LEARNING EFFICIENT TENSOR REPRESENTATIONS WITH RING-STRUCTURED NETWORKS

Qibin Zhao[1], *Masashi Sugiyama*[1,2], *Longhao Yuan*[3,1], *Andrzej Cichocki*[4]

[1]RIKEN AIP, [2]The University of Tokyo, [3]Saitama Institute of Technology, [4]Skoltech

ABSTRACT

Tensor train decomposition is a powerful representation for high-order tensors, which has been successfully applied to various machine learning tasks in recent years. In this paper, we study a more generalized tensor decomposition with a ring-structured network by employing circular multilinear products over a sequence of lower-order core tensors. We refer to such tensor decomposition as *tensor ring (TR) representation*. Our goal is to introduce learning algorithms including sequential singular value decompositions and blockwise alternating least squares with adaptive tensor ranks. Experimental results demonstrate the effectiveness of the TR model and the learning algorithms. In particular, we show that the structure information and high-order correlations within a 2D image can be captured efficiently by employing an appropriate tensorization and TR decomposition.

Index Terms— Tensor decomposition, tensor train decomposition, tensorization, tensor network

1. INTRODUCTION

Tensor decompositions aim to represent a higher-order (or multi-dimensional) data as a multilinear product of several latent factors, which attracted considerable attentions in machine learning [1, 2, 3, 4, 5] and signal processing [6] in recent years. For a dth-order tensor of size $n \times \cdots \times n$, the standard tensor decompositions are the *canonical polyadic decomposition (CPD)* [7] which represents data as a sum of r rank-one tensors with $\mathcal{O}(dnr)$ parameters and *Tucker decomposition* [8, 9, 10] which represents data as a core tensor and several factor matrices with $\mathcal{O}(dnr + r^d)$ parameters. In general, CP decomposition provides a compact representation but with difficulties in finding the optimal solution, while Tucker decomposition is stable and flexible but its number of parameters scales exponentially to the tensor order.

Recently, *tensor networks* have emerged as a powerful tool for analyzing very high-order tensors [11]. A powerful tensor network is *tensor train / matrix product states* (TT/MPS) representation [12], which requires $\mathcal{O}(dnr^2)$ parameters and avoids the curse of dimensionality through a particular geometry of low-order contracted tensors. TT representation has been applied to modeling weight parameters in deep neural networks and nonlinear kernel learning [13, 14, 15], achieving a significant compression factor and scalability. It has also been successfully used for feature learning and classification [16]. To fully explore the advantages of tensor algebra, the key step is to efficiently represent real-world datasets by tensor networks, which has not been well studied yet. In addition, there are some limitations of TT, for example, i) the constraint on TT-ranks, i.e., $r_1 = r_{d+1} = 1$, leads to the limited representation ability and flexibility; ii) TT-ranks are bounded by the rank of a particularly unfolded matrix; iii) the permutation of the data tensor will yield an inconsistent solution, i.e., TT representations and TT-ranks are sensitive to the order of tensor dimensions. Hence, finding the optimal permutation remains a challenging problem.

In this paper, we study a new structure of tensor networks, which can be considered as a generalization of TT representations. First of all, we relax the condition over TT-ranks, i.e., $r_1 = r_{d+1} = 1$, leading to an enhanced representation ability. Secondly, the strict ordering of multilinear products between cores are alleviated. Third, the cores are treated equally by making the model symmetric. To this end, we add a new connection between the first and the last core tensors, yielding a circular tensor products of a set of cores (see Fig. 1). More specifically, we consider that each tensor element is approximated by performing a trace operation over the sequential multilinear products of cores. Since the trace operation ensures a scalar output, $r_1 = r_{d+1} = 1$ is not necessary. In addition, the cores can be circularly shifted and treated equally due to the properties of the trace operation. We call this model *tensor ring (TR) decomposition* and its cores *tensor ring (TR) representations*. Several works on tensor networks [17, 18, 19] have introduced the similar structure, however, learning algorithms for TR cores are not well explored. To learn TR representations, we firstly develop a non-iterative singular value decomposition (SVD) algorithm that is similar to TT-SVD algorithm [12]. To find the optimal TR-ranks, a block-wise alternating least squares (ALS) algorithms is presented.

Another interesting property is that we show the intrinsic structure or higher-order correlations within a 2D image can be captured more efficiently than SVD by converting a

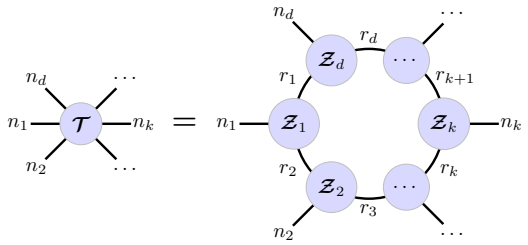

Fig. 1: A graphical representation of tensor ring decomposition.

2D matrix into a higher-order tensor. For example, given an image of size $I \times J$, we can apply an appropriate tensorization operation (see Sec. 4.1 for details) to obtain a fourth order tensor, each mode of which controls one specific scale of resolution. To demonstrate this, Fig. 2 illustrates the effects of noise corruption of specific tensor cores. As we can see, the first mode corresponds to the small-scale patches, while the 4th-mode corresponds to large-scale partitions. We will show in Sec. 4.1 that TR model can represent images more efficiently than the standard SVD.

2. TENSOR RING DECOMPOSITION

The TR decomposition aims to represent a high-order (or multi-dimensional) tensor by a sequence of 3rd-order tensors that are multiplied circularly. Specifically, let $\boldsymbol{\mathcal{T}}$ be a dth-order tensor of size $n_1 \times n_2 \times \cdots \times n_d$, denoted by $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$. Then the purpose of TR representation is to decompose it into a sequence of latent tensors $\boldsymbol{\mathcal{Z}}_k \in \mathbb{R}^{r_k \times n_k \times r_{k+1}}, k = 1, 2, \ldots, d$, which can be expressed in an element-wise form given by

$$T(i_1, i_2, \ldots, i_d) = \mathrm{Tr}\left\{\mathbf{Z}_1(i_1)\mathbf{Z}_2(i_2)\cdots\mathbf{Z}_d(i_d)\right\}. \quad (1)$$

$T(i_1, i_2, \ldots, i_d)$ denotes the $(i_1, i_2, \ldots, i_d)$th element of the tensor. $\mathbf{Z}_k(i_k)$ denotes the i_kth lateral slice matrix of the latent tensor $\boldsymbol{\mathcal{Z}}_k$, which is of size $r_k \times r_{k+1}$. Note that any two adjacent latent tensors, $\boldsymbol{\mathcal{Z}}_k$ and $\boldsymbol{\mathcal{Z}}_{k+1}$, have a common dimension r_{k+1} on their corresponding modes. The last latent tensor $\boldsymbol{\mathcal{Z}}_d$ is of size $r_d \times n_d \times r_1$, i.e., $r_{d+1} = r_1$, which ensures the product of these matrices is a square matrix. These prerequisites play key roles in TR decomposition, resulting in some important numerical properties. For simplicity, the latent tensor $\boldsymbol{\mathcal{Z}}_k$ can also be called the kth-*core* (or *node*). The minimum size of cores, $r_k, k = 1, 2, \ldots, d$, collected and denoted by a vector $\mathbf{r} = [r_1, r_2, \ldots, r_d]^T$, are called *TR-ranks*. From (1), we can observe that $T(i_1, i_2, \ldots, i_d)$ is equivalent to the trace of a sequential product of matrices $\{\mathbf{Z}_k(i_k)\}$. Based on (1), we can also express TR decomposition by

$$\boldsymbol{\mathcal{T}} = \sum_{\alpha_1,\ldots,\alpha_d=1}^{r_1,\ldots,r_d} \mathbf{z}_1(\alpha_1, \alpha_2) \circ \mathbf{z}_2(\alpha_2, \alpha_3) \circ \cdots \circ \mathbf{z}_d(\alpha_d, \alpha_1),$$

where the symbol "$\circ$" denotes the outer product of vectors and $\mathbf{z}_k(\alpha_k, \alpha_{k+1}) \in \mathbb{R}^{n_k}$ denotes the (α_k, α_{k+1})th mode-2 fiber of tensor $\boldsymbol{\mathcal{Z}}_k$. By assuming $n_1 = \cdots = n_d = n$ and $r_1 = \cdots = r_d = r$, the number of parameters in TR representation is $\mathcal{O}(dnr^2)$, which is linear to the tensor order d as in TT representation.

The TR representation can also be illustrated graphically by a linear tensor network as shown in Fig. 1. A node represents a tensor (including a matrix and a vector) whose order is denoted by the number of edges. The number by an edge specifies the size of each mode (or dimension). The connection between two nodes denotes a multilinear product operator between two tensors on a specific mode. This is also called *tensor contraction*, which corresponds to the summation over the indices of that mode. It should be noted that $\boldsymbol{\mathcal{Z}}_d$ is connected to $\boldsymbol{\mathcal{Z}}_1$ by the summation over the index α_1, which is equivalent to the trace operation. For simplicity, we denote TR decomposition by $\boldsymbol{\mathcal{T}} = \Re(\boldsymbol{\mathcal{Z}}_1, \boldsymbol{\mathcal{Z}}_2, \ldots, \boldsymbol{\mathcal{Z}}_d)$.

Property 1 (Circular dimensional permutation invariance)**.** *Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_d}$ be a dth-order tensor and its TR decomposition is given by $\boldsymbol{\mathcal{T}} = \Re(\boldsymbol{\mathcal{Z}}_1, \boldsymbol{\mathcal{Z}}_2, \ldots, \boldsymbol{\mathcal{Z}}_d)$. If we define $\overleftarrow{\boldsymbol{\mathcal{T}}}^k \in \mathbb{R}^{n_{k+1} \times \cdots \times n_d \times n_1 \times \cdots \times n_k}$ as the circularly shifted version along the dimensions of $\boldsymbol{\mathcal{T}}$ by k, then we have $\overleftarrow{\boldsymbol{\mathcal{T}}}^k = \Re(\boldsymbol{\mathcal{Z}}_{k+1}, \ldots, \boldsymbol{\mathcal{Z}}_d, \boldsymbol{\mathcal{Z}}_1, \ldots \boldsymbol{\mathcal{Z}}_k)$.*

Proof. It is obvious that (1) can be rewritten as

$$\begin{aligned} T(i_1, i_2, \ldots, i_d) &= \mathrm{Tr}(\mathbf{Z}_2(i_2), \mathbf{Z}_3(i_3), \ldots, \mathbf{Z}_d(i_d), \mathbf{Z}_1(i_1)) \\ &= \cdots = \mathrm{Tr}(\mathbf{Z}_d(i_d), \mathbf{Z}_1(i_1), \ldots, \mathbf{Z}_{d-1}(i_{d-1})). \end{aligned}$$

Therefore, we have $\overleftarrow{\boldsymbol{\mathcal{T}}}^k = \Re(\boldsymbol{\mathcal{Z}}_{k+1}, \ldots, \boldsymbol{\mathcal{Z}}_d, \boldsymbol{\mathcal{Z}}_1, \ldots, \boldsymbol{\mathcal{Z}}_k)$. □

3. LEARNING ALGORITHMS

In this section, we describe learning algorithms for TR decomposition.

3.1. Sequential SVDs

We propose an algorithm for computing the TR decomposition using d sequential SVDs, which is termed as *TR-SVD*. TR decomposition can be written as

$$T_{\langle k \rangle}(\overline{i_1 \cdots i_k}, \overline{i_{k+1} \cdots i_d}) = \mathrm{Tr}\left\{\prod_{j=1}^{k} \mathbf{Z}_j(i_j) \prod_{j=k+1}^{d} \mathbf{Z}_j(i_j)\right\}. \quad (2)$$

We define subchains by merging multiple linked cores as $\mathbf{Z}^{<k}(\overline{i_1 \cdots i_{k-1}}) = \prod_{j=1}^{k-1} \mathbf{Z}_j(i_j)$ and $\mathbf{Z}^{>k}(\overline{i_{k+1} \cdots i_d}) = \prod_{j=k+1}^{d} \mathbf{Z}_j(i_j)$. Hence, we can obtain $\mathbf{T}_{\langle k \rangle} = \mathbf{Z}_{(2)}^{\leq k}(\mathbf{Z}_{[2]}^{>k})^T$, where the subchain $\mathbf{Z}_{(2)}^{\leq k}$ is of size $\prod_{j=1}^{k} n_j \times r_1 r_{k+1}$, and $\mathbf{Z}_{[2]}^{>k}$ is of size $\prod_{j=k+1}^{d} n_j \times r_1 r_{k+1}$.

According to (2), TR decomposition can be performed from mode-1 by

$$T_{\langle 1\rangle}(i_1, \overline{i_2 \cdots i_d}) = \sum_{\alpha_1,\alpha_2} Z^{\leq 1}(i_1, \overline{\alpha_1 \alpha_2}) Z^{>1}(\overline{\alpha_1 \alpha_2}, \overline{i_2 \cdots i_d}).$$

Since the low-rank approximation of $\mathbf{T}_{\langle 1\rangle}$ can be obtained by the truncated SVD, which is $\mathbf{T}_{\langle 1\rangle} = \mathbf{U}\Sigma\mathbf{V}^T + \mathbf{E}_1$, the first core $\boldsymbol{\mathcal{Z}}_1 (i.e., \boldsymbol{\mathcal{Z}}^{\leq 1})$ of size $r_1 \times n_1 \times r_2$ can be obtained by the proper reshaping and permutation of $\mathbf{U}$. Similarly, the subchain $\boldsymbol{\mathcal{Z}}^{>1}$ of size $r_2 \times \prod_{j=2}^{d} n_j \times r_1$ can be obtained by the proper reshaping and permutation of $\mathbf{\Sigma V}^T$, which corresponds to the remaining $d-1$ dimensions of $\boldsymbol{\mathcal{T}}$. Note that this algorithm uses a similar strategy to TT-SVD [12], but the reshaping and permutations are different between them. Subsequently, we can further reshape the subchain $\boldsymbol{\mathcal{Z}}^{>1}$ as a matrix $\mathbf{Z}^{>1} \in \mathbb{R}^{r_2 n_2 \times \prod_{j=3}^{d} n_j r_1}$ which can be written as

$$Z^{>1}(\overline{\alpha_2 i_2}, \overline{i_3 \cdots i_d \alpha_1}) = \sum_{\alpha_3} Z_2(\overline{\alpha_2 i_2}, \alpha_3) Z^{>2}(\alpha_3, \overline{i_3 \cdots i_d \alpha_1}).$$

By applying truncated SVD, i.e., $\mathbf{Z}^{>1} = \mathbf{U}\Sigma\mathbf{V}^T + \mathbf{E}_2$, we can obtain the second core $\boldsymbol{\mathcal{Z}}_2$ of size $r_2 \times n_2 \times r_3$ by appropriately reshaping $\mathbf{U}$ and the subchain $\boldsymbol{\mathcal{Z}}^{>2}$ by proper reshaping of $\mathbf{\Sigma V}^T$. This procedure can be performed sequentially to obtain all d cores $\boldsymbol{\mathcal{Z}}_k, k = 1, \ldots, d$.

As proved in [12], the approximation error by using such sequential SVDs is given by $\|\boldsymbol{\mathcal{T}} - \Re(\boldsymbol{\mathcal{Z}}_1, \boldsymbol{\mathcal{Z}}_2, \ldots, \boldsymbol{\mathcal{Z}}_d)\|_{\mathrm{F}} \leq \sqrt{\sum_{k=1}^{d-1} \|\mathbf{E}_k\|_{\mathrm{F}}^2}$, where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. Hence, given a prescribed relative error ϵ_p, the truncation threshold δ can be set to $\frac{\epsilon_p}{\sqrt{d-1}}\|\boldsymbol{\mathcal{T}}\|_{\mathrm{F}}$. However, $\|\mathbf{E}_1\|_{\mathrm{F}}$ corresponds to two ranks including both r_1 and r_2, while $\|\mathbf{E}_k\|_{\mathrm{F}}, \forall k > 1$ correspond to only one rank r_{k+1}. Therefore, we modify the truncation threshold as

$$\delta_k = \begin{cases} \sqrt{2}\epsilon_p \|\boldsymbol{\mathcal{T}}\|_{\mathrm{F}}/\sqrt{d} & k = 1, \\ \epsilon_p \|\boldsymbol{\mathcal{T}}\|_{\mathrm{F}}/\sqrt{d} & k > 1. \end{cases} \tag{3}$$

Note that the cores obtained by the TR-SVD algorithm are left-orthogonal, which is $\mathbf{Z}_{k\langle 2\rangle}^T \mathbf{Z}_{k\langle 2\rangle} = \mathbf{I}$ for $k = 2, \ldots, d-1$.

3.2. Block-Wise Alternating Least-Squares (ALS)

The ALS algorithm has been widely applied to various tensor decomposition models such as CP and Tucker decompositions [20, 21]. Given a dth-order tensor $\boldsymbol{\mathcal{T}}$, our goal is to optimize the error function as

$$\min_{\boldsymbol{\mathcal{Z}}_1,\ldots,\boldsymbol{\mathcal{Z}}_d} \|\boldsymbol{\mathcal{T}} - \Re(\boldsymbol{\mathcal{Z}}_1, \ldots, \boldsymbol{\mathcal{Z}}_d)\|_F. \tag{4}$$

According to the TR definition in (1), we have

$$T_{[k]}(i_k, \overline{i_{k+1} \cdots i_d i_1 \cdots i_{k-1}}) = \sum_{\alpha_k \alpha_{k+1}} \left\{ Z_k(i_k, \overline{\alpha_k \alpha_{k+1}}) Z^{\neq k}(\overline{\alpha_k \alpha_{k+1}}, \overline{i_{k+1} \cdots i_d i_1 \cdots i_{k-1}}) \right\},$$

where $\mathbf{Z}^{\neq k}(\overline{i_{k+1} \cdots i_d i_1 \ldots i_{k-1}}) = \prod_{j=k+1}^{d} \mathbf{Z}_j(i_j) \prod_{j=1}^{k-1} \mathbf{Z}_j(i_j)$ denotes the slice matrix of the subchain tensor by merging all cores except kth core $\boldsymbol{\mathcal{Z}}_k$. The objective function in (4) can be optimized by solving d subproblems alternatively. More specifically, having fixed all but one core, the problem is reduced to a linear least squares problem, which is

$$\min_{\mathbf{Z}_{k(2)}} \left\| \mathbf{T}_{[k]} - \mathbf{Z}_{k(2)} \left(\mathbf{Z}_{[2]}^{\neq k} \right)^T \right\|_F, \quad k = 1, \ldots, d.$$

This optimization procedure is repeated till convergence, which is called TT-ALS.

Here, we propose a computationally efficient block-wise ALS (BALS) algorithm by utilizing truncated SVD, which facilitates the self-adaptation of ranks. The main idea is to perform block-wise optimization followed by the separation of a block into individual cores. To achieve this, we consider merging two linked cores, e.g., $\boldsymbol{\mathcal{Z}}_k, \boldsymbol{\mathcal{Z}}_{k+1}$, into a block (or subchain) $\boldsymbol{\mathcal{Z}}^{(k,k+1)} \in \mathbb{R}^{r_k \times n_k n_{k+1} \times r_{k+2}}$. Thus, the subchain $\boldsymbol{\mathcal{Z}}^{(k,k+1)}$ can be optimized while leaving all cores except $\boldsymbol{\mathcal{Z}}_k, \boldsymbol{\mathcal{Z}}_{k+1}$ fixed. Subsequently, the subchain $\boldsymbol{\mathcal{Z}}^{(k,k+1)}$ can be reshaped into $\tilde{\mathbf{Z}}^{(k,k+1)} \in \mathbb{R}^{r_k n_k \times n_{k+1} r_{k+2}}$ and separated into a left-orthonormal core $\boldsymbol{\mathcal{Z}}_k$ and $\boldsymbol{\mathcal{Z}}_{k+1}$ by a truncated SVD: $\tilde{\mathbf{Z}}^{(k,k+1)} = \mathbf{U}\Sigma\mathbf{V}^T = \mathbf{Z}_{k\langle 2\rangle}\mathbf{Z}_{k+1\langle 1\rangle}$, where $\mathbf{Z}_{k\langle 2\rangle} \in \mathbb{R}^{r_k n_k \times r_{k+1}}$ is the 2-unfolding matrix of core $\boldsymbol{\mathcal{Z}}_k$, which can be set to $\mathbf{U}$, while $\mathbf{Z}_{k+1\langle 1\rangle} \in \mathbb{R}^{r_{k+1} \times n_{k+1} r_{k+2}}$ is the 1-unfolding matrix of core $\boldsymbol{\mathcal{Z}}_{k+1}$, which can be set to $\mathbf{\Sigma V}^T$. This procedure then moves on to optimize the next block cores $\boldsymbol{\mathcal{Z}}^{(k+1,k+2)}, \ldots, \boldsymbol{\mathcal{Z}}^{(d-1,d)}, \boldsymbol{\mathcal{Z}}^{(d,1)}$ successively in a similar way. Note that since the TR model is circular, the dth core can also be merged with the first core yielding the block core $\boldsymbol{\mathcal{Z}}^{(d,1)}$.

The key advantage of our BALS algorithm is the rank adaptation ability which can be achieved simply by separating the block core into two cores via truncated SVD. The truncated rank r_{k+1} can be chosen such that the approximation error is below a certain threshold. One possible choice is to use the same threshold as in the TR-SVD algorithm, i.e., δ_k described in (3). However, our empirical experience showed that this threshold often leads to overfitting and the truncated rank tends to be higher than the optimal rank. This is because the updated block $\boldsymbol{\mathcal{Z}}^{(k,k+1)}$ during ALS iterations is not a closed form solution and many iterations are necessary for convergence. To relieve this problem, we choose the truncation threshold based on both the current and the desired approximation errors, which is $\delta = \max\left\{\epsilon\|\boldsymbol{\mathcal{T}}\|_F/\sqrt{d},\ \epsilon_p\|\boldsymbol{\mathcal{T}}\|_F/\sqrt{d}\right\}$.

4. EXPERIMENTAL RESULTS

In this section, we report experimental results.

4.1. Image Representation by Higher-order Tensor Decompositions

An image is naturally represented by a 2D matrix, on which SVD can provide the best low-rank approximation. Here, we show the tensorization of an image yields a higher-order tensor, and TR decomposition enables us to represent the image more efficiently than SVD. Given an image (e.g., "Peppers") denoted by $\boldsymbol{\mathcal{Y}}$ of size $I \times J$, we can reshape it as $I_1 \times I_2 \times \ldots \times I_d \times J_1 \times J_2 \times \ldots \times J_d$, where $I = \prod_{k=1}^{d} I_k$, $J = \prod_{k=1}^{d} J_k$, followed by an appropriate permutation to $I_1 \times J_1 \times I_2 \times J_2 \ldots \times I_d \times J_d$ and thus reshape it again

Table 1: Image representation by using tensorization and TR decomposition. The number of parameters is compared for SVD, TT and TR given the same approximation errors.

Data	$\epsilon = 0.1$		$\epsilon = 0.01$		$\epsilon = 9e-4$		$\epsilon = 2e-15$	
$n = 256, d = 2$	SVD	TT/TR	SVD	TT/TR	SVD	TT/TR	SVD	TT/TR
	9.7e3	9.7e3	7.2e4	7.2e4	1.2e5	1.2e5	1.3e5	1.3e5
Tensorization	$\epsilon = 0.1$		$\epsilon = 0.01$		$\epsilon = 2e-3$		$\epsilon = 1e-14$	
	TT	TR	TT	TR	TT	TR	TT	TR
$n = 16, d = 4$	5.1e3	3.8e3	6.8e4	6.4e4	1.0e5	7.3e4	1.3e5	7.4e4
$n = 4, d = 8$	4.8e3	4.3e3	7.8e4	7.8e4	1.1e5	9.8e4	1.3e5	1.0e5
$n = 2, d = 16$	7.4e3	7.4e3	1.0e5	1.0e5	1.5e5	1.5e5	1.7e5	1.7e5

Fig. 2: Effects of noise corrupted tensor cores. From left to right, each figure shows a reconstructed image under corruption by adding noise to one specific tensor core.

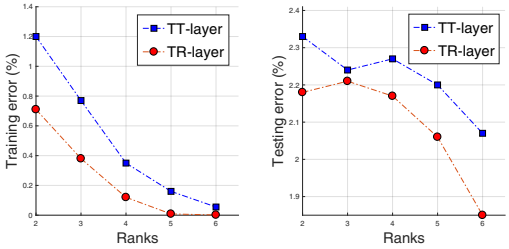

Fig. 3: The classification performances of tensorizing neural networks by using TR representation.

to $I_1J_1 \times I_2J_2 \times \ldots \times I_dJ_d$, which is a dth-order tensor. The first mode corresponds to small-scale patches of size $I_1 \times J_1$, while the dth-mode corresponds to large-scale partitions of the whole image as $I_d \times J_d$. To illustrate how each core tensor corresponds to the different resolutions, Fig. 2 shows reconstructed images by adding noise to one specific core tensor. Based on this tensorization operations, TR decomposition is able to capture the intrinsic structure information and provides a more compact representation. As shown in Table 1, for the 2D matrix case, SVD, TT and TR give exactly the same results. In contrast, for the 4th-order tensorization cases, TT needs only half the number of parameters (2 times compression rate) while TR achieves 3 times higher compression rate, given the same approximation error 0.1. Hence, TR representation can obtain enhanced presentation ability.

4.2. Tensorizing Neural Networks using TR representation

TT representations have been successfully applied to deep neural networks [13], which can significantly reduce the number of model parameters and improve computational efficiency. To investigate the properties of TR representation, we applied the TR framework to approximate the weight matrix of a fully-connected layer and compared it with TT representation. We run the experiment on the MNIST dataset [22]. The same setting of the neural network (two fully-connected layers with ReLU activation functions) as that in [13] was applied for comparison. The input layer is tensorized to a 4th-order tensor of size $4 \times 8 \times 8 \times 4$, the weight matrix of size 1024×625 is represented by a TR format of size $4 \times 8 \times 8 \times 4 \times 5 \times 5 \times 5 \times 5$. Through deriving the gradients over each core tensor, all computations can be performed on small core tensors instead of the dense weight matrix, yielding significant improvements of computational efficiency. The experimental results are shown in Fig. 3. We observe that the TR-layer provides much better flexibility than the TT-layer, leading to much lower training and testing errors under the same compression level (i.e., TT/TR ranks). In addition, TR can achieve a much better compression rate under the same level of the test error. When $r_1 = \cdots = r_4 = 2$, the compression rate of the dense weight matrix is up to 1300 times.

We tested the tensorizing neural networks with the same architecture on the SVHN dataset[1]. By setting all the TT-ranks in the network to 4, we achieved the test error of 0.13 with compression rate of 444 times, while we can achieve the same test error by setting all the TR-ranks to 3 with compression rate of 592 times. Hence, the TR representation can obtain a significantly higher compression rate under the same level of the test error.

5. CONCLUSION

We have proposed a novel tensor decomposition model, which provides an efficient representation for a very high-order tensor by a sequence of low-dimensional cores. The number of parameters in our model scales only linearly to the tensor order. To optimize the latent cores, we have presented several different algorithms: TR-SVD is a non-recursive algorithm that is stable and efficient, while TR-BALS can learn a more compact representation with adaptive TR-ranks. The experimental results verified the effectiveness of our proposed algorithms.

Acknowledgments

This work was partially supported by JSPS KAKENHI (Grant No. 17K00326) and JST CREST (Grant No. JPMJCR1784). MS was supported by JST CREST Grant Number JPMJCR18A2.

[1] http://ufldl.stanford.edu/housenumbers/

6. REFERENCES

[1] Rose Yu and Yan Liu, "Learning from multiway data: Simple and efficient tensor regression," in *International Conference on Machine Learning*, 2016, pp. 373–381.

[2] Animashree Anandkumar, Rong Ge, Daniel J Hsu, Sham M Kakade, and Matus Telgarsky, "Tensor decompositions for learning latent variable models.," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2773–2832, 2014.

[3] Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil, "Multilinear multitask learning," in *International Conference on Machine Learning*, 2013, pp. 1444–1452.

[4] Heishiro Kanagawa, Taiji Suzuki, Hayato Kobayashi, Nobuyuki Shimizu, and Yukihiro Tagami, "Gaussian process nonparametric tensor estimator and its minimax optimality," in *International Conference on Machine Learning (ICML2016)*, 2016, pp. 1632–1641.

[5] Yinchong Yang, Denis Krompass, and Volker Tresp, "Tensor-train recurrent neural networks for video classification," *arXiv preprint arXiv:1707.01786*, 2017.

[6] G. Zhou, Q. Zhao, Y. Zhang, T. Adali, S. Xie, and A. Cichocki, "Linked component analysis from matrices to high-order tensors: Applications to biomedical data," *Proceedings of the IEEE*, vol. 104, no. 2, pp. 310–331, 2016.

[7] J. Goulart, M. Boizard, R. Boyer, G. Favier, and P. Comon, "Tensor cp decomposition with structured factor matrices: Algorithms and performance," *IEEE Journal of Selected Topics in Signal Processing*, 2015.

[8] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(R1,R2,. . .,RN) approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1324–1342, 2000.

[9] Zenglin Xu, Feng Yan, and Alan Qi, "Infinite Tucker decomposition: Nonparametric Bayesian models for multiway data analysis," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 1023–1030.

[10] Shandian Zhe, Kai Zhang, Pengyuan Wang, Kuang-chih Lee, Zenglin Xu, Yuan Qi, and Zoubin Ghahramani, "Distributed flexible nonlinear tensor factorization," in *Advances in Neural Information Processing Systems*, 2016, pp. 928–936.

[11] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P Mandic, et al., "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Foundations and Trends® in Machine Learning*, vol. 9, no. 4-5, pp. 249–429, 2016.

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

[13] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 442–450.

[14] Edwin Stoudenmire and David J Schwab, "Supervised learning with tensor networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., pp. 4799–4807. Curran Associates, Inc., 2016.

[15] Chuan-Yung Tsai, Andrew M Saxe, and David Cox, "Tensor switching networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2038–2046.

[16] J. A. Bengua, H. N. Phien, and H. D. Tuan, "Optimal feature extraction and classification of tensors via matrix product state decomposition," in *2015 IEEE International Congress on Big Data*, June 2015, pp. 669–672.

[17] Mike Espig, Wolfgang Hackbusch, Stefan Handschuh, and Reinhold Schneider, "Optimization problems in contracted tensor networks," *Computing & Visualization in Science*, vol. 14, no. 6, pp. 271–285, 2011.

[18] Mike Espig, Kishore Kumar Naraparaju, and Jan Schneider, "A note on tensor chain approximation," *Computing & Visualization in Science*, vol. 15, no. 6, pp. 331–344, 2012.

[19] Boris N. Khoromskij, "O(d log N)-quantics approximation of - tensors in high-dimensional numerical modeling," *Constructive Approximation*, vol. 34, no. 2, pp. 257–280, 2011.

[20] T.G. Kolda and B.W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[21] S. Holtz, T. Rohwedder, and R. Schneider, "The alternating linear scheme for tensor optimization in the tensor train format," *SIAM J. Scientific Computing*, vol. 34, no. 2, 2012.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.