Deep Convolutional Neural Network Compression via Coupled Tensor Decomposition

Weize Sun, *Member, IEEE,* Shaowu Chen*, Lei Huang, *Senior Member, IEEE,* Hing Cheung So, *Fellow, IEEE*, Min Xie

***Abstract*—Large neural networks have aroused impressive progress in various real world applications. However, the expensive storage and computational resources requirement for running deep networks make them problematic to be deployed on mobile devices. Recently, matrix and tensor decompositions have been employed for compressing neural networks. In this paper, we develop a simultaneous tensor decomposition technique for network optimization. The shared network structure is first discussed. Sometimes, not only the structure but also the parameters are shared to form a compressed model at the expense of degraded performance. This indicates that the weight tensors between layers within one network contain both identical components and independent components. To utilize this characteristic, two new coupled tensor train decompositions are developed for fully and partly structure sharing cases, and an alternating optimization approach is proposed for low rank tensor computation. Finally, we restore the performance of the neural network model by fine-tuning. The compression ratio of the devised approach can then be calculated. Experimental results are also included to demonstrate the benefits of our algorithm for both applications of image reconstruction and classification, using the well known datasets such as Cifar-10/Cifar-100 and ImageNet and widely used networks such as ResNet. Comparing to the state-of-the-art independent matrix and tensor decomposition based methods, our model can obtain a better network performance under the same compression ratio.**

***Index Terms*—Deep Neural Network, Network Compression, Low Rank Approximation, Coupled Tensor Decomposition.**

I. Introduction

In recent years, deep learning has received significant attention, and deep neural networks (DNNs) have been widely applied in various fields of science and engineering, e.g., speech and audio signal processing [1] [2], natural language processing [3] [4], radio signal processing [5], object detection [6], video classification [7], barcode detection [8], synthetic aperture radar [9], emotion recognition [10] and bioinformatics [11] [12]. The success of these deep learning methods mainly rely on the deep network architecture with a huge amount of parameters trained by large-scale datasets and powerful hardware such as GPUs. This leads to the problem of parameter redundancy [13] as well as prohibitive memory and computational resources, and limiting their deployment on resource constrained appliances such as mobile or wearable devices [14]. In order to reduce the redundancy, network compression methods are introduced.

It has been found that the network parameters have low rank property in matrix or tensor structure, and by substituting the weight matrices or tensors using low rank matrix or tensor approximation, the number of parameters can be reduced. In [15], singular value decomposition (SVD) is proposed to compress the DNNs, and Denton $et\ al.$ also included filter clustering to achieve a 2-times ($2\times$) acceleration of the convolutional layers [16], while Jaderberg $et\ al.$ proposed to approximate the convolutional neural networks using rank-1 filters [17]. To achieve better network performance, the sparse characteristic of the network weights is exploited together with the low rank property to approximate the weights [13] under the 'pre-train - decomposition - fine-tune' procedure. Two low rank sub matrices and one sparse matrix are first calculated from the pre-trained weights using iterative optimization, and then fine-tuned to yield a satisfactory model performance. Furthermore, by setting the compression ratio as one objective and image classification error rate as another, [18] proposed to compute the compressed low rank and sparse structure and parameters using multi-objective evolutionary algorithm, and achieving a satisfying Pareto Front [19] result with a good trade-off between network performance and compression ratio. These methods assumed that the network weights before compression contains both low rank and sparse terms. On the other hand, [20] [21] assumed that the compressed low rank network weights are of sparse structure, and suggested approximation methods accordingly to attain a more optimized structure and higher compression ratio.

In convolutional neural networks, the weights are usually 4-dimensional (4-D) tensors. Nevertheless, the above-mentioned methods will fold the tensors into matrices for low rank matrix approximation, and fail to utilize the tensor based network structure. To achieve higher compression, tensor decomposition methods, such as Tucker [22] and CANDECOMP/PARAFAC (CP) [23] decompositions, are developed for compressing DNNs. Inspired by the Kronecker tensor decomposition (KTD), Zhou $et\ al.$ proposed to approximate the weights with a sum of several sub-tensors linked by Kronecker product [24]. These methods mainly rely on the procedure of 'pre-train - decomposition - fine-tune', and many different fine-tuning algorithms are developed. Some research works, on the other hand, propose to substitute one large layer with multiple small layers sequentially, and train the 'low rank approximation of the large layer' from the beginning [25]. Furthermore, the tensor train (TT) decomposition is also

Weize Sun, Shaowu Chen, Lei Huang, and Min Xie are with the Guangdong Key Laboratory of Intelligent Information Processing, College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: proton198601@hotmail.com; shaowu-chen@foxmail.com; lhuang8sasp@hotmail.com; 370558989@qq.com).

H.C. So is with Department of Electrical Engineering, City University of Hong Kong, Hong Kong 999077, China.

The work described in this paper was supported in part by the National Natural Science Foundation of China (NSFC) under Grant U1713217.

(*Corresponding author: Shaowu Chen.)

utilized for network optimization, and a $30\times$ of compression result is attained [26]. Usually, these low rank optimization methods can provide a high compression, but might lead to a performance degradation, especially when the network compression ratio is very high. To alleviate this drawback, random shuffling technique is introduced to restore the effective tensor structure of the weights, and experiments have confirmed that it can be applied in many matrix and tensor based decomposition methods [27].

In order to get satisfying model performance, a deep network model is usually applied. Generally speaking, the network is constructed using the idea of modular design, and thus giving some network layers sharing the same or similar tensor structure, and/or even with the same parameters [28]. In this case, the weight tensors with the same or similar structure might share part of the parameters in the low rank decomposition model. However, the state-of-the-art network optimization methods will compress the weight tensors independently, and fail to explore this property.

To utilize the shared or common information contained in a set of matrices or tensors, simultaneous or coupled matrix and/or tensor decomposition methods are proposed. Using the symmetric property of the matrices, an approximate joint SVD approach is proposed for electroencephalography (EEG) signal analysis [29]. On the other hand, to discover the inner correlation of the toxicology data, a simultaneous non-negative matrix factorization is introduced [30], and its tensor version is used for analyzing the Yelp dataset [31]. Furthermore, the simultaneous CP and Tucker decompositions are developed and applied in many applications such as array signal processing [32], traffic data analysis [33], image fusion [34] and hyperspectral imaging [35]. Although the formulation varies, the basic objective of these coupled decomposition methods is the same, which is to decompose several original tensors simultaneously into a set of sub-tensors, where some sub-tensors are included in all original tensors, and are referred to as the common components, while other sub-tensors are included in one original tensor only, and are referred to as independent components. These coupled decomposition methods perform well in identifying the common information or independent information or both from these original tensors with partly implied common information. However, they have not been applied to deep neural network compression so far. In this paper, we will first extend the idea of coupled factorization to achieve simultaneous TT decomposition. The relationship between the simultaneous TT decomposition and network model compression is then discussed, and two optimization methods, for fully and partly coupled TT decomposition are devised to compress the DNN with fully and partly identical network structures.

The remainder of this paper is organized as follows. Section II introduces the notations, definitions and preliminaries. In Section III, the details of the shared network structure as well as the coupled tensor decomposition are described. The optimization methods for network compression under coupled TT decomposition are then presented in Section IV, and the experimental results are provided in Section V. Finally, conclusions are drawn in Section VI.

II. Notations, Definitions and Preliminaries

The notation used in this paper is introduced as follows. Scalars, vectors, matrices and tensors are denoted by italic, bold lower-case, bold upper-case and bold calligraphic symbols, respectively. The transpose of a vector or matrix is written as ^{T}, and the $i \times i$ identity matrix is symbolized as $\mathbf{I}_i$. To write A as B, we use the symbol $\doteq$, and the inverse and pseudoinverse are represented by $^{-1}$ and †. To refer to the $(m_1, m_2, \ldots, m_R)$ entry of an R-D tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_R}$, we use $\boldsymbol{\mathcal{A}}_{(m_1,m_2,\ldots,m_R)}$. Furthermore, the Frobenius-norm of a tensor $\boldsymbol{\mathcal{A}}$ is defined as $||\boldsymbol{\mathcal{A}}||_2 = \sqrt{\sum_{m_1=1}^{M_1}\sum_{m_2=1}^{M_1}\cdots\sum_{m_R=1}^{M_R}\boldsymbol{\mathcal{A}}^2_{(m_1,m_2,\ldots,m_R)}}$, and the number of elements of $\boldsymbol{\mathcal{A}}$ is defined as $\text{size}(\boldsymbol{\mathcal{A}}) = M_1 \times M_2 \times \cdots \times M_R$. The vectorization operation is written as $\text{vec}(\cdot)$, and the symbol $\sqcup$ represents the concatenation operator where $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}_1 \sqcup_r \boldsymbol{\mathcal{A}}_2$ is obtained by stacking $\boldsymbol{\mathcal{A}}_2 \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_{r-1} \times L_2 \times M_{r+1} \times \cdots \times M_R}$ to the end of $\boldsymbol{\mathcal{A}}_1 \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_{r-1} \times L_1 \times M_{r+1} \times \cdots \times M_R}$ along the r-th dimension. The one vector of dimensions $n \times 1$ is written as $\mathbf{1}_n$.

Now we go on to the definitions and preliminaries.

Definition 2.1. **Unfolding and general unfolding.** The r-th or r-mode **unfolding** of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_R}$ is written as $[\boldsymbol{\mathcal{A}}_{(r)}] \in \mathbb{R}^{M_r \times (M_1 M_2 \cdots M_{r-1} M_{r+1} \cdots M_R)}$, where the r-th dimension of $\boldsymbol{\mathcal{A}}$ is stacked to the first dimension of $[\boldsymbol{\mathcal{A}}_{(r)}]$, while the other dimensions of $\boldsymbol{\mathcal{A}}$ are stacked to the second dimension of $[\boldsymbol{\mathcal{A}}_{(r)}]$ under the order $\{r+1 \to r+2 \to \cdots \to R \to 1 \to \cdots \to r-1\}$ [22]. For example, for $R = 3$, we have $\{[\boldsymbol{\mathcal{A}}_{(1)}]\}_{(m_1,n_2)} = \{\boldsymbol{\mathcal{A}}\}_{(m_1,m_2,m_3)}$ with $n_2 = (m_3 - 1)M_2 + m_2$. Note that this definition is different from that in [36].

The $\{r_1, r_2, \cdots, r_K\}$-mode **general unfolding** of $\boldsymbol{\mathcal{A}}$ is written as $[\boldsymbol{\mathcal{A}}_{(r_1,r_2,\cdots,r_K)}] \in \mathbb{R}^{\prod_{k=1}^{K} M_{r_k} \times \prod_{j=1, j\neq k}^{R} M_j}$ for $1 \le r_1 \le r_2 \le \cdots \le r_K \le R$, and $\{r_1, r_2, \cdots, r_K\}$ is a set selected from $\{1, 2, \cdots, R\}$. It folds the tensor $\boldsymbol{\mathcal{A}}$ by arranging its $\{r_1, r_2, \cdots, r_K\}$ dimensions to the first dimension of the matrix, and the remainder to the second one, following the same order as that in the above unfolding process with $r = r_K$. It is worth noting that the unfolding process is a special case of the general unfolding procedure, and we have $[\boldsymbol{\mathcal{A}}_{(r)}] = [\boldsymbol{\mathcal{A}}_{(1,2,\cdots,r-1,r+1,\cdots,R)}]^T$ and $\text{vec}(\boldsymbol{\mathcal{A}}) = [\boldsymbol{\mathcal{A}}_{(1,2,\cdots,\text{R})}]^T$. Furthermore, we refer to the opposite of the general unfolding process as **general folding**.

The r-mode product [36] of tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_R}$ and matrix $\mathbf{U} \in \mathbb{R}^{N_r \times M_r}$ along the r-th dimension is expressed as $\boldsymbol{\mathcal{B}} = \boldsymbol{\mathcal{A}} \times_r \mathbf{U} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_{r-1} \times N_r \times M_{r+1} \times \cdots \times M_R}$ where $[\boldsymbol{\mathcal{B}}_{(r)}] = \mathbf{U}[\boldsymbol{\mathcal{A}}_{(r)}]$. Now we proceed to the Tucker decomposition.

Definition 2.2. The Tucker decomposition [37] of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_R}$ is

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{S}} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_R \mathbf{U}_R \tag{1}$$

where $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_R}$ is the core tensor satisfying the all-orthogonality and $\mathbf{U}_r \in \mathbb{R}^{M_r \times M_r}, r = 1, 2, \cdots, R$, are the unitary matrices [38] of the r-mode singular vectors.

Recently, a new tensor decomposition, referred to as TT decomposition, is proposed. To better present it, we first define

the TT product $*$ between two 3-D tensors $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{r_1 \times M \times r_2}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{r_2 \times N \times r_3}$ as $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}} \in \mathbb{R}^{r_1 \times M \times N \times r_3}$ where $[\boldsymbol{\mathcal{C}}_{(1,2)}] = [\boldsymbol{\mathcal{A}}_{(1,2)}][\boldsymbol{\mathcal{B}}_{(1)}]$. Note that the TT product follows the associative property, namely, $(\boldsymbol{\mathcal{A}} * \boldsymbol{\mathcal{B}}) * \boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{A}} * (\boldsymbol{\mathcal{B}} * \boldsymbol{\mathcal{C}})$.

Definition 2.3. The TT decomposition [39] of $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_R}$ is

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}}_1 * \boldsymbol{\mathcal{G}}_2 * \cdots * \boldsymbol{\mathcal{G}}_R \tag{2}$$

where $\boldsymbol{\mathcal{G}}_r \in \mathbb{R}^{r_{r-1} \times M_r \times r_r}$ is the R-th core tensor, and r_f for $f = 0, 1, \cdots, R$ is the f-th TT rank, where $r_0 = r_R = 1$. That is, the TT decomposition factorizes the R-D tensor $\boldsymbol{\mathcal{A}}$ to R 3-D core tensors $\boldsymbol{\mathcal{G}}_r$, with $R-1$ ranks $\mathrm{rank}(\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{G}}_1 * \boldsymbol{\mathcal{G}}_2 * \cdots * \boldsymbol{\mathcal{G}}_R) = \{r_1, r_2, \cdots, r_{R-1}\}$. Furthermore, by writing $rank(\boldsymbol{\mathcal{A}}) \leq \{r_1^0, r_2^0, \cdots, r_{R-1}^0\}$, we refer to factorizing $\boldsymbol{\mathcal{A}}$ using (2) under the constraint $r_f \leq r_f^0$ for $f = 1, 2, \cdots, R-1$. It is worth noting that the TT decomposition is first presented in sub-tensors summation format in [39], and here we write it in a sequential TT product form for ease of presentation of the neural network compression techniques in the next section. We can now go on to present the relationship between TT decomposition and low rank network compression in convolutional neural network.

Usually, given an input tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{M_1 \times M_2 \times L_1}$ and output $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{N_1 \times N_2 \times L_2}$, a convolutional layer of a deep neural network is represented by a weight tensor Θ of dimensions $F_1 \times F_2 \times L_1 \times L_2$, where $F_1 \times F_2$ is the filter size, L_1 and L_2 are input and output depths, and the output sizes N_1 and N_2 will also be affected by other parameters like stride and zero padding [40]. To compress the weight tensor, tensor decomposition methods can be applied, and we follow the TT decomposition. However, the tensor Θ is not directly decomposed via TT as F_1 and F_2 are usually small values, for example, 3, and thus leading to a weak low rank property via TT. Here we combine the first and second dimensions of Θ to form a new tensor $\Theta' \in \mathbb{R}^{F \times L_1 \times L_2}$, where $F = F_1 F_2$, and swap the first and second dimensions of Θ' to get an appropriate tensor $\boldsymbol{\mathcal{W}} = g_{\text{fold}}(\Theta) \in \mathbb{R}^{L_1 \times F \times L_2}$, where $g_{\text{fold}}(\dot{)}$ refers to the folding and swapping process, for factorization:

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{G}}_1 * \boldsymbol{\mathcal{G}}_2 * \boldsymbol{\mathcal{G}}_3 \tag{3}$$

Note that according to **Definition 2.3**, the first dimension of $\boldsymbol{\mathcal{G}}_1$ and the third dimension of $\boldsymbol{\mathcal{G}}_3$ are of length 1, and they are in fact matrices. For compact presentation, we will write them as matrices of dimensions $L_1 \times r_1$ and $r_2 \times L_2$, where $\{r_1, r_2\}$ is the TT rank set, and $\boldsymbol{\mathcal{G}}_2 \in \mathbb{R}^{r_1 \times F \times r_2}$.

By decomposing the $\boldsymbol{\mathcal{W}}$ to these 3 core tensors, the original convolutional layer $\Theta \in \mathbb{R}^{F_1 \times F_2 \times L_1 \times L_2}$ is divided into three independent convolutional layers $\Theta_1 \in \mathbb{R}^{1 \times 1 \times L_1 \times r_1}$, $\Theta_2 \in \mathbb{R}^{F_1 \times F_2 \times r_1 \times r_2}$ and $\Theta_3 \in \mathbb{R}^{1 \times 1 \times r_2 \times L_2}$ as shown in Figure 1, where $\Theta_1 = g_{\text{fold}}(\boldsymbol{\mathcal{G}}_1)$ and $\Theta_2 = g_{\text{fold}}(\boldsymbol{\mathcal{G}}_2)$, $\Theta_3 = g_{\text{fold}}(\boldsymbol{\mathcal{G}}_3)$, which means the core tensors from the TT decomposition on $\boldsymbol{\mathcal{W}}$ are in fact three independent convolutional layers, and thus a new network structure is generated. By defining convolutional operator with input $\boldsymbol{\mathcal{X}}$ and filter Θ as $\mathrm{Conv}(\boldsymbol{\mathcal{X}}, \Theta)$, this relationship can be proved as follows.

Proof. For a given input $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{F_1 \times F_2 \times L_1}$ and convolutional layer $\Theta \in \mathbb{R}^{F_1 \times F_2 \times L_1 \times L_2}$, the output $\boldsymbol{\mathcal{Y}} = Conv(\boldsymbol{\mathcal{X}}, \Theta) \in \mathbb{R}^{1 \times 1 \times L_2}$ is $\mathbf{y} = \mathrm{vec}(\boldsymbol{\mathcal{Y}}) = [\Theta_{(4)}]\mathrm{vec}(\boldsymbol{\mathcal{X}}) = [\boldsymbol{\mathcal{W}}_{(3)}]\mathrm{vec}(\mathbf{X}) = \boldsymbol{\mathcal{G}}_3{}^T[(\boldsymbol{\mathcal{G}}_{12})_{(3)}]\mathrm{vec}(\mathbf{X}) = \boldsymbol{\mathcal{G}}_3^T \mathbf{z}_2 \in \mathbb{R}^{\mathrm{L}_2 \times 1}$ where $\boldsymbol{\mathcal{W}} = g_{\text{fold}}(\Theta)$, $\mathbf{X} = g_{\text{fold}}(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{L_1 \times F}$, $\boldsymbol{\mathcal{G}}_{12} = \boldsymbol{\mathcal{G}}_1 * \boldsymbol{\mathcal{G}}_2$ and $\mathbf{z}_2 = [(\boldsymbol{\mathcal{G}}_{12})_{(3)}]\mathrm{vec}(\mathbf{X}) \in \mathbb{R}^{\mathrm{r}_2 \times 1}$.

Suppose $\mathbf{z}_1 = \mathrm{Conv}(\boldsymbol{\mathcal{X}}, \Theta_1)$, $\mathbf{z}_2 = \mathrm{Conv}(\mathbf{z}_1, \Theta_2)$ and $\dot{y} = \mathrm{Conv}(\mathbf{z}_2, \Theta_3)$, which is, $\dot{\boldsymbol{\mathcal{Y}}} = \mathrm{Conv}(\mathrm{Conv}(\mathrm{Conv}(\boldsymbol{\mathcal{X}}, \Theta_1), \Theta_2), \Theta_3)$, then $\mathrm{vec}(\mathbf{z}_1) = \mathrm{vec}(\boldsymbol{\mathcal{G}}_1^T \mathbf{X})$, $\mathrm{vec}(\mathbf{z}_2) = [\boldsymbol{\mathcal{G}}_{2(3)}]\mathrm{vec}(\mathbf{z}_1)$. Defining $\mathbf{G}_{2,i} = \boldsymbol{\mathcal{G}}_2(:, :, i)$, we have

$$\begin{aligned}
\dot{\mathbf{y}} &= \mathrm{vec}(\dot{\boldsymbol{\mathcal{Y}}}) \\
&= \boldsymbol{\mathcal{G}}_3^T \mathrm{vec}(\mathbf{z}_2) \\
&= \boldsymbol{\mathcal{G}}_3^T [\boldsymbol{\mathcal{G}}_{2(3)}] \mathrm{vec}(\mathbf{z}_1) \\
&= \boldsymbol{\mathcal{G}}_3^T [\boldsymbol{\mathcal{G}}_{2(3)}] \mathrm{vec}(\boldsymbol{\mathcal{G}}_1^T \mathbf{X}) \\
&= \boldsymbol{\mathcal{G}}_3^T \begin{bmatrix} \mathrm{vec}(\mathbf{G}_{2,1})^T \\ \mathrm{vec}(\mathbf{G}_{2,2})^T \\ \cdots \\ \mathrm{vec}(\mathbf{G}_{2,\mathrm{r}_2})^T \end{bmatrix} \mathrm{vec}(\boldsymbol{\mathcal{G}}_1^T \mathbf{X}) \\
&= \boldsymbol{\mathcal{G}}_3^T \begin{bmatrix} \mathbf{1}_{r_1}^T \boldsymbol{\mathcal{G}}_1^T \mathbf{X} \mathbf{G}_{2,1}^T \mathbf{1}_{r_I} \\ \mathbf{1}_{r_1}^T \boldsymbol{\mathcal{G}}_1^T \mathbf{X} \mathbf{G}_{2,2}^T \mathbf{1}_{r_I} \\ \cdots \\ \mathbf{1}_{r_1}^T \boldsymbol{\mathcal{G}}_1^T \mathbf{X} \mathbf{G}_{2,\mathrm{r}_2}^T \mathbf{1}_{r_I} \end{bmatrix} \\
&= \boldsymbol{\mathcal{G}}_3^T \begin{bmatrix} \mathrm{vec}(\boldsymbol{\mathcal{G}}_1 \mathbf{G}_{2,1})^T \\ \mathrm{vec}(\boldsymbol{\mathcal{G}}_1 \mathbf{G}_{2,2})^T \\ \cdots \\ \mathrm{vec}(\boldsymbol{\mathcal{G}}_1 \mathbf{G}_{2,\mathrm{r}_2})^T \end{bmatrix} \mathrm{vec}(\mathbf{X}) \\
&= \boldsymbol{\mathcal{G}}_3{}^T [\boldsymbol{\mathcal{G}}_{12(3)}] \mathrm{vec}(\mathbf{X}) \\
&= \mathbf{y}
\end{aligned} \tag{4}$$

therefore $\dot{\boldsymbol{\mathcal{Y}}} = \boldsymbol{\mathcal{Y}}$. □

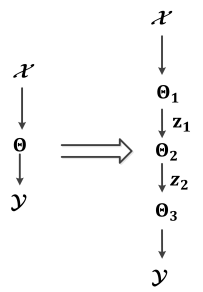

Fig. 1: Compressing one convolutional layer.

It is worth noting that the special 3-D TT (3) can also be represented in Tucker form as $\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{G}}_2 \times_1 \boldsymbol{\mathcal{G}}_1 \times_3 \boldsymbol{\mathcal{G}}_3$ according to (1). Furthermore, other folding and swapping approach can be applied to Θ or Θ' to generate other forms of tensors for TT decomposition. For example, we can produce a new tensor $\Theta'' \in \mathbb{R}^{F \times L_{1,1} \times L_{1,2} \times \cdots \times L_{1,R} \times L_{2,1} \times \cdots \times L_{2,R}}$ from $\Theta' \in \mathbb{R}^{F \times L_1 \times L_2}$ by folding the second and third dimensions of Θ' into R dimensions, respectively. Then the $(2r)$-th and $(2r+1)$-th dimensions of Θ'' can be unfolded to one dimension for $r = 1, 2, \cdots, R$, giving a new tensor

$\mathcal{W}' \in \mathbb{R}^{F\times L_{1,1}L_{2,1}\times L_{1,2}L_{2,2}\times\cdots\times L_{1,R}L_{2,R}}$. Factorizing $\mathcal{W}'$ using TT without the first dimension yields

$$\mathcal{W}' = \mathcal{G}_1 * \mathcal{G}_2 * \cdots * \mathcal{G}_R \tag{5}$$

where $\mathcal{G}_1 \in \mathbb{R}^{F\times L_{1,1}L_{2,1}\times r_1}$ according to (2). Such decomposition might yield an outstanding network compression result under an appropriate selected set of parameters $\{L_{1,r}, L_{2,r}, R\}$ and TT ranks [41]. However, the core tensors generated from $\mathcal{W}'$ might not be able to be regarded as independent convolutional layers except in some special situations. Therefore, $\mathcal{W}'$ is used for the convolution operation instead of $\{\mathcal{G}_r\}$, $r = 1, 2, \cdots, R$. For presentation compactness, we will use (3) in this study. Note that factorizing the weights using (5) is straightforward and thus is not shown here.

III. Shared Network Structure

Usually, in DNN design, similar or consistent subnetwork structure appears several times for better extraction of feature and simplicity of network design. One typical work is the iterative shrinkage-thresholding algorithm network plus (ISTA-Net$^+$) [28], whose structure is shown in Figure 2. This network aims at reconstructing the data $\mathbf{X}$ from its down-sampled data $\mathbf{Y}$. It is initialized by a linear estimation of $\mathbf{X}^{(0)}$ from $\mathbf{Y}$, and then $\mathbf{X}^{(0)}$ is fed into the DNN. By applying the idea of iterative optimization, ISTA-Net$^+$ optimizes the reconstruction result several times, and for each time, a small but consistent network is used. In other words, this DNN consists of N subnetworks with exactly the same structure, and the weight tensors are written as $\mathcal{W}_{n,m}$, where $m = 1, 2, \cdots, M$, and M is the number of layers in one subnetwork. For this DNN, we have $\text{size}(\mathcal{W}_{n_1,m}) = \text{size}(\mathcal{W}_{n_2,m})$ for $n_1, n_2 = 1, 2, \cdots, N$, where $\text{size}(\mathcal{W}_{n_1,m})$ corresponds to the dimensions of $\mathcal{W}_{n,m}$, and this can be referred to as shared network structure.

This shared network structure can also be observed in other network models. In the well-known ResNet model [42], a structure like Figure 3 can usually be observed. In this case, we might have $\text{size}(\mathcal{W}_{n_1,m}) = \text{size}(\mathcal{W}_{n_2,m})$ for $n_1, n_2 = 1, 2, \cdots, N$, $m = 1, 2$. However, the weights in different blocks under the same m are different, and the effect of making them equal to each other has not been discussed in the literature yet.

Usually, to compress a DNN, a low rank decomposition is applied to the layers of the DNN independently, without considering the property of the shared network structure and/or weights, leading to an unsatisfactory performance. It is worth noting that for the ISTA-Net$^+$ [28], not only the structure but also the weights can be shared, which means that we have $\mathcal{W}_{n_1,m} = \mathcal{W}_{n_2,m} = \mathcal{W}_m$ for any n_1, n_2, and $m = 1, 2, \cdots, M$. Comparing to $\mathcal{W}_{n_1,m} \neq \mathcal{W}_{n_2,m}$ for any $n_1 \neq n_2$ case, the former will have worse reconstruction performance but only $1/N$ of parameters, which is a good compression result when N is sufficiently large. Therefore, it is straightforward to assume that the weight tensors contain both identical part or common components and non-identical part or independent components, which is:

$$\mathcal{W}_{n,m} = \mathcal{W}_m + \mathcal{W}'_{n,m} \tag{6}$$

where $\mathcal{W}_m$ represents the common component of the weights and $\mathcal{W}'_{n,m}$ is the independent component. Using $\mathcal{W}_{n,m}$, that is, both $\mathcal{W}_m$ and $\mathcal{W}'_{n,m}$, will give a better network performance than using $\mathcal{W}_m$ only, but resulting in N times of parameters or weights in the DNN. Here we propose to compress the tensors with the novel coupled TT decomposition:

Definition 3.1. The coupled TT decomposition of 3-D tensors $\mathcal{W}_{n,m}$ is

$$\mathcal{W}_{n,m} = \mathcal{G}_{m,1} * \mathcal{G}_{m,2} * \mathcal{G}_{m,3} + \mathcal{G}_{n,m,1} * \mathcal{G}_{n,m,2} * \mathcal{G}_{n,m,3} \tag{7}$$

for $n = 1, 2, \cdots, N$, where $\{\mathcal{G}_{m,i} \in \mathbb{R}^{r_{m,i-1}\times M_r\times r_{m,i}}\}_{i=1}^3$ are identical components and $\{\mathcal{G}_{n,m,i} \in \mathbb{R}^{r_{n,m,i-1}\times M_r\times r_{n,m,i}}\}_{i=1}^3$ are independent components. Note that $r_{m,0} = r_{m,3} = r_{n,m,0} = r_{n,m,3} = 1$. The formulation for r-D ($r > 3$) cases is straightforward and thus is not shown here. Furthermore, when only part of dimensions of $\mathcal{W}_{n,m}$ are the same, a modification is required case by case. In next section, we propose two joint tensor factorization methods under the coupled TT decomposition to make use of the shared structure in network optimization.

IV. Main Results

In this section, simultaneous or coupled tensor factorization methods are proposed for the optimization of deep convolutional neural networks with shared structure and/or weights. The mathematical models of the coupled tensor factorization and the network structure after the decomposition are first introduced, and then the algorithms for the network compression will be derived.

A. Coupled Tensor Decomposition for Network Compression

In this section, the problem of coupled tensor decomposition for network optimization is introduced. The task is to approximate the tensors $\{\mathcal{W}_{n,m}\}$, $n = 1, 2, \cdots, N$, $m = 1, 2, \cdots, M$ using coupled TT decomposition and the $\{\mathcal{W}_{n,m}\}$ are referred to as original or baseline network weights. Note that we follow the definition in (3) and write $\mathcal{W}_{n,m} = g_{\text{fold}}(\Theta_{n,m} \in \mathbb{R}^{F_{n,m,1}\times F_{n,m,2}\times L_{n,m,in}\times L_{n,m,out}}) \in \mathbb{R}^{L_{n,m,in}\times F_{n,m}\times L_{n,m,out}}$, where $F_{n,m} = F_{n,m,1}F_{n,m,2} \doteq M_2$ is the filter size, $L_{n,m,in} \doteq M_1$ and $L_{n,m,out} \doteq M_3$ are input and output depths of the layer. The actual convolution is operated in $\Theta_{n,m}$ but we write $\mathcal{W}_{n,m}$ for compact presentation. According to Figures 2 and 3, $\{\mathcal{W}_{n,m}\}$ can be pre-trained using the training dataset, and we have $\mathcal{W}_{n_1,m} \neq \mathcal{W}_{n_2,m}$ when $n_1 \neq n_2$, that is, only the structure but not weights are shared. Furthermore, the network in Figures 2 or 3 is trained under the shared weights, and the weight tensors $\{\mathcal{W}_{n,m,\text{shared}}\}$, where $\mathcal{W}_{n_1,m,\text{shared}} = \mathcal{W}_{n_2,m,\text{shared}} = \mathcal{W}_m$ for $n_1, n_2 = 1, 2, \cdots, N$, can be computed. Once $\mathcal{W}_{n,m}$ and $\mathcal{W}_m$ in (6)

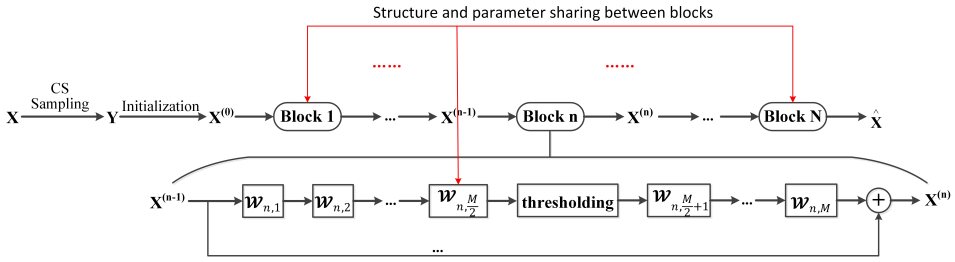

Fig. 2: Illustration of structure and parameter sharing in ISTA-Net$^+$.

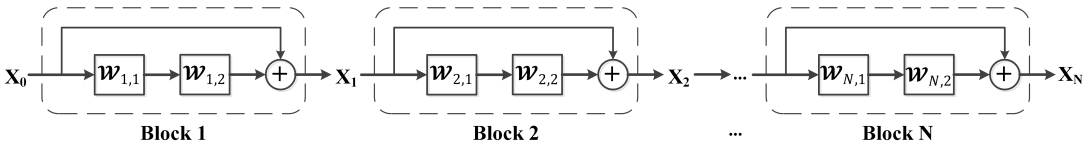

Fig. 3: A common shared network structure in ResNet with $M = 2$.

are calculated, the coupled TT decomposition is applied for network compression:

$$\begin{aligned} \min_{\{\boldsymbol{\mathcal{G}}_{m,i},\boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=1}^{3}} \quad & ||\boldsymbol{\mathcal{W}}_{n,m} - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3} \\ & - \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}||_2^2 \qquad (8) \\ for \quad & n = 1, 2, \cdots, N \\ s.t. \quad & ||\boldsymbol{\mathcal{W}}_m - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}||_2^2 < \epsilon \\ & rank(\boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}) \leq \{r_{m,1}^0, r_{m,2}^0\} \\ & rank(\boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}) \\ & \leq \{r_{n,m,1}^0, r_{n,m,2}^0\} \end{aligned}$$

where $\{\boldsymbol{\mathcal{G}}_{m,i} \in \mathbb{R}^{r_{m,i-1} \times M_r \times r_{m,i}}\}_{i=1}^3$ and $\{\boldsymbol{\mathcal{G}}_{n,m,i} \in \mathbb{R}^{r_{n,m,i-1} \times M_r \times r_{n,m,i}}\}_{i=1}^3$ are defined in (7). The basic principle of this formulation is to approximate the weight tensors $\boldsymbol{\mathcal{W}}_{n,m}$ by $\{\boldsymbol{\mathcal{G}}_{m,i}, \boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=1}^3$ with low TT ranks $\{r_{m,1}, r_{m,2}\} \leq \{r_{m,1}^0, r_{m,2}^0\}$, $\{r_{n,m,1}, r_{n,m,2}\} \leq \{r_{n,m,1}^0, r_{n,m,2}^0\}$ and $\boldsymbol{\mathcal{W}}_m \approx \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}$ for supplement of shared structure and weights computing. To illustrate the new network structure, we show the optimized DNN under $N = 2$ and $M = 1$ in Figure 4 as an example.

However, the optimization of the rank is an NP-hard problem. Assuming that the ranks $\{r_{m,1}, r_{m,2}\}$ and $\{r_{n,m,1}, r_{n,m,2}\}$ are known and according to the augmented Lagrangian method [43] [44], we relax (8) to:

$$\begin{aligned} \min_{\{\boldsymbol{\mathcal{G}}_{m,i},\boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=1}^{3}} \quad & \alpha||\boldsymbol{\mathcal{W}}_m - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}||_2^2 \\ & + \frac{\lambda}{N}\sum_{n=1}^{N} ||\boldsymbol{\mathcal{W}}_{n,m} - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3} \\ & - \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}||_2^2 \qquad (9) \end{aligned}$$

A joint minimization of $\{\boldsymbol{\mathcal{G}}_{m,i}, \boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=1}^3$ will be highly inefficient and thus we propose to optimize the parameters alternatively as follows.

In the k-th iteration, given $\boldsymbol{\mathcal{G}}_{m,i}^{(k-1)}$, the problem is reduced to

$$\min_{\{\boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=1}^{3}} ||\boldsymbol{\mathcal{W}}'_{n,m} - \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}||_2^2 \qquad (10)$$

where $\boldsymbol{\mathcal{W}}'_{n,m} = \boldsymbol{\mathcal{W}}_{n,m} - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}$ for $n = 1, 2, \cdots, N$ and $m = 1, 2, \cdots, M$, and the new $\boldsymbol{\mathcal{G}}_{n,m,i}^{(k)}$ are updated from the previous iteration results $\boldsymbol{\mathcal{G}}_{n,m,i}^{(k-1)}$ for $i = 1, 2, 3$. The optimization problem can be solved by TT-SVD or iteratively as follows. By writing $\boldsymbol{\mathcal{G}}_{n,m,(23)}^{(k-1)} = \boldsymbol{\mathcal{G}}_{n,m,2}^{(k-1)} * \boldsymbol{\mathcal{G}}_{n,m,3}^{(k-1)} \in \mathbb{R}^{r_{m,1} \times M_2 \times M_3}$, the problem (10) becomes $\min_{\boldsymbol{\mathcal{G}}_{n,m,1}^{(k)}} |||[\boldsymbol{\mathcal{W}}'_{n,m_{(1)}}] - [\boldsymbol{\mathcal{G}}_{n,m,1_{(1,2)}}^{(k)}][\boldsymbol{\mathcal{G}}_{n,m,(23)_{(1)}}^{(k-1)}]||_2^2$ whose solution is:

$$[\boldsymbol{\mathcal{G}}_{n,m,1_{(1,2)}}^{(k)}] = [\boldsymbol{\mathcal{G}}'_{n,m_{(1)}}]\mathbf{C}_3^T(\mathbf{C}_3\mathbf{C}_3^T)^{-1} \qquad (11)$$

where $\mathbf{C}_3 \doteq [\boldsymbol{\mathcal{G}}_{n,m,(23)_{(1)}}^{(k-1)}]$.

To update $\boldsymbol{\mathcal{G}}_{n,m,1}$, we rewrite (10) as $\min_{\boldsymbol{\mathcal{G}}_{n,m,2}^{(k)}} ||\boldsymbol{\mathcal{W}}'_{n,m} - \boldsymbol{\mathcal{G}}_{n,m,2}^{(k)} \times_1 \boldsymbol{\mathcal{G}}_{n,m,1}^{(k)} \times_3 \boldsymbol{\mathcal{G}}_{n,m,3}^{(k-1)}||_2^2$ and realize that it can be solved by $\boldsymbol{\mathcal{G}}_{n,m,1}^{(k)} = \boldsymbol{\mathcal{W}}'_{n,m} \times_1 \boldsymbol{\mathcal{G}}_{n,m,1}^{(k)\dagger} \times_3 \boldsymbol{\mathcal{G}}_{n,m,3}^{(k-1)\dagger}$, which is

$$\boldsymbol{\mathcal{G}}_{n,m,2}^{(k)} = \boldsymbol{\mathcal{G}}_{n,m,1}^{(k)\dagger} * \boldsymbol{\mathcal{W}}'_{n,m} * \boldsymbol{\mathcal{G}}_{n,m,3}^{(k-1)\dagger} \qquad (12)$$

Similar to (11), $\boldsymbol{\mathcal{G}}_{n,m,3}^{(k)}$ is calculated as

$$[\boldsymbol{\mathcal{G}}_{n,m,3_{(1)}}^{(k)}] = (\mathbf{C}_1^T\mathbf{C}_1)^{-1}\mathbf{C}_1^T[\boldsymbol{\mathcal{W}}'_{n,m_{(1,2)}}] \qquad (13)$$

where $\mathbf{C}_1 \doteq [(\boldsymbol{\mathcal{G}}_{n,m,1}^{(k)} * \boldsymbol{\mathcal{G}}_{n,m,2}^{(k)})_{(1,2)}]$.

Given $\boldsymbol{\mathcal{G}}_{n,m,i}^{(k-1)}$ and assigning $\alpha \leftarrow \alpha/\lambda$, the problem (9) becomes

$$\begin{aligned} \min_{\{\boldsymbol{\mathcal{G}}_{m,i}\}_{i=1}^{3}} \quad & \alpha||\boldsymbol{\mathcal{W}}_m - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}||_2^2 \\ & + \frac{1}{N}\sum_{n=1}^{N} ||\boldsymbol{\mathcal{W}}''_{n,m} - \boldsymbol{\mathcal{G}}_{m,1} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}||_2^2 \qquad (14) \end{aligned}$$

where $\boldsymbol{\mathcal{W}}''_{n,m} = \boldsymbol{\mathcal{W}}_{n,m} - \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}$. Taking the derivative of (14), setting the result to zero, and following

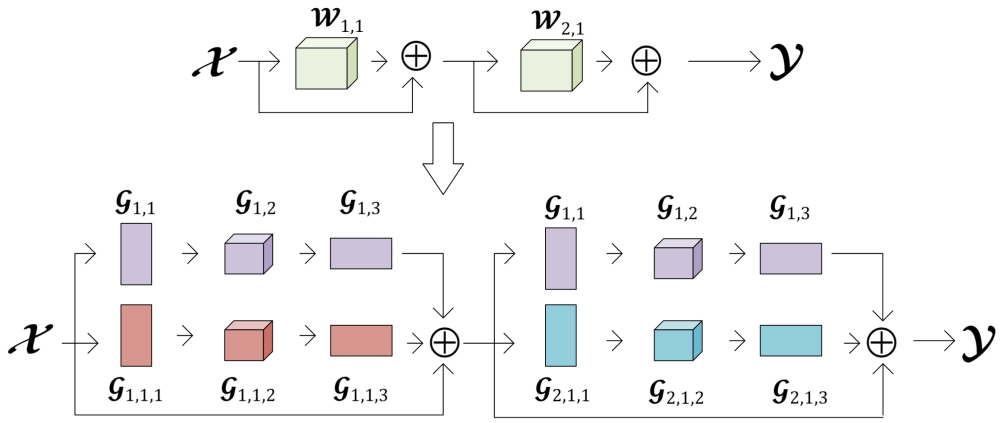

Fig. 4: Optimizing DNN via coupled tensor decomposition.

the same procedure as in (11) to (13), $\{\mathcal{G}_{m,i}^{(k)}\}_{i=1}^{3}$ are updated alternatively as

$$[\mathcal{G}_{m,1}^{(k)}{}_{(1,2)}] = [\mathcal{W}'_{m(1)}]\mathbf{C}_3'^{T}(\mathbf{C}_3'\mathbf{C}_3'^{T})^{-1} \tag{15}$$

$$\mathcal{G}_{m,2}^{(k)} = \mathcal{G}_{m,1}^{(k)\ \dagger} * \mathcal{W}'_m * \mathcal{G}_{m,3}^{(k-1)\dagger} \tag{16}$$

$$[\mathcal{G}_{m,3}^{(k)}{}_{(1)}] = (\mathbf{C}_1'^{T}\mathbf{C}_1')^{-1}\mathbf{C}_1'^{T}[\mathcal{W}'_{m(1,2)}] \tag{17}$$

where $\mathbf{C}_3' \doteq [\mathcal{G}_{m,(23)}^{(k-1)}{}_{(1)}]$, $\mathbf{C}_1' \doteq [(\mathcal{G}_{m,1}^{(k)} * \mathcal{G}_{m,2}^{(k)})_{(1,2)}]$, and $\mathcal{W}'_m = \frac{1}{\alpha+1}(\alpha\mathcal{W}_m + \frac{1}{N}\sum_{n=1}^{N}\mathcal{W}''_{n,m})$. Then $\{\mathcal{G}_{m,i}, \mathcal{G}_{n,m,i}\}_{i=1}^{3}$ can be computed alternatively with random initialization or TT-SVD [39] initialization, until a stopping criterion is reached.

Once the new weight tensors are computed, they can be fed back to the new neural network, which is generated from the original network with weights $\mathcal{W}_{n,m}$ according to Figure 4, and a compressed network is obtained. However, the concept of this optimization is to approximate the weight tensors $\mathcal{W}_{n,m}$ with low rank subtensors $\{\mathcal{G}_{m,i}, \mathcal{G}_{n,m,i}\}_{i=1}^{3}$, and it is different from the original target of the DNN, which might be aiming at image reconstruction [28] or classification [42]. Therefore, a fine-tuning step is required based on the new neural network structure with $\{\mathcal{G}_{m,i}, \mathcal{G}_{n,m,i}\}_{i=1}^{3}$ from the above mentioned optimization being the initial values for network training [13]. The whole Network Compression via fully Coupled Tensor Decomposition (NC-CTD) algorithm for the fully identical tensors is now summarized in Algorithm 1. It is worth noting that both $\mathcal{W}_m$ and $\mathcal{W}_{n,m}$ are pre-trained under the DNN before compression, with or without weight sharing. In some cases, the pre-trained $\mathcal{W}_m$ might not be observed, for example, in large neural networks, training $\mathcal{W}_{n,m}$ might take several days and thus training $\mathcal{W}_m$ again will be highly inefficient. It those cases, we simply assign $\alpha = 0$ in Step 3 of Algorithm 1. The compression ratio of the proposed method is now studied. Given a DNN with N blocks and there are M layers in each block, and weights $\mathcal{W}_{n,m} = g_{\text{fold}}(\Theta_{n,m} \in \mathbb{R}^{F_{n,m,1}\times F_{n,m,2}\times L_{n,m,in}\times L_{n,m,out}}) \in \mathbb{R}^{M_{m,1}\times M_{m,2}\times M_{m,3}}$, the number of parameters in the original network is

Algorithm 1: NC-CTD algorithm.

Input: The DNN to be compressed and its weights $\mathcal{W}_m$ and $\mathcal{W}_{n,m}$, for $n = 1, 2, \cdots, N$, $m = 1, 2, \cdots, M$:

Step 1: Initialize $\{\mathcal{G}_{m,i}^{(0)}, \mathcal{G}_{n,m,i}^{(0)}\}_{i=1}^{3}$ via TT-SVD.

do

Step 2: Update $\{\mathcal{G}_{n,m,i}^{(k)}\}_{i=1}^{3}$ according to (11) – (13).

Step 3: Update $\{\mathcal{G}_{m,i}^{(k)}\}_{i=1}^{3}$ according to (15) – (17).

until a maximum number of K iterations is reached.

Step 4: Generate the new network structure according to Figure 4, and fine-tune the weights $\{\mathcal{G}_{m,i}, \mathcal{G}_{n,m,i}\}_{i=1}^{3}$. Output the results.

$$P_o = N(\sum_{m=1}^{M} \#\mathcal{W}_{n,m}) = N(\sum_{m=1}^{M} M_{m,1}M_{m,2}M_{m,3}) \tag{18}$$

where $\#\mathcal{W}_{n,m}$ is the number of parameters in $\mathcal{W}_{n,m}$. The parameter number after compression is

$$\begin{aligned} P_c &= \sum_{m=1}^{M}\sum_{i=1}^{3} \#\mathcal{G}_{m,i} + \sum_{n=1}^{N}\sum_{m=1}^{M}\sum_{i=1}^{3} \#\mathcal{G}_{n,m,i} \\ &= \sum_{m=1}^{M}(M_1 r_{m,1} + M_2 r_{m,1} r_{m,2} + M_3 r_{m,2}) + \\ &\quad \sum_{n=1}^{N}\sum_{m=1}^{M}(M_1 r_{n,m,1} + M_2 r_{n,m,1} r_{n,m,2} + M_3 r_{n,m,2}) \end{aligned} \tag{19}$$

Then the compression ratio (CR) is

$$CR = \frac{P_o}{P_c} \tag{20}$$

B. Modification for Partly Coupled Tensor Decomposition

In the previous section, it is assumed that $\text{size}(\mathcal{W}_{n_1,m}) = \text{size}(\mathcal{W}_{n_2,m})$ for $n_1, n_2 = 1, 2, \cdots, N$. However, this assumption might be conflicted in some layers in DNNs. For example, to solve the image classification problem, deep networks are employed. However, the depth of the input image is 3 because of the RGB channels. To achieve better feature extraction performance, the depth of the network layers is increased layer by layer or block by block, making the size

of weight tensors different in the connection point when the depth changes. Such technique is widely used in CNNs. For example, for the ResNet-34 model [42], the size of the 8-th and the 10-th layers are of dimensions $3 \times 3 \times 64 \times 128$ and $3 \times 3 \times 128 \times 128$. Therefore, they cannot be decomposed by the coupled decomposition method in Algorithm 1.

Generally speaking, the problem can be modeled as compressing the network weights $\boldsymbol{\mathcal{W}}_{n,m} \in \mathbb{R}^{M_{n,m,1} \times M_{m,2} \times M_{m,3}}$ for $n = 1, 2, \cdots, N$. The $M_{n,m,1}$, $M_{m,2}$, $M_{m,3}$ are input depth, filter size and output depth, respectively. This time, the input depths of different $\boldsymbol{\mathcal{W}}_{n,m}$ are different, and we propose to decompose these partly identical tensors using the partly coupled tensor decomposition similarly to **Definition 3.1** as:

$$\begin{aligned}\boldsymbol{\mathcal{W}}_{n,m} &\approx \boldsymbol{\mathcal{W}}^{(c)}_{n,m} + \boldsymbol{\mathcal{W}}'_{n,m} \\ &= \boldsymbol{\mathcal{P}}_{n,m} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3} + \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}\end{aligned} \tag{21}$$

where $\boldsymbol{\mathcal{W}}'_{n,m} = \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}$ is the independent component as in (7). For each $\boldsymbol{\mathcal{W}}^{(c)}_{n,m}$, an independent $\boldsymbol{\mathcal{P}}_{n,m}$ and shared $\boldsymbol{\mathcal{G}}_{m,2}$ and $\boldsymbol{\mathcal{G}}_{m,3}$ are calculated, thus it is referred to as partly coupled tensor decomposition. It is worth noting that a common $\boldsymbol{\mathcal{W}}_m$ cannot be constructed in this case. Furthermore, for some DNNs, the weight tensor might have the same input depth but different output depths. A similar decomposition can be applied and the modification is straightforward, and thus is not discussed here.

Similar to (9), the network compression problem (21) is formulated as:

$$\begin{aligned}\min_{\{\boldsymbol{\mathcal{G}}_{m,i}, \boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=1}^{3}} \frac{1}{N} \sum_{n=1}^{N} ||\boldsymbol{\mathcal{W}}_{n,m} - \boldsymbol{\mathcal{P}}_{n,m} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3} \\ - \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3}||_2^2\end{aligned} \tag{22}$$

and can be solved by alternating optimization.

In the k-th iteration, given $\boldsymbol{\mathcal{P}}^{(k-1)}_{n,m}$, $\boldsymbol{\mathcal{G}}^{(k-1)}_{m,2}$ and $\boldsymbol{\mathcal{G}}^{(k-1)}_{m,3}$, the $\boldsymbol{\mathcal{G}}^{(k)}_{n,m,i}$ are calculated according to (11)–(13). Then (22) becomes

$$\min_{\{\boldsymbol{\mathcal{P}}_{n,m}, \boldsymbol{\mathcal{G}}_{m,2}, \boldsymbol{\mathcal{G}}_{m,3}\}} ||\boldsymbol{\mathcal{W}}''_m - \boldsymbol{\mathcal{P}}_{n,m} * \boldsymbol{\mathcal{G}}_{m,2} * \boldsymbol{\mathcal{G}}_{m,3}||_2^2 \tag{23}$$

where

$$\boldsymbol{\mathcal{W}}''_m = \boldsymbol{\mathcal{W}}''_{1,m} \sqcup_1 \boldsymbol{\mathcal{W}}''_{2,m} \sqcup_1 \cdots \sqcup_1 \boldsymbol{\mathcal{W}}''_{N,m} \tag{24}$$

$$\boldsymbol{\mathcal{W}}''_{n,m} = \boldsymbol{\mathcal{W}}_{n,m} - \boldsymbol{\mathcal{G}}_{n,m,1} * \boldsymbol{\mathcal{G}}_{n,m,2} * \boldsymbol{\mathcal{G}}_{n,m,3} \tag{25}$$

$$\boldsymbol{\mathcal{P}}_m = \boldsymbol{\mathcal{P}}_{1,m} \sqcup_1 \boldsymbol{\mathcal{P}}_{2,m} \sqcup_1 \cdots \sqcup_1 \boldsymbol{\mathcal{P}}_{N,m}. \tag{26}$$

The new $\boldsymbol{\mathcal{P}}^{(k)}_{n,m}$, $\boldsymbol{\mathcal{G}}^{(k)}_{m,2}$ and $\boldsymbol{\mathcal{G}}^{(k)}_{m,3}$ can be computed following the same steps in (15)–(17) or TT-SVD. To estimate the low rank sub-tensors, an alternating optimization approach as in Algorithm 1 can be employed, and the network structure of the whole algorithm, referred to as Network Compression via Partly Coupled Tensor Decomposition (NC-PCTD), is now summarized in Algorithm 2. As an example, Figure 5 illustrates the NC-PCTD algorithm using $N = 2$ and $M = 1$.

Algorithm 2: NC-PCTD algorithm.

Input: The DNN to be compressed and its weights $\boldsymbol{\mathcal{W}}_{n,m}$, for $n = 1, 2, \cdots, N$, $m = 1, 2, \cdots, M$:
Step 1: Initialize $\boldsymbol{\mathcal{P}}^{(0)}_{n,m}$, $\{\boldsymbol{\mathcal{G}}^{(0)}_{m,i}\}_{i=2}^{3}$, $\{\boldsymbol{\mathcal{G}}^{(0)}_{n,m,i}\}_{i=1}^{3}$ via TT-SVD.
do
Step 2: Update $\{\boldsymbol{\mathcal{G}}^{(k)}_{n,m,i}\}_{i=1}^{3}$ according to (11) – (13).
Step 3: Update $\{\boldsymbol{\mathcal{G}}^{(k)}_{m,i}\}_{i=0}^{3}$ with $\alpha = 0, \boldsymbol{\mathcal{W}}'_m = \boldsymbol{\mathcal{W}}''_m$ according to (15) – (17).
Step 4: Obtain $\{\boldsymbol{\mathcal{P}}^{(k)}_{n,m}\}_{n=1}^{N}$ by reversing the concatenation of (26).
until a maximum number of K iterations is reached.
Step 5: Generate the new network structure according to Figure 5, and fine-tune the weights $\boldsymbol{\mathcal{P}}_{n,m}$, $\{\boldsymbol{\mathcal{G}}_{m,i}\}_{i=2}^{3}$, $\{\boldsymbol{\mathcal{G}}_{n,m,i}\}_{i=2}^{3}$. Output the results.

Finally, fine-tuning is carried out under the new network structure initialized by the optimized $\boldsymbol{\mathcal{P}}_{n,m}$, $\boldsymbol{\mathcal{G}}_{m,2}$, $\boldsymbol{\mathcal{G}}_{m,3}$ and $\boldsymbol{\mathcal{G}}_{n,m,i}$ for better network performance.

Similar to Section IV-A, the parameter number after compression is

$$\begin{aligned}P_{c2} &= \sum_{m=1}^{M} \sum_{i=2}^{3} \#\boldsymbol{\mathcal{G}}_{m,i} + \sum_{n=1}^{N} \sum_{m=1}^{M} (\#\boldsymbol{\mathcal{G}}_{n,m} + \sum_{i=1}^{3} \#\boldsymbol{\mathcal{G}}_{n,m,i}) \\ &= \sum_{m=1}^{M} (M_{m,2} r_{m,1} r_{m,2} + M_{m,3} r_{m,2}) + \sum_{n=1}^{N} \sum_{m=1}^{M} \\ &\quad [M_{n,m,1}(r_{n,m} + r_{n,m,1}) + M_2 r_{n,m,1} r_{n,m,2} + M_3 r_{n,m,2}]\end{aligned} \tag{27}$$

where $\{r_{m,1}, r_{m,2}\}$, $\{r_{n,m,1}, r_{n,m,2}\}$ are the TT ranks, and the compression ratio of the NC-PCTD approach is

$$CR = \frac{\sum_{m=1}^{M} \sum_{n=1}^{N} \#\boldsymbol{\mathcal{W}}_{n,m}}{P_{c2}} \tag{28}$$

V. Experimental Results

We evaluate our proposed network compression methods across several popular neural architectures on both image reconstruction and image classification benchmarks:

- **ISTA-Net**$^+$ [28] for image compressive sensing (CS) and reconstruction.
- **ResNet-18** [42] for image classification on Cifar-10 dataset [45]. Then it is modified to ResNet-8 for the evaluation of the NC-PCTD approach only.
- **ResNet-34** [42] for image classification on ImageNet dataset [46]. Then the compressed network models are transferred to Cifar-10 and Cifar-100 datasets under the standard transfer learning technique [47] to test the compatibility of the compressed model.

All of the following experiments are based on Tensorflow [48] and with Adam optimizer [49] in the back propagation (BP) training. Other training techniques are not included here. The approach using independent TT decomposition for network compression is included for comparison, and note that in this special 3-D case, it becomes the Tucker compression approach [22]. The compression method via SVD [15], which decomposes a convolutional layer of dimensions $F_1 \times F_2 \times L_1 \times L_2$ into two sub-layers $F_1 \times 1 \times L_1 \times r$ and $1 \times F_2 \times r \times L_2$, is also used in some experiments. Note that the SVD can be regarded as a special 2-D TT decomposition.

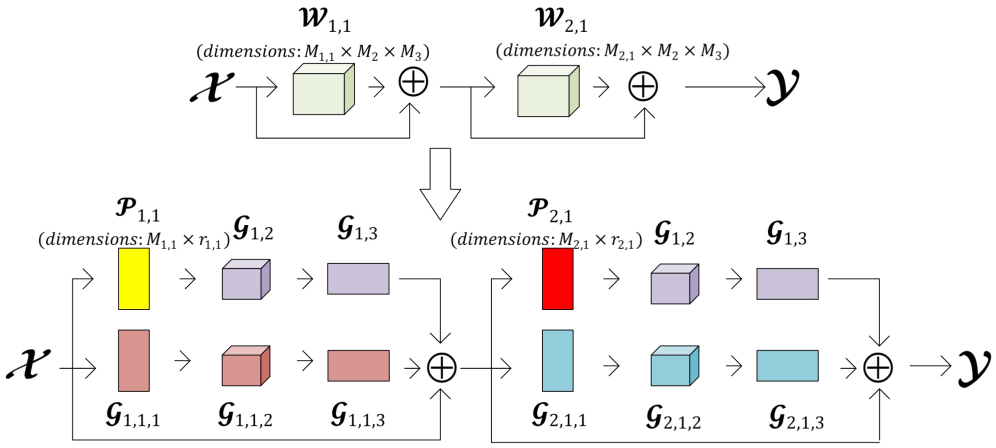

Fig. 5: Optimizing DNN via partly coupled tensor decomposition.

A. Image Reconstruction

For the image CS and reconstruction problem, we consider the dataset of 91 images in [50] for network training, and during the training, we follow the common CS and reconstruction practice as in [50] [51] and split the images to small sub-images $\boldsymbol{\mathcal{X}}$ of dimensions $33 \times 33 = 1089$. Then, for one CS ratio $CS_{\text{ratio}} \in [0.01, 0.5]$, a corresponding orthogonal Gaussian measurement matrix is generated to sample the sub-image $\boldsymbol{\mathcal{X}}$ to get the observation $\mathbf{y}$, where $\text{size}(\mathbf{y}) = (1089 CS_{\text{ratio}})$. Then the initial $\boldsymbol{\mathcal{X}}^{(0)}$ is generated using linear mapping from $\mathbf{y}$, and fed into the DNN for final result reconstruction. The full network structure is shown in Figure 2, and two cases, $(N, M) = (5, 6)$ and $(N, M) = (7, 6)$ are considered. The dimensions of all the layers are $3 \times 3 \times 128 \times 128$ except the first and the last layers, which are $\boldsymbol{\mathcal{W}}_{n,1} \in \mathbb{R}^{3\times3\times1\times128}$ and $\boldsymbol{\mathcal{W}}_{n,6} \in \mathbb{R}^{3\times3\times128\times1}$, of each block $n = 1, 2, \cdots, N$. As the numbers of parameters in $\boldsymbol{\mathcal{W}}_{n,1}$ and $\boldsymbol{\mathcal{W}}_{n,6}$ are small, they will not be compressed. Furthermore, for $\boldsymbol{\mathcal{W}}_{n,m}$, $m \neq 1, 6$, two weights, namely, $\boldsymbol{\mathcal{W}}_{n_1,m} \neq \boldsymbol{\mathcal{W}}_{n_2,m}$ without parameter sharing, and $\boldsymbol{\mathcal{W}}'_{n_1,m} = \boldsymbol{\mathcal{W}}'_{n_2,m} = \boldsymbol{\mathcal{W}}_m$ with parameter sharing, are both trained with a learning rate of 0.0001 (200 epochs), and a batch size of 64. After the compression, fine-tuning is carried out for all the methods with 10 epoches to obtain the final results. For testing, we utilize Set11 (11 gray images) [50] as shown in Figure 6, and average peak signal-to-noise ratio (PSNR) over the test images for reconstruction evaluation.

Parameter settings. In this test, as both $\boldsymbol{\mathcal{W}}_{n,m}$ and $\boldsymbol{\mathcal{W}}_m$ are given and all the weight tensors to be optimized are of the same dimensions, the proposed NC-CTD method in Algorithm 1 is applied, and α in (14) is set to 1. Furthermore, for ease of presentation, we set $r_c = r_{m,1} = r_{m,2}$ and $r_i = r_{n,m,1} = r_{n,m,2}$ for $n = 1, 2, \cdots, N$ and $m = 2, 3, 4, 5$, where r_c and r_i refer to the ranks of the common components and independent components. This means that for all the layers, the ranks (r_c, r_i) are the same. For the independent TT decomposition, we follow the same idea and set the rank to be r. Given (r_c, r_i) and r, the CR values can be computed according to (20). Note that for the remaining experiments, we follow the same protocol to design the ranks of all the methods unless stated otherwise.

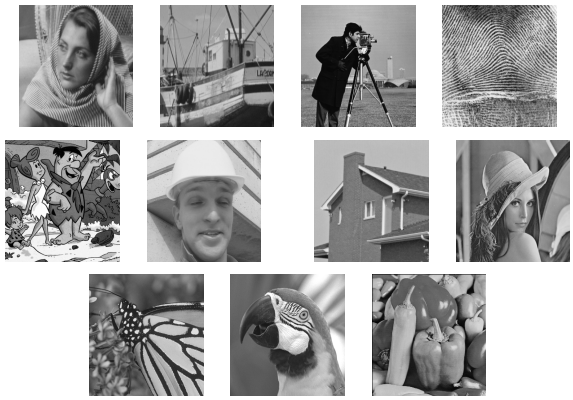

Fig. 6: Test images of Set11.

Test 1: $(N, M) = (5, 6)$ and $CS_{\text{ratio}} = 0.25$. The total number of parameters of the network when $(N, M) = (5, 6)$ without parameter sharing is 2.961million (m). At $CS_{\text{ratio}} = 0.25$, the PSNR results for the original network with and without parameter sharing (using $\boldsymbol{\mathcal{W}}_m$ and $\boldsymbol{\mathcal{W}}_{n,m}$) are 29.30dB and 31.88dB, respectively. Note that the compression ratio of the former over the latter (baseline) is approximately $1/N = 20\%$ in this test. Four different groups of CR values in $[7, 99]$ are used, and different (r_c, r_i) and r are selected to make the compression models approaching these CR values, and the results are tabulated in Table I.

Test 2: $(N, M) = (7, 6)$ and $CS_{\text{ratio}} = 0.10$. The total number of parameters of the network when $(N, M) = (7, 6)$ without parameter sharing is 4.145m. At $CS_{\text{ratio}} = 0.1$, the PSNR results for the original network with and without parameter sharing (using $\boldsymbol{\mathcal{W}}_m$ and $\boldsymbol{\mathcal{W}}_{n,m}$) are 24.47dB and 26.25dB, respectively. Note that the compression ratio of the former over the latter (baseline) is approximately $1/N = 14.29\%$ in this test. The same (r_c, r_i) and r are used as in **Test 1** for this experiment, and the PSNR as well as the CR results are shown in Table II.

Generally speaking, under the same or similar compression

Group	TT			NC-CTD		
	r	PSNR(dB)	CR(×)	$(r_c,\ r_i)$	PSNR(dB)	CR(×)
1	4	28.79	84.88	(8, 2)	29.03	89.78
				(16, 1)	**29.29**	69.79
2	8	29.15	46.26	(8, 6)	29.07	50.00
				(16, 5)	**29.28**	44.04
				(32, 2)	**29.44**	32.14
3	16	**29.46**	21.22	(8, 14)	29.42	22.95
				(16, 13)	29.39	22.08
				(32, 10)	29.41	19.69
				(64, 3)	29.41	12.26
4	32	**29.57**	8.23	(8, 30)	29.53	8.77
				(16, 29)	29.55	8.79
				(32, 26)	29.55	8.81
				(64, 19)	**29.57**	7.65

TABLE I: Network compression for ISTA-Net$^+$ when $(N, M) = (5, 6)$ and $CS_{\text{ratio}} = 0.25$.

Group	TT			NC-CTD		
	r	PSNR(dB)	CR(×)	$(r_c,\ r_i)$	PSNR(dB)	CR(×)
1	4	24.27	84.88	(8, 2)	24.47	98.77
				(16, 1)	**24.63**	84.34
2	8	24.54	46.26	(8, 6)	24.36	52.66
				(16, 5)	24.58	49.42
				(32, 2)	**24.65**	41.00
3	16	24.68	21.22	(8, 14)	24.67	23.50
				(16, 13)	24.63	23.35
				(32, 10)	**24.71**	22.69
				(64, 3)	24.71	16.39
4	32	24.81	8.23	(8, 30)	24.74	8.85
				(16, 29)	**24.83**	8.98
				(32, 26)	24.81	9.37
				(64, 19)	24.72	9.08

TABLE II: Network compression for ISTA-Net$^+$ when $(N, M) = (7, 6)$ and $CS_{\text{ratio}} = 0.1$.

ratio, the proposed coupled decomposition based compression model can give higher PSNR than the independent TT decomposition in most cases, especially in the high CR value region. The PSNR of the NC-CTD model is 0.36dB and 0.26dB better than the TT model for $N = 7$ and $N = 5$, respectively, for CR around $100\times$. In other words, to achieve a specified PSNR performance, the CR value of the proposed method is higher than the TT method for sufficiently high CR cases. Take the $N = 7$ case as an example, to get a PSNR$\geq$ 24.54dB, the maximum CR for the proposed and TT methods are $84.34\times$ and $46.26\times$. In these high CR value cases, there are sufficient parameters in each compressed layer of the NC-CTD models because of the large value of r_c (i.e., $r_c = 16$ for NC-CTD versus $r = 4$ for TT) of the shared common components. This leads to performance improvement of the joint tensor decomposition approach over the independent tensor decomposition one. In small CR value cases ($CR \leq 22$), it is different. When $N = 7$, the NC-CTD method performs slightly better than that of the TT approach, however, it ends up with opposite conclusions when $N = 5$. In the latter scenario, the independent components of the NC-CTD approach can provide enough representation ability of the network model, implying the less importance of the common components, and thus making the superiority of proposed NC-CTD approach becomes less distinct comparing to the independent tensor decomposition method.

It is worth noting that according to Group 3 in both Tables I and II, a moderate r_c should be selected in order to balance the performance of image reconstruction and compression ratio. A very small r_c will reduce the ability to exploit common components of the compression model, however, in order to achieve a certain level of compression ratio, as r_c increases, r_i should decrease, and a very small r_i will also reduce the network model presentation ability. Furthermore, comparing to the original parameter sharing model, which uses $\boldsymbol{\mathcal{W}}_m$ instead of $\boldsymbol{\mathcal{W}}_{n,m}$, the proposed model gives a much higher compression under the same PSNR performance: $7\times$ of CR for the former, and $98.77\times$ of CR for the latter in $N = 7$ case, which is 14 times better, indicating the effectiveness of the joint tensor factorization method over simple parameter sharing.

Model Name	ResNet-18	ResNet-34
conv1	7×7, 64, stride 2	
conv2_x	3×3 max pool, stride 2	
	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$
conv3_x	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$
conv4_x	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$
conv5_x	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$
average pool, 1000-d fc, softmax		

TABLE III: Architecture of ResNet-18 and ResNet-34 for ImageNet dataset.

B. Image Classification

1) ResNet-18: In this section, the ResNet-18 model is optimized, and Cifar-10 dataset is used for evaluation. The structure of the original model is shown in Table III. To modify it for Cifar-10, we substitute the '7×7, stride 2' in the sub-network 'conv1' by '3×3, stride 1', and delete the 'max pooling' in sub-network 'conv2_x'. The output is also modified to 10, and the other settings remain unchanged as in [42]. We train the original weights $\boldsymbol{\mathcal{W}}_{n,m}$ to achieve an accuracy of 94.87%, and the total number of parameters of the network before compression is 11.164m.

Parameter settings. In this test, the original $\boldsymbol{\mathcal{W}}_m$ does not exist and thus we set $\alpha = 0$ in (14). For the five sub-networks of ResNet-18 as shown in Table III, only 'conv$i$_x', $i = 3, 4, 5$, are optimized by the proposed and state-of-the-art methods. The others are not compressed because of their small parameter number. As the input depths of the first and third layers of each sub-network are different, they are optimized by the 'NC-PCTD' approach, while the second and fourth layers can be compressed by 'NC-CTD' as they have same shape. Therefore, we refer it to as 'NC-CTD + NC-PCTD'. For sub-networks 'conv$i$_x', $i = 3, 4, 5$, the output depth is $128q$, $q = 1, 2, 4$, respectively, and the ranks (r_c, r_i) are written as $(r_c q, r_i q)$, with $q = 1, 2, 4$ for $i = 3, 4, 5$, where r_c refers to the rank of the common or partly common components, and r_i is the rank of independent components. Furthermore, as the input depth of the first layer of all the sub-networks is only one half of the other layers, the respective rank decreases by half in this test. The independent TT and SVD [15] compression

Group	SVD				TT				NC-CTD + NC-PCTD			
	$r'q$	Accuracy(%)		CR(×)	rq	Accuracy(%)		CR(×)	$(r_cq,\ r_iq)$	Accuracy(%)		CR(×)
		w/o fine-tune	fine-tune			w/o fine-tune	fine-tune			w/o fine-tune	fine-tune	
1	$21q$	77.48	91.95	7.41	$32q$	78.84	92.83	7.40	$(4q, 31q)$	79.47	92.71	8.59
									$(8q, 30q)$	**80.34**	**92.99**	8.55
									$(16q, 29q)$	78.09	92.85	8.56
2	$40q$	90.98	93.75	4.18	$48q$	90.06	93.74	4.16	$(8q, 47q)$	90.26	93.84	4.50
									$(16q, 45q)$	**90.41**	**93.94**	4.53
									$(32q, 42q)$	89.19	93.75	4.50
3	$65q$	92.57	93.93	2.67	$64q$	93.03	94.06	2.66	$(8q, 63q)$	93.07	94.18	2.80
									$(16q, 62q)$	**93.14**	**94.28**	2.80
									$(24q, 61q)$	92.83	94.16	2.80
4	$95q$	93.33	94.3	1.85	$80q$	94.07	94.38	1.84	$(16q, 78q)$	93.88	94.5	1.91
									$(32q, 78q)$	**94.18**	**94.66**	1.85
									$(48q, 75q)$	93.81	94.61	1.85
									$(64q, 69q)$	93.31	94.31	1.91
5	$131q$	93.46	94.55	1.35	$96q$	94.6	94.68	1.35	$(16q, 94q)$	**94.61**	94.65	1.39
									$(32q, 93q)$	94.56	**94.72**	1.38
									$(64q, 87q)$	94.57	94.71	1.39

TABLE IV: Network compression for ResNet-18 on Cifar-10, w/o means without.

methods are included for comparison with ranks rq and $r'q$. Note that the SVD decomposition is treated as a special 2-D TT decomposition, although its coupled decomposition format can be derived straightforwardly, the results are not included here for compact presentation.

Results: In this test, we choose $rq = 32q$, $64q$ and $96q$ for the TT compression model to get high, moderate and low CR values (approximately $8\times$ to $1.35\times$) for comparison. Different (r_cq, r_iq) and $r'q$ are used to approximate the same CR values for the proposed method and the SVD compression one [15]. The results of all the methods before and after fine-tuning are tabulated in Table IV. It is observed that comparing to the state-of-the-art methods, the proposed 'NC-CTD + NC-PCTD' model can achieve higher image classification accuracy before and after fine-tuning under the same compression ratio. Specifically, to achieve a classification accuracy around 94.66%, the minimum CR for the proposed and TT methods are $1.85\times$ and $1.35\times$, while the SVD approach fails to obtain such high accuracy in all cases. Furthermore, by a fixed CR value, the best classification performance is usually attained under $r_c/r_i \approx 1/3$ for the proposed method in ResNet-18. It is worth noting that for low CR case, the result given by 'NC-CTD + NC-PCTD' model under $(r_cq, r_iq) = (32q, 93q)$ before fine-tuning outperforms the TT and SVD methods with fine-tuning, showing the effectiveness of exploring appropriate initial weight parameters of the alternating optimization algorithm for the coupled low rank compressed neural network.

2) ResNet-8: In this part, we design a ResNet-8 network for Cifar-10 data classification in order to investigate the performance of NC-PCTD only. The structure of the network is modified from the Cifar-10-based ResNet-18 used above. We delete the entire conv2_x sub-network and the second layer of each residual block, i.e., the second and fourth layers of sub-networks 'conv$i$_x', $i = 3, 4, 5$. There are only 8 layers in the network now, and thus is referred to as ResNet-8. The original network with weights $\boldsymbol{\mathcal{W}}_{n,m}$ is trained from scratch to achieve an accuracy of 93.09%, and the total number of parameters of the uncompressed network is 4.82m.

Parameter settings for 'NC-PCTD'. As there are only two layers with different dimensions of input channel in each sub-network 'conv$i$_x', $i = 3, 4, 5$, NC-PCTD is applied only to make use of shared information among the sub-networks. Similar to Table IV, rq and (r_cq, r_iq), with $q = 1, 2, 4$ for 'conv$i$_x', $i = 3, 4, 5$ are used to represent the ranks. We choose $rq = 32q$, $48q$, $64q$, $80q$ and $96q$ for the TT compression model to get different CR values for comparison. The r_cq is set to be $16q$ or $24q$, and different r_iq are calculated according to (28) in order to obtain similar CR values for the proposed 'NC-PCTD' approach. In the fine-tuning step, compressed networks are fine-tuned with 15 epochs.

Results:The results of both methods before and after fine-tuning are shown in Table V. We see that the proposed NC-PCTD model gives a superior performance over TT in all cases. Specifically, the accuracy of the NC-PCTD approach is 1.23% and 0.55% higher than that of the TT approach before and after fine-tuning, respectively, when $CR \approx 7.4\times$. It is worth noting that for the NC-PCTD approach, the cases with smaller $r_c (r_c = 16q)$ perform better than those with larger $r_c (r_c = 24q)$ under similar CR values after fine-tuning. This is due to the reason that there are only 2 original $\boldsymbol{\mathcal{W}}_{n,m}$ used to build the shared structure, limiting the shared information, thus an increase of r_c for $r_c \geq 16q$ might not be able to provide useful common information.

3) ResNet-34: In this section, the ResNet-34 model is optimized, and ImageNet and Cifar-10/Cifar-100 datasets are used for evaluation. The structure of the original model is shown in Table III. The original network model with weights $\boldsymbol{\mathcal{W}}_{n,m}$ are downloaded from GitHub[1] and transferred from Pytorch to Tensorflow. A fine-tuning with a few epochs is applied to $\boldsymbol{\mathcal{W}}_{n,m}$ to achieve an accuracy of 71.09%, and the total number of parameters of the original network is 21.78m.

Parameter settings for 'NC-CTD'. Similar to the second experiment, the original $\boldsymbol{\mathcal{W}}_m$ for Algorithm 1 does not exist and thus we set $\alpha = 0$ in (14), and only 'conv$i$_x', $i = 3, 4, 5$, are compressed. Here we apply the NC-CTD only for the network optimization. As the input depth of the first block (2 layers) of all the sub-networks 'conv$i$_x' is different from the others, they are not included for coupled decomposition. Therefore, in sub-networks 'conv$i$_x', we have $N = 3, 5,$

[1]https://github.com/Cadene/pretrained-models.pytorch#torchvision

Group	TT				NC-PCTD			
	rq	Accuracy(%)		CR(×)	$(r_cq,\ r_iq)$	Accuracy(%)		CR(×)
		w/o finetune	finetune			w/o finetune	finetune	
1	$32q$	56.85	89.89	7.39	$(16q, 28q)$	**58.08**	**90.34**	7.48
					$(24q, 26q)$	21.50	90.06	7.46
2	$48q$	80.03	91.33	4.49	$(16q, 45q)$	86.81	**91.58**	4.51
					$(24q, 44q)$	**86.89**	91.47	4.51
3	$64q$	87.37	91.79	2.98	$(16q, 62q)$	**89.98**	**92.21**	3.00
					$(24q, 61q)$	89.79	91.82	2.99
4	80q	90.82	92.27	2.12	$(16q, 78q)$	**91.52**	**92.38**	2.12
					$(24q, 77q)$	91.27	92.38	2.12
5	96q	92.16	92.65	1.57	$(16q, 94q)$	92.13	**92.73**	1.58
					$(24q, 93q)$	**92.22**	92.69	1.58

TABLE V: Network compression for ResNet-8 on Cifar-10, w/o means without.

Group	TT					NC-CTD			
	rq	Accuracy(%)		Time(ms)	CR(×)	$(r_cq,\ r_iq)$	Accuracy(%)		Time(ms)	CR(×)
		w/o fine-tune	fine-tune				w/o fine-tune	fine-tune		
1	$48q$	31.65	66.05	1.04	3.21	$(24q, 44q)$	39.53	**67.20**	1.12	3.21
						$(32q, 41q)$	**40.53**	67.05	1.11	3.21
						$(48q, 35q)$	34.27	66.87	1.24	3.22
2	$64q$	57.01	68.33	1.11	2.31	$(32q, 59q)$	**59.71**	**68.80**	1.11	2.31
						$(48q, 54q)$	59.09	68.76	1.25	2.32
						$(64q, 47q)$	55.80	68.49	1.25	2.32
3	96q	68.42	70.10	1.22	1.32	$(32q, 92q)$	69.36	70.39	1.29	1.32
						$(48q, 89q)$	69.37	**70.51**	1.45	1.32
						$(64q, 85q)$	**69.55**	70.49	1.43	1.32

TABLE VI: Network compression for ResNet-34 on ImageNet, w/o means without.

	Group	Original Net		TT				NC-CTD			
		Accuracy(%)	Time(ms)	rq	Accuracy(%)	Time(ms)	CR(×)	$(r_cq,\ r_iq)$	Accuracy(%)	Time(ms)	CR(×)
Cifar-10	1	88.62	0.67	$48q$	89.78	0.49	3.21	$(32q, 41q)$	**90.39**	0.56	3.21
								$(48q, 35q)$	90.32	0.58	3.22
	2			$64q$	89.36	0.54	2.31	$(48q, 54q)$	90.01	0.57	2.32
								$(64q, 47q)$	**90.59**	0.60	2.32
Cifar-100	1	52.22	0.65	$48q$	63.11	0.49	3.21	$(32q, 41q)$	**63.52**	0.53	3.21
								$(48q, 35q)$	62.20	0.54	3.22
	2			$64q$	63.08	0.59	2.31	$(48q, 54q)$	63.02	0.57	2.32
								$(64q, 47q)$	**64.19**	0.60	2.32

TABLE VII: Transferring ResNet-34 to Cifar-10/Cifar-100.

2, respectively, for coupled tensor decomposition. Similar to Table IV, rq and (r_cq, r_iq), with $q = 1, 2, 4$ for 'conv$i$_x', $i = 3, 4, 5$ are used to represent the ranks. We choose $rq = 48q$, $64q$ and $96q$ for the TT compression model to get the high, moderate and low CR values for comparison, and different (r_cq, r_iq) are employed to approximate the same CR values for the proposed 'NC-CTD' approach.

The ImageNet training database contains 1.2m pictures from 1000 classes. In the fine-tuning step, we use only half of training data, that is 600 pictures of each class, and fine-tuned it with 4 epochs for all the methods.

Since ResNet-34 is deep enough to show the distinct difference of running time among different models, the average running time for an image is tested with a GPU TITAN Xp in this experiment by using the whole test set with a batch-size of 250. The time for the original network is 1.45ms.

Results: The results of all the methods before and after fine-tuning are shown in Table VI. It is shown that comparing to the state-of-the-art methods, the proposed models can achieve higher image classification accuracy before and after fine-tuning under the same compression ratio. Specifically, the 'NC-CTD' approach under $(48q, 89q)$ gives the best result with only 0.58% accuracy loss and a compression ratio of $1.32\times$ comparing to the original model, while the TT approach under the same compression ratio gives approximately 1% accuracy loss. Besides, by a fixed CR value, the best classification performance is usually given under $r_c/r_i \approx 1/2$ for the proposed method in ResNet-34. This ratio is larger than that from the ResNet-18 case. It means that for a larger N, that is, more blocks sharing the same structure, the ratio r_c/r_i should be higher, because a higher percentage of common information can be discovered from a higher number of networks sharing the same structure.

The running times of all the cases are also shown in Table VI. It is observed that both the TT and proposed models can accelerate the network faster compared to the original one because of the reduction of parameters. In most cases, the proposed approach is more complicated than the TT model under similar CR values but with better performance. In fact, the number of parameters in each layer of the proposed approach is larger than that of the TT model and thus it is reasonable. This effect can be alleviated as r_c decreases (and thus r_i increases to maintain the same CR). According to the first row of each group in Table VI, the runtime of the proposed methods is only slightly higher than or sometimes equal to that of the TT method. For example, the NC-CTD

approach under $(r_c, r_i) = (32q, 59q)$ runs as fast as TT under $rq = 64$, while the running time of the proposed model under $(r_c, r_i) = (64q, 47q)$ is 0.14ms higher.

Finally, we take some of the compressed networks in Table VI as well as the original ResNet-34 and transfer them for the classification of Cifar-10/Cifar-100 datasets. Similar to ResNet-18, we substitute '7×7, stride2' in the sub-network 'conv1' by '3×3, stride1', and delete the 'max pooling' in sub-network 'conv2_x'. The output is also modified to 10/100, and the other settings remain unchanged. Note that for transfer learning, the parameters of the first and last layers of all the models are trained from the beginning by fixing the remaining layers unchanged as in Table V, and then the whole network will be fine-tuned with a few epoches. Other network training techniques, such as data augmentation and regularization are not considered and therefore the commonly used networks are not included for comparison [52]. In order to evalute the acceleration effect in a more general case, the models are run under the same setting as in the previous test but with an Intel Xeon E5-2690 CPU instead of a GPU. All the results are tabulated in Table VII. We see that the NC-CTD method outperforms the original network with higher classification accuracy and less computational complexity, and can achieve better image classification accuracy under similar compression ratio comparing to the TT model. In some cases, the proposed algorithm can attain similar degree of acceleration comparing to TT. For example, in the $(32q, 41q)$ case, the time cost of the proposed model is 0.12ms and 0.06ms smaller than the original model and TT one under $rq = 64q$, and also provides better classification accuracy as well as higher CR result. This demonstrates the compatibility of the proposed method.

VI. Conclusion

In this paper, the fully and partly coupled tensor train decomposition methods are proposed and employed for deep neural network compression. By utilizing the inter layer common information of neural networks, the weight tensors can be approximated by several low rank sub-tensors with both identical components and independent components. Two alternating optimization schemes are developed for the factorization. Experimental results show that comparing to the state-of-the-art independent matrix or tensor decomposition based compression methods, our approach can achieve better performance under the same compression ratio.

In future work it would be interesting to include other characteristics of the network weights to improve the performance. For example, the sparse structure of the weight tensors can be used to improve the network model representation ability, and the random shuffle technique can also be employed to alleviate the performance degradation. Other tensor decomposition methods, such as Kronecker tensor decomposition and tensor ring decomposition, can be applied for coupled decomposition. Furthermore, a joint estimation of the ranks of both the common and independent components is a very important issue.

References

[1] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[2] H. Purwins, B. Li, T. Virtanen, J. Schluter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[3] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland, Jun. 2014, pp. 655–665.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, Long Beach, CA, USA, Jun. 2017, pp. 5998–6008.

[5] T. J. O'Shea, T. Roy, and T. C. Clancy, "Over-the-air deep learning based radio signal classification," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 168–179, 2018.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, Ohio, USA, Jun. 2014, pp. 580–587.

[7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, Ohio, USA, Jun. 2014, pp. 1725–1732.

[8] J. Jia, G. Zhai, P. Ren, J. Zhang, Z. Gao, X. Min, and X. Yang, "Tiny-BDN: An efficient and compact barcode detection network," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 688–699, 2020.

[9] B. Yonel, E. Mason, and B. Yazici, "Deep learning for passive synthetic aperture radar," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 90–103, 2018.

[10] P. Tzirakis, G. Trigeorgis, M. A. Nicolaou, B. W. Schuller, and S. Zafeiriou, "End-to-end multimodal emotion recognition using deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1301–1309, 2017.

[11] N. P. Bidargaddi, M. Chetty, and J. Kamruzzaman, "Combining segmental semi-Markov models with neural networks for protein secondary structure prediction," *Neurocomputing*, vol. 72, no. 16-18, pp. 3943–3950, 2009.

[12] P. Mamoshina, A. Vieira, E. Putin, and A. Zhavoronkov, "Applications of deep learning in biomedicine," *Molecular Pharmaceutics*, vol. 13, no. 5, pp. 1445–1454, 2016.

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

[14] A. Jain, P. Goel, S. Aggarwal, A. Fell, and S. Anand, "Symmetric k-means for deep neural network compression and hardware acceleration on fpgas," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 4, pp. 737–749, 2020.

[15] C. Tai, T. Xiao, Y. Zhang, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," in *4th International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[16] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, Montreal, QC, Canada, Dec. 2014, pp. 1269–1277.

[17] A. V. M. Jaderberg and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proceedings of British Machine Vision Conference (BMVC 2014)*, Nottingham, UK, Sep. 2014.

[18] J. Huang, W. Sun, and L. Huang, "Deep neural networks compression learning based on multiobjective evolutionary algorithms," *Neurocomputing*, vol. 378, pp. 260–269, 2020.

[19] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[20] J. Huang, W. Sun, L. Huang, and S. Chen, "Deep compression with low rank and sparse integrated decomposition," in *2019 IEEE 7th International Conference on Computer Science and Network Technology*, Dalian, China, Oct. 2019, pp. 289–292.

[21] S. Sridhar, G. Deepak, K. Rajkumar, and A. Frederic, "Sparse low rank factorization for deep neural network compression," *Neurocomputing*, vol. 398, pp. 185–196, 2020.

[22] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *4th International Conference on Learning Representations*, San Juan, Puerto Rico, 2016.

[23] M. Astrid and S.-I. Lee, "CP-decomposition with tensor power method for convolutional neural networks compression," in *2017 IEEE International Conference on Big Data and Smart Computing*, Jeju, South Korea, Feb. 2017, pp. 115–118.

[24] S. Zhou and J. N. Wu, "Compression of fully-connected layer in neural network by kronecker product," in *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, Chiang Mai, Thailand, 2016, pp. P173–179.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[26] T. Garipov, D. Podoprikhin, A. Novikov, and D. Vetrov, "Ultimate tensorization: Compressing convolutional and FC layers alike," *arXiv Preprint arXiv:1611.03214*, 2016.

[27] C. Li, Z. Sun, J. Yu, M. Hou, and Q. Zhao, "Low-rank embedding of kernels in convolutional neural networks under random shuffling," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, Brighton, United Kingdom, 2019, pp. 3022–3026.

[28] Z. Jian and G. Bernard, "ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Salt Lake City, UT, USA, Jun. 2018, pp. 1828–1837.

[29] M. Congedo, R. Phlypo, and D. Pham, "Approximate joint singular value decomposition of an asymmetric rectangular matrix set," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 415–424, 2011.

[30] C. M. Lee, M. A. V. Mudaliar, D. R. Haggart, C. R. Wolf, G. Miele, J. K. Vass, D. J. Higham, and D. Crowther, "Simultaneous non-negative matrix factorization for multiple large scale gene expression datasets in toxicology," *PLOS ONE*, vol. 7, no. 12, pp. 1–21, 2012.

[31] T. Koh, T. Ryota, I. Katsuhiko, K. Akisato, and S. Hiroshi, "Non-negative multiple tensor factorization," in *IEEE 13th International Conference on Data Mining*, Dallas, TX, USA, Dec. 2013, pp. 1199–1204.

[32] S. Mikeal and D. L. Lieven, "Coupled tensor decompositions for applications in array signal processing," in *2013 5th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, St. Maritn, France, Dec. 2013, pp. 228–231.

[33] L. Lu, X. Ren, K.-H. Yeh, and Z. Tan, "Exploring coupled images fusion based on joint tensor decomposition," *Human-Centric Computing and Information Sciences*, vol. 10, no. 10, 2020.

[34] Y. Ke, H. Lifang, S. Y. Philip, Z. Wenkai, and L. Yue, "Coupled tensor decomposition for user clustering in mobile internet traffic interaction pattern," *IEEE Access*, vol. 7, pp. 18 113–18 124, 2019.

[35] L. Hong, L. Weibin, H. Gaining, and L. Fang, "Coupled tensor decomposition for hyperspectral pansharpening," *IEEE Access*, vol. 6, pp. 34 206–34 213, 2018.

[36] D. L. Lieven, D. M. Bart, and V. Joos, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[37] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep. 1966.

[38] W. Sun and H. C. So, "Accurate and computationally effcient tensor-based subspace approach for multi-dimensional harmonic retrieval," *IEEE Transactions on Signal Processing*, vol. 60, no. 10, pp. 5077–5088, 2012.

[39] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[40] D. Learning, *Ian, Goodfellow and Yoshua, Bengio and Aaron, Courville*. MIT Press, 2016.

[41] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems 28*, Montreal, Quebec, Canada, Dec. 2015, pp. 442–450.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, Las Vegas, NV, USA, 2016, pp. 770–778.

[43] M. S. Kim and D. H. Choi, "A new penalty parameter update rule in the augmented Lagrange multiplier method for dynamic response optimization," *KSME International Journal*, vol. 14, no. 10, pp. 1122–1130, 2000.

[44] F. E. Curtis, H. Jiang, and D. P. Robinson, "An adaptive augmented Lagrangian method for large-scale constrained optimization," *Mathematical Programming*, vol. 152, no. 1-2, pp. 201–245, 2015.

[45] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.

[46] O. Russakovsky, J. Denga, H. Su, J. Krausen, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F. F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[47] S. J. Pan and Y. Qiang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[48] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Largescale machine learning on heterogeneous systems," *Software available from tensorflow.org*, 2015.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations (ICLR 2015)*, San Diego, USA, May 2015.

[50] K. Kulkarni, S. Lohit, P. Turaga, R. Kerviche, and A. Ashok, "ReconNet: Non-iterative reconstruction of images from compressively sensed random measurements," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 449–458.

[51] H. Yao, F. Dai, S. Zhang, Y. Zhang, Q. Tian, and C. Xu, "DR2-Net: Deep residual reconstruction network for image compressive sensing," *Neurocomputing*, vol. 359, pp. 483–493, 2019.

[52] C. Zhuge, "pytorch-imagenet-cifar-coco-voc-training," *https://github.com/zgcr/pytorch-ImageNet-CIFAR-COCO-VOC-training*, 2020.