Information Sciences 658 (2024) 120024

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A neural tensor decomposition model for high-order sparse data recovery

Tianchi Liao [a], Jinghua Yang [c], Chuan Chen [b,*], Zibin Zheng [a]

[a] School of Software Engineering, Sun Yat-sen University, Zhuhai 519000, China
[b] School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China
[c] School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China

ARTICLE INFO

Keywords:
Tensor completion
Convolutional neural networks
Tensor-ring decomposition
Rank robustness

ABSTRACT

Tensor decomposition has attracted wide attention in the low-rank tensor completion (LRTC) problem because of its marvelous recovering ability to missing entries. However, previous LRTC methods are generally based on linear and shallow models, which are prone to overfitting when the data is sparse, resulting in significantly degraded performance. Meanwhile, the models are highly sensitive and suffer from the difficult rank selection problem. To address these issues, we propose an effective and novel tensor-ring (TR) decomposition method based on the convolutional computation (ConvTR), which can be regarded as a natural extension of deep learning models for the LRTC problem. Specifically, ConvTR employs a multi-layer convolutional neural network (CNN) to model the complex interactions between TR factors. Each element in the index vector of the observation tensor can be embedded as a corresponding tensor slice in the factor tensor decomposed by the TR model. These individual slice matrices are then concatenated to get a wider matrix used for extracting the nonlinear features by feeding them into a 2D convolutional layer. A fully-connected layer is utilized to aggregate the final convoluted features to a scalar value, which is the desired missing entry indexed by the original index vector exactly. Extensive experiments on various common datasets verified the effectiveness of the proposed method and demonstrated its superior to the traditional TR-based completion methods and other state-of-the-art network-based methods.

1. Introduction

Tensors, also known as N-way arrays, provide a natural structure to high-order data [1]. Multi-modal and large-scale data displayed in the tensor form are common in various types, such as color images, video, social networks, knowledge graphs, etc. The definition and computation of tensor are not only popular in mathematics [2], but also a fundamental problem in many applications, such as computer vision [3], machine learning [4], data mining [5]. However, real-world tensors are often sparse and incompleteness due to unexpected human or natural factors. Recently tensor completion, which predicts missing entries from partial observations, has gained widespread interest. In the real world, many multi-modal data tend to exhibit low-rankness because of their inherent local similarity, global symmetry, and sample redundancy. Utilizing known observations to recover missing entries of targeted tensor

* Corresponding author.
E-mail address: chenchuan@mail.sysu.edu.cn (C. Chen).

https://doi.org/10.1016/j.ins.2023.120024
Received 23 September 2023; Received in revised form 10 November 2023; Accepted 15 December 2023

Available online 22 December 2023

together with low-rank constraints is often called the low-rank tensor completion (LRTC) problem. LRTC has become the key issue in many applications, including visual image completion [6], recommender systems [7], and link prediction [8], etc.

Since the key to tensor completion is to find out the relationship between the observations and missed entries, which can be captured by possible latent factors decomposed from the target tensor. Therefore, the most commonly accepted method in LRTC problems is based on tensor decomposition or its variations. By assuming that the tensor has a compact underlying structure, tensor decomposition allows entries to be reconstructed from low-rank factors by multilinear multiplication, thus effectively transforming higher-order data into latent factors. Whereupon various among different tensor decomposition models are proposed by designing different latent factor structures. Among them, the classical decomposition methods include CANDECOMP/PARAFAC (CP) [9] and Tucker [10]. In CP decomposition, the target tensor is decomposed into linear combinations of rank-1 tensors, in which the rank is defined as the minimum number of outer products of the vectorized factors. Whereas, the CP rank is NP-hard to calculate in general and it is not flexible enough to portray the correlation between the different factors of the tensor [11]. Compared with CP, the Tucker decomposition can utilize more correlations, in this case, the rank is defined as the set of matrix ranks after the tensor matricization along all modes. However, the Tucker rank [12] remains unbalanced and the decomposition does not apply to high-order tensor, whose storage capacity grows exponentially with the number of data dimensions. Moreover, the advancement of information technology has led to increasingly abundant information and growing volume of data, posing great challenges for conventional decomposition methods to handle high-dimensional data, which results in their rapidly deteriorating performance.

Tensor networks [13][14], considered as a generalization of tensor decompositions, have emerged as a promising powerful tool for analyzing large-scale datasets. At the moment, the most popular tensor networks are based on the Tensor-Train (TT) [15] and Tensor-Ring (TR) [16] representations. TT decomposition has been widely used in research as a basic building block of complex tensor networks. It decomposes the target tensor into two matrices and several third-order tensors, which may not be the optimum for specific data [17]. As an improved version, TR decomposition can overcome this issue, where TR approximates a higher-order tensor using a cyclic multi-linear product of a sequence of third-order core tensors. Furthermore, TR decomposition is now shown to outperform other decomposition methods in capturing the relevance of real data [18]. Due to the excellent expressive performance for higher-order tensors, TR decomposition has been increasingly applied to LRTC problems in recent years. Nevertheless, the TR decomposition method inherits the inherent drawbacks of tensor decomposition. First, the optimal rank as the input of the algorithm needs to be determined in advance, while the optimal TR rank remains unknown in most realistic scenarios. Therefore, the model behaves sensitively and the performance is largely influenced by the rank [19]. Additionally, these mathematics-based low-rank decomposition methods rely more heavily on observed values. When faced with high missing ratios or sparse observed sets, the recovery results become less ideal [20]. More importantly, the nonlinear information in the data may obscure the low rankness and the model performance may be hindered by the multi-linear hypothesis in the decomposition, making it fail to capture the nonlinear features [21].

To learn the complex interaction within the tensor, the model needs to have the ability to capture and express the potential nonlinearity of data. Recently, many scholars have applied the existing deep learning methods to the tensor completion problem, and obtained better results than the multi-linear models [22]. Xu et al. [23] proposed a nonlinear Tucker model (InfTucker), which uses Tensor-variate Gaussian to extend the Tucker model to infinite feature space, so it has better performance in modeling complex nonlinear interactions. However, the Kronecker product involved in the solving process of InfTucker and its variant [24], is likely to bring about a high computational cost. Besides, due to parameter redundancy, existing tensor completion algorithms based on deep learning are often prone to overfitting in sparse or limited training data. Meanwhile, the model generalization is also suboptimal due to the inefficient parameters.

To tackle the above limitations, we propose a new nonlinear tensor decomposition model ConvTR (**Conv**olutional **T**ensor **R**ing Decomposition). Exploiting the expressive power of Convolutional Neural Network (CNN), the model is parameter efficient and can avoid overfitting to sparse and limited training data. The CNN architecture is used to reconstruct the factors of TR decomposition to model the non-linearity in relational data and eliminate the linear limitation of dot product used in conventional tensor decomposition. Specifically, the ConvTR model concatenates the resulting decomposition factors and relates the rank R to the size of the convolution filter. In this way, while solving the problem of rank sensitivity, the model learns more nonlinear information. Therefore, ConvTR can be excellently used for tensor completion of sparse tensors with complex underlying structures. We evaluate the proposed model, and perform extensive experiments on a variety of common sparse tensors. Our model invariably outperforms state-of-the-art tensor completion models (both linear and nonlinear), while being robust for a different choice of rank. In summary, the contributions of this paper are as follows:

1. We define the generalized TR decomposition method and propose a new CNN-based nonlinear tensor decomposition model, ConvTR, to characterize the nonlinearity and correlation of decomposition factors.
2. ConvTR has excellent robustness and can always obtain a relatively stable solution even if the TR rank is improperly given.
3. Extensive completion experiments are conducted on three different standard datasets, demonstrating the leading performance capabilities regardless of the type of the data.

2. Related works

Recently, TR-based tensor decomposition models have been extensively studied and have shown the ability to handle high-order tensors. Many works [25][26] designed different completion algorithms based on TR and total variance regularization, all of them achieved good results on remote sensing images. Meanwhile, sparse regularization is also considered for model generalizability, and

Asante et al. [27] completed the data tensor by learning a core tensor with sparsely constrained TR representation. Wang et al. [17] first extended alternating least squares (ALS) to TR decomposition (TRALS). Then, Yuan et al. [28] further proposed TR weighted optimization (TRWOPT), which deploys the gradient descent algorithm to optimize the model, resulting in slow convergence. Although ALS and gradient-based algorithms do not require tedious parameter tuning, the performance of these algorithms is quite sensitive to rank selection. A new rank selection method was proposed by Sedighin et al. [29] for automatically searching the optimal TR rank (TRAR). In fact, TRAR consists of an internal TRALS iteration and an external rank-increasing iteration, which reduces the efficiency of the model. To speed up the convergence, exploring data topologies using TR decomposition with tensor networks [30] is a current hot direction. Zheng et al. [12] proposed the fully connected tensor network (FCTN), which describes the correlation by establishing the relationship between any decomposition factors, but its computational complexity grows exponentially with the increase of the tensor order. Then, several works [31][32] improved the FCTN to reduce the computational complexity of the tensor network.

Whereas, these TR-based completion methods encounter severe degradation problems when the chosen rank is far from the true rank. Thus, some robust TR completion methods have been widely proposed. To avert artificial rank selection, Long et al. [33] proposed a Bayesian TR decomposition without parameter-tuning, where ranks can be acquired by Bayesian deduce. Methods based on rank minimization usually convert into convex surrogates to minimize tensor rank. In recent years, newly-designed norms have been used to approximate tensor rank functions. Yuan et al. [18] carved the relation between tensor rank and factor rank and proposed a new method called tensor ring lower-order factor (TRLRF), which effectively reduced the burden of rank selection by introducing nuclear norm regularization to the potential TR factor. Yu et al. [34] first designed a tensor circle unfolding method and applied the tensor nuclear norms to the model. Li et al. [35] combined TR rank and $\ell_{p,\epsilon}$-norm to form a robust tensor completion method that has strong generalization for all types of data. Huang et al. [36] proposed a robust tensor-ring completion method, which separates the data from potential low-order components and sparse components, and added nuclear norm and ℓ_1 norm to constrain the model, respectively. Yu et al. [37] used the Frobenius norm of the latent TR nuclear to express the target low rank and sparsity of the tensor, which makes the model more robust while reducing its computational effort. Nevertheless, to the best of our knowledge, most of the existing tensor norm-based completion methods claim expensive computational expenses for each iteration in the optimization process. Moreover, the aforementioned methods can only alleviate rank sensitivity to a fairly small extent, especially for large-scale tensors.

Since the above methods are linear-oriented regardless of TR decomposition or norm minimization based, they are powerless for processing the incomplete tensors with nonlinearity. Facing this challenge, many scholars apply deep learning to tensor completion. Wu et al. [38] proposed a deep tensor decomposition network with bias, which takes the horizontal and lateral vectors constructs to observation tensors as inputs, and constructing them into a multilayer perceptron (MLP) network, respectively. Liu et al. [21] used convolution to perform CP decomposition accurately and efficiently, and proposing a completion method (CoSTCo) for large-scale, highly sparse data. Wu et al. [39] proposed a neural network based tensor factorization model (NTF) by replacing the multi-linear operation of CP decomposition with MLP. Chen et al. [40] proposed a novel nonlinear tensor machine, which establishes the relationship between neural network and tensor algebra to describe the nonlinear information of data. A series of network models based on tensor decomposition have been proposed, and they are widely used in various situations. However, the majority of these network models remind tuning abundant parameters to avoid overfitting. In addition, the above completion methods are mostly designed for a specific application, e.g., image/video recovery, link prediction, spatio-temporal analysis, etc., which leads to a lack of generalization of these models. Till now, few models can handle data of different types robustly.

Therefore, facing the common defects of traditional tensor decomposition methods and the overfitting problems caused by the complex parameters of network models, it is necessary to explore a valid and general completion method to address these limitations. In this work, we propose a neural tensor decomposition model ConvTR with a multi-layer convolution network, which is almost immune to rank fluctuation and robust to overfitting problems. Meanwhile, the proposed model does not demand to convert the objective function into a preview design neural, nor involves expensive computational optimization.

3. Tensors and notations

For the tensor completion, the details about some definitions and tensor properties can be found in [1]. Throughout this paper, Table 1 lists the relevant notations commonly used in this paper. And beyond that, for a third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, we denote its (i,j)-th mode-1, mode-2, and mode-3 fibers as $\mathcal{X}(:,i,j)$, $\mathcal{X}(i,:,j)$, and $\mathcal{X}(i,j,:)$. We use the Matlab notation $\mathcal{X}(i,:,:)$, $\mathcal{X}(:,i,:)$, and $\mathcal{X}(:,:,i)$ to denote the i-th horizontal, lateral and frontal slice, respectively. Moreover, the superscript (k) is used to denote the k-th feature factor of the tensor decomposition, e.g., $\mathcal{X}^{(k)}$ represents the k-th latent tensor of the TR decomposition.

3.1. Tensor decomposition

Definition 1 *(CP Decomposition).* [1] The CP decomposition factorizes a tensor into the sum of R component rank-one tensors. For example, given an Nth-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we write it as:

$$\mathcal{X} = \sum_{r=1}^{R} U_r^{(1)} \circ U_r^{(2)} \circ, \ldots, \circ U_r^{(N)}, \tag{1}$$

where R is an integer, $U_r^{(k)} \in \mathbb{R}^{I_k}$. The factor matrices refer to the combination of the vectors from the rank-one components, i.e., $U^{(k)} = [U_1^{(k)}, U_2^{(k)}, \ldots, U_R^{(k)}] \in \mathbb{R}^{I_k \times R}$. Therefore, the entry value of tensor $\mathcal{X}$ can be expressed as:

Table 1
Tensor Notation.

Symbol	Definition
$\mathcal{X}$, X, $\boldsymbol{x}$, x	tensor, matrix, vector, scalar
R	rank of tensor decomposition
$x_{i_1 i_2 \cdots i_n}$	tensor entry value at index $(i_1, i_2, \cdots, i_n)$
$U^{(k)}$	k-th CP decomposition factor matrix $\in \mathbb{R}^{R_{k-1} \times I_k \times R_k}$
$\mathcal{U}^{(k)}$	k-th TR decomposition factor tensor $\in \mathbb{R}^{R_{k-1} \times I_k \times R_k}$
$U_r^{(k)}$, $U_{i_k}^{(k)}$	k-th CP decomposition factor vector $\in \mathbb{R}^{I_k}, \in \mathbb{R}^R$
$\mathcal{U}_{i_k}^{(k)}$, $\mathcal{U}^{(k)}(:, i_k, :)$	k-th TR decomposition factor matrix $\in \mathbb{R}^{R \times R}$
$\bigtriangledown_{\mathcal{X}}$	gradient operator with respect to $\mathcal{X}$
$\sigma(\cdot)$	activation function
$\circ$	vector outer product
$\odot$	the entry-wise product
$\times_n$	tensor n-mode product
$\mathcal{J}$	tensor entry index set
$\|\mathcal{X}\|_F$	Frobenius norm of a tensor

$$x_{i_1 i_2 \cdots i_n} = \sum_{r=1}^{R} U_{i_1,r}^{(1)} U_{i_2,r}^{(2)}, \cdots, U_{i_n,r}^{(N)}. \tag{2}$$

Therefore, CP rank is defined as the minimum number of rank-1 tensors needed to represent $\mathcal{X}$ in the decomposition process, i.e., $\boldsymbol{rank_{CP}} = \min_R \{R | \mathcal{X}\}$.

Definition 2 *(Tensor-Ring Decomposition).* [16] Tensor-Ring decomposition represents a higher-order tensor by a sequence of third-order latent tensors multiplied circularly.

$$\mathcal{X} = \sum_{r_1, \cdots, r_n = 1}^{R_1, \cdots, R_n} \mathcal{U}^{(1)}(r_n, :, r_1) \circ \mathcal{U}^{(2)}(r_1, :, r_2) \circ \cdots \circ \mathcal{U}^{(N)}(r_{n-1}, :, r_n), \tag{3}$$

where $\mathcal{U}^{(k)}(r_k - 1, :, r_k) \in \mathbb{R}$ is the vertical fiber and the syntax $[R_1, R_2, \cdots, R_N]$ denotes the TR-rank which controls the model complexity of TR decomposition. The TR factors are denoted by $\left\{\mathcal{U}^{(k)} \in \mathbb{R}^{R_{k-1} \times I_k \times R_k}\right\}_{k=1}^{N}$, where $R_0 = R_N$. Thus, the entries of the tensor $\mathcal{X}$ can be TR represented as:

$$x_{i_1 i_2 \cdots i_n} = \sum_{r_1 \cdots r_n = 1}^{R_1 \cdots R_N} \mathcal{U}^{(1)}(r_n, i_1, r_1) \times \cdots \times \mathcal{U}^{(N)}(r_{n-1}, i_n, r_n). \tag{4}$$

To further describe the concept, we can also rewrite it in the index form, which is

$$x_{i_1 i_2 \cdots i_n} = Trace\left(\prod_{k=1}^{n} \mathcal{U}^{(k)}(:, i_k, :)\right), \tag{5}$$

where $\mathcal{U}^{(k)}(:, i_k, :) \in \mathbb{R}^{R_{k-1} \times 1 \times R_k}$ for $k = 1 \cdots N$ can be regarded as a matrix $\mathcal{U}_{i_k}^{(k)}$, which denotes the i_k-th lateral slice with dimensions $R_{k-1} \times R_k$. The $Trace(\cdot)$ is the matrix trace operation. For convenience, we simply denote the TR decomposition of the tensor $\mathcal{X}$ by $\mathcal{X} = \mathfrak{R}_{TR}(\mathcal{U}^{(1)}, \cdots, \mathcal{U}^{(N)})$.

3.2. Generalized tensor decomposition

As shown in the equations above, the tensor decomposition model can interpret multi-directional interactions by its multi-linear multiplication. Here, taking CP decomposition as an example, we generalize the tensor decomposition model to learn nonlinear feature interactions.

Suppose we obtain the features of tensors by embedding, i.e., $\left\{U_{i_1}^{(1)}, U_{i_2}^{(2)}, \cdots, U_{i_n}^{(N)}\right\} \in \mathbb{R}^R$. We design an operation $\phi(\cdot)$, which converts a set of embedded information into a vector:

$$\phi(U_{i_1}^{(1)}, U_{i_2}^{(2)}, \cdots, U_{i_n}^{(N)}) = U_{i_1}^{(1)} \odot U_{i_2}^{(2)} \odot \cdots \odot U_{i_n}^{(N)}, \tag{6}$$

here $\odot$ is the element-wise product. Clearly, the layer $\phi(\cdot)$ is a simple operation with no additional parameters introduced. Then, we can project the vector ϕ into the output layer of the neural network:

$$x_{i_1 i_2 \cdots i_n} = \sigma\left(\boldsymbol{w}^T\left(\phi(U_{i_1}^{(1)}, U_{i_2}^{(2)}, \cdots, U_{i_n}^{(N)}) + \boldsymbol{b}\right)\right), \tag{7}$$

where σ denotes the activation function, $\boldsymbol{w}$ and $\boldsymbol{b}$ denote the weights and biases.

Lemma 1. *Traditional CP decomposition in Eq. (2) is a special case of generalized CP decomposition in Eq. (7)*

Proof. Let σ be an identity function i.e., $\sigma(x) = x$, the weight $\boldsymbol{w}$ be a uniform vector of 1 $\left(\boldsymbol{w} = [1, \cdots, 1]^T \in \mathbb{R}^R\right)$, the bias $\boldsymbol{b}$ be a zero vector, and $U^{(N)}_{i_1,r}$ be the r-th element in the column vector $U^{(N)}_{i_1}$. Therefore, we have Eq. (2) equals Eq. (7).

Lemma 2. *CP decomposition can be viewed as a special case of TR decomposition.*

Proof. Given an Nth-order tensor $\mathcal{X}$ with its CP decomposition as Eq. (1), it can also be written in TR decomposition form as,

$$\begin{aligned} &\mathcal{X} = \sum_{r=1}^{R_{cp}} U^{(1)}_r \circ, \cdots, \circ U^{(N)}_r = \mathfrak{R}_{TR}(\mathcal{U}^{(1)}, \cdots, \mathcal{U}^{(N)}) \\ &s.t. \quad \mathcal{U}^{(k)}\left(:, i_k, :\right) = diag\left(U^{(k)}_{i_k}\right), \forall k = 1, 2, \cdots, N, \end{aligned} \tag{8}$$

where $diag(\cdot)$ is turning the vector into a diagonal matrix. Hence, CP decomposition can be viewed as a special case of TR decomposition, where the factors $\mathcal{U}^{(k)}, k = 1, \cdots, N$ are of size $R_{cp} \times I_k \times R_{cp}$ and each lateral slice matrix $U^{(k)}_{i_k}$ is a diagonal matrix of size $R_{cp} \times R_{cp}$. For detailed proof, see [16].

By applying the nonlinear scheme to tensor completion, the nonlinear TR decomposition is shown in Theorem 1.

Theorem 1. *Assume that $\mathcal{X}$ is a Nth-order tensor. The TR decomposition of $\mathcal{X}$ can be written as the following generalized tensor decomposition form:*

$$x_{i_1 i_2 \cdots i_n} = \sigma\left(W^T\left(\phi(\mathcal{U}^{(1)}_{i_1}, \mathcal{U}^{(2)}_{i_2}, \cdots, \mathcal{U}^{(N)}_{i_n}) + B\right)\right), \tag{9}$$

where $\mathcal{U}^{(k)}_{i_k} = \mathcal{U}^{(k)}\left(:, i_k, :\right)$ is the factor tensor obtained by TR decomposition, σ denotes the activation function, and the matrix W and matrix B represent the weights and biases, respectively. Furthermore, $\phi(\cdot)$ is represented as a linear operation, which converts a set of embedded factors into a matrix as follows:

$$\phi(\mathcal{U}^{(1)}_{i_1}, \mathcal{U}^{(2)}_{i_2}, \cdots, \mathcal{U}^{(N)}_{i_n}) = \mathcal{U}^{(1)}_{i_1} \times \mathcal{U}^{(2)}_{i_2} \times \cdots \times \mathcal{U}^{(N)}_{i_n}. \tag{10}$$

Proof. According to Lemmas 1 - 2, since CP decomposition is a special case of TR decomposition and has the generalized tensor decomposition form, TR decomposition can also be described in the generalized form Eq. (9). □

Based on Theorem 1, we have the following definition.

Definition 3 *(Generalized Tensor-Ring Decomposition).* Given an Nth-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the indexes of all entries in the $\mathcal{X}$ are represented by the set $\mathcal{J}_y$, i.e., $\mathcal{J}_y = \left\{\left(i_1, \cdots, i_n\right) | \forall i_1 \in \left\{1, \cdots, I_1\right\}, \cdots, i_n \in \left\{1, \cdots, I_N\right\}\right\}$. The decomposition model consists of a non-linear mapping function $f : \mathcal{J}_y \to \mathbb{R}$ with the whole set of parameters Θ:

$$x_{i_1 i_2 \cdots i_n} = f\left(i_1, \cdots, i_n; \mathcal{U}^{(1)}, \cdots, \mathcal{U}^{(N)}; \Theta\right), \tag{11}$$

where $\left\{\mathcal{U}^{(k)}\right\}_{k=1}^{N}$ are TR decomposition factors.

4. The proposed model and algorithm

In this section, we first introduce a neural TR decomposition framework, which is capable of learning nonlinear interactions in the data and demonstrate the details of the ConvTR model. Then discuss the process of model optimization and show the pseudo-code of the algorithm. Finally, the complexity of the model is investigated.

4.1. The ConvTR model

The hard-core of ConvTR employs a convolutional neural network to implement the computational process of TR decomposition.

For an Nth-order tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the input to the ConvTR model is a set of vectors consisting of index values $\boldsymbol{x} = index(i_1, i_2, \cdots, i_n)$. Each element of the index vector can be embedded as a corresponding tensor slice in a factor tensor decomposed by the TR method. These factor slices are stitched together into a convolutional neural network for operation, and the output is the tensor entry $y_{i_1 i_2 \cdots i_n}$ corresponding to each index vector. The model improves the nonlinear processing capability by combining multiple convolutional layers. The master framework of the proposed model is shown in Fig. 1, which mainly contains three modules: including a TR decomposition module, an embedding module, and finally a nonlinear mapping module.

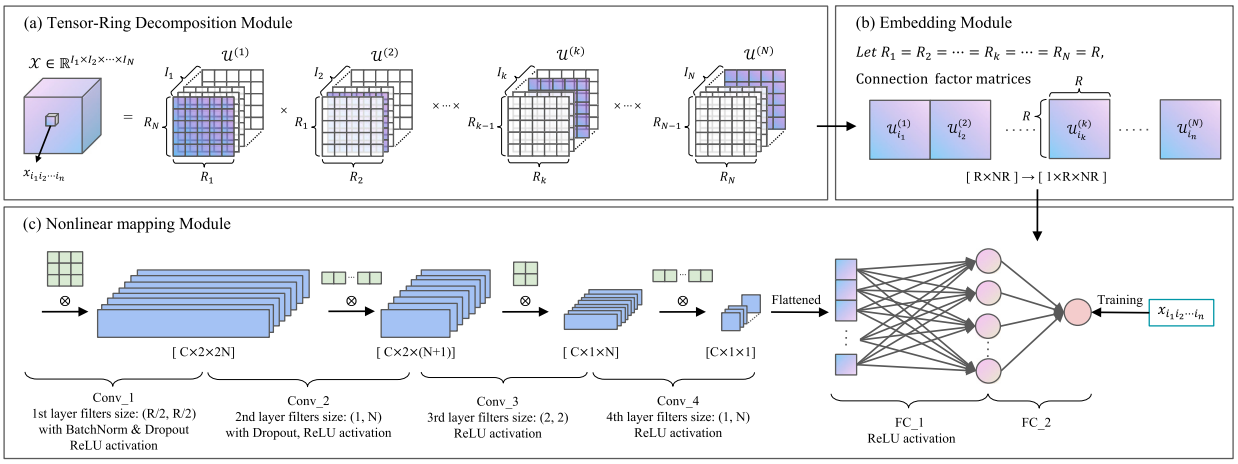

Fig. 1. The main framework of ConvTR model.

4.1.1. Tensor-ring decomposition module

TR decomposition represents an Nth-order tensor $\mathcal{Y}$ by multiple third-order tensors as shown in Eq. (5):

$$y_{i_1 i_2 \cdots i_n} = Trace\left(\prod_{k=1}^{n} \mathcal{U}^{(k)}\left(:, i_k, :\right)\right), \tag{12}$$

where $\mathcal{U}^{(k)}\left(:, i_k, :\right) \in \mathbb{R}^{R_{k-1} \times 1 \times R_k}$ for $k = 1 \cdots N$, and $R = [R_1, \cdots, R_N]$ is called TR rank. Fig. 1(a) illustrates the TR decomposition process. Note that all R_k with $k \in [1, N]$ can be different in the standard TR decomposition. In our work, the TR decomposition has the same rank, i.e., $R_1 = R_2 = \cdots = R_N = R$. Therefore there is $\mathcal{Y} = \mathfrak{R}_{TR}(\mathcal{U}^{(1)}, \cdots, \mathcal{U}^{(N)})$.

4.1.2. Embedding module

The embedding module has N third-order tensors $\mathcal{U}$ obtained by TR decomposition. The dimension of $\mathcal{U}$ is $R \times I_k \times R$, which is since the common rank R is utilized in all factor tensors. Each tensor entry can be represented as an N-way index vector. The embedding module extracts the corresponding embedding matrix $\mathcal{U}_{i_k}^{(k)} \in \mathbb{R}^{R \times R}$ from the factor tensor based on each element in the index vector $\boldsymbol{x}$ as input. Then, these embedding matrices are spliced to get the corresponding feature matrix of the index, where we define $Cat(\cdot)$ as the matrix splicing operation. More specifically, we have

$$\mathcal{H}_{emb} = Cat\left(\mathcal{U}^{(1)}\left(:, i_1, :\right), \cdots, \mathcal{U}^{(N)}\left(:, i_n, :\right)\right) = \left[\mathcal{U}_{i_1}^{(1)}, \mathcal{U}_{i_2}^{(2)}, \cdots, \mathcal{U}_{i_k}^{(k)}, \cdots, \mathcal{U}_{i_n}^{(n)}\right] \in \mathbb{R}^{R \times NR}. \tag{13}$$

Since a convolutional layer is required, we reshape $\mathcal{H}_{emb}$ into a tensor of size $1 \times R \times NR$.

4.1.3. Nonlinear mapping module

The module is a multi-layer representation learning, which mainly includes convolutional and fully-connected layers.

Convolution layer: As CNNs are increasingly used to deal with complex nonlinearities in datasets [41], it has become more intuitive to explore nonlinear interactions in relational data. Arguably, two key contributors to CNNs are their nonlinearity and multilayer stacking. To prevent the CNNs from being affected by the change of internal covariance during training, we apply batch normalization (BN) [42] to each small batch of trained datas of CNNs to avoid performance degradation. Again, we denote by σ the activation function ReLU, $Conv(\cdot)$ represents the 2D convolution operation, and the convolution filter as $\theta_n, n = 1 \cdots L$, stacking multiple 2D convolutional blocks:

$$\begin{aligned} \mathcal{H}_{conv}^{1} &= \sigma\left(Conv\left(\mathcal{H}_{emb}; \theta_1\right)\right), \cdots, \\ \mathcal{H}_{conv}^{L} &= \sigma\left(Conv\left(\mathcal{H}_{conv}^{(L-1)}; \theta_L\right)\right). \end{aligned} \tag{14}$$

Suppose C is the channel number in the convolution layer. The mapping module consists of two filters that model the TR reconstruction process. One is a filter of size $1/k(R, R)$, where k is the coefficient that controls the size of the square filter so that the filter size can be adjusted to better learn the interaction between the data. Note that for the first convolution, we need to set the step size to R/k. Correspondingly, the size of the other filter is set to $(1, N)$. We further utilize Dropout [43] as a regularization to the convolution layer to avoid the overfitting. Alternating the two filters during the convolution process is intended to eventually produce an output of size $C \times 1 \times 1$.

Fully-Connected layer: The resulting $C \times 1 \times 1$ tensor is used as the next input, which is flattened into a vector of length C, denoted as

$$\mathcal{H}_{flat} = Flatten(\mathcal{H}_{conv}^{L}) \in \mathbb{R}^{1 \times C}. \tag{15}$$

It is aggregated by a fully-connected layer and a scalar is obtained as the output. For purpose of learning the nonlinear structure in the tensor more efficiently, two fully connected layers are chosen here, and again, ReLU is used as the activation function. θ_f is denoted as the parameter. Thus, we have

$$\mathcal{H}_{fc} = FC\left(\sigma\left(FC(\mathcal{H}_{flat};\theta_f)\right)\right) \in \mathbb{R}, \tag{16}$$

where $FC(\cdot)$ is fully-connected layer operation. In summary, the output of the nonlinear mapping module can be written as $\hat{y} = \mathcal{H}_{fc}$.

It follows that the ConvTR model is a generalized TR decomposition model with the factor tensor stored in the embedding module and represented by a neural network. Here we should mention that, when $L = 2$, the model can be completely degraded to simulate the TR reconstruction process, denoted as follows

$$\mathcal{H}^1_{conv} = \sigma\left(Conv\left(\mathcal{H}_{emb};\theta_1\right)\right) \in \mathbb{R}^{C\times 1\times N},$$
$$\mathcal{H}^2_{conv} = \sigma\left(Conv\left(\mathcal{H}^1_{conv};\theta_2\right)\right) \in \mathbb{R}^{C\times 1\times 1}.$$

The 2-layer convolution will lead to unsatisfactory results due to the large filter size, so we set $L = 4$ here, and the convolution process is shown in Fig. 1(b). Furthermore, in case the factor tensor is reduced in size to a matrix, i.e., $\mathcal{U}^{(k)} \in \mathbb{R}^{R\times I_k\times R} \to U^{(k)} \in \mathbb{R}^{I_k\times R}$, the model degenerates to the CoSTCo [21].

To simplify the model, the nonlinear module is represented by having a mapping function

$$\hat{y} = f\left(\boldsymbol{x};\mathcal{U}^{(1)},\cdots,\mathcal{U}^{(N)};\Theta\right), \tag{17}$$

here we employ a set $\Theta = \left\{\theta_1,\cdots,\theta_L,\theta_f\right\}$ to represent all the parameters of the model.

4.2. Optimization of ConvTR

Suppose $\mathcal{Y}$ is the tensor of observed missing entries, $\hat{\mathcal{Y}}$ is the tensor approximated by the core tensor, and the number of all observed entries is M. Define the index set of the observed entries as $\mathcal{J}_{\mathcal{Y}} = \left\{\left(i^m_1, i^m_2,\cdots,i^m_n\right) | m = 1,\cdots,M\right\}$. Since $\boldsymbol{x}$ is an entry of the set that represents the index vector, we have $y_m = \mathcal{Y}(\boldsymbol{x})$, $\hat{y}_m = f(\boldsymbol{x};\cdot)$.

The loss function is the Frobenius norm of the difference between the reconstructed tensor $\hat{\mathcal{Y}}$ and the target tensor $\mathcal{Y}$, i.e., $\|\hat{\mathcal{Y}} - \mathcal{Y}\|^2_F$. Under the sparse observation condition, the training set only observes part of the tensor entries, whereupon we have the same loss function as the mean square error, denoted as

$$\mathcal{L} = \|\hat{\mathcal{Y}} - \mathcal{Y}\|^2_F = \sum_{m=1}^{M}\left(\hat{y}_m - y_m\right)^2. \tag{18}$$

We express the reconstruction loss of factor tensor as $\mathcal{R}\left(\mathcal{U}^{(1)},\cdots,\mathcal{U}^{(N)},\Theta\right)$ in regularized form.

$$\mathcal{R}\left(\mathcal{U}^{(1)},\cdots,\mathcal{U}^{(N)},\Theta\right) = \sum_{k=1}^{N}\left\|\mathcal{U}^{(k)}\right\|^2_F + \|\Theta\|^2_2. \tag{19}$$

We introduce the reconstruction loss of the factor tensor as a regularization term $\mathcal{R}$. This regularization helps prevent overfitting arising from the model complexity and thereby improves the model's generalization capability. Thus, the overall objective function of ConvTR is

$$\mathcal{L} = \sum_{m=1}^{M}\left(\hat{y}_m - y_m\right)^2 + \mathcal{R}(\mathcal{U}^{(1)},\cdots,\mathcal{U}^{(N)},\Theta). \tag{20}$$

The ConvTR can be taught by minimizing the loss function between the prediction term obtained from the interaction and the tensor term obtained from the observation.

Thus, ConvTR is trained by measuring all observed variables and the entry value of each completion. Algorithm 1 presents the pseudo-code of the model training process. The index set and observations are used as inputs, and the number of convolutional channels C and the rank R of TR decomposition are the model parameters. The parameters of the TR potential tensor and the convolutional layer are first initialized randomly before training. During the training process, a small batch J_{batch} of size m_k is sampled, each observation is considered for training. Specifically, for each index vector and observation in the J_{batch}, the algorithm obtains the embedding matrix of the index vectors after Eq. (13). Then, the nonlinear operation is used to calculate the predicted values Eq. (17), and further Eq. (20) is applied to calculate the loss. Finally, the model parameters Θ update according to the calculated loss gradient.

4.3. Complexity analysis

The core idea of ConvTR is to construct an efficient nonlinear mapping module to emulate the interaction function between factor tensors, and the computational complexity of the model primarily comes from the TR decomposition module and the nonlinear mapping module. For an N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$, we simply assume that the rank of the tensor decomposition is R. It is straightforward to know the complexity of the TR decomposition as $\mathcal{O}\left(NIR^2\right)$. The nonlinear module mainly consists of

Algorithm 1 Training Algorithm for ConvTR.

Input: training set $\mathcal{J}_y$, original data $\mathcal{Y}$, n_{epoch}, m_k;
Parameters: channels C, TR-ranks R;
Initialization: TR factors tensors $\left\{\mathcal{U}^{(k)}\right\}_{k=1}^{N}$ and model parameters Θ;

```
for t = 1, 2, ⋯, n_epoch do
    Sample a batch-size J_batch ⊆ J_y of size m_k ;
    Loss function L = 0;
    for (i_1, i_2 ⋯, i_n) ∈ J_batch do
        Index vector x = (i_1, i_2, ⋯, i_n);
        Gather embeddings for all dimensions by Eq. (13), U^(k)_{i_k} = U^(k)(:, i_k, :) for k = 1 ⋯ N;
        Calculation of nonlinear layers by f(x; ·) Eq. (17);
        Calculation of the loss function L by Eq. (20);
        L = L + L(f(x; ·), Y(x));
    end for
    Update TR factors tensors and parameters of convolution. w.r.t the gradients using ▽L;
end for
```

convolutional and fully connected layers, where the convolutional layer is composed of convolutional filter of size (R, R) and $(1, N)$, their complexity can be denoted as $1 \times C \times R \times R$, and $C \times C \times 1 \times N$, respectively, and the complexity of the fully-connected layer is $C \times C$. Hence, the overall complexity of the model is $\mathcal{O}\left(NIR^2 + CR^2 + NC^2\right)$.

Compared with ConvTR, the three typical TR completion methods TRWOPT, TRALS and TRLRF require computational complexity of $\mathcal{O}\left(NR^2I^N + NR^4I^{N-1}\right)$, $\mathcal{O}\left(PNR^4I^N + NR^6\right)$ and $\mathcal{O}\left(NR^2I^N + NR^6\right)$, respectively, where P is the sampling rate. Although, the computational complexity of these three algorithms is comparable, the practical application often increases the workload in rank selection since the real rank size usually remains unknown. It can be seen that among the above TR decomposition models, ConvTR has the lowest model complexity. Meanwhile, ConvTR has better rank robustness, which can effectively reduce the workload caused by rank selection, thus reducing the computational cost of the algorithm.

5. Numerical experiments

In this section, we validate the performance of the proposed ConvTR model and compare it with the existing advanced linear/nonlinear tensor decomposition models.

All experiments are performed in pytorch 1.9.0 in Linux, using an Intel(R) Core(TM) i9-10940X CPU at 3.30GHz and 128GB RAM. To speed up the operation, we ran the ConvTR model on the graphics (GPU) GeForce RTX 3090 processing unit.

5.1. Experimental setup

5.1.1. Datasets

We test our ConvTR model with seven popular color image datasets (e.g., Lena, Peppers, Starfish, etc.) and six video datasets (e.g., Akiyo, Container, News, etc.).[1] Meanwhile, four publicly available datasets with different specifications collected from real traffic systems are selected to evaluate our proposed model to show the generality of the ConvTR model. Two small-scale data are Guangzhou city traffic speed data[2] and Hangzhou subway passenger flow data.[3] To demonstrate the advantages of ConvTR in handling large-scale high-order traffic data, we specifically select another two publicly available datasets collected from the California Transportation System (i.e., PEMS)[4] as our benchmark datasets. All the above datasets are normalized to $[0, 1]$.

5.1.2. Metrics

For model evaluation, we mask the tensor entries according to the proportion of missing rates, and then estimate the completion of these missing entries. We calculate the metrics root mean square error (RMSE) and mean absolute percentage error (MAPE) [44] using the true values of these entries, i.e.,

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)}, \quad MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100.$$

Lower values of these two quality metrics indicate better reconstruction performance. For both image and video data, we weigh the performance of each method using the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) [45]. Conversely, higher values of these two quality measures reflect better completion scores.

[1] https://media.xiph.org/video/derf/.
[2] https://doi.org/10.5281/zenodo.1205229.
[3] https://tianchi.aliyun.com/competition/entrance/231708/information.
[4] https://github.com/wanhuaiyu/ASTGCN/tree/master/data.

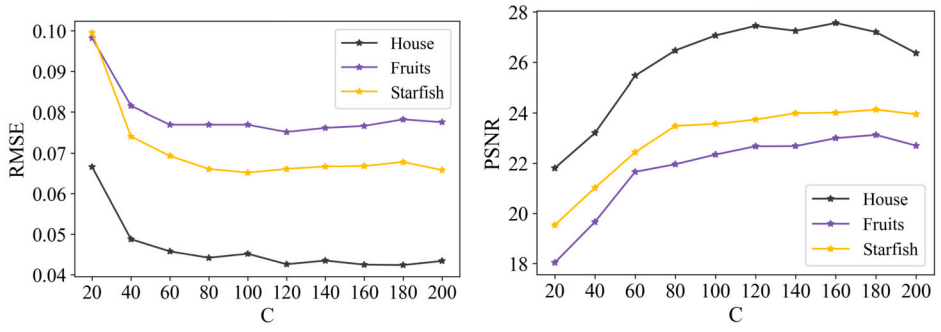

Fig. 2. The RMSE and PSNR values of recovered color images by ConvTR with different C values and the missing rate 85%.

5.1.3. Baselines

Several state-of-the-art tensor decomposition methods are compared under different experimental settings, which contain both linear and nonlinear models.

TRWOPT [28]: The gradient descent algorithm is used to find out the potential factor of the incomplete tensor so as to complete the missing entries of the tensor.

TRALS [17]: It is mainly through the alternating optimization of each core in turn until the optimization process reaches a certain convergence condition.

TRLRF [18]: The nuclear norm regularization is applied to the TR decomposition factor, each step is optimized by SVD decomposition, and the whole model is solved by ADMM algorithm.

GETD [8]: A generalized model unites Tucker decomposition and Tensor Ring decomposition. After Tucker nonlinear decomposition, the core tensor is decomposed by TR, to alleviate the dimensional disaster caused by the Tucker decomposition of the higher-order tensor.

CoSTCo [21]: This model designs a shallow neural network structure based on CP decomposition and uses CNN expression to simulate the CP decomposition process. Such an approach can guarantee the low rank of the model while handling high-order sparse data.

5.1.4. Implementation

In the model ConvTR training, the detailed settings of the network hyperparameters are determined as follows. The batch size of all data training samples is fixed at 256. In addition, batch normalization and dropout are used to control overfitting. In addition to the TR decomposition embedding size, the initial learning rate during network training is varied from 0.001, with learning rate decay in the range $\{0.90, 0.995, 1\}$ and dropout in the range $[0.0, 0.5]$. The entire number of epochs is set to 200 for network training and the Adam [46] technique is used as the gradient descent algorithm for backpropagation. We use the Python language and the Pytorch platform to train and test our network in a Windows 10 environment and GPU acceleration mode.

We assume all values of TR rank are identical, i.e., $R_1 = \cdots = R_N = R$, where the TR factors are randomized from the standard Gaussian distribution $\mathcal{N}(0, 1)$. All the methods are stopped when the maximum number of iterations or convergence criterion is reached, and the parameters are set according to the corresponding papers to attain the best results.

5.2. Parameters analysis

As embedding size is an important factor in connecting predictive models with expressivity [47], and the number of convolutional channels C and the rank R of the TR decomposition are the only hyperparameters of ConvTR that determine the complexity and performance of the model, we now study the impact of these parameters.

5.2.1. Effect of C

The number of channels inside the convolutional layer represents the depth of the layer, which illustrates how many counts are used to represent each pixel point. In considering the sensitivity of the channel C of the convolution layer, experiments are conducted on color images with different C values. We select three color images with a random missing rate (MR) of 85%, and the results are shown in Fig. 2. In the beginning, the RMSE/PSNE values decrease/increase with increasing C. This is attributed to the increase of parameters in the convolution process, which makes the captured information increase accordingly, indicating that the amount of C affects the model. However, as C continues to increase, the RMSE/PSNR values tend to stabilize. In the later experiments of this paper, the number of convolution layer channels C can be adjusted according to this result.

5.2.2. Effect of R

Optimal rank selection is a momentous problem in tensor decomposition, particularly for TR decomposition. Usually, tensor decomposition models are extremely sensitive to rank and the choice of rank often has a great influence on the effectiveness of the algorithm. Whereas, the true rank of the real datasets is unknown. It usually requires a large number of experiments to determine

Fig. 3. Recovered results of the TRWOPT, TRALS, TRLRF, GETD, CoSTCo, and ConvTR (proposed) on image "Lena" with different TR-ranks for the missing rate are 90%. From the first row to the last row, the selected TR-ranks are 3, 5, 10, 20, and 40, respectively.

the appropriate rank. Therefore, rank robustness is a crucial feature for tensor decomposition. We will demonstrate that ConvTR has excellent robustness in the next experiments.

We evaluate the rank robustness of the model ConvTR on an image named "Lena". Fig. 3 visualizes the completion results of each model on example "Lena" when the chosen rank varies from 3 to 40. Clearly, the proposed method remains stable and gives the best visualization of the recovered image when the selected rank increases.

Obviously, the traditional TR-based completion methods (i.e., TRALS and TRWOPT) cannot maintain sound stability performance with increasing the selected rank since the algorithm does not have any regularization term, resulting in a sharp drop in recovery performance. Such a result is due to the overfitting of the model when the selected rank is larger than the true one. In contrast, the proposed method invariably provides the optimal recovery results while the model can ignore the effects of inappropriate choice of rank due to the excellent tuning capability of the convolutional layer of the model ConvTR.

Fig. 4 further shows the recovery performance of various algorithms at high missing rates, with different ranks. It is known that the higher the deletion rate is, the easier it is for the model to overfitting, so we set the image missing rate to 0.9 and 0.95 respectively. As you can see in Fig. 4(a), the RMSE value of the TRWOPT and TRALS algorithms is the lowest when $rank = 3$, which is the expected value of the algorithm. As the TR-rank increases, the recovery ability of TRWOPT and TRALS decreases due to redundant model. As a network, GETD and CoSTCo learn more features with the increase in rank, and the performance improves gradually until it is stable. In Fig. 4(b), we find that TRWOPT and TRALS algorithms have lost their effectiveness under high sparsity, and GETD needs more features in the completion process. No matter what the TR rank is, TRLRF, CoSTCo, and ConvTR are robust, whereas ConvTR is optimal in most cases. Therefore, ConvTR has stronger robustness to rank selection, which can extremely reduce computational costs.

In addition, we run ConvTR on the color image "Lena", using RTX 3090 GPU. With the MR of 90%, regardless of the value of R and C, each epoch training takes approximately 0.48 s, the total training takes approximately 2 min, and the inference takes only 2 s. The outcomes show that the ConvTR method has high prediction efficiency and robustness in color images.

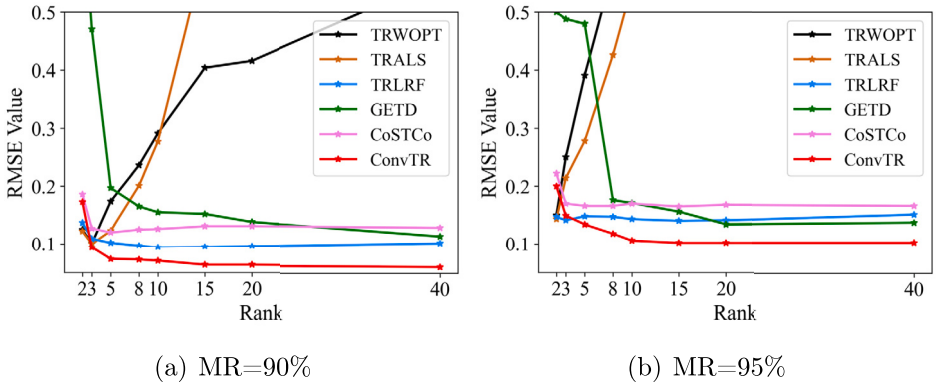

Fig. 4. The RMSE values of the recovered color image "Lena" by all algorithms with different ranks when the missing rate is 90% and 95%.

Table 2
Evaluation results of recovered color images by different methods, **bold** and underline indicate optimal and sub-optimal results, respectively.

Image	MR	PSNR						SSIM					
		TRWOPT	TRALS	TRLRF	GETD	CoSTCo	ConvTR	TRWOPT	TRALS	TRLRF	GETD	CoSTCo	ConvTR
Lena $256\times256\times3$	85%	21.064	21.723	23.018	19.222	19.280	**26.224**	0.8608	0.8813	0.9019	0.8101	0.8436	**0.9537**
	90%	19.971	20.034	20.651	16.243	18.154	**24.223**	0.8398	0.8412	0.8486	0.6510	0.7790	**0.9303**
	95%	11.950	16.860	17.177	15.042	15.349	**20.512**	0.5769	0.7457	0.7173	0.5952	0.7210	**0.8545**
Airplane $256\times256\times3$	85%	19.751	19.870	20.872	17.427	20.738	**23.119**	0.5384	0.5426	0.5510	0.2826	0.6181	**0.8231**
	90%	17.077	17.131	18.952	16.407	17.877	**21.096**	0.3902	0.4439	0.4845	0.2033	0.4599	**0.7341**
	95%	14.918	14.750	17.358	15.023	15.323	**18.983**	0.2251	0.2186	0.3527	0.1603	0.2529	**0.5882**
Peppers $256\times256\times3$	85%	19.895	20.142	20.989	18.185	18.478	**25.899**	0.8518	0.8558	0.8719	0.7928	0.8141	**0.9587**
	90%	18.051	17.762	17.896	16.934	15.774	**23.045**	0.8023	0.7886	0.7731	0.7185	0.7245	**0.9248**
	95%	14.328	14.413	14.915	13.576	13.199	**17.956**	0.6445	0.6677	0.6555	0.5539	0.5513	**0.8049**
Starfish $256\times256\times3$	85%	19.684	19.872	20.628	18.132	17.307	**24.211**	0.7968	0.8068	0.8142	0.7203	0.7077	**0.9311**
	90%	18.002	18.094	17.898	16.300	16.324	**22.334**	0.7315	0.7366	0.7236	0.6362	0.6940	**0.9047**
	95%	11.225	13.990	15.617	14.324	14.319	**17.793**	0.4072	0.5418	0.5967	0.5440	0.5174	**0.7833**
House $256\times256\times3$	85%	22.224	21.104	25.180	19.696	21.472	**27.642**	0.7985	0.7901	0.8678	0.6978	0.8001	**0.9345**
	90%	19.881	19.914	20.621	16.083	19.810	**26.031**	0.7518	0.7531	0.7708	0.5548	0.7220	**0.9005**
	95%	18.160	18.268	17.819	15.375	17.813	**21.194**	0.6244	0.6273	0.5253	0.3702	0.6059	**0.8082**
Baboon $256\times256\times3$	85%	19.751	19.998	19.871	17.378	18.737	**21.987**	0.6433	0.6448	0.6320	0.4843	0.5396	**0.7474**
	90%	18.792	18.884	18.410	16.215	17.827	**20.994**	0.5985	0.5995	0.5983	0.4290	0.5072	**0.6846**
	95%	17.277	16.674	15.538	14.977	16.792	**18.705**	0.4918	0.4655	0.3151	0.3242	0.4217	**0.5657**
Fruits $256\times256\times3$	85%	18.842	18.572	19.838	17.651	18.064	**23.171**	0.7290	0.7093	0.7403	0.6157	0.7157	**0.9015**
	90%	17.331	17.094	17.605	15.988	17.658	**21.396**	0.5763	0.5858	0.5306	0.5196	0.6643	**0.8484**
	95%	14.586	14.494	15.063	14.788	15.615	**17.919**	0.5342	0.5522	0.5422	0.4867	0.5853	**0.7076**

5.3. Color image completion

In this subsection, we test our ConvTR against the state-of-the-art algorithms on seven benchmark images. The size of each Color image is $256\times256\times3$ which can be considered as a third-order tensor.

Before starting the experiment, we perform missing processing on the original image. Specifically, all elements in the color image will be erased randomly according to a certain ratio to obtain the observed data. For traditional TR-based mathematical methods, TR-rank can be selected from $[2, 20]$ by a cross-validation method, and the parameters are tuned according to the corresponding literature (The following video experiments are identical to the Spatio-temporal traffic experiments).

We test these algorithms on all the seven benchmark images with different missing rates: 85%, 90%, and 95%. We report the PSNR and SSIM values of the recovery results of different algorithms for 7 color images in Table 2. ConvTR performs the best and stable, followed by TRLRF, which establishes the relationship between the multilinear tensor rank and TR factor rank, allowing the low-rank constraint to be implicitly carried out on the TR latent space, thus ensuring that the model has better recovery results in the case of robustness. Considering the PSNR, the ConvTR outperforms the TRLRF by about 3 dB. It is observed that the model degrades substantially when the missing rate changes from 90% to 95%, compared to increasing the missing rate from 85% to 90%. When the missing rate reaches 95%, the baseline models appear to be overfitted and the completion performance drops significantly due to too few observations. However, our method still provides the best accuracy in entries of PSNR and SSIM at high sparsity.

Fig. 5 shows the results of the recovery of color images for all compared models with an 85% missing rate. Obviously, the images reconstructed by our method are the clearest with the best visual effect. GETD uses Tucker to decompose the target tensor, and

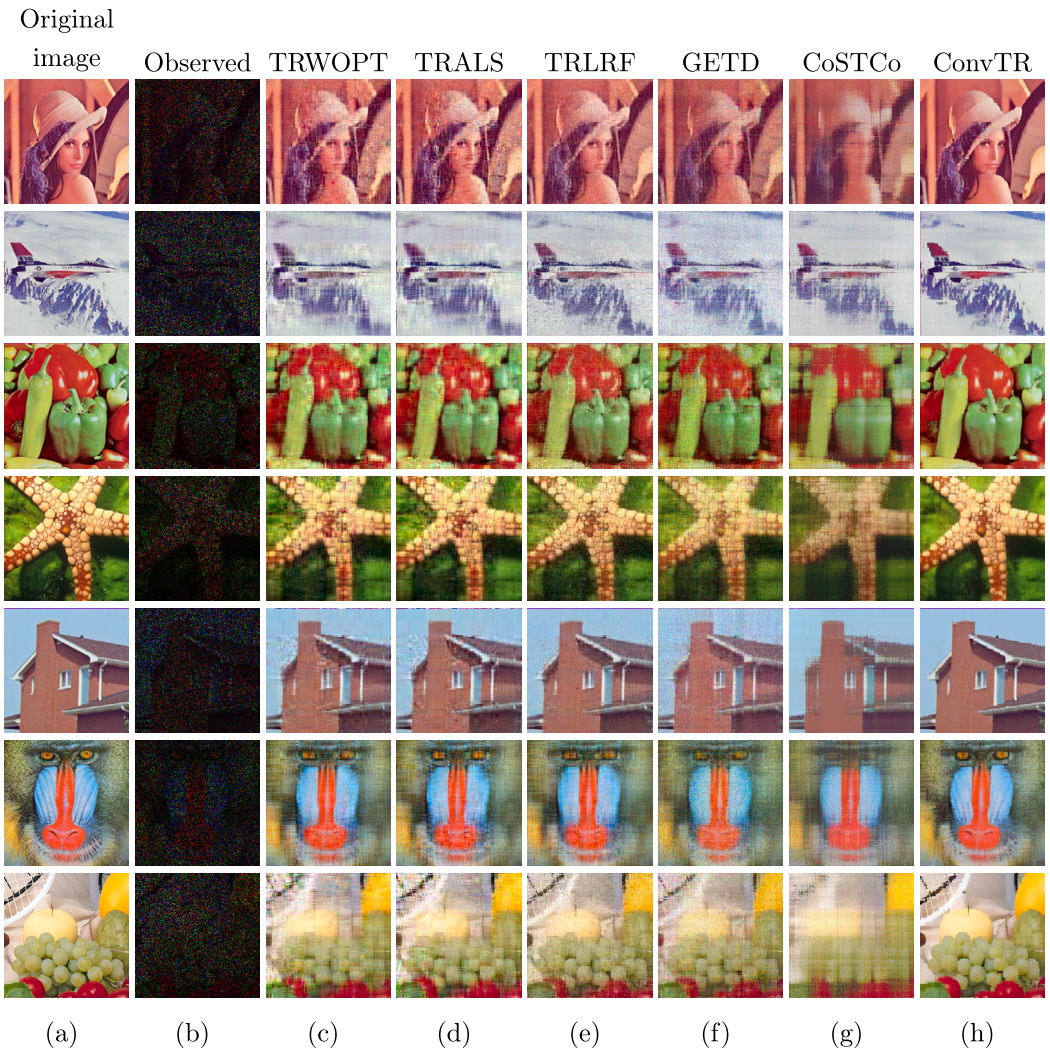

Fig. 5. The recovered color images by different methods with the missing rate 85%. From (a) to (h): original image, observed image, recovered images by TRWOPT, TRALS, TRLRF, GETD, CoSTCo, and ConvTR, respectively.

then the resulting core tensor is decomposed by TR. Although this algorithm reduces the complexity of Tucker decomposition, the lack of a convolutional layer makes the learned features worse than ConvTR. CoSTCo is a convolutional network method that uses CP decomposition and the image looks darker because the reconstruction produces more negative entries. Although both are in the form of convolutional networks, the best recovery performances of CoSTCo are significantly lower than that of ConvTR based on TR decomposition, which can indicate that the expressiveness of CP decomposition is inferior to that of TR decomposition. Among the traditional TR-based completion methods (i.e., TRWOPT, TRALS, and TRLRF), TRLRF complements most effectively but does not mitigate the blurring due to the fact that using only the global low-rank prior is not sufficient to recover the underlying image. The gradient descent-based TRWOPT algorithm is unstable during optimization and the reconstruction performance is inferior to that of TRALS, but they all outperform the network-structured GETD and CoSTCo models. In contrast, the ConvTR proposed in this paper can capture the intrinsic structure of multidimensional visual data, and thus has clean and sharp spatial details with the best visual results.

5.4. Video completion

The video completion task is to reconstruct the frames of a video based on a given partial observation. In the experiments, we test six grayscale videos, each named “Akiyo”, “Container”, “News”, “Suzie”, and “Seafish”, with size denoted as ($height \times width \times frame$). Unlike color images with only three channels, gray video data usually consists of a dozen or hundreds of frames. As a result, the relationship between different frames is much tighter and the adjacent frames of the video sequence are highly similar, so there is also more redundant information.

Table 3
Evaluation results of recovered videos by different methods, **bold** and underline indicate optimal and sub-optimal results, respectively.

Video	MR	PSNR						SSIM					
		TRWOPT	TRALS	TRLRF	GETD	CoSTCo	ConvTR	TRWOPT	TRALS	TRLRF	GETD	CoSTCo	ConvTR
Akiyo $144\times176\times30$	90%	30.811	30.420	30.189	26.298	27.736	**33.022**	0.8881	0.9137	0.8921	0.7801	0.9062	**0.9616**
	95%	26.728	26.861	28.572	24.675	27.260	**31.486**	0.7639	0.7867	0.8485	0.7643	0.8336	**0.9366**
	99%	20.448	11.033	19.134	22.391	21.604	**24.725**	0.4511	0.1201	0.5896	0.5994	0.5532	**0.7792**
Container $144\times176\times30$	90%	28.485	27.582	29.018	22.624	26.466	**31.663**	0.8671	0.8538	0.9144	0.7269	0.8745	**0.9386**
	95%	23.572	23.383	25.864	21.841	24.021	**27.508**	0.7321	0.7282	0.8234	0.6463	0.7985	**0.8985**
	99%	17.192	12.212	17.621	20.394	20.017	**21.978**	0.4251	0.2061	0.3541	0.5665	0.5845	**0.7218**
News $144\times176\times30$	90%	29.286	29.287	29.813	24.388	25.143	**32.893**	0.8628	0.8729	0.9038	0.7418	0.8313	**0.9642**
	95%	24.965	25.724	26.089	22.522	23.566	**28.306**	0.7183	0.7551	0.7903	0.6818	0.7528	**0.9229**
	99%	11.727	11.567	19.392	18.085	18.721	**21.747**	0.2263	0.1218	0.4634	0.4442	0.4353	**0.7435**
Suzie $144\times176\times30$	90%	30.146	30.169	30.954	27.988	28.287	**32.808**	0.8127	0.8224	0.8312	0.7723	0.8125	**0.9013**
	95%	27.435	27.896	26.105	26.468	25.919	**29.901**	0.7299	0.7532	0.7237	0.7073	0.7209	**0.8469**
	99%	19.735	16.269	20.095	23.548	21.799	**24.569**	0.4421	0.1364	0.4501	0.6035	0.5071	**0.6519**
Seafish $185\times290\times30$	90%	24.285	24.661	25.718	18.018	18.497	**26.813**	0.8207	0.8323	0.8708	0.5378	0.5646	**0.8874**
	95%	20.177	20.483	20.697	17.067	18.158	**23.521**	0.6388	0.6571	0.6633	0.4143	0.4717	**0.8201**
	99%	13.855	15.231	14.836	14.602	14.352	**16.968**	0.1182	0.1835	0.1883	0.1675	0.1938	**0.4087**
Bus $256\times256\times30$	90%	19.444	19.705	19.108	18.618	18.740	**21.135**	0.4588	0.4817	0.4238	0.3896	0.3835	**0.6186**
	95%	18.357	18.446	18.183	17.348	17.734	**19.223**	0.3534	0.3531	0.3348	0.3281	0.3315	**0.4838**
	99%	15.626	16.217	15.669	15.733	16.873	**17.239**	0.1646	0.1948	0.1859	0.1760	0.2120	**0.3017**

We randomly remove 90%, 95%, and 99% data from 6 videos. In the experiment, the average PSNR and SSIM of 30 bands are used to evaluate all methods. Table 3 shows the PSNR and SSIM values of the completed videos with different missing rates. Despite the increased dimensionality of the tensor data compared to the images, the same range of rank selection yields excellent PSNR values. Unlike the baseline, for different deletion rates, our method achieves optimal results in all cases. Grayscale videos have high linear correlation, so GETD and CoSTCo are less effective than traditional mathematical methods in video completion. In addition, the TRLRF method is almost similar to TRWOPT and TRALS in entries of PSNR and does not show outstanding recovery as color images.

When MR = 90%, the first frame visualization results are displayed in Fig. 6. The proposed method recovers the best results under all videos, which is consistent with the results shown in Table 3. Our ConvTR recovers the edges well in successive videos, with the most pronounced effect on the "Bus". It can be clearly seen that ConvTR recovers coral bush details on "Seafish", boat outlines on "Container", and facial expressions on "Akiyo" and "Suzie".

Fig. 7 plots the numerical PSNR curves for each frame of the six video datasets, respectively. The complete method based on TR decomposition works better than the other decomposition methods, indicating that the TR structure has advantages in representing grayscale video data. Numerically, our method obtains the highest value in entries of PSNR on each frame.

Choose the "News" dataset as an example. The results are shown in Fig. 8, where the random missing rate is 90%, 95%, and 99% respectively. Intuitively, ConvTR performs the best among these models, while GETD performs the worst. In particular, when the missing rate reaches 99%, the video images recovered by the baseline methods are all obscured, while ConvTR is still able to delineate the boundaries of the edges of the people in the video frame and has higher quality results. All these results show that ConvTR can capture more information from incomplete videos than its comparison methods regardless of data sparsity, reflecting the superiority of ConvTR in video data completion.

5.5. Spatio-temporal traffic data completion

Spatio-temporal traffic prediction is an essential task with wide application prospects [44]. In this subsection, we test all methods on the following four traffic data. Guangzhou city traffic speed data (GZ), which collects the average traffic speed of 214 road sections in Guangzhou, China for two months (from August 1 to September 30, 2016) with a interval of 10 minutes i.e., 144 data per day. Hence, we constitute it as a third-order tensor of size $214\times61\times144$. Hangzhou metro passenger flow data (HZ) provides the inbound passenger flow for 80 metro stations for 25 days (from January 1, 2019, to January 25, 2019), which is collected every 10 minutes. Unlike the GZ dataset, metro passenger flow data is meaningless from 0:00 a.m. to 6:00 a.m., i.e., only 108 intervals are considered in a day, so group it into a third-order tensor of $80\times25\times108$. Essentially, both datasets follow the same tensor structure $sensor\times day\times minute$.

For large-scale data, we adopt the reference [48] provided flow datasets PEMS, including PEMS04 and PEMS08. PEMS04 is the flow data collected by 307 detectors for 59 consecutive days since January 1, 2018, and it is collected every 5 minutes. The shape after reading all the original traffic data is $16992\times307\times3$. Each tensor index is a tuple of (sequence-length, sensor, features), of which the three-dimensional features are flow, occupy, and speed. Similarly, PEMS08 is the traffic data of 170 nodes for 62 consecutive days in San Bernardino since July 1, 2016, and its data shape is $17856\times170\times3$. To simulate the periodicity of traffic data, the original 3rd-order ordered tensor is reconstructed into a 5th-order tensor by decomposing the time index into the daily index, hour index, and minute index. The advantage of this division is that the information under each dimension is easier to distinguish, so the

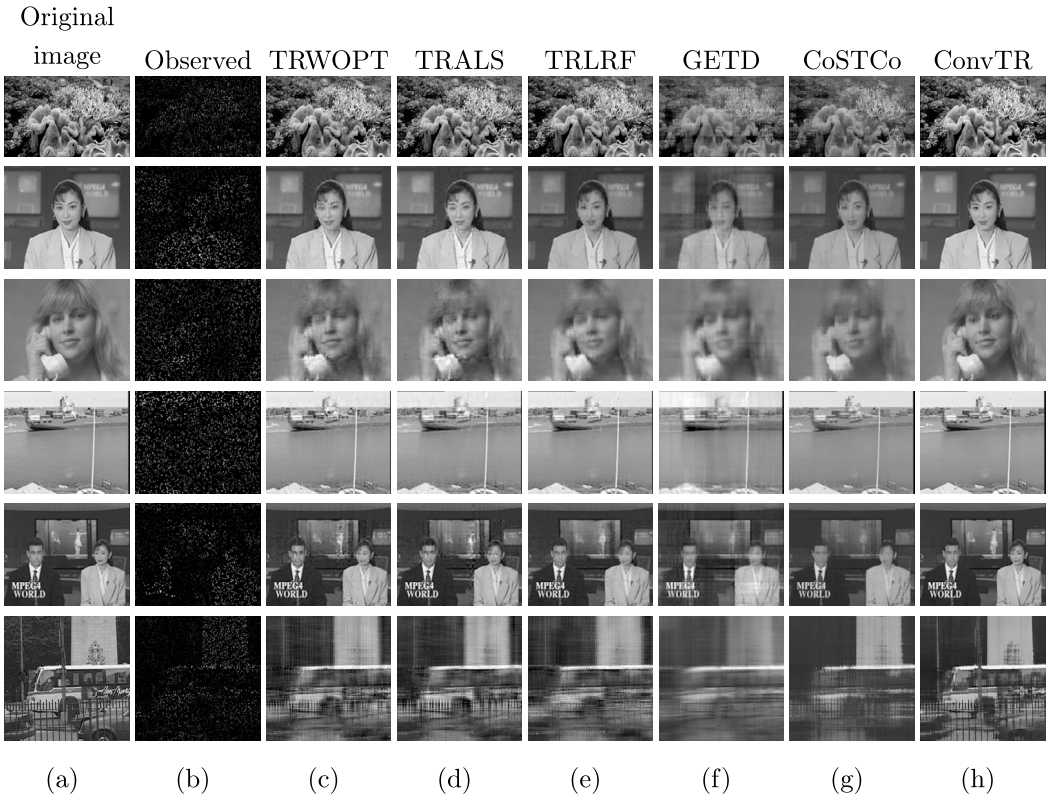

Fig. 6. The 1-th frame of recovered videos by different methods with the missing rate 90%. From (a) to (h): original video, observed video, recovered videos by TRWOPT, TRALS, TRLRF, GETD, CoSTCo and ConvTR, respectively.

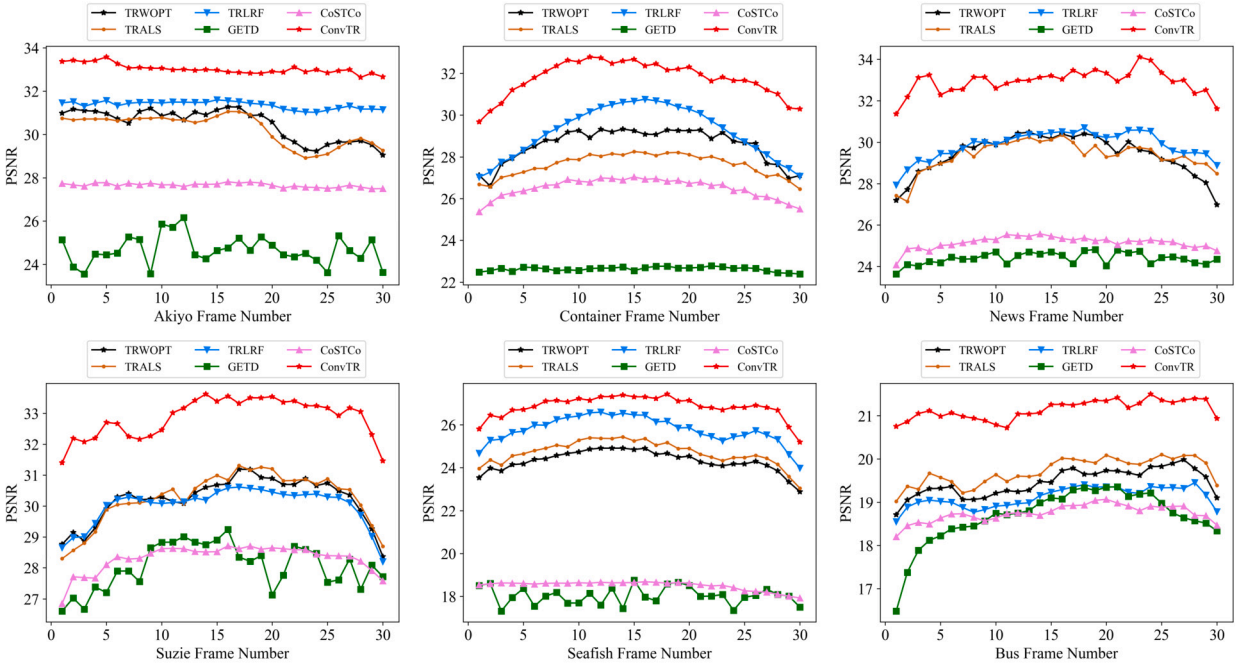

Fig. 7. The PSNR values of recovered frames form six videos by different methods with the missing rate 90%.

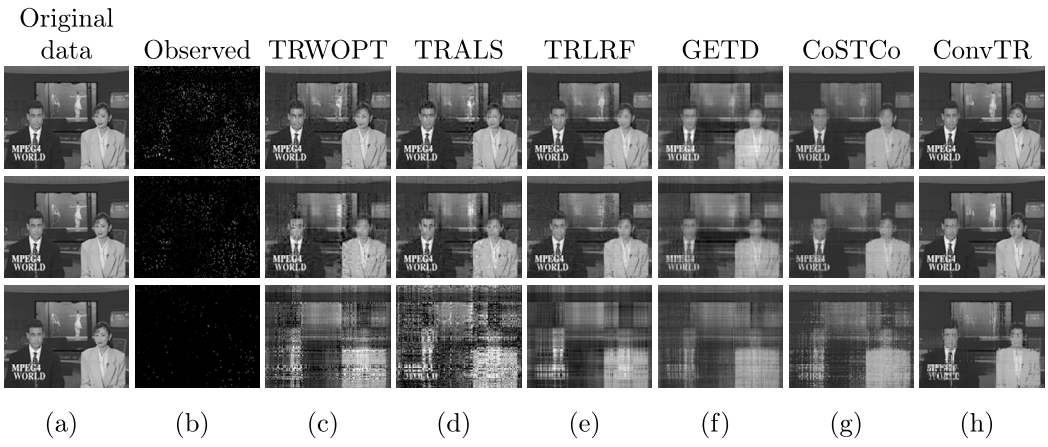

Fig. 8. The 1-th frame of recovered "News" by all methods for different missing rates. From (a) to (h): original image, observed image, recovered images by TRWOPT, TRALS, TRLRF, GETD, CoSTCo and ConvTR, respectively. From the top down: missing rate 90%, 95% and 99%, respectively.

Table 4
Evaluation results of recovered spatiotemporal traffic by different methods, **bold** and underline indicate optimal and sub-optimal results, respectively.

Traffic	MR	RMSE						MAPE(%)					
		TRWOPT	TRALS	TRLRF	GETD	CoSTCo	ConvTR	TRWOPT	TRALS	TRLRF	GETD	CoSTCo	ConvTR
HZ $80\times25\times108$	90%	0.0138	0.0132	0.0168	0.0211	0.0152	**0.0096**	25.34	25.61	57.42	45.55	55.67	**22.86**
	95%	0.0383	0.0236	0.0304	0.0198	0.0184	**0.0136**	35.33	30.33	68.82	42.79	46.11	**23.27**
	99%	0.0712	0.0565	0.0436	0.0562	0.0310	**0.0263**	190.71	108.69	97.99	102.59	107.91	**52.01**
GZ $214\times61\times144$	90%	0.0338	0.0341	0.0336	0.0397	0.0416	**0.0315**	10.11	10.01	10.23	11.97	11.75	**8.67**
	95%	0.0374	0.0371	0.0367	0.0398	0.0431	**0.0342**	10.98	10.85	10.96	11.87	12.59	**9.91**
	99%	0.0485	0.0750	0.0479	0.0447	0.0489	**0.0437**	14.47	16.67	14.61	13.14	14.41	**12.27**
PEMS04 $59\times24\times12\times307\times3$	99.9%	0.7985	0.1311	0.3116	0.0986	0.0668	**0.0595**	112.78	44.56	86.18	51.08	46.89	**44.11**
	99.99%	0.5411	0.4288	0.4685	0.2928	0.1040	**0.0885**	168.98	153.13	100.01	83.49	77.53	**58.48**
PEMS08 $62\times24\times12\times170\times3$	99.9%	1.0454	0.1721	0.2403	0.3187	0.0585	**0.0492**	118.62	52.03	393.75	99.18	31.48	**23.29**
	99.99%	0.5256	0.7714	0.3425	0.4214	0.0878	**0.0672**	142.73	186.12	647.95	149.56	65.90	**50.20**

interaction between factors is easier to capture. Each tensor is indexed as a tuple(day, hour, minute, sensor, variable), and it is not difficult to see that both datasets are massively high-dimensional.

We choose RMSE and MAPE to quantitatively measure the quality of the results, and lower RMSE and MAPE values indicate better reconstruction. A random sampling of data with different specifications is performed, and Table 4 gives the results of different methods at different sampling rates. We note that the ability of the models TRWOPT, TRALS, and TRLRF of traditional mathematics is limited, especially on large-scale data. When the missing rate increases and the sampling rate is too weak, the RMSE/MAPE values of the traditional methods increase sharply. The network structure has better adaptability to different data sizes, which shows that the network structure has better model extensibility. GETD is less effective than CoSTCo in the process of feature learning due to the lack of a corresponding convolutional structure. Although CoSTCo and ConvTR have similar network structures, the TR decomposition we use in the embedding layer has better expressibility than the CP decomposition and is stronger than the CP decomposition in this periodic sequence.

With a missing rate of 90%, we visualize the GZ traffic data recovery using a heat map, showing the observed 1st side slice in Fig. 9. As can be seen, the effect we recovered is the closest to the original data. The three traditional TR mathematical models are more effective in small data recovery. Since GETD and CoSTCo mainly use low-rank constraints, the low-rankness of the complete results is too abrupt. Overall, the effect of our method on small-scale traffic data is excellent in entries of completion, a phenomenon that is consistent with the demonstrated results shown in Table 4.

We mask a certain amount of observations as missing values in the PEMS dataset (i.e., 99.9%, 99.99%), and the remainder of the observations are used as input data to estimate these masked entries. In general, the completion issue becomes increasingly challenging as the missing rate increases. Next, we examine the capabilities of various models under severe and extreme missing scenarios. When the missing rate increases, the traditional mathematical methods show severe overfitting, and the network structure approach can still guarantee a better complementary effect under parameter control. We visualize the time series with an extreme missing rate of 99.9% by picking two weeks of data from PEMS08. Fig. 10 shows the corresponding time series recovered by ConvTR, and we demonstrate the recovery structure of the 3 features flow, occupy, and speed in the 5th-order tensor. It can be seen that ConvTR can achieve very high accuracy for PEMS08 with only 0.1% input. This can be attributed to our ingeniously designed network architecture. Our model can not only efficiently extract nonlinear interactions within the data, but also better characterize the similarities between neighboring data points by exploiting low-rank information. Take PEMS08 as an example. Despite the

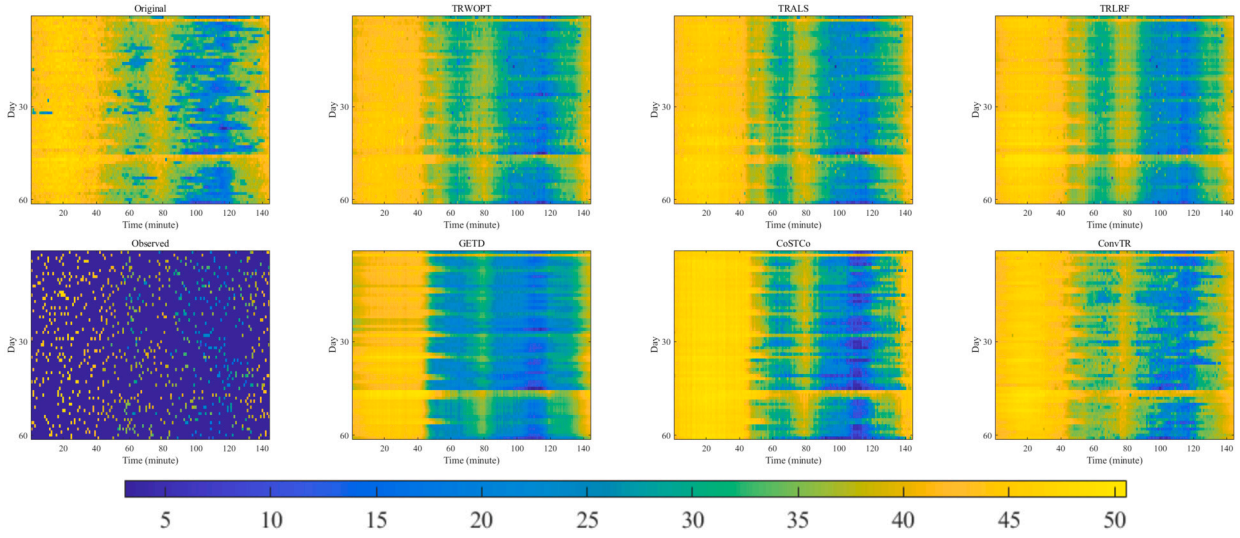

Fig. 9. The reconstructed 1-th road by different methods on the GZ traffic data (MR = 90%). The picture contains 61 days with 144 intervals per day.

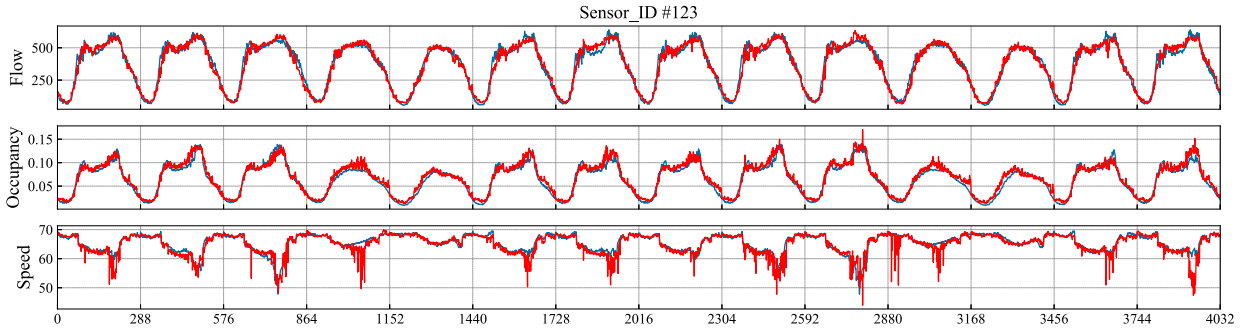

Fig. 10. Example of the recovered PEMS08 traffic dataset (MR = 99.9%). Selecting the sensor data visualization with number 123, the red curve is the predicted value, while the blue curve is the ground fact.

massive dataset size with nearly 10 million data points, ConvTR's low-rank representation can well capture the periodicity in the data. With only 0.1% observed values, there are still nearly 10,000 data points remaining, sufficient for convolutional neural networks to fully extract distinctive features from the data and complete missing values.

In summary, the above experimental results show that the ConvTR proposed in this paper is more capable of capturing higher-order tensor intrinsic information compared with other tensor decompositions.

5.6. Convergence analysis

In Fig. 11, we show the test error of our algorithm in each iteration on three different types of datasets (i.e., image, video, and spatio-temporal traffic data). Here, the total error is defined as the RMSE value of the convergence condition in the algorithm. According to Fig. 11, we can see that the error decreases rapidly as the number of iterations increases. On both image and video datasets, our algorithm can converge within 50 iterations. Even on two larger-scale traffic datasets, it similarly converges within 100 iterations. At the same time, the error after the algorithm converges does not increase with the number of iterations, indicating that the model does not exhibit overfitting and confirms very excellent robustness. The experiments also verify that our algorithm ConvTR not only has a fast convergence rate but also has a good complementary effect.

6. Conclusions

We propose a convolutional expressive TR decomposition model for tensor completion, ConvTR. Based on the expressiveness and flexibility of TR decomposition, we define a new convolutional neural network model to fit large-scale sparse data with fast convergence guarantee. ConvTR can effectively capture nonlinear interactions, thus expressing the underlying information between data even when there are few observations. Extensive experiments on different datasets show that ConvTR outperforms current state-of-the-art linear/nonlinear methods in high-order sparse tensor completion, and achieves the satisfied performance on standard

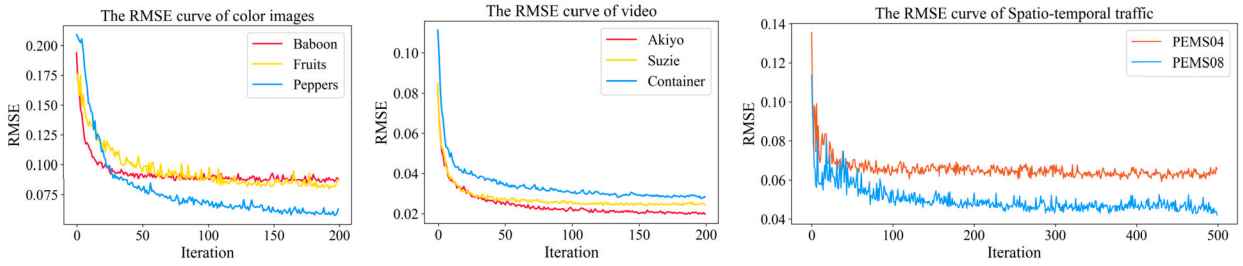

Fig. 11. The convergence behavior regarding the number of iterations for color images, grayscale videos, and spatio-temporal traffic data, respectively.

color image datasets and grayscale video datasets. Also, the model has higher effectiveness and speed in large-scale spatio-temporal traffic data completion than the compared methods.

For future work, we will study the theoretical properties of the ConvTR model and explore its relationship with deep-seated neural networks. At the same time, we will extend the model to more applications, such as recommendation system, knowledge graph and so on.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

The research is supported by the National Key Research and Development Program of China (2023YFB2703700), the National Natural Science Foundation of China (62176269), the Guangzhou Science and Technology Program (2023A04J0314).

References

[1] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.
[2] J.M. Landsberg, Tensors: geometry and applications, Represent. Theory 381 (402) (2012) 3.
[3] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2012) 208–220.
[4] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E.E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Trans. Signal Process. 65 (13) (2017) 3551–3582.
[5] T.G. Kolda, J. Sun, Scalable tensor decompositions for multi-aspect data mining, in: 2008 Eighth IEEE International Conference on Data Mining, IEEE, 2008, pp. 363–372.
[6] C. Lu, X. Peng, Y. Wei, Low-rank tensor completion with a new tensor nuclear norm induced by invertible linear transforms, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 5996–6004.
[7] V.N. Ioannidis, A.S. Zamzam, G.B. Giannakis, N.D. Sidiropoulos, Coupled graphs and tensor factorization for recommender systems and community detection, IEEE Trans. Knowl. Data Eng. 33 (3) (2019) 909–920.
[8] Y. Liu, Q. Yao, Y. Li, Generalizing tensor decomposition for n-ary relational knowledge bases, in: Proceedings of the Web Conference 2020, 2020, pp. 1104–1114.
[9] J.D. Carroll, J.-J. Chang, Analysis of individual differences in multidimensional scaling via an n-way generalization of “Eckart-Young” decomposition, Psychometrika 35 (3) (1970) 283–319.
[10] L.R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.
[11] V. De Silva, L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 1084–1127.
[12] Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Q. Zhao, T.-X. Jiang, Fully-connected tensor network decomposition and its application to higher-order tensor completion, in: Proc. AAAI, vol. 35, 2021, pp. 11071–11078.
[13] R. Orús, A practical introduction to tensor networks: matrix product states and projected entangled pair states, Ann. Phys. 349 (2014) 117–158.
[14] T. Huckle, K. Waldherr, T. Schulte-Herbrüggen, Computations in quantum tensor networks, Linear Algebra Appl. 438 (2) (2013) 750–781.
[15] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317.
[16] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor ring decomposition, preprint, arXiv:1606.05535, 2016.
[17] W. Wang, V. Aggarwal, S. Aeron, Efficient low rank tensor ring completion, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5697–5705.
[18] L. Yuan, C. Li, D. Mandic, J. Cao, Q. Zhao, Tensor ring decomposition with rank minimization on latent space: an efficient approach for tensor completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 9151–9158.
[19] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, S. Yan, Tensor robust principal component analysis: exact recovery of corrupted low-rank tensors via convex optimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5249–5257.
[20] J. Xue, Y. Zhao, W. Liao, J.C.-W. Chan, S.G. Kong, Enhanced sparsity prior model for low-rank tensor completion, IEEE Trans. Neural Netw. Learn. Syst. 31 (11) (2019) 4567–4581.
[21] H. Liu, Y. Li, M. Tsang, Y. Liu, Costco: a neural tensor completion model for sparse tensors, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 324–334.

[22] N. Kargas, N.D. Sidiropoulos, Nonlinear system identification via tensor completion, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 4420–4427.
[23] Z. Xu, F. Yan, et al., Infinite Tucker decomposition: nonparametric Bayesian models for multiway data analysis, preprint, arXiv:1108.6296, 2011.
[24] S. Zhe, K. Zhang, P. Wang, K.-c. Lee, Z. Xu, Y. Qi, Z. Ghahramani, Distributed flexible nonlinear tensor factorization, Adv. Neural Inf. Process. Syst. 29 (2016).
[25] W. He, N. Yokoya, L. Yuan, Q. Zhao, Remote sensing image reconstruction using tensor ring completion and total variation, IEEE Trans. Geosci. Remote Sens. 57 (11) (2019) 8998–9009.
[26] M. Wang, Q. Wang, J. Chanussot, D. Hong, Total variation regularized weighted tensor ring decomposition for missing data recovery in high-dimensional optical remote sensing images, IEEE Geosci. Remote Sens. Lett. 19 (2021) 1–5.
[27] M.G. Asante-Mensah, S. Ahmadi-Asl, A. Cichocki, Matrix and tensor completion using tensor ring decomposition with sparse representation, Mach. Learn.: Sci. Technol. 2 (3) (2021) 035008.
[28] L. Yuan, J. Cao, X. Zhao, Q. Wu, Q. Zhao, Higher-dimension tensor completion via low-rank tensor ring decomposition, in: 2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2018, pp. 1071–1076.
[29] F. Sedighin, A. Cichocki, A.-H. Phan, Adaptive rank selection for tensor ring decomposition, IEEE J. Sel. Top. Signal Process. 15 (3) (2021) 454–463.
[30] C. Li, Z. Sun, Evolutionary topology search for tensor network decomposition, in: International Conference on Machine Learning, PMLR, 2020, pp. 5947–5957.
[31] M. Hashemizadeh, M. Liu, J. Miller, G. Rabusseau, Adaptive learning of tensor network structures, preprint, arXiv:2008.05437, 2020.
[32] C. Nie, H. Wang, Z. Lai, Multi-tensor network representation for high-order tensor completion, preprint, arXiv:2109.04022, 2021.
[33] Z. Long, C. Zhu, J. Liu, Y. Liu, Bayesian low rank tensor ring for image recovery, IEEE Trans. Image Process. 30 (2021) 3568–3580.
[34] J. Yu, C. Li, Q. Zhao, G. Zhao, Tensor-ring nuclear norm minimization and application for visual: data completion, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 3142–3146.
[35] X.P. Li, H.C. So, Robust low-rank tensor completion based on tensor ring rank via $\ell_{p,\epsilon}$-norm, IEEE Trans. Signal Process. 69 (2021) 3685–3698.
[36] H. Huang, Y. Liu, Z. Long, C. Zhu, Robust low-rank tensor ring completion, IEEE Trans. Comput. Imaging 6 (2020) 1117–1126.
[37] J. Yu, G. Zhou, W. Sun, S. Xie, Robust to rank selection: low-rank sparse tensor-ring completion, IEEE Trans. Neural Netw. Learn. Syst. (2021).
[38] Q. Wu, A.-B. Xu, A biased deep tensor factorization network for tensor completion, preprint, arXiv:2105.09629, 2021.
[39] X. Wu, B. Shi, Y. Dong, C. Huang, N.V. Chawla, Neural tensor factorization for temporal interaction learning, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 537–545.
[40] H. Chen, J. Li, Neural tensor model for learning multi-aspect factors in recommender systems, in: IJCAI, 2020, pp. 2449–2455.
[41] L. Wu, C. Shen, A.v.d. Hengel, Personnet: person re-identification with deep convolutional neural networks, preprint, arXiv:1601.07255, 2016.
[42] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[43] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[44] X. Chen, J. Yang, L. Sun, A nonconvex low-rank tensor completion model for spatiotemporal traffic data imputation, Transp. Res., Part C, Emerg. Technol. 117 (2020) 102673.
[45] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.
[46] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, preprint, arXiv:1412.6980, 2014.
[47] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 675–678.
[48] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 922–929.