Tensor Decomposition via Core Tensor Networks

Jianfu Zhang[1], Zerui Tao[2], Qibin Zhao[3], and Liqing Zhang[1]

[1]Shanghai Jiao Tong University
[2]Lanzhou University
[3]RIKEN AIP, Tensor Learning Team
c.sis@sjtu.edu.cn, zerui.tao@foxmail.com, qibin.zhao@riken.jp, zhang-lq@cs.sjtu.edu.cn

Abstract

Tensor decomposition has shown promising performance in image completion and denoising. Existing methods always aim to decompose one tensor into latent factors or core tensors by optimizing a particular cost function based on a specific tensor model. These algorithms iteratively learn the optima from random initialization given any individual tensor, resulting in slow convergence and low efficiency. In this paper, we propose an efficient tensor decomposition algorithm which aims to learn a global mapping from input tensors to latent core tensors, under the assumption that the mappings of a bunch of tensors might be shared or highly correlated. To this end, we train a deep neural network (DNN) to model the global mapping and then apply it to decompose a newly given tensor with high efficiency. Furthermore, the initial values of DNN are learned based on meta-learning methods. By leveraging the pretrained meta-learning based core tensor DNN, our proposed method enables us to perform tensor decomposition efficiently and accurately. Experimental results demonstrate the significant improvements of our method over other tensor decomposition methods in terms of speed and accuracy.

1 Introduction

Tensors are extensions of vectors or matrices to high-order cases. Tensor decomposition, which aims to represent high-order tensors by multi-linear operations over low-order tensors, have shown compelling performance in a variety of artificial intelligence application tasks such as image and video completion [Oseledets, 2011; Zhao *et al.*, 2016], multi-modal signal processing [Jia and Gong, 2005; Zadeh *et al.*, 2017] and model compression[Yu *et al.*, 2017]. In particular, besides directly decomposing of high-order data, we can also tensorize data of lower-order into a high-order tensor which is called tensorization operation, and then apply tensor decomposition based on an appropriate assumption, such as low-rankness, smoothness, or sparsity, to handle different tasks.

In this paper, we focus on tensor decomposition for image completion and denoising. Particularly, our objective is to find a low-rank approximation of the observed signal. Following the popular tensor decomposition models such as [Harshman and others, 1970; Carroll and Chang, 1970; Tucker, 1966; Oseledets, 2011; Zhao *et al.*, 2016], these methods search for the best low-rank decomposition by optimizing algorithms like gradient descent or alternating least square. Since the optimization problem is non-convex, ill initialization may lead to a spurious local minimum and slow convergence. Moreover, traditional methods always treat every image individually without considering any correlations. Individually decomposing the tensors means more time-consuming when the number of input tensors is large. Consequently, when the dataset contains a large amount of data, existing methods cannot give out results fast and accurately. These drawbacks make tensor decomposition cumbersome for more flexible applications that have a high demand for time efficiency. Also, different input tensors may share some common structures, learning tensor decomposition individually will ignore these correlations.

To address the problems abovementioned, we proposed a tensor decomposition method based on Neural Networks (NN) called Core Tensor Networks (CTN). Given a large dataset consists of many tensors (*e.g.*, images), we use the networks to learn the main core tensors for all the tensors and leverage bias core tensors to adjust each core tensor. Specifically, we vectorize the input tensor and learn the core tensors with a few fully-connected layers. With the help of CTN, we can intuitively learn the structure information shared by all the input tensors. Leveraging the scalability of NN and computing ability of GPU, our proposed method can decompose the tensors fast and accurately. We also train CTN with a small amount of auxiliary tensor data and transfer to the test tensors. We designed a framework based on meta-learning [Finn *et al.*, 2017; Nichol *et al.*, 2018] to learn the main core tensors, which can adaptively fit different bias core tensors. With the help of meta-learning, CTN can decompose test tensor in significant fast speed and still compatible accuracy.

Our contribution can be summarized as the following:

- We proposed a model that directly learns the core tensors for tensor decomposition via a neural network framework, to our knowledge, this is the first paper to

learn tensor decomposition with neural networks.

- Our proposed method can learn the correlations among all input tensors yielding better accuracy. Furthermore, combining with meta-learning, our proposed method can adaptively fit different data and significantly improve the speed of tensor decomposition.
- We evaluate our proposed model on unsupervised image completion and denoising tasks. Experimental results show that our model achieves competitive performance at a significantly fast speed.

2 Related Works

Tensor decompositions are widely used in completion and denoising tasks. Traditional tensor completion/denoising models assume that the true signal lies in some low-rank space, which can be approximated by low-rank tensor decompositions. A wide variety of tensor decomposition models have been established, e.g., CP [Harshman and others, 1970], Tucker [Tucker, 1966], Tensor-Train (TT) [Oseledets, 2011] and Tensor-Ring (TR) [Zhao *et al.*, 2016], among many others. To pursue low-rank approximation of observed data, one way is to use convex surrogation of tensor rank, *i.e.*, overlapped nuclear norm [Liu *et al.*, 2012], TT nuclear norm [Bengua *et al.*, 2017], TR nuclear norm [Yu *et al.*, 2019], tubal nuclear norm [Semerci *et al.*, 2014]. However, nuclear norm based methods suffer from heavy computation and can hardly be applied to large-scale datasets. Another track is to search low-rank factorizations of the data. Basically, there are two strategies for tensor decompositions. One is to use Alternating Least Square (ALS) based algorithms, namely, to alternatively solve least square problems for different factors. For example, [Wang *et al.*, 2017] proposed TR-ALS for tensor completion. [Yuan *et al.*, 2019a] designed rTRD for large tensors, using sketching technique. Another is to use gradient-based algorithms, which is more scalable to large scale datasets. The idea is to compute the gradient for weighted objective functions, then apply gradient descent (GD) or stochastic gradient descent (SGD), e.g., CP-WOPT [Acar *et al.*, 2011], Tucker-WOPT [Filipović and Jukić, 2015], TT-WOPT [Yuan *et al.*, 2017]. Besides, [Yuan *et al.*, 2019b] used nuclear norm regularization in TR, in order to get low-rank factors.

It should be noted that our proposed algorithm differs from all the above traditional approaches in that we train core tensor networks (CTN) based on different images, while traditional tensor decomposition methods only use the information of one single image. It is reasonable to assume that there exists some typical structure among different images, and using the learned cores as initialization should be helpful for further learning on different test images. To the best of our knowledge, it is the first attempt to combine meta-learning with tensor decomposition. Moreover, our proposed model can be easily applied to enormous images, with the help of stochastic optimization algorithms like gradient descent.

There are existing works which are related to both tensor decomposition and NN. Methods like [Lebedev *et al.*, 2015; Wang *et al.*, 2018] utilize tensor decomposition methods to reduce the number of parameters in NN. CoSTCo [Liu *et al.*, 2019] leverages NN to preserve the high-rank information as supplementary for sparse tensor decomposition. In [Maruhashi *et al.*, 2018], Maruhashi *et al.* proposed a method utilizing decomposed core tensor as fixed input for NN to learn multi-way relations. In [Wu *et al.*, 2019], Wu *et al.* proposed an approach to learn temporal relationship with a recurrent neural network, then learn the non-linearities between different modes with multi-linear perceptron. Our proposed method is distinguished from these approaches that we directly generate the decomposed core tensors for tensor decomposition in an end-to-end manner, instead of using tensor decomposition methods as a submodule in NN.

3 Methodology

In this section, we will first define what is tensor decomposition and the tasks we are focusing on. Then our proposed methods will be demonstrated.

3.1 Preliminaries

A tensor with order $N \geq 3$ is denoted by Euler script letter, *e.g.*, $\mathcal{T} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$. Each dimension I_k is called a mode. An element of tensor $\mathcal{T}$ of index $(i_1, ..., i_n)$ is denoted by $t_{i_1,\ldots,i_n}$. In this paper we apply tensor decomposition to unsupervised tensor reconstruction, which consists two different tasks: tensor completion and tensor denoising.

Tensor Completion For tensor completion, some of the entries in $\hat{\mathcal{T}}$ for the input tensor are missing. Our goal is to complete these entries based on the entries which are observed (*i.e.*, not missing). We define a binary tensor $\mathcal{W}$ which represents the entries are not missing. $\mathcal{W}$ has the same shape with $\hat{\mathcal{T}}$. $w_{i_1,\ldots,i_N}$ equals to 1 (*resp.*, 0) means $\hat{t}_{i_1,\ldots,i_N}$ is observed (*resp.*, missing). Assume $\mathcal{T}$ is the tensor we obeserved, we have

$$t_{i_1,\ldots,i_N} = \hat{t}_{i_1,\ldots,i_N} \times w_{i_1,\ldots,i_N}. \tag{1}$$

Tensor Denoising For tensor denoising, the original tensor $\hat{\mathcal{T}}$ is observed with random noise. We define a noise tensor $\mathcal{E}$, which represents the random noise (white Gaussian noise in our case). $\mathcal{E}$ has the same shape with $\hat{\mathcal{T}}$. Similarly to image completion, assume $\mathcal{T}$ is the tensor we observed, we have

$$t_{i_1,\ldots,i_N} = \hat{t}_{i_1,\ldots,i_N} + e_{i_1,\ldots,i_N}. \tag{2}$$

Tensor Decomposition Tensor decomposition methods have shown its powerful performance for both tensor completion and denoising [Kolda and Bader, 2009]. Tensor decomposition is to decompose $\mathcal{T}$ into several linear combinations of smaller core tensors as an approximation estimation, which can be expressed as follow:

$$\mathcal{T} \approx \mathcal{X} = \ll \mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)} \gg, \tag{3}$$

where $\mathcal{G}^{(k)}$ are called core tensors on k-th-mode and $\mathcal{X}$ is the approximated tensor. Here we focus on the tensor decomposition methods that the model parameters will not grow exponentially by the increase of the tensor order of $\mathcal{T}$ like CANDECOM/PARAFAC (CP) [Harshman and others,

Algorithm 1 Gradient Descent for Tensor Decomposition.

Require: Input data $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$.
Ensure: Model parameters $\{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M$.
1: Randomly initialize $\{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M$.
2: **while** Not converged **do**
3: Calculate loss function L based on Eq. 5.
4: Update $\{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M$ based on Eq. 6.
5: **end while**N

1970], Tensor Train (TT) [Oseledets, 2011] and Tensor Ring (TR) [Zhao *et al.*, 2016] decompsition models, which means excluding Tucker decomposition [Tucker, 1966] models. For example, with TR decomposition, all the core tensors are three-ordered tensor (*i.e.*, $\mathcal{G}^{(k)} \in \mathbb{R}^{R_{k-1} \times I_k \times R_k}$). Let $\mathbf{G}_{i_k}^k$ be the i_k-th lateral slice of the k-th core tensor, TR decomposition can be expressed as:

$$x_{i_1,\ldots,i_N} = Tr\left(\prod_{k=1}^{N} \mathbf{G}_{i_k}^{(k)}\right), \tag{4}$$

where $Tr(\cdot)$ denotes calculating the trace on matrices. Here we call R_k as the rank for the k-th mode and we have $R_0 = R_N$ for TR decomposition. For the details of CP and TT decomposition methods, please refer to the references.

Objective Function When applying different tensor decomposition methods, assume we have M different groundtruth tensors $\{\hat{\mathcal{T}}_1, \ldots, \hat{\mathcal{T}}_M\}$, the corresponding binary tensors $\{\mathcal{W}_1, \ldots, \mathcal{W}_M\}$ (we set $\mathcal{W} = \mathbf{1}$ for tensor denoising task), the input tensors $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$ which generated by the groundtruth and binary tensors, and the reconstructed tensors $\{\mathcal{X}_1, \ldots, \mathcal{X}_M\}$ where $\mathcal{X}_i = \ll \mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)} \gg$. The objective function for tensor decomposition is to optimize all the core tensors $\{\mathcal{G}_i^{(1)}, \ldots, \mathcal{G}_i^{(N)}\}_{i=1}^M$ with:

$$L = \frac{1}{M}\sum_{i=1}^{M} \|\mathcal{W}_i * (\mathcal{T}_i - \mathcal{X}_i)\|_F^2, \tag{5}$$

where $\|\cdot\|_F$ means Frobenius norm and $*$ denotes element-wise product.

3.2 Gradient Descent for Tensor Decomposition

Since tensor decomposition is to approximate the original tensor with the core tensors and we defined the objective function Eq. 5 for tensor decomposition, we can learn the core tensors with Gradient Descent (GD) algorithm by calculating the partial derivatives of the core tensors on the objective function and update the core tensors. The optimizing process for each iteration can be depicted by:

$$\mathcal{G}_i^{(k)} \leftarrow \mathcal{G}_i^{(k)} - \lambda \nabla_{\mathcal{G}_i^{(k)}} L, \tag{6}$$

where λ stands for learning rate of each iteration.

The overall learning process for GD can be summarized in Alg. 1. We omit the binary mask $\mathcal{W}$ for the input data to simplify the representation. We run GD with maximum of 10000 iterations or if difference of the loss function between two adjacent iterations are close enough, *i.e.*, for iteration t, $|L_t - L_{t-1}| < thd$, where $thd = 1e-4$.

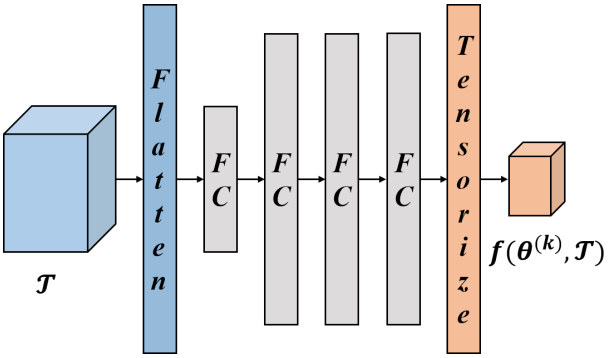

Figure 1: An illustration of the proposed neural network framework which learn one of the core tensors. We first flatten the input tensor by vectorization and then use four fully-connected (FC) layer to learn the core tensors.

3.3 Tensor Decomposition by Core Tensor Networks (CTN)

Traditional tensor decomposition methods always treat every tensor as an individual decomposition process without searching for the correlation among them. We designed a simple neural network to learn the core tensors based on the input tensor called Core Tensor Networks (CTN), which can build global network models for all the input tensors and learn the core tensors based on the networks.

Assume $f(\theta^{(k)}, \mathcal{T})$ is the function representation for the network to learn the k-th core tensor with network parameter θ, our framework can be illustrated as in Figure 1. We first vectorize the input tensor and then feed the input vector to three fully-connected (FC) layers with an output size of 16, 32 and 32 correspondingly. The output size of the first layer is relatively small compared with the corresponding input size for a low-rank compression. Each of the above three FC layers is followed by an ELU [Clevert *et al.*, 2015] activation. Finally, we add another FC layer which has the same size of output with the model parameters for $\mathcal{G}^{(k)}$ and tensorize the output to the same shape to $\mathcal{G}^{(k)}$. All the input tensor $\mathcal{T}_i$ share the same network parameters while for different cores the parameters differ (*i.e.*, $\theta^{(a)} \neq \theta^{(b)}$ for $a \neq b$). We call $f(\theta, \mathcal{T}_i)$ as the *main core tensors*. We replace the original k-th mode core tensor for input tensor $\mathcal{T}_i$ with:

$$\mathcal{G}_i^{(k)} = f(\theta^{(k)}, \mathcal{T}_i) + \mathcal{B}_i^{(k)}, \tag{7}$$

where $\mathcal{B}_i^{(k)}$ is called as *bias core tensors* which are added to the main core tensors $f(\theta^{(k)}, \mathcal{T}_i)$, which for different input tensors we have different bias core tensors.

Similar to Algorithm 1, we can learn CTN by GD. We define each iteration for the main and bias core tensors as:

$$\theta^{(k)} \leftarrow \theta^{(k)} - \lambda \nabla_{\theta^{(k)}} L, \mathcal{B}_i^{(k)} \leftarrow \mathcal{B}_i^{(k)} - \lambda \nabla_{\mathcal{B}_i^{(k)}} L. \tag{8}$$

The whole algorithm to learn CTN is depicted in Alg. 2. The advantage of the proposed network structure is two-folded. On the one hand, algorithms for tensor decomposition models like Alternating Least Squares (ALS)[Harshman and

Algorithm 2 Gradient Descent for Core Tensor Networks.

Require: Input data $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$.
Ensure: Model parameters $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
Ensure: Model parameters $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^M$.
1: Randomly initialize $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
2: Initialize $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^M$ with zeros.
3: **while** Not converged **do**
4: Calculate loss function L based on Eq. 5.
5: Update $\theta^{(k)}, \{\mathcal{B}_i^{(k)}\}_{i=1}^M$ for all k based on Eq. 6.
6: **end while**

others, 1970; Carroll and Chang, 1970] and Stochastic Gradient Descent (SGD)[Yuan *et al.*, 2019c; Huang *et al.*, 2013] always randomly initialize the core tensors and update the core tensors iteratively. The values of the core tensors should be related to the original values of the tensors because we can compute the core tensors with the inverse tensor and the original tensor. With CTN we can initialize the core tensors with random projections of the input tensors, which can improve the learning speed to converge faster for tensor decomposition. On the other hand, the main core tensors share the same model parameter for all the input tensors, which means we can combine different tensor decomposition process for better performance and robust results.

3.4 Transfer Learning for CTN

The most powerful advantage for neural networks (NN) is the scalability and generalizability of different data. NN can learn models on training data and memorize the data patterns then infer the results with fast or even without re-training on new data. Assume we have another training set contains M' tensors $\{\mathcal{T}'_1, \ldots, \mathcal{T}'_{M'}\}$, the goal of transfer learning for CTN (tCTN) is to pretrain the model parameters for main core tensors with the training set and finetune the model to the test set $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$. Note that the bias core tensors $\mathcal{B}^{(k)}$ are learned for the input tensors individually. They are re-initialized and not finetuned for testing.

The simplest way to implement tCTN is to ignore the bias core tensors and let the composition of main core tensors be the reconstructed tensors on the training set, which means:

$$\mathcal{T}'_i \approx \mathcal{X}_i = \ll f(\theta^{(1)}, \mathcal{T}'_i), \ldots, f(\theta^{(N)}, \mathcal{T}'_i) \gg . \quad (9)$$

However, due to the possible data distribution shift between the training set and test set, the training process above may be harmful to the test set. Also, ignoring the bias core tensors will make the objective too difficult for CTN, which may cause a failure of convergence. Inspired by the Meta-Learning (or learning to learn) methods [Finn *et al.*, 2017; Nichol *et al.*, 2018], which learn models applicable to different tasks, we decompose the learning process for main core tensors and bias core tensors into two different tasks. For each iteration, we sample a batch of tensors from the training set, we initialize the bias core tensors and optimize the bias core tensors with the current main core tensors computed with the current network parameters. Then update the parameters for the main core tensors based on the learned bias core tensors for the current batch.

Algorithm 3 Transfer Learning for Core Tensor Networks.

Require: Training data $\{\mathcal{T}'_1, \ldots, \mathcal{T}'_{M'}\}$.
Require: Test data $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$.
Ensure: Model parameters $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
Ensure: Model parameters $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^M$.
1: Randomly initialize $\{\theta^{(1)}, \ldots, \theta^{(N)}\}$.
2: **for** $iter$ in $1, \ldots, iter_{max}$ **do**
3: Sample a batch $\{\mathcal{T}'_{b_1}, \ldots, \mathcal{T}'_{b_m}\}$ from training set.
4: Initialize $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^M$ with zeros.
5: **for** p in $1, \ldots, \gamma$ **do**
6: Calculate L for $\{\mathcal{T}'_{b_1}, \ldots, \mathcal{T}'_{b_m}\}$ based on Eq. 5.
7: Update $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^M$ based on Eq. 6.
8: **end for**
9: Calculate L for $\{\mathcal{T}'_{b_1}, \ldots, \mathcal{T}'_{b_m}\}$ based on Eq. 5.
10: Update $\theta^{(k)}$ for all $k \in [N]$ based on Eq. 6.
11: **end for**
12: Initialize $\{\mathcal{B}_i^{(1)}, \ldots, \mathcal{B}_i^{(N)}\}_{i=1}^M$ with zeros.
13: **while** Not converged **do**
14: Calculate L for $\{\mathcal{T}_1, \ldots, \mathcal{T}_M\}$ based on Eq. 5.
15: Update $\theta^{(k)}, \{\mathcal{B}_i^{(k)}\}_{i=1}^M$ for all k based on Eq. 6.
16: **end while**

The detailed algorithm is depicted in Alg. 3. γ is a hyper-parameter that determines how precisely we want to learn the bias core tensors in each batch update. When $\gamma = 0$, the algorithm equals to the method in Eq. 9. With the help of transfer learning and meta-learning, CTN can learn tensor decomposition with significantly fewer iterations and better performance.

4 Experiments

In this section, we first introduce the details and ablative study for our proposed CTN (Alg. 2) and tCTN (Alg. 3). Then compare them with state-of-the-art image completion and denoising methods. Finally. we visualize the outputs of CTN to show insights of CTN.

4.1 Implementation Details

To prove the effectiveness of our proposed method, we test CTN and tCTN on unsupervised image completion and denoising tasks. Peak Signal-to-Noise Ratio (PSNR, the higher the better) and Relative Square Error (RSE, the lower the better) are used for evaluation. We run all the experiments on images with resolution $256 \times 256 \times 3$. The images are reshaped to a 9-order tensor with size $4\times4\times4\times4\times4\times4\times4\times4\times3$. We use Adam optimizer with a learning rate of 0.0001. β_1 and β_2 are set to be 0.9 and 0.999. All these parameters are determined based on the best results by evaluating GD (Alg. 1). The image is down-scaled to 128×128 before feeding into CTN. The whole CTN contains about 14.5M parameters.

We use DIV2K [Agustsson and Timofte, 2017], which is a popular open-access dataset for image denoising and enhancing, to evaluate our proposed method. All the experiments in the rest of the paper is evaluated on the validation set of DIV2K, which contains 100 high-quality

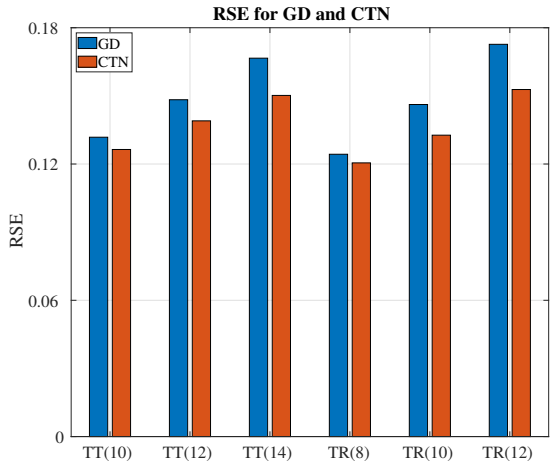

Figure 2: Comparison of GD and CTN under different settings of tensor decomposition.

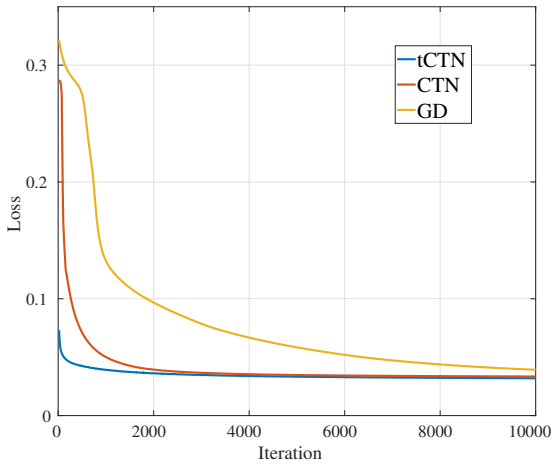

Figure 3: Covergence performance of GD, CTN and tCTN.

Metrics/Settings	0	5	10	20	50
RSE↓	0.1284	0.1246	0.1212	0.1201	0.1211
Batch per Second↑	26.53	19.53	13.44	9.93	4.68

Table 1: Comparison of different choices of γ for tCTN.

images. We use X2 images collection in DIV2K and reshape the images to 256×256 for the evaluation.

4.2 Ablative Studies

Effect of CTN (Alg. 2) Rank is the most important hyperparameter to select for tensor decomposition methods. To evaluate the effect of proposed CTN, we test TT and TR decomposition on image completion with 0.9 missing rate under different fixed rank settings (performance for CP decomposition is not good enough so we omit the results). For the sake of fair comparisons, we evaluate both GD (Alg. 1) and CTN (Alg. 2) on TT with rank in $\{10, 12, 14\}$ and TR with rank in $\{8, 10, 12\}$. RSE results can be found in Table 2. We can see that CTN always performs better than GD on the same setting of decomposition format and rank. We claim that is because CTN can learn the structure information shared among all the images in the test set. Note that for the rest part of the paper, we choose to use TR decomposition for our proposed model due to results in Table 2 and also some other cross-validation tests.

The choice of γ in tCTN (Alg. 3) We train tCTN on the training set of DIV2k which contains 800 images. We train tCTN for 40 epoches with batch size equals to 4. The number of images is relatively small for training a deep neural network model. Also, our proposed model is based on unsupervised learning, which means the quality for training data is not that important and the data seeking much easier.

γ is the most important hyperparameter for tCTN to determine how many steps to learn the bias core tensor before updating the main core tensors. Larger γ will reduce the difficulty to learn CTN for better performance while introducing more time to train the network. We vary γ in $\{0, 5, 10, 20, 50\}$ for tCTN and evaluate the performance on image completion task with missing rate 0.9. For a fair comparison, we evaluate the results after training 300 iterations on the test set. The results can be found in Table 1. Here we also put the training time for each batch with different settings of γ. As we can see, when γ goes larger, RSE performance becomes better for tCTN. While when γ is large enough, RSE remains consistent and does not grow. $\gamma = 20$ and $\gamma = 50$ have close RSE but $\gamma = 50$ consumes much more training time. We found that $\gamma = 20$ is the best choice based on the test results. γ in Alg. 3 is set to be 20 in the rest of the paper. Note that the whole training process for tCTN takes about 2.24 hours.

Speed Analyses In this section, we discuss the speed of our proposed method. We evaluate GD (1), Alternating Least Square (ALS), CTN (2) and tCTN (3) on the test set with rank 10 TR decomposition. GD and ALS are two of the most popular algorithm to solve tensor decomposition. All the algorithms are tested on the same platform (Tensorflow implementation, Titan Xp GPU). The results are shown in Table 2. We can see that CTN and tCTN outperform GD and ALS on both speed and accuracy. Besides, tCTN uses much less time to give out a close performance compared with CTN. These results proved the power of CTN in processing multiple tensor decomposition.

In Figure 3, we show the change of loss function in Eq. 5 with the number of iterations during the optimizing process for convergence anylysis. We can see that CTN converges much faster (about four times faster) compared with GD, while tCTN can sharply reduce the loss function with a few iterations. Here both CTN and tCTN takes about 1.16 unit of time for each iteration compared with GD. Note that CTN does not use auxiliary data to train but still performs so well, while tCTN uses a small amount of auxiliary data to pretrain but converges much more faster.

Metrics/Algorithms	GD	ALS	CTN	tCTN
RSE↓	0.1243	0.1229	0.1205	0.1201
Second per Image↓	22.2	109.3	6.55	0.57

Table 2: Comparison of GD, ALS, CTN and tCTN for speed and accuracy.

Rate	Metric	BCPF	TT-WOPT	TR-ALS	TRLRF	CTN	tCTN
0.9	PSNR↑	19.69	23.21	22.22	22.27	23.50	23.53
	RSE↓	0.1868	0.1262	0.1396	0.1388	0.1205	0.1201
0.7	PSNR↑	25.18	25.36	24.51	26.82	27.78	27.79
	RSE↓	0.0993	0.0972	0.1072	0.0822	0.0736	0.0735

Table 3: Comparison with state-of-the-art image completion methods.

4.3 Comparison with State-of-the-Art Methods

In this section, we compare our proposed CTN and tCTN with state-of-the-art (SOTA) methods that have the same objective function in Eq. 5, for image completion and denoising, e.g., BCPF [Zhao *et al.*, 2015], TT-WOPT [Yuan *et al.*, 2017], TR-ALS [Wang *et al.*, 2017], TRLRF [Yuan *et al.*, 2019b]. We evaluate both image completion and denoising task with PSNR and RSE.

Results for image completion are shown in Table 3. We evaluate all the SOTA methods for two different missing rates: 0.7 and 0.9. For both CTN and tCTN, we use TR decomposition with rank 16 and 8 for the missing rate of 0.7 and 0.9 correspondingly.

Results for image denoising are shown in Table 4. Note that we implement the algorithms by ourselves for both TT-WOPT and TR-ALS for image denoising. We use TR decomposition with rank 16 to test the images with 10dB and 20dB white Gaussian noise, correspondingly.

We can see that CTN and tCTN achieve better accuracy on both image completion and denoising tasks. All of these methods are unsupervised and based on CPU implementation. Among them, the fastest algorithm is BCPF, which takes about 70 seconds for each 0.9 missing image completion. Since the platform for these methods and our proposed method are different, but we can still see the speedup obtained by CTN.

4.4 Qualitative Results

We visualize the network outputs for the first FC layer 1 in CTN to show the details of CTN learning process. The size of the weights for the first layers are $(W \times H \times 3) \times 16$, where W and H are the input size of the images. We average these weights on the last dimension, then reshape them to $W \times H \times 3$. Finally, we multiply this weight to the reconstructed image and whiten the results to visualize the outputs. Each of the pixels is related to the corresponding pixel of the original images. We can see how CTN learns the main core tensors with different weights from different pixels.

Results are shown in Figure 4. We have four groups of examples, two for image completion and two for image denoising. For each group, we put the groundtruth images, observed images and the reconstructed images on the first row and the visualization for the outputs on the second row. Since the images are reshaped into nine-order tensor, we formulate the outputs for each order into 3×3 squared images.

Level	Metrics	TT-WOPT	TR-ALS	CTN	tCTN
10dB	PSNR↑	19.24	19.69	20.17	20.33
	RSE↓	0.0849	0.0774	0.0686	0.0682
20dB	PSNR↑	19.48	20.03	20.23	20.34
	RSE↓	0.0804	0.0723	0.0682	0.0675

Table 4: Comparison with state-of-the-art image denoising methods.

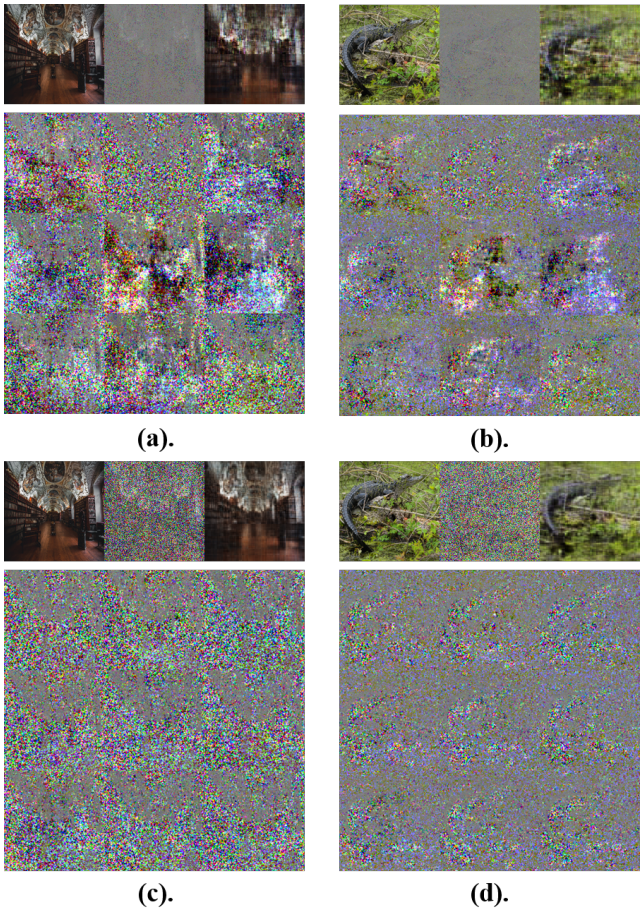

Figure 4: Visualization of the average output for the first layer of CTN. (a) and (b) are for image completion while (c) and (d) are for image denoising.

For image completion, the outputs of the first layer are more sharp compare with these outputs for image denoising. We can see for image completion the outputs change significantly with different core tensor networks. Both image completion and denoising outputs pay more attention to the foreground (*e.g.*, the reptile in (b). and (d).) and less attention on the background (*e.g.*, the grass in (b). and (d).).

5 Conclusion

In this paper, we propose an efficient tensor decomposition algorithm that aims to learn a global mapping from input tensors to latent core tensors. We train a deep neural network to model the global mapping, and then it can be applied to decompose a newly given tensor with high efficiency. Furthermore, we learn the initial values of the network based on meta-learning methods. With the help of pretrained meta-learning based core tensor networks, our proposed method can learn tensor decomposition extremely fast and accurately. Experimental results for image completion and denoising show that our proposed method significantly improves tensor decomposition speed and accuracy.

References

[Acar *et al.*, 2011] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[Agustsson and Timofte, 2017] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *CVPR Workshops*, 2017.

[Bengua *et al.*, 2017] Johann A Bengua, Ho N Phien, Hoang Duong Tuan, and Minh N Do. Efficient tensor completion for color image and video recovery: Low-rank tensor train. *IEEE Transactions on Image Processing*, 26(5):2466–2479, 2017.

[Carroll and Chang, 1970] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[Clevert *et al.*, 2015] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[Filipović and Jukić, 2015] Marko Filipović and Ante Jukić. Tucker factorization with missing data with application to low-n-rank tensor completion. *Multidimensional Systems and Signal Processing*, 26(3):677–692, Jul 2015.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

[Harshman and others, 1970] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an "explanatory" multimodal factor analysis. 1970.

[Huang *et al.*, 2013] Furong Huang, UN Niranjan, Mohammad Umar Hakeem, and Animashree Anandkumar. Fast detection of overlapping communities via online tensor methods. *arXiv preprint arXiv:1309.0787*, 40:43, 2013.

[Jia and Gong, 2005] Kui Jia and Shaogang Gong. Multi-modal tensor face for simultaneous super-resolution and recognition. In *ICCV*, 2005.

[Kolda and Bader, 2009] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[Lebedev *et al.*, 2015] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan V. Oseledets, and Victor S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *CoRR*, abs/1412.6553, 2015.

[Liu *et al.*, 2012] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2012.

[Liu *et al.*, 2019] Hanpeng Liu, Yaguang Li, Michael Tsang, and Yan Liu. Costco: A neural tensor completion model for sparse tensors. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[Maruhashi *et al.*, 2018] Koji Maruhashi, Masaru Todoriki, Takuya Ohwa, Keisuke Goto, Yu Hasegawa, Hiroya Inakoshi, and Hirokazu Anai. Learning multi-way relations via tensor decomposition with neural networks. In *AAAI*, 2018.

[Nichol *et al.*, 2018] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

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

[Semerci *et al.*, 2014] Oguz Semerci, Ning Hao, Misha E Kilmer, and Eric L Miller. Tensor-based formulation and nuclear norm regularization for multienergy computed tomography. *IEEE Transactions on Image Processing*, 23(4):1678–1693, 2014.

[Tucker, 1966] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[Wang *et al.*, 2017] Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Efficient low rank tensor ring completion. In *ICCV*, 2017.

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

[Wu *et al.*, 2019] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh V. Chawla. Neural tensor factorization for temporal interaction learning. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019.

[Yu *et al.*, 2017] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *CVPR*, 2017.

[Yu *et al.*, 2019] Jinshi Yu, Chao Li, Qibin Zhao, and Guoxu Zhou. Tensor-ring nuclear norm minimization and application for visual: Data completion. In *ICASSP*, 2019.

[Yuan *et al.*, 2017] Longhao Yuan, Qibin Zhao, and Jianting Cao. Completion of high order tensor data with missing entries via tensor-train decomposition. In *ICONIP*, 2017.

[Yuan *et al.*, 2019a] Longhao Yuan, Chao Li, Jianting Cao, and Qibin Zhao. Randomized tensor ring decomposition and its application to large-scale data reconstruction. In *ICASSP*, 2019.

[Yuan *et al.*, 2019b] Longhao Yuan, Chao Li, Danilo Mandic, Jianting Cao, and Qibin Zhao. Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion. In *AAAI*, 2019.

[Yuan *et al.*, 2019c] Longhao Yuan, Qibin Zhao, Lihua Gui, and Jianting Cao. High-order tensor completion via gradient-based optimization under tensor train format. *Signal Processing: Image Communication*, 73:53–61, 2019.

[Zadeh *et al.*, 2017] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250*, 2017.

[Zhao *et al.*, 2015] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1751–1763, 2015.

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