\mathcal{Y} \in \mathbb{R}^{n_1 \times \dots \times n_d}$  are the tensorized input and output corresponding to  $\mathbf{x}$  and  $\mathbf{y}$ , respectively. The details about TT-FC layer is introduced in [37].

**TT Convolutional Layer.** For a conventional convolutional layer, its forward computation is to perform convolution between a 3-order input tensor  $\tilde{\mathcal{X}} \in \mathbb{R}^{W \times H \times N}$  and a 4-order weight tensor  $\tilde{\mathcal{W}} \in \mathbb{R}^{K \times K \times M \times N}$  to produce the 3-order output tensor  $\tilde{\mathcal{Y}} \in \mathbb{R}^{(W-K+1) \times (H-K+1) \times M}$ . In a TT convolutional (TT-CONV) layer, the input tensor  $\tilde{\mathcal{X}}$  is reshaped to a tensor  $\mathcal{X} \in \mathbb{R}^{W \times H \times n_1 \times \dots \times n_d}$ , while the weight tensor  $\tilde{\mathcal{W}}$  is reshaped and transposed to a tensor  $\mathcal{W} \in \mathbb{R}^{(K \times K) \times (m_1 \times n_1) \times \dots \times (m_d \times n_d)}$  and then decomposed to TT-format:

$$\mathcal{W}_{((k_1, k_2), (i_1, j_1), \dots, (i_d, j_d))} = \mathcal{G}_0(k_1, k_2) \mathcal{G}_1(:, i_1, j_1, :) \dots \mathcal{G}_d(:, i_d, j_d, :), \quad (4)$$

where  $M = \prod_{k=1}^d m_k$  and  $N = \prod_{k=1}^d n_k$ . Similar with TT-FC layer, here  $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times m_k \times n_k \times r_k}$  is a 4-order tensor except  $\mathcal{G}_0 \in \mathbb{R}^{K \times K}$ . Then the new output tensor  $\mathcal{Y} \in \mathbb{R}^{(W-K+1) \times (H-K+1) \times m_1 \times \dots \times m_d}$  is obtained by

$$\begin{aligned} \mathcal{Y}_{(w, h, i_1, \dots, i_d)} &= \sum_{k_1=1}^K \sum_{k_2=1}^K \sum_{j_1, \dots, j_d} \mathcal{X}_{(k_1+w-1, k_2+h-1, j_1, \dots, j_d)} \\ &\quad \mathcal{G}_0(k_1, k_2) \mathcal{G}_1(:, i_1, j_1, :) \dots \mathcal{G}_d(:, i_d, j_d, :). \end{aligned} \quad (5)$$

The detailed description of TT-CONV layer is in [12].

**Training on TT-format DNN.** As TT-FC layer, TT-CONV layer and the corresponding forward propagation schemes are formulated, standard stochastic gradient descent (SGD) algorithm can be used to update the TT-cores

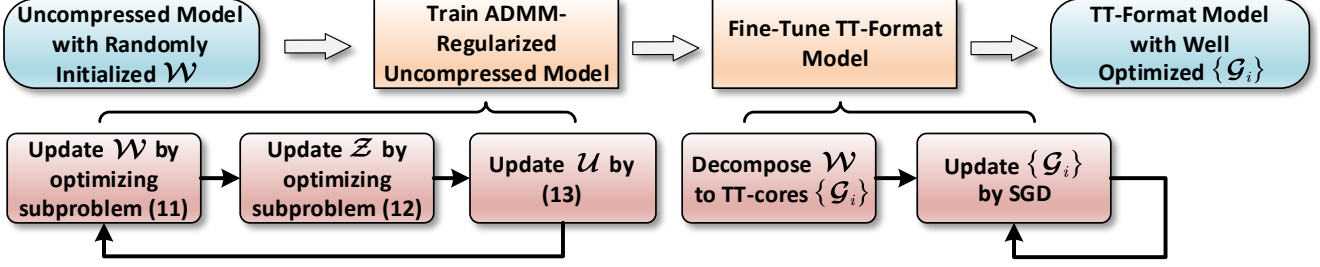


Figure 2: Procedure of the proposed compression framework using ADMM for a TT-format DNN model.

with the rank set  $\mathbf{r}$ , which determines the target compression ratio. The initialization of the TT-cores can be either randomly set or obtained from directly TT-decomposing a pre-trained uncompressed model.

#### 4. Systematic Compression Framework

**Analysis on Existing TT-format DNN Training.** As mentioned in the last paragraph, currently a TT-format DNN is either 1) trained from with randomly initialized tensor cores; or 2) trained from a direct decomposition of pre-trained model. For the first strategy, it does not utilize any information related to the high-accuracy uncompressed model; while other model compression methods, e.g. pruning and knowledge distillation, have shown that proper utilization of the pre-trained models are very critical for DNN compression. For the second strategy, though the knowledge of the pre-trained model is indeed utilized, because the pre-trained model generally lacks low TT-rank property, after direct low-rank tensor decomposition the approximation error is too significant to be properly recovered even using long-time re-training. Such inherent limitations of the existing training strategies, consequently, cause significant accuracy loss for the compressed TT-format DNN models.

**Our Key Idea.** We believe the key to overcome these limitations is to maximally retain the knowledge contained in the uncompressed model, or in other words, minimize the approximation error after tensor decomposition with given target tensor ranks. To achieve that, we propose to formulate an optimization problem to minimize the loss function of the uncompressed model with low tensor rank constraints. With proper advanced optimization technique (e.g. ADMM)-regularized training procedure, the uncompressed DNN models can gradually exhibit low tensor rank properties. After the ADMM-regularized training phase, the approximation error brought by the explicit low-rank tensor decomposition becomes negligible, and can be easily recovered by the SGD-based fine-tuning. Figure 2 shows the main steps of our proposed overall framework.

##### 4.1. Problem Formulation

As mentioned above, the first phase of our framework is to gradually impose low tensor rank characteristics onto a

high-accuracy uncompressed DNN model. Mathematically, this goal can be formulated as a optimization problem to minimize the loss function of the object model with constraints on TT-ranks of each layer (convolutional or fully-connected):

$$\begin{aligned} \min_{\mathcal{W}} \quad & \ell(\mathcal{W}), \\ \text{s.t.} \quad & \text{rank}(\mathcal{W}) \leq \mathbf{r}^*, \end{aligned} \quad (6)$$

where  $\ell$  is the loss function of the DNN,  $\text{rank}(\cdot)$  is a function that returns the TT-ranks  $\mathbf{r} = [r_0, \dots, r_d]$  of the weight tensor cores, and  $\mathbf{r}^* = [r_0^*, \dots, r_d^*]$  are the desired TT-ranks for the layer. To simplify the notation, here  $\mathbf{r} \leq \mathbf{r}^*$  means  $r_i \leq r_i^*, i = 0, \dots, d$ , for each  $r_i$  in  $\mathbf{r}$ .

##### 4.2. Optimization Using ADMM

Obviously, solving the problem (6) is generally difficult via using normal optimization algorithms since  $\text{rank}(\cdot)$  is non-differentiable. To overcome this challenge, we first rewrite it as

$$\begin{aligned} \min_{\mathcal{W}} \quad & \ell(\mathcal{W}), \\ \text{s.t.} \quad & \mathcal{W} \in \mathcal{S}, \end{aligned} \quad (7)$$

where  $\mathcal{S} = \{\mathcal{W} \mid \text{rank}(\mathcal{W}) \leq \mathbf{r}^*\}$ . Hence, the objective form (7) is a classic non-convex optimization problem with constraints, which can be properly solved by ADMM [1]. Specifically, we can first introduce an auxiliary variable  $\mathcal{Z}$  and an indicator function  $g(\cdot)$  of  $\mathcal{S}$ , i.e.

$$g(\mathcal{W}) = \begin{cases} 0 & \mathcal{W} \in \mathcal{S}, \\ +\infty & \text{otherwise.} \end{cases} \quad (8)$$

And then the problem (7) is equivalent to the following form:

$$\begin{aligned} \min_{\mathcal{W}, \mathcal{Z}} \quad & \ell(\mathcal{W}) + g(\mathcal{Z}), \\ \text{s.t.} \quad & \mathcal{W} = \mathcal{Z}. \end{aligned} \quad (9)$$

To ensure convergence without assumptions like strict convexity or finiteness of  $\ell$ , instead of Lagrangian, the corresponding augmented Lagrangian in the scaled dual form of the above problem is given by

$$\begin{aligned} \mathcal{L}_\rho(\mathcal{W}, \mathcal{Z}, \mathcal{U}) = & \ell(\mathcal{W}) + g(\mathcal{Z}) \\ & + \frac{\rho}{2} \|\mathcal{W} - \mathcal{Z} + \mathcal{U}\|_F^2 + \frac{\rho}{2} \|\mathcal{U}\|_F^2, \end{aligned} \quad (10)$$



where  $\mathbf{U}$  is the dual multiplier, and  $\rho > 0$  is the penalty parameter. Thus, the iterative ADMM scheme can be explicitly performed as

$$\mathbf{W}^{t+1} = \underset{\mathbf{W}}{\operatorname{argmin}} \mathcal{L}_\rho(\mathbf{W}, \mathbf{Z}^t, \mathbf{U}^t), \quad (11)$$

$$\mathbf{Z}^{t+1} = \underset{\mathbf{Z}}{\operatorname{argmin}} \mathcal{L}_\rho(\mathbf{W}^{t+1}, \mathbf{Z}, \mathbf{U}^t), \quad (12)$$

$$\mathbf{U}^{t+1} = \mathbf{U}^t + \mathbf{W}^{t+1} - \mathbf{Z}^{t+1}, \quad (13)$$

where  $t$  is the iterative step. Now, the original problem (9) is separated to two subproblems (11) and (12), which can be solved individually. Next, we introduce the detailed solution of each subproblem.

**W-subproblem.** The  $\mathbf{W}$ -subproblem (11) can be reformulated as follows:

$$\min_{\mathbf{W}} \ell(\mathbf{W}) + \frac{\rho}{2} \|\mathbf{W} - \mathbf{Z}^t + \mathbf{U}^t\|_F^2, \quad (14)$$

where the first term is the loss function, e.g. cross-entropy loss in classification tasks, of the DNN model, and the second term is the  $L_2$  regularization. This subproblem can be directly solved by SGD since both these two terms are differentiable. Correspondingly, the partial derivative of (14) with respect to  $\mathbf{W}$  is calculated as

$$\frac{\partial \mathcal{L}_\rho(\mathbf{W}, \mathbf{Z}^t, \mathbf{U}^t)}{\partial \mathbf{W}} = \frac{\partial \ell(\mathbf{W})}{\partial \mathbf{W}} + \rho(\mathbf{W} - \mathbf{Z}^t + \mathbf{U}^t). \quad (15)$$

And hence  $\mathbf{W}$  can be updated by

$$\mathbf{W}^{t+1} = \mathbf{W}^t - \eta \frac{\partial \mathcal{L}_\rho(\mathbf{W}, \mathbf{Z}^t, \mathbf{U}^t)}{\partial \mathbf{W}}, \quad (16)$$

where  $\eta$  is the learning rate.

**Z-subproblem.** To solve  $\mathbf{Z}$ -subproblem (12), we first explicitly formulate it as follows:

$$\min_{\mathbf{Z}} g(\mathbf{Z}) + \frac{\rho}{2} \|\mathbf{W}^{t+1} - \mathbf{Z} + \mathbf{U}^t\|_F^2, \quad (17)$$

where the indicator function  $g(\cdot)$  of the non-convex set  $\mathcal{S}$  is non-differentiable. Then, according to [1], in this format updating  $\mathbf{Z}$  can be performed as:

$$\mathbf{Z}^{t+1} = \Pi_{\mathcal{S}}(\mathbf{W}^{t+1} + \mathbf{U}^t), \quad (18)$$

where  $\Pi_{\mathcal{S}}(\cdot)$  is the projection of singular values onto  $\mathcal{S}$ , by which the TT-ranks of  $(\mathbf{W}^{t+1} + \mathbf{U}^t)$  are truncated to target ranks  $\mathbf{r}^*$ . Algorithm 1 describes the specific procedure of this projection in the TT-format scenario.

In each ADMM iteration, upon the update of  $\mathbf{W}$  and  $\mathbf{Z}$ , the dual multiplier  $\mathbf{U}$  is updated by (13). In overall, to solve (9), the entire ADMM-regularized training procedure is performed in an iterative way until convergence or reaching the pre-set maximum iteration number. The overall procedure is summarized in Algorithm 2.

### 4.3. Fine-Tuning

After ADMM-regularized training, we first decompose the trained uncompressed DNN model into TT format. Here the decomposition is performed with the target TT-ranks  $\mathbf{r}^*$

---

#### Algorithm 1 TT-SVD-based Projection for Solving (17)

---

**Input:**  $d$ -order tensor  $\mathcal{A} \in \mathbb{R}^{n_1 \times \dots \times n_d}$ , target TT-ranks  $\mathbf{r}^*$ .

**Output:**  $\hat{\mathcal{A}} = \Pi_{\mathcal{S}}(\mathcal{A})$ .

```

1: Temporary tensor  $\mathcal{T} = \mathcal{A}$ ;
2: for  $k = 1$  to  $d - 1$  do
3:    $\mathcal{T} := \operatorname{reshape}(\mathcal{T}, [r_{k-1}^*, n_k, -1])$ ;
4:   Compute matrix SVD:  $\mathbf{U}, \mathbf{S}, \mathbf{V} := \operatorname{SVD}(\mathcal{T})$ ;
5:    $\mathbf{U} := \mathbf{U}_{(1:r_k^*, :)}$ ;
6:    $\mathbf{S} := \mathbf{S}_{(1:r_k^*, 1:r_k^*)}$ ;
7:    $\mathbf{V} := \mathbf{V}_{(:, 1:r_k^*)}$ ;
8:    $\mathcal{G}_k := \operatorname{reshape}(\mathbf{U}, [r_{k-1}^*, n_k, r_k^*])$ ;
9:    $\mathcal{T} := \mathbf{S}\mathbf{V}^T$ ;
10:  $\mathcal{T} := \mathcal{G}_1$ ;
11: for  $k = 1$  to  $d - 1$  do
12:    $\mathcal{T}_1 := \operatorname{reshape}(\mathcal{T}, [-1, r_k^*])$ ;
13:    $\mathcal{T}_2 := \operatorname{reshape}(\mathcal{G}_{k+1}, [r_k^*, -1])$ ;
14:    $\mathcal{T} := \mathcal{T}_1 \mathcal{T}_2$ ;
15:  $\hat{\mathcal{A}} = \operatorname{reshape}(\mathcal{T}, [n_1, \dots, n_d])$ .
```

---



---

#### Algorithm 2 ADMM-Regularized Training Procedure

---

**Input:** Weight tensor  $\mathbf{W}$ , target TT-ranks  $\mathbf{r}^*$ , penalty parameter  $\rho$ , feasibility tolerance  $\epsilon$ , maximum iterations  $T$ .

**Output:** Optimized  $\mathbf{W}$ .

```

1: Randomly initialize  $\mathbf{W}$ ;
2:  $\mathbf{Z} := \mathbf{W}$ ,  $\mathbf{U} := \mathbf{0}$ ;
3: while  $\|\mathbf{W}^t - \mathbf{Z}^t\| > \epsilon$  and  $t \leq T$  do
4:   Updating  $\mathbf{W}$  via (16);
5:   Updating  $\mathbf{Z}$  via (18) (Algorithm 1);
6:   Updating  $\mathbf{U}$  via (13);
7: end
```

---

for tensor cores. Because the ADMM optimization procedure has already imposed the desired low TT-rank structure to the uncompressed model, such direction decomposition, unlike their counterpart in the existing TT-format DNN training, will not bring significant approximation error (More details will be analyzed in Section 5.1). Then, the decomposed TT-format model is fine-tuned using standard SGD. Notice that in the fine-tuning phase the loss function is  $\ell(\{\mathcal{G}_i\})$  without other regularization term introduced by ADMM. Typically this fine-tuning phase is very fast with requiring only a few iterations. This is because the decomposed TT model at the starting point of this phase already has very closed accuracy to the original uncompressed model.

## 5. Experiments

To demonstrate the effectiveness and generality of the proposed compression framework, we evaluate different

DNN models in different computer vision tasks. For image classification tasks, we evaluate multiple CNN models on MNIST, CIFAR-10, CIFAR-100 and ImageNet datasets [31, 27, 8]. For video classification tasks, we evaluate different LSTM models on UCF11 and HMDB51 datasets [34, 29]. We follow the same rank selection scheme adopted in prior works – set ranks to satisfy the need of the targeted compression ratio. To simplify selection procedure, most of the ranks in the same layer are set to equal.

### 5.1. Convergence and Sensitivity Analysis

As shown in (10),  $\rho$  is the additional hyperparameter introduced in the ADMM-regularized training phase. To study the effect of  $\rho$  on the performance as well as facilitating hyperparameter selection, we study the convergence and sensitivity of the ADMM-regularized training for ResNet-32 model with different  $\rho$  settings on CIFAR10 dataset.

**Convergence.** Figure 3a shows the loss curves in the ADMM-regularized training phase. It is seen that different curves with very different  $\rho$  values (e.g. 0.001 vs 0.02), exhibit very similar convergence speed. This phenomenon therefore demonstrates that  $\rho$  has little impact on the convergence of ADMM-regularized training.

**Sensitivity.** Considering the similar convergence behavior does not necessarily mean that different  $\rho$  would bring the similar accuracy, we then analyze the performance sensitivity of ADMM-regularized training with respect to  $\rho$ . Notice that ideally after ADMM-regularized training,  $\mathcal{W}$ , though in the uncompressed format, should exhibit strong low TT-rank characteristics and meanwhile enjoy high accuracy. Once  $\mathcal{W}$  meets such two criteria simultaneously, that means TT-cores  $\{\mathcal{G}_i\}$ , whose initialization is decomposed from  $\mathcal{W}$ , will already have high accuracy even before fine-tuning.

To examine the required low TT-rank behavior of  $\mathcal{W}$ , we observe  $\|\mathcal{W} - \mathcal{Z}\|_F^2$ , which measures the similarity between  $\mathcal{W}$  and  $\mathcal{Z}$ , in the ADMM-regularized training (see Figure 3b). Since according to (18)  $\mathcal{Z}$  is always updated with low TT-rank constraints, the curves shown in Figure 3b reveal that  $\mathcal{W}$  indeed quickly exhibits low TT-rank characteristics during the training, except when  $\rho = 0.001$ . This phenomenon implies that to ensure the weight tensors are well regularized to the target TT-ranks by ADMM,  $\rho$  should not be too small (e.g. less than 0.001). On the other hand, Figure 3c shows the test accuracy of  $\mathcal{W}$  as training progresses. Here it is seen that smaller  $\rho$  tends to bring better performance. Based on these observations,  $\rho = 0.005$  can be an appropriate choice to let the trained  $\mathcal{W}$  meet the aforementioned two criteria.

### 5.2. Image Classification

**MNIST.** Table 1 shows the experimental results of LeNet-5 model [31] on MNIST dataset. We compare our

Model	Comp. Method	Top-1 (%)	Comp. Ratio
Uncompressed	-	99.21	1.0×
Standard TR[43]	TR	99.10	10.5×
PSTRN-M[32]		99.43	16.5×
PSTRN-S[32]		99.51	6.5×
Standard TT[12]	TT	99.07	17.9×
<b>Ours</b>		99.48	<b>17.9×</b>
<b>Ours</b>		<b>99.51</b>	<b>8.3×</b>

Table 1: LeNet-5 on MNIST dataset using different TT/TR-format compression approaches.

Model	Comp. Method	Top-1 (%)	Comp. Ratio
ResNet-20			
Uncompressed	-	91.25	1.0×
Standard TR[43, 32]	TR	87.5	5.4×
TR-RL[5]		88.3	6.8×
PSTRN-M[32]		88.50	6.8×
PSTRN-S[32]		90.80	2.5×
Standard TT[12]	TT	86.7	5.4×
<b>Ours</b>		91.03	<b>6.8×</b>
<b>Ours</b>		<b>91.47</b>	<b>4.5×</b>
ResNet-32			
Uncompressed	-	92.49	1.0×
Standard TR[43]	TR	90.6	5.1×
PSTRN-M[32]		90.6	5.8×
PSTRN-S[32]		91.44	2.7×
Standard TT[12, 43]	TT	88.3	4.8×
<b>Ours</b>		91.96	<b>5.8×</b>
<b>Ours</b>		<b>92.87</b>	<b>4.8×</b>

Table 2: ResNet-20 and ResNet-32 on CIFAR-10 dataset using different TT/TR-format compression approaches.

ADMM-based TT-format model with the uncompressed model as well as the state-of-the-art TT/TR-format works. It is seen that our ADMM-based compression can achieve the highest compression ratio and the best accuracy.

**CIFAR-10.** Table 2 compares our ADMM-based TT-format ResNet-20 and ResNet-32 models with the state-of-the-art TT/TR-format works on CIFAR-10 dataset. For ResNet-20, it is seen that standard training on TT/TR-format models causes severe accuracy loss. Even for the state-of-the-art design using some advanced techniques, such as heuristic rank selection (PSTRN-M/S) and reinforcement learning (TR-RL), the performance degradation is still huge, especially with high compression ratio 6.8×. On the other side, with the same high compression ratio our ADMM-based TT-format model has only 0.22% accuracy drop, which means 2.53% higher than the state-of-the-art PSTRN-M. Furthermore, with moderate compression ratio 4.5× our method can even outperform the uncompressed

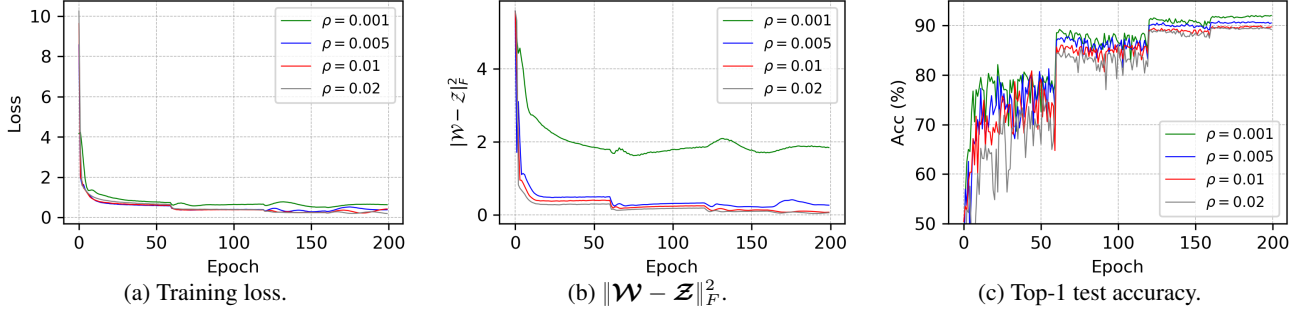


Figure 3: Training loss, Frobenius norm and test accuracy in ADMM-regularized training procedure with different  $\rho$ .

Model	Comp. Method	Top-1 (%)	Comp. Ratio
ResNet-20			
Uncompressed	-	65.4	1.0×
Standard TR[43, 32]	TR	63.55	4.7×
PSTRN-M[32]		63.62	4.7×
PSTRN-S[32]		66.13	2.3×
Standard TT[12]	TT	61.64	5.6×
Ours		64.92	5.6×
Ours		67.36	2.3×
ResNet-32			
Uncompressed	-	68.10	1×
Standard TR[43]	TR	66.70	4.8×
PSTRN-M[32]		66.77	5.2×
PSTRN-S[32]		68.05	2.4×
Standard TT[12, 43]	TT	62.90	4.6×
Ours		67.17	5.2×
Ours		70.31	2.4×

Table 3: ResNet-20 and ResNet-32 on CIFAR-100 dataset using different TT/TR-format compression approaches.

model with 0.22% accuracy increase.

For ResNet-32, again, standard training on compressed models using TT or TR decomposition causes huge performance degradation. The state-of-the-art PSTRN-S/M indeed brings performance improvement, but the test accuracy is still not satisfied. Instead, our highly compressed ( $5.8\times$ ) TT-format model only has 0.53% accuracy loss, which means it has 1.36% higher accuracy than PSTRN-M with the same compression ratio. More importantly, when compression ratio is relaxed to  $4.8\times$ , our ADMM-based TT-format model achieves 92.87%, which is even 0.38% higher than the uncompressed model.

**CIFAR-100.** Table 3 shows the experimental results on CIFAR-100 dataset. Again, our ADMM-based TT-format model outperforms the state-of-the-art work. For ResNet-20, with even higher compression ratio (Our  $5.6\times$  vs  $4.7\times$  in PSTRN-M), our model achieves 1.3% accuracy increase. With  $2.3\times$  compression ratio, our model achieves 67.36% Top-1 accuracy, which is even 1.96% higher than the uncompressed model. For ResNet-32, with the same  $5.2\times$

Model	Comp. Method	Top-5 (%)	FLOPs↓
ResNet-18			
Uncompressed	-	89.08	1.00×
Standard TR[43]	TR	86.29	4.28×
TRP[46]	Matrix SVD	86.74	2.60×
TRP+Nu[46]		86.61	3.18×
DACP[52]	Pruning	87.60	1.89×
FBS[11]		88.22	1.98×
FPGM[21]		88.53	1.72×
DSA[36]		88.35	1.72×
Standard TT[12]	TT	85.64	4.62×
<b>Ours</b>		87.47	<b>4.62×</b>
<b>Ours</b>		<b>89.08</b>	2.47×

Table 4: ResNet-18 on ImageNet dataset using compression approaches. We do not list PSTRN-M/S since [32] does not report results on ImageNet. Also the listed pruning and SVD works do not report compression ratios in their papers. The uncompressed baseline model is from Torchvision. Note that the reported Top-5 accuracy of [11, 21] in this table are obtained from pruning the baselines with higher accuracy.

compression ratio, our approach brings 0.4% accuracy increase over the state-of-the-art PSTRN-M. With the same  $2.4\times$  compression ratio, our approach has 2.26% higher accuracy than PSTRN-S. Our model even outperforms the uncompressed model with 2.21% accuracy increase.

**ImageNet.** Table 4 shows the results of compressing ResNet-18 on ImageNet dataset. Because no prior TT/TR compression works report results on this dataset, we use standard TT and TR-based training in [43, 12] for comparison. We also compare our approach with other compression methods, including pruning and matrix SVD. Since these works report FLOPs reduction instead of compression ratio, we also report FLOPs reduction brought by tensor decomposition. It is seen that with the similar FLOPs reduction ratio ( $4.62\times$ ), our ADMM-based TT-format model has 1.83% and 1.18% higher accuracy than standard TT and TR, respectively. Compared with other compression approaches with non-negligible accuracy loss, our ADMM-

Model	Comp. Method	Top-1 (%)	# Para.	Comp. Ratio
Uncompressed	-	69.7	59M	1.0×
TR-LSTM[38]	TR	86.9	1,725	34.2K×
TT-LSTM[48]	TT	79.6	3,360	17.6K×
<b>Ours</b>		<b>89.0</b>	<b>1,656</b>	<b>35.6K×</b>

Table 5: LSTM on UCF11 dataset using different TT/TR-format compression approaches.

based TT-format models achieve much better accuracy with more FLOPs reduction. In particular, with  $2.47\times$  FLOPs reduction, our model has the same accuracy as the uncompressed baseline model.

### 5.3. Video Recognition

**UCF11.** In this experiment, we use the same uncompressed LSTM model, data pre-processing and experimental settings adopted in [49, 38]. To be consistent with [49, 38], only the ultra-large input-to-hidden layer is compressed for fair comparison. Table 5 compares our ADMM-based TT-format LSTM with the uncompressed model and the existing TT-LSTM [48] and TR-LSTM [38]. Note that [32] does not report the performance of PSTRN-M/S on UCF11 dataset.

From Table 5, it is seen that both TT-LSTM and TR-LSTM provide remarkable performance improvement and excellent compression ratio. As analyzed in [48], such huge improvement over the uncompressed model mainly comes from the excellent feature extraction capability of TT/TR-format LSTM models on the ultra-high-dimensional inputs. Compared with these existing works, our ADMM-based TT-format model achieves even better performance. With fewer parameters, our method brings 2.1% higher top-1 accuracy than the state-of-the-art TR-LSTM.

**HMDB51.** To be consistent with the setting adopted in [32, 38], in this experiment we use the same Inception-V3 as the front-end pre-trained CNN model, and the same back-end uncompressed LSTM model. For fair comparison, we follow the compression strategy adopted in [32, 38] as only compressing the ultra-large input-to-hidden layer of LSTM.

Table 6 summarizes the experimental results. It is seen that comparing with the state-of-the-art TT/TR-format designs, our ADMM-based TT-format model shows excellent performance. With the highest compression ratio ( $84.0\times$ ), our model achieves 64.09% top-1 accuracy. Compared with the state-of-the-art TR-LSTM, our model brings  $3.35\times$  more compression ratio with additional 0.29% accuracy increase.

### 5.4. Discussion on Tensor Format and Generality

**Why Choosing TT-format.** Recently several state-of-the-art tensor decomposition-based compression works [43, 38, 49] report that TT decomposition is inferior to

Model	Comp. Method	Top-1 (%)	# Para.	Comp. Ratio
Uncompressed	-	62.9	16.8M	1.0×
TR-LSTM[38]	TR	63.8	0.67M	25.0×
PSTRN-M[32]		59.67	0.36M	46.7×
PSTRN-S[32]		60.04	0.48M	34.7×
TT-LSTM[48]	TT	62.24	0.67M	25.0×
<b>Ours</b>		<b>64.09</b>	<b>0.20M</b>	<b>84.0×</b>

Table 6: LSTM on HMDB51 dataset using different TT/TR-format compression approaches.

other advanced approach (e.g. TR) on DNN compression, in terms of compression ratio and test accuracy. To fully demonstrate the excellent effectiveness of our approach, in this paper we choose TT, the tensor format that is believed to be not the best for model compression, and adapt the ADMM-regularized compression framework to TT-format. As presented in the experimental results, all the ADMM-based TT-format models consistently outperform the existing TT/TR-format models with higher accuracy and higher compression ratio over different datasets, thereby comprehensively demonstrating the huge benefits brought by our proposed framework.

**Generality of Our Framework.** Although in this paper our focus is to compress TT-format DNN models, because ADMM is a general optimization technique, our proposed framework is very general and can be easily applied for model compression using other tensor decomposition approaches, such as Tensor Ring (TR), Block-term (BT), Tucker etc. To adapt to other tensor decomposition scenario, the main modification on our proposed framework is to modify the Euclidean projection (Algorithm 1) to make the truncating methods being compatible to the corresponding tensor decomposition methods.

## 6. Conclusion

In this paper, we present a systematic compression framework for tensor-format DNNs using ADMM. Under the framework, the tensor decomposition-based DNN model compression is formulated to a nonconvex optimization problem with constraints on target tensor ranks. By performing ADMM to solve this problem, a uncompressed but low tensor-rank model can be obtained, thereby finally bringing the decomposed high-accuracy TT-format model. Extensive experiments for image and video classification show that our ADMM-based TT-format models consistently outperform the state-of-the-art works in terms of compression ratio and test accuracy.

## Acknowledgements

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



## References

- [1] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011. 2, 4, 5
- [2] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 1
- [3] Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH Computer Architecture News*, 44(3):367–379, 2016. 2
- [4] Yuan Cheng, Guangya Li, Ngai Wong, Hai-Bao Chen, and Hao Yu. Deepeye: A deeply tensor-compressed neural network hardware accelerator. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019. 1
- [5] Zhiyu Cheng, Baopu Li, Yanwen Fan, and Yingze Bao. A novel rank selection scheme in tensor ring decomposition based on reinforcement learning for deep neural networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3292–3296. IEEE, 2020. 6
- [6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015. 2
- [7] 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
- [8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009. 6
- [9] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015. 1
- [10] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1933–1941, 2016. 1
- [11] Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. In *International Conference on Learning Representations*, 2019. 7
- [12] Timur Garipov, Dmitry Podoprikin, Alexander Novikov, and Dmitry Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv preprint arXiv:1611.03214*, 2016. 1, 3, 6, 7
- [13] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 1
- [14] Julia Gusk, Maksym Kholiavchenko, Evgeny Ponomarev, Larisa Markeeva, Philip Blagoveschensky, Andrzej Cichocki, and Ivan Oseledets. Automated multi-stage compression of neural networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019. 3
- [15] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. Eie: efficient inference engine on compressed deep neural network. *ACM SIGARCH Computer Architecture News*, 44(3):243–254, 2016. 2
- [16] 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, 2
- [17] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015. 1, 2
- [18] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an “explanatory” multi-modal factor analysis. 1970. 3
- [19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1
- [20] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2234–2240, 2018. 2
- [21] 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 Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 7
- [22] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 2
- [23] Hantao Huang, Leibin Ni, and Hao Yu. Ltnn: An energy-efficient machine learning accelerator on 3d cmos-rram for layer-wise tensorized neural network. In *2017 30th IEEE International System-on-Chip Conference (SOCC)*, pages 280–285. IEEE, 2017. 1
- [24] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014. 3
- [25] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pages 1–12, 2017. 2

- [26] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015. 3
- [27] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6
- [28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1
- [29] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *International Conference on Computer Vision*, pages 2556–2563. IEEE, 2011. 6
- [30] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014. 3
- [31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6
- [32] Nannan Li, Yu Pan, Yaran Chen, Zixiang Ding, Dongbin Zhao, and Zenglin Xu. Heuristic rank selection with progressively searching tensor ring network. *arXiv preprint arXiv:2009.10580*, 2020. 2, 3, 6, 7, 8
- [33] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015. 2
- [34] Jingen Liu, Jiebo Luo, and Mubarak Shah. Recognizing realistic actions from videos “in the wild”. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1996–2003. IEEE, 2009. 6
- [35] 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, 2
- [36] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In *The European Conference on Computer Vision (ECCV)*, 2020. 7
- [37] Alexander Novikov, Dmitrii Podoprikin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015. 1, 3
- [38] 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, 3, 8
- [39] Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavský, 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. 3
- [40] 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, 2
- [41] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 1
- [42] Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in Measuring Change*, 15:122–137, 1963. 3
- [43] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018. 1, 3, 6, 7, 8
- [44] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2074–2082, 2016. 2
- [45] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015. 1
- [46] 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
- [47] Yuhui Xu, Yongzhuang Wang, Aojun Zhou, Weiyao Lin, and Hongkai Xiong. Deep neural network compression with single and multiple level quantization. *arXiv preprint arXiv:1803.03289*, 2018. 1, 2
- [48] 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, 3, 8
- [49] 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. 8
- [50] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–199, 2018. 1, 2
- [51] Hao Zhou, Jose M Alvarez, and Fatih Porikli. Less is more: Towards compact cnns. In *European Conference on Computer Vision*, pages 662–677. Springer, 2016. 2
- [52] Zhuangwei Zhuang, Minghui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018. 2, 7

## Appendix

**Implimentation Details.** All TT operations are implemented by reshape, transpose and matrix multiplication offered by PyTorch.

For ResNet-18 on ImageNet, the original weight tensor of size  $O \times I \times K \times K$  is reshaped and transposed to size  $O \times K^2 \times I$ . The TT-ranks for FLOPs reduction  $2.47\times$  are summarized in Table 7.

Layer	TT-ranks
layer1.0.conv1	[1, 45, 45, 1]
layer1.0.conv2	[1, 36, 36, 1]
layer1.1.conv1	[1, 32, 32, 1]
layer1.1.conv2	[1, 32, 32, 1]
layer2.0.conv1	[1, 80, 40, 1]
layer2.0.conv2	[1, 70, 70, 1]
layer2.1.conv1	[1, 65, 65, 1]
layer2.1.conv2	[1, 64, 64, 1]
layer3.0.conv1	[1, 130, 65, 1]
layer3.0.conv2	[1, 120, 120, 1]
layer3.1.conv1	[1, 120, 120, 1]
layer3.1.conv2	[1, 115, 115, 1]
layer4.0.conv1	[1, 250, 122, 1]
layer4.0.conv2	[1, 240, 240, 1]
layer4.1.conv1	[1, 230, 230, 1]
layer4.1.conv2	[1, 220, 220, 1]

Table 7: TT-ranks for ResNet-18.

For other DNN models, the settings keep similar to prior works.