Neural Tensor Completion for Accurate Network Monitoring

Kun Xie[1,2,3], Huali Lu[1], Xin Wang[4], Gaogang Xie[5,6], Yong Ding[2,7], Dongliang Xie[8], Jigang Wen[5], Dafang Zhang[1]
[1] College of Computer Science and Electronics Engineering, Hunan University, China
[2] Cyberspace Security Research Center, Peng Cheng Laboratory, China
[3] Purple Mountain Laboratories, China
[4] Department of Electrical and Computer Engineering, State University of New York at Stony Brook, USA
[5] Computer Network Information Center, Chinese Academy of Sciences, China
[6] School of Computer Science and Technology, University of Chinese Academy of Sciences, China
[7] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, China
[8] State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China

***Abstract*—Monitoring the performance of a large network is very costly. Instead, a subset of paths or time intervals of the network can be measured while inferring the remaining network data by leveraging their spatio-temporal correlations. The quality of missing data recovery highly relies on the inference algorithms. Tensor completion has attracted some recent attentions with its capability of exploiting the multi-dimensional data structure for more accurate missing data inference. However, current tensor completion algorithms only model the three-order interaction of data features through the inner product, which is insufficient to capture the high-order, nonlinear correlations across different feature dimensions. In this paper, we propose a novel Neural Tensor Completion (NTC) scheme to effectively model three-order interaction among data features with the outer product and build a 3D interaction map. Based on which, we apply 3D convolution to learn features of high-order interaction from the local range to the global range. We demonstrate this will lead to good learning ability. We conduct extensive experiments on two real-world network monitoring datasets, Abilene and WS-DREAM, to demonstrate that NTC can significantly reduce the error in missing data recovery. When the sampling ratio is low at 1%, the recovery error ratios on the testing data are around 0.05 (Abilene) and 0.13 (WS-DREAM) when using NTC, but are 0.99 (Abilene) and 0.99 (WS-DREAM) using the best current tensor completion algorithms, which are 21 times and 8 times larger.**

***Index Terms*—Tensor completion, Sparse network monitoring, Neural network**

I. Introduction

It is very important to efficiently monitor the network to track the network status as well as troubleshoot network incidents and performance issues such as packet losses, latency spikes, and traffic burst [1], [2]. Several network monitoring systems are proposed to monitor networks in the past few years, where Pingmesh [3] and NetNORAD [4] are two examples. These monitoring systems generally send probe packets between each pair of network nodes, which creates a monitoring cost at $O(n^2)$ for a network of n nodes. It is common for a modern data center to contain hundreds of thousands of servers and the scale is growing rapidly at an exponential rate. The network monitoring becomes a challenge as the size of network constantly increases.

Relying on the knowledge of routing paths and network topology, recently deTector [5] proposes to detect and localize network failures with the probing of selected paths to reduce the measurement cost. However, it requires the paths selected well cover all transmission links in the data center, so the measurement cost is still high.

Some recent studies show that network monitoring data such as end-to-end latency and flow traffic have hidden spatio-temporal correlations and thus the matrix of monitoring data has the low rank feature [6]–[9]. This inspires the development of novel sparse network monitoring technique [10]. Sample-based network monitoring is applied in this technique, where measurements are only taken between some random node pairs or at some intervals for a given node pair, and the data of other paths are inferred leveraging the spatio-temporal correlations in network monitoring data. As only a few paths need to be probed, the measurement cost can be largely reduced under the sparse network monitoring scheme.

As part of the applications of network state tracking and forecasting [11], anomaly detection [12] and failure recovery are highly sensitive to the missing (unmeasured) performance data. This makes the accurate recovery of missing monitoring data from partially observed network states an important procedure in sparse network monitoring.

The quality of data recovery highly relies on the inference technique to recover the complete data from partial measurements. Various studies have been made to handle and recover the missing monitoring data. Designed based on purely spatial [13], [14] or temporal [15] information, the data recovery performance of most known approaches is low. To utilize both spatial and temporal information, recent studies exploit matrix completion [6], [8]–[10], [16]–[19] to recover the missing data from a low-rank matrix. Although these approaches present good performance under low data missing ratio, when the data missing ratio is large, the recovery performance suffers as the two-dimensional matrix-based data analysis has the limitation in extracting information.

For better data recovery, very few recent studies [23]–[25] proposes to apply the tensor completion to the network monitoring data recovery. Although these initial efforts demonstratstart to model the network monitoring data as a higher dimensional array called *tensor*, and propose tensor

completion based algorithms for more accurate missing data recovery. Tensors are the high-order generalization of vectors and matrices. Tensor-based multilinear data analysis [20], [21] has shown that tensor models can take full advantages of the multilinear structures to provide better data understanding and information precision.

Despite its effectiveness, we argue that the recovery performance under current tensor completion algorithms can be limited by its linearity, its ability of only modeling three-order feature interaction, and its inability of capturing the correlations across different feature dimensions. To model the network interaction function, current tensor completion techniques project origins, destinations and time into a shared latent feature space, with the interaction between an origin and a destination at a specific time modeled as the inner product of their latent feature vectors in the space. Existing tensor completion algorithms face a few issues:

- Inner product only considers the linear features of data while ignoring the possible non-linearity. Unfortunately, the underlying structure of real-world network monitoring data are often highly non-linear and cannot be accurately approximated by linear models [22].
- Inner product only captures the correlations among the same dimensions of latent features, while the correlations among different feature dimensions are not captured at all. This inability leads to incomplete data acquisition and low accuracy data modeling.
- Inner product assumes that feature dimensions are independent from each other and just multiplies the features with the same weight. In the network field, latent features may explicitly or implicitly represent the features of the type of nodes, Internet access patterns, online behaviors of nodes, working time and so on. These feature dimensions will have different impact on user's Internet access behavior thus the network performance data. So, it is unreasonable to assume that each dimension of latent features has an equal contribution.
- Inner product can only capture the three-order feature interaction, but not the complex structure of the monitoring data that may have high-order correlations.

As current tensor completion algorithms are insufficient in capturing the complex structure of network monitoring data, it will lead to poor missing data recovery performance when the measurement ratio is low.

To address the limitations of existing tensor completion solutions, we propose a novel Neural Tensor Completion model (NTC), which enhances current tensor completion by extracting high-order and non-linear feature interaction across different feature dimensions through neural networks. NTC approach adopts a multi-layer architecture to model an origin-destination-time interaction, where the output of one layer serves as the input of the next layer. From the bottom up, NTC includes input layer, embedding layer, interaction map layer, feature extraction layer based on 3D Convolutional Neural Network (3D CNN), and inferring layer based on Multi-Layer Perception (MLP). The main contributions are summarized as follows.

- We propose to utilize an 3D interaction map to explicitly represent feature interaction among origins, destinations and time, which models tensor completion with an outer product operation to capture complex correlations among feature dimensions.
- To extract the hidden features for more accurate missing data recovery, we propose a novel framework to employ 3D CNN on top of the 3D interaction map.
- We conduct comprehensive experiments on two publicly real-world network monitoring datasets to comparatively evaluate and demonstrate the effectiveness of the proposed method. The experiments results demonstrate that our NTC can achieve significantly better recovery accuracy even when the sampling ratio is very low.
- To the best of our knowledge, this is the first work to propose a novel neural tensor completion framework that combines outer product and 3D CNN to solve the monitoring data recovery problem in the tensor form. We demonstrate that current tensor completion can be interpreted as a specialization of NTC. We expect that 3D CNN's powerful learning ability and tensor's ability of representing hidden structures of data in high dimensions open a new venue for high performance data recovery and analysis.

The rest of the paper is organized as follows. We introduce the related work in Section II. The system model and problem are presented in Section III. We present solution overview, detailed NTC, and analysis of NTC in Sections IV, V, and VI, respectively. Finally, we evaluate the performance of the proposed NTC through extensive experiments in Section VII, and conclude the work in Section VIII.

II. Related work

We are not aware of any other work that provides practical techniques to exploit 3D CNN to enhance tensor completion, and apply efficient neural tensor completion to accurately recover the network monitoring data.

To capture more spatio-temporal features in the network monitoring data, very few recent work [23]–[25] proposes to apply the tensor completion to the network monitoring data recovery. Although these initial efforts demonstrate the potential of applying tensor-based schemes for better monitoring data recovery, we find the performance suffers when the sampling ratio is low. As tensor factorization, such as CANDECOMP PARAFAC (CP) decomposition [26], [27], usually projects different tensor domains (origins, destinations, and time) into a shared latent feature space using a fixed and data independent function, i.e., the inner product as the interaction function of latent features. We argue that the recovery performance under these designs is limited by inner product's linearity, its ability of only modeling of three-order feature interaction, and its inability of capturing the correlations among different feature dimensions in the network performance data.

With the rapid progress of sparse representation techniques, following the compressive sensing [28] and matrix completion [29]–[31] to process one-dimensional and two-dimensional data arrays, tensor completion has attracted lots of research interests recently. In terms of methodologies, many variants [32]–[36] of tensor completion method have been developed to capture the global data structure for recovering the missing data via tensor factorization. These variants are all linear extensions of CP and model the independent three order feature interaction only. As such, current tensor completion algorithms may suffer from insufficient ability of representing real network monitoring data with complex inherent structure and regularities.

To conquer the limitations in current studies, we propose a novel NTC which explores the use of deep neural networks for learning the interaction function, rather than using a simple inner product as the interaction function in current tensor completion algorithms. The deep neural networks (DNNs) have been proven to be capable of approximating any continuous function [37], and more recently they have been found to be effective in several domains, such as speech recognition, computer vision, and text processing [38]. However, we have not found existing work that exploits 3D CNN and outer product to enhance tensor completion algorithms.

To well utilize the learning power of CNN, we build 3D interaction map to explicitly capture correlations between feature dimensions. Above which, we exploit 3D convolution to extract features from the map. From theoretical way, we demonstrate that our design of feeding 3D interaction map to 3D CNN can leverage CNN to learn high-order correlations among embedding dimensions from the local range to the global range in a hierarchical way, thus bringing in significant recovery performance.

III. System Model and Problem

We model the network monitoring data as a 3-way tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$(Fig.1(a)), where I, J, and K correspond to the number of origin nodes, destination nodes, and time intervals monitored, respectively. Each entry x_{ijk} indicates the end-to-end network monitoring data from the origin node i to the destination node j in the time slot k.

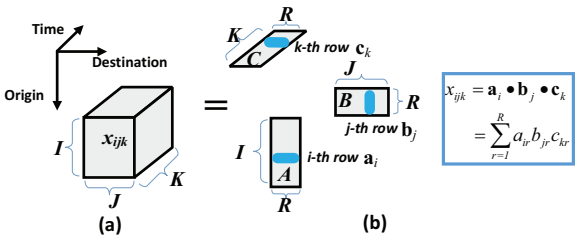


Fig. 1. Tensor model. (a) 3-way monitoring tensor (b) CP decomposition of the 3-way tensor

If there are no network measurement data between a pair of nodes in a given time slot, it leaves the corresponding entry in $\mathcal{X}$ empty. All observed entries are denoted by $\Omega = \{i, j, k | x_{ijk} \ is \ known\}$. To reduce the cost, partial node pairs are often monitored to reduce the measurement load. In addition, there are unavoidable data losses upon severe communication conditions. Therefore, $\mathcal{X}$ is generally a sparse and incomplete tensor. As many network engineering tasks require the complete network performance information, the accurate reconstruction of unmeasured data from partial traffic measurements becomes a key problem, and we refer this problem as the network monitoring data recovery problem.

The aim of this paper is to obtain all data entries of the network monitoring tensor $\mathcal{X}$ based on its measurement samples through the tensor factorization. CP decomposition [26], [27] is a typical tensor factorization approach. Under CP decomposition, we project origins and destinations and time into a shared latent feature space $\mathbb{R}^{1\times R}$, using a vector of latent features to represent an origin or a destination or a time slot. Let $\mathbf{a}_i \in \mathbb{R}^{1\times R}$, $\mathbf{b}_j \in \mathbb{R}^{1\times R}$, and $\mathbf{c}_k \in \mathbb{R}^{1\times R}$ denote the feature vector of origin i, destination j, and time slot k, where R is the dimensionality of the feature vector.

By collecting the vectors $\mathbf{a}_i$, $\mathbf{b}_j$ and $\mathbf{c}_k$, we have tensor $\mathcal{X}$'s feature matrices $\mathbf{A} = \left[\mathbf{a}_1^T, \cdots, \mathbf{a}_I^T\right]^T \in \mathbb{R}^{I\times R}$, $\mathbf{B} = \left[\mathbf{b}_1^T, \cdots, \mathbf{b}_J^T\right]^T \in \mathbb{R}^{J\times R}$, and $\mathbf{C} = \left[\mathbf{c}_1^T, \cdots, \mathbf{c}_K^T\right]^T \in \mathbb{R}^{K\times R}$. As shown in Fig.1, using the feature matrices, CP decomposition of the monitoring tensor $\mathcal{X}$ can be written as

$$\mathcal{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{1}$$

where each entry x_{ijk} can be calculated by x_{ijk}=$\mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k$= $\sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}$.

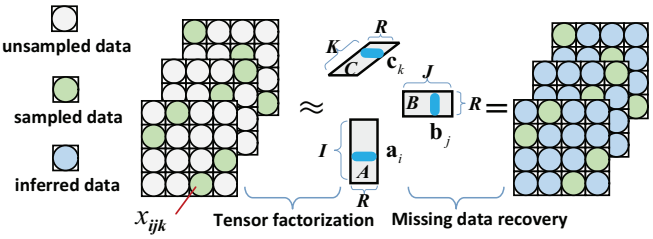


Fig. 2. Overview of tensor completion

To recover the missing network monitoring data, the data recovery problem can be formulated as a tensor completion problem with the goal of learning the latent feature vectors ($\mathbf{a}_i$, $\mathbf{b}_j$, and $\mathbf{c}_k$) to minimize the regularized squared error on the set of observed entries:

$$\min_{\mathbf{a}_i, \mathbf{b}_j, \mathbf{c}_k} \sum_{i,j,k \in \Omega} (x_{ijk} - \mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k)^2 + \alpha \|\mathbf{a}_i\|^2 + \beta \|\mathbf{b}_j\|^2 + \gamma \|\mathbf{c}_k\|^2 \tag{2}$$

where $(x_{ijk} - \mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k)^2$ is the squared error on the observed data x_{ijk}. In Eq.(2), α, β and γ are the regularization coefficients. $\alpha\|\mathbf{a}_i\|^2 + \beta\|\mathbf{b}_j\|^2 + \gamma\|\mathbf{c}_k\|^2$ is added in the formulation to prevent overfitting problem, where the error corresponding to the tensor elements with observed entries are very small while the errors for the inferred data entries are large.

From the view of factorization machine, Eq.(2) can be considered as a bilinear regression model that generates a network monitoring data for each origin-destination-time triple based on the interaction among the origin latent features, destination latent features, and time latent features.

Fig.2 illustrates the main steps involved to solve the CP based tensor completion problem: 1) training the latent feature vectors using the partial measurement samples based on tensor factorization, 2) inferring missing tensor entries using the trained latent feature vectors. After getting three latent feature matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, the network monitoring tensor can be recovered by

$$\hat{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{3}$$

with the recovered entry $\hat{x}_{ijk}=\mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k = \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr}$.

Under CP decomposition model, both feature training (in Eq.(2)) and missing data recovery (in Eq.(3)) use a fixed and data-independent function, the inner product, to capture the interaction among origins, destinations, and time.

Despite its effectiveness and many subsequent developments, CP decomposition [32]–[36] based tensor completion has an inherent limitation in its model design. Let look at Fig.3, under the inner product, the entry value is $\mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k = \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr}$. It is a linear combination of correlations among the same dimension of latent features, i.e., a_{ir}, b_{jr} and c_{kr}, where a_{ir}, b_{jr} and c_{kr} are the r-th dimension of latent features of the origin i, destination j, and time slot k. Using the inner product as the interaction function, tensor completion based on the CP decomposition has the following four drawbacks:

$$\hat{x}_{ijk}=\mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k=\sum_{r=1}^{3} a_{ir}b_{jr}c_{kr} = a_{i1}b_{j1}c_{k1} + a_{i2}b_{j2}c_{k2} + a_{i3}b_{j3}c_{k3}$$

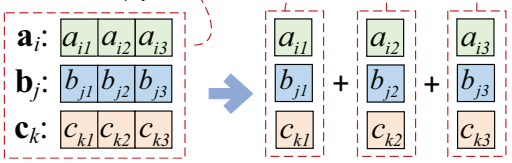


Fig. 3. Description of CP decomposition

(a) Inner product $\mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k = \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr}$ can only capture the linear correlations, but not the non-linearity in the interaction.

(b) Inner product losses the information of correlations across different feature dimensions. As shown in Fig.3, inner product only considers the three-order correlations of the same feature dimension, that is, $a_{i1}b_{j1}c_{k1}$, $a_{i2}b_{j2}c_{k2}$ and $a_{i3}b_{j3}c_{k3}$, while other correlations among different feature dimensions $a_{i1}b_{j1}c_{k2}$, $a_{i1}b_{j1}c_{k3}$, $a_{i1}b_{j2}c_{k1}$, $a_{i1}b_{j2}c_{k2}$, $a_{i1}b_{j2}c_{k3}$, $a_{i1}b_{j3}c_{k1}$, $a_{i1}b_{j3}c_{k2}$, $a_{i1}b_{j3}c_{k3}$, $a_{i2}b_{j1}c_{k1}$, $a_{i2}b_{j1}c_{k2}$, $a_{i2}b_{j1}c_{k3}$, $a_{i2}b_{j2}c_{k1}$, $a_{i2}b_{j2}c_{k3}$, $a_{i2}b_{j3}c_{k1}$, $a_{i2}b_{j3}c_{k2}$, $a_{i2}b_{j3}c_{k3}$, $a_{i3}b_{j1}c_{k1}$, $a_{i3}b_{j1}c_{k2}$, $a_{i3}b_{j1}c_{k3}$, $a_{i3}b_{j2}c_{k1}$, $a_{i3}b_{j2}c_{k2}$, $a_{i3}b_{j2}c_{k3}$, $a_{i3}b_{j3}c_{k1}$, $a_{i3}b_{j3}c_{k2}$ are lost. Such information lost may impact the model accuracy and thus the accuracy of the missing data recovery.

(c) Inner product cannot capture different impacts brought by different feature dimensions. $\mathbf{a}_i \bullet \mathbf{b}_j \bullet \mathbf{c}_k = \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr}$ represents each correlation among the same dimension of latent features (a_{ir}, b_{jr} and c_{kr} are the r-th dimension of feature vector $\mathbf{a}_i, \mathbf{b}_j, \mathbf{c}_k$) equally. As introduced in Section I, latent features may explicitly or implicitly represent the network context information. Different feature dimensions (for example, a_{i1}, a_{i2}, a_{i3} are three feature dimensions of the origin i) have different importance level and impact on the user's network access behavior, thus resulting in different performance data.

(d) Inner product can only capture the three-order feature interaction. It cannot be sufficient to capture the complex structure of real monitoring data, which may have high-order correlations in the feature space.

Therefore, using inner product as the interaction function, only very few correlations among the features of different domains (origin, destination, time slot) are taken into account in the tensor completion design, which leads to low recovery performance.

IV. Solution Overview

To address the limitations of existing solutions, we propose a novel Neural Tensor Completion (NTC) method. It applies multi-layer architecture to model the origin-destination-time interaction x_{ijk}, with the output of one layer being the input of the next layer, as shown in Fig.4. From the bottom up, we introduce the main function modules in our NTC approach.

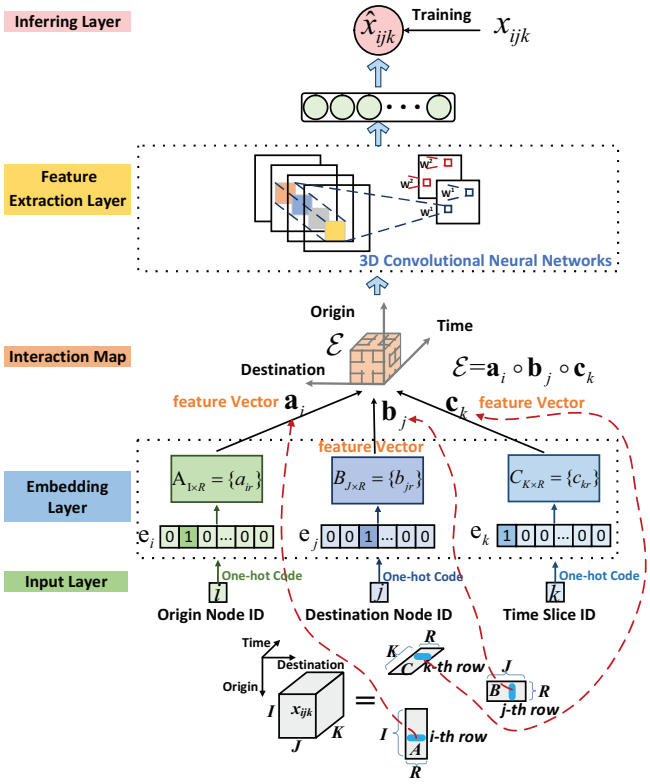


Fig. 4. Multi-layer architecture of NTC

At the bottom input layer, three indication vectors $\mathbf{e}_i$, $\mathbf{e}_j$, and $\mathbf{e}_k$ describe the origin i, destination j, and time slot k, respectively. The next embedding layer is a fully connected layer that projects origin, destination, and time slot to a feature vector. The obtained origin, destination, and time slot

embeddings can be seen as the latent feature vectors in the context of CP decomposition model, which form the input of a 3D interaction map layer to model the three-order interaction among origin, destination, and time slot domains with outer product. It is more effective to capture the correlations among feature dimensions compared with inner product.

To explicitly learn high-order signals, the feature extraction layer employs 3D CNN on the 3D interaction map. Above the feature extraction layer is the inferring layer for the monitoring data, where a single perception layer with a Sigmoid function is used to generate the final inferred monitoring data $\hat{x}_{ijk}$.

In the NTC framework, the neural network based tensor completion problem can be formulated as the training process to minimize the pointwise loss between the recovered data entry $\hat{x}_{ijk}$ and its measurement data value x_{ijk}.

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\Theta_f^C,\Theta_f^M} \sum_{i,j,k\in\Omega} (x_{ijk} - f(\mathbf{e}_i\mathbf{A},\ \mathbf{e}_j\mathbf{B},\ \mathbf{e}_k\mathbf{C}|\Theta_f^C,\Theta_f^M))^2 + \alpha\|\mathbf{A}\| + \beta\|\mathbf{B}\| + \gamma\|\mathbf{C}\| \tag{4}$$

where $\hat{x}_{ijk}{=}f(\mathbf{e}_i\mathbf{A},\ \mathbf{e}_j\mathbf{B},\ \mathbf{e}_k\mathbf{C}|\Theta_f^C,\Theta_f^M)$ is the recovered monitoring data of the entry (i, j, k). $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$, and $\mathbf{C} \in \mathbb{R}^{K\times R}$ represent the latent feature matrices for origins, destinations and time, respectively. $\mathbf{e}_i$, $\mathbf{e}_j$, and $\mathbf{e}_k$ are one-hot code vectors that indicate the origin i, destination j and time k, respectively. Θ_f^C and Θ_f^M denote parameters of the 3D CNN and the MLP. α, β and γ are the regularization coefficients.

Similar to the CP-based tensor completion algorithm (in Section III), NTC includes two steps to recover the missing monitoring data:

(1) Training the latent feature matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and the parameters Θ_f^C and Θ_f^M of the 3D CNN and the MLP by using the partial measurements.

(2) The missing monitoring tensor entry x_{ijk} can be recovered using the trained parameters with $\hat{x}_{ijk}{=}f(\mathbf{e}_i\mathbf{A},\ \mathbf{e}_j\mathbf{B},\ \mathbf{e}_k\mathbf{C}|\Theta_f^C,\Theta_f^M)$.

Specially, in the training phase, we optimize NTC with the regression objective (in Eq.(4)) with minibatch Adam. In each epoch, we first shuffle all observed measurement samples, and then get a mini-batch in a sequential way.

V. Detailed Solution

This section presents the detailed solutions of the main layers in Section IV.

A. Input and Embedding Layer

To solve the problem in Eq.(4), we first need to find a way to represent an origin, a destination and a time slot with a real-valued vector of latent features in a neural network. Embedding-based model is a typical example of representation learning [39], which aims to learn features from raw data for prediction. Inspired by this, we design our neural input and embedding layers to get the latent feature vectors of an origin, a destination and a time slot, as shown in the Fig.4.

Given IDs of an origin i, a destination j, and a time slot k, we can obtain their embeddings $\mathbf{a}_i$, $\mathbf{b}_j$, and $\mathbf{c}_k$ via

$$\mathbf{a}_i = \mathbf{e}_i\mathbf{A},\ \ \mathbf{b}_j = \mathbf{e}_j\mathbf{B},\ \ \mathbf{c}_k = \mathbf{e}_k\mathbf{C} \tag{5}$$

where $\mathbf{e}_i \in \mathbb{R}^{1\times I}$, $\mathbf{e}_j \in \mathbb{R}^{1\times J}$, and $\mathbf{e}_k \in \mathbb{R}^{1\times K}$ are one-hot ID indication vectors, respectively. $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$, and $\mathbf{C} \in \mathbb{R}^{K\times R}$ are the embedding matrices for origins, destinations, and time slots, respectively.

Actually, the embedding size R is equivalent to the rank R in CP decomposition. Embedding matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are equivalent to the latent feature matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ in the CP decomposition. From the input layer to the embedding layer is essentially the process of obtaining the initial three feature matrices in the CP decomposition.

B. Interaction Map Layer

Definition 1. *The outer product of two vectors* $\mathbf{a}\circ\mathbf{b}$ *is the matrix defined by:* $(\mathbf{a}\circ\mathbf{b})_{ij} = a_i b_j$.

Definition 2. *The outer product of three vectors* $\mathbf{a}\circ\mathbf{b}\circ\mathbf{c}$ *is a rank one 3-way tensor defined by:* $(\mathbf{a}\circ\mathbf{b}\circ\mathbf{c})_{ijk} = a_i b_j c_k$.

Above the embedding layer, we propose to use a 3D interaction map to represent feature correlations. Specifically, the interaction among origin i, destination j, and time slot k is modeled by the outer product of their feature vectors

$$\mathcal{E}{=}\mathbf{a}_i\circ\mathbf{b}_j\circ\mathbf{c}_k \tag{6}$$

where $\mathcal{E}$ is a $R\times R\times R$ 3D tensor with its (x, y, z)-th entry is $e_{xyz} = a_{ix}b_{jy}c_{kz}$.

This is the core design of our NTC framework to ensure the effectiveness of NTC. The major advantages of using this 3D interaction map to represent feature interaction are:

(A) The outer product to generate the 3D interaction map is more effective to capture the dimension correlations compared with conventional inner product, as embedding dimensions are independent from each other.

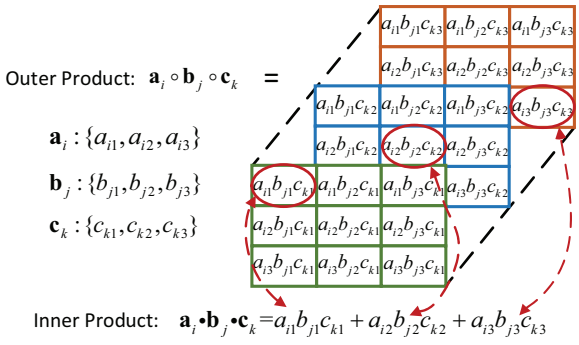


Fig. 5. An example of outer product with 3 dimensions

Given the embedding dimension being 3, Fig.5 compares the outer product $(\mathbf{a}_i\circ\mathbf{b}_j\circ\mathbf{c}_k)$ with inner product $(\mathbf{a}_i\bullet\mathbf{b}_j\bullet\mathbf{c}_k)$. The output of outer product is the 3D interaction map, while the output of inner product is the uniform sum of diagonal elements in our 3D interaction map. Therefore, 3D interaction map facilitates our NTC with more feature signals than CP by accounting for the correlations between among feature dimensions.

(B) While the 3D interaction map itself only considers three-order correlations, its 3D architecture can benefit the modeling of high-order correlations, which can be regarded as convolution in the depth direction. Therefore, a 3D interaction map also allows the powerful 3D CNN to be applied to learn

complex structure hidden in the monitoring data for more accurate recovery of missing data. We will further demonstrate this good properties in Section VI.

C. Feature Extraction Layer

1) Motivation of exploiting 3D CNN

To extract the hidden signals in the network monitoring data, a straightforward solution is to use the MLP network. Despite that MLP is theoretically guaranteed to have a strong representation ability [40], its main drawback of having a large number of parameters cannot be ignored.

We give an example to illustrate that MLP based solution can not work well in our scenario. To apply MLP on the 3D interaction map $\mathcal{E} \in \mathbb{R}^{R\times R\times R}$, we can flat $\mathcal{E}$ to a vector of size R^3, which results in R^3 nodes in the input layer of MLP. If we design MLP following the half-size tower structure, there are $\frac{1}{2}R^6 = R^3 \times (\frac{1}{2}R^3)$ parameters between the first input layer and the second layer, where R^3 corresponds to R^3 nodes on the input layer and $\frac{1}{2}R^3$ corresponds to $\frac{1}{2}R^3$ nodes on the second layer. Given $R = 32$, the first two layers has 536870912 (i.e., $32^3 \times (\frac{1}{2}32^3)$) parameters. Such a large number of parameters make MLP prohibitive to be used in NTC because of following two reasons: 1) It requires powerful machines with large memories to store the model; and 2) It needs a large number of training data to learn the model well.

To address the drawback of MLP, we propose to employ 3D CNN above the 3D interaction map to extract hidden signals. As 3D CNN stacks layers in a locally connected and parameter sharing manner, it utilizes much fewer parameters than MLP, which allows us to build deeper 3D CNN model compared to MLP. Deeper 3D CNN model can bring good benefits of learning the higher-order correlations among embedding dimensions, which further brings the significant gain to the missing data recovery. We will show the benefit in Section VI.

2) The design of feature extraction based on 3D CNN

In our 3D CNN design, there are L convolutional layers followed by one full connected layer. The input tensor to 3D CNN is the 3D interaction map $\mathcal{E} \in \mathbb{R}^{R\times R\times R}$. In every convolutional layer, there exist T convolutional kernels (or filters). In each layer, we first perform 3D convolution between a kernel and the input tensor, after which we add a bias and perform a nonlinear transformation by using ReLU [41] as the activation function to obtain a new output tensor.

The architecture of 3D CNN is designed to extract features from 3D interaction map. This is achieved through 3D convolution operations between convolutional kernels and the input tensor of each layer. As one convolutional kernel can only extract one type of features from the input tensor, to extract more types of features, our 3D CNN adopts T convolutional kernels with $T > 1$. As a result, there are T feature maps in each convolutional layer, where each map stores the feature data extracted by one convolutional kernel. In the experiment, we will show that how T impacts the performance and will set the parameter according to the experiment.

In every convolutional layer, the input tensor of layer l can be represented as a tensor $\mathcal{E}^{l-1}$. When $l = 1$, its input tensor is the 3D interaction map, we have $\mathcal{E}^0 = \mathcal{E}$. We use $\mathcal{G}_i^l \in \mathbb{R}^{l_R\times l_R\times l_R}$ to denote the i-th convolutional kernel in l-th convolutional layer. After the convolution operation, the i-th feature map on the l-th convolutional layer extracted by the convolutional kernel can be expressed as

$$\mathcal{S}_i^l = \mathrm{ReLU}(\mathcal{G}_i^l \otimes \mathcal{E}^{l-1} + b_l) \tag{7}$$

where $\mathcal{S}_i^l$ denotes the i-th feature map in the l-th convolutional layer, $\otimes$ is the convolution operator, and b_l is the bias. In Eq.(7), the entry(x, y, z) in i-th feature map of layer l is given by

$$\begin{cases} e_{xyz}^{li} = \mathrm{ReLU}(b_{li} + \sum_{p=0}^{l_R-1}\sum_{q=0}^{l_R-1}\sum_{r=0}^{l_R-1} g_{pqr}^{li} e^0_{(x+p)(y+q)(z+r)}), l=1 \\ e_{xyz}^{li} = \mathrm{ReLU}(b_{li} + \sum_{m=0}^{T-1}\sum_{p=0}^{l_R-1}\sum_{q=0}^{l_R-1}\sum_{r=0}^{l_R-1} g_{pqr}^{lim} e^{(l-1)m}_{(x+p)(y+q)(z+r)}), l>1 \end{cases} \tag{8}$$

where e_{xyz}^0 denotes the entry (x, y, z) of the 3D interaction map $\mathcal{E}$, b_{li} denotes the bias term for the i-th feature map on the layer l, l_R is the size of the kernel along the origin, destination and time direction. g_{pqr}^{li} and g_{pqr}^{lim} are (p, q, r)-th entry of the i-th kernel on the layer l. When $l = 1$, the input of the convolutional layer l is the 3D interaction map. Therefore, its basic convolution operation is $g_{pqr}^{li} e^0_{(x+p)(y+q)(z+r)}$ where $e^0_{(x+p)(y+q)(z+r)}$ is the entry $(x + p, y + q, z + r)$ in the 3D interaction map. When $l > 1$, as multiple feature maps are all involved in convolution operations to calculate the next layer, the basic convolution operation is denoted by $g_{pqr}^{lim} e^{(l-1)m}_{(x+p)(y+q)(z+r)}$, where $e^{(l-1)m}_{(x+p)(y+q)(z+r)}$ denotes the entry $(x + p, y + q, z + r)$ in the m-th feature map of layer l-1.

For each of the convolutional layers of the CNN, its input is the output of the preceding layer. So the input tensor $\mathcal{E}^l$ of layer l+1 is shown by

$$\mathcal{E}^l = [\mathcal{S}_1^l \mathcal{S}_2^l \cdots \mathcal{S}_T^l] \tag{9}$$

where $\mathcal{S}_i^l (1 \le i \le T)$ is i-th feature map in layer l.

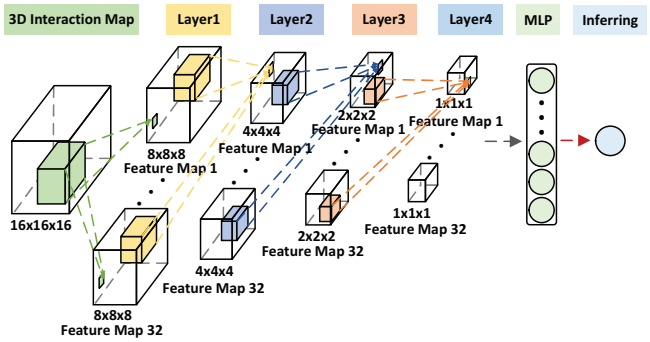


Fig. 6. An example of the architecture of our NTC framework

In Fig.6, the feature dimension $R = 16$. The size of the input 3D interaction map is $16 \times 16 \times 16$. The model has $L = 4$ convolutional layers and $T = 32$ convolutional kernels. Because each convolutional kernel corresponds to a feature map, therefore each convolutional layer has 32 feature maps. In this example, NTC model is designed following half-size tower structure with the size of each convolutional kernel being $2\times2\times2$. As a result, given the input 3D interaction map being

$16 \times 16 \times 16$, the size of each feature map obtained in the 1^{st} convolutional layer is $8 \times 8 \times 8$.

Note that convolutional kernel can be seen as the "locally connected parameter tensor" for a layer, since it is shared in generating all entries of the feature maps of the next layer. This significantly reduces the number of parameters of a convolutional layer compared to that of a fully connected layer. Specifically, in contrast to the 1-layer MLP that has over 8 millions parameters (i.e., $16^3 \times (\frac{1}{2}16^3)$), the above 4-layer CNN has only about 1 thousand parameters, which are several magnitudes smaller. This makes our NTC more stable and generalizable than MLP.

D. Inferring Layer

The last layer of our NTC framework is the inferring layer, which accepts the output of the preceding feature extraction layer i.e., $\mathbf{s}$, to generate the final inferring data $\hat{x}_{ijk}$. In this layer, we feed $\mathbf{s}$ into a single layer perception and use Sigmoid function $\sigma(\cdot)$ to compute $\hat{x}_{ijk}$ as follows

$$\hat{x}_{ijk}=\sigma(\mathbf{w}_{out}^T\mathbf{s}+b) \tag{10}$$

where $\mathbf{w}_{out}^T \in \mathbb{R}^R$ is the weight vector, $\mathbf{s} \in \mathbb{R}^R$ is the flat vector of output of 3D CNN, and b is a bias.

VI. Analysis

From the low convolutional layer to high convolutional layer, 3D CNN extracts signals in the monitoring data from the local range to the global range in a hierarchical way. The depth (i.e., L) of convolutional layers affects the accuracy of neural network training. Generally, the more convolutional layers there are, the stronger their expression ability will be.

In this section, we first give some intuitions on how high-order correlations can be captured by NTC with multiple convolutional layers, then demonstrate that current tensor completion can be interpreted as a specialization of NTC without 3D CNN.

A. NTC's Ability to Capture High-order Correlations

Theorem 1. *Following half-size tower structure, the convolutional kernel size is $2 \times 2 \times 2$ in our NTC model. The l-th convolutional layer can capture the K_l-order feature of embedding dimensions where*

$$K_l = 3(l+1) \tag{11}$$

Due to limited space, the proof is abbreviated. Theorem 1 demonstrates the NTC's strong expression ability of capturing high-order correlations among feature dimensions.

B. Current Tensor Completion is A Specialization of NTC

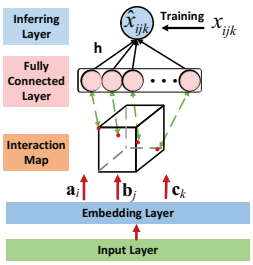


Fig. 7. Current tensor completion is a specialization of NTC

CP is the most popular tensor factorization model for tensor completion and has been investigated extensively in the literature. In Fig.7, we show how CP decomposition can be interpreted as a special case of our NTC framework. CP adopts the inner product as the interaction function. The inner product of $\mathbf{a}_i$, $\mathbf{b}_j$ and $\mathbf{c}_k$ is the elements on the main diagonal of the 3D interaction map obtained by the outer product of $\mathbf{a}_i$, $\mathbf{b}_j$ and $\mathbf{c}_k$, which is also demonstrated in Fig.5. To represent a standard CP in NTC, the following operations can be performed:

(A) Directly connecting the 3D interaction map layer with the inferring layer in our NTC by deleting the 3D CNN based feature extraction layer.

(B) Flatting the diagonal 3D interaction map to form the vector of the inferring layer, then projecting the vector on the inferring layer:

$$\hat{x}_{ijk} = a_{out}(\mathbf{h}^T\mathbf{v}) \tag{12}$$

where $\mathbf{v}$ denotes the vector flatted. a_{out} and $\mathbf{h}$ denote the activation function and edge weights of the inferring layer, respectively. Intuitively, if we use an identity function for a_{out} and $\mathbf{h}$ enforce to be a uniform vector of 1, the modified NTC is tensor completion model based on CP decomposition.

VII. Performance Evaluations

We conduct extensive experiments to answer the following questions: **RQ1** How do the key hyperparameters (i.e., the dimensions of the latent feature space (R), the number of convolutional kernels (T), and the number of convolutional layers (L)) affect NTC's performance? **RQ2** Are the proposed 3D CNN based feature extraction helpful in learning signal in the monitoring data and improving the missing data recovery performance? **RQ3** Can our NTC outperform the state of the art CP based tensor completion methods?

A. Simulation Setup

Data Descriptions. To evaluate the performance of our proposed NTC model, we use the real network monitoring traces from two sources: the U. S. Internet2 Network (**Abilene** [42]) and the real Web service QoS dataset (**WS-DREAM** [43]). The Abilene network consists of 12 nodes thus 144 OD pairs. It collects monitoring data every 5 minutes in 168 days. WS-DREAM records the traffic volume between 142 users and 4,500 Web services over 64 consecutive time slices, at an interval of 15 minutes. We denote the raw trace data as $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$. Given x_{ijk}, for more efficient data process, we adopt ERFIINV transformation to convert data to gaussian data in Eq.(13), where $erfinv(\mathrm{x})$ returns the inverse error function calculated for each element of x, μ and σ represent the mean and the variance, respectively. In our experiments, we set $\mu = 0$, and $\sigma = 1$.

$$x_{ijk} = \mu + \sqrt{2}\sigma erfinv(2x_{ijk} - 1) \tag{13}$$

Metrics. In Table I, four metrics are utilized to evaluate NTC. x_{ijk} and $\hat{x}_{ijk}$ denote the raw data and the recovered data at (i,j,k)-th element of the monitoring data tensor $\mathcal{X}$, where $1 \le i \le I$, $1 \le j \le J$, and $1 \le k \le K$. Ω denotes the sampled entries, $\bar{\Omega}$ denotes the unsampled entries (also called testing entries), and $\Omega+\bar{\Omega}$ denotes the total data entries. m denotes the number of total data entries, $m = \left|\Omega+\bar{\Omega}\right|$.

Only sampled entries are counted in SER, unsampled entries are counted in TER, the total data entries are counted in the MAE and RMSE. MAE is an average of the absolute errors after the interpolation, RMSE is the standard deviation of the differences between recovered values and raw values.

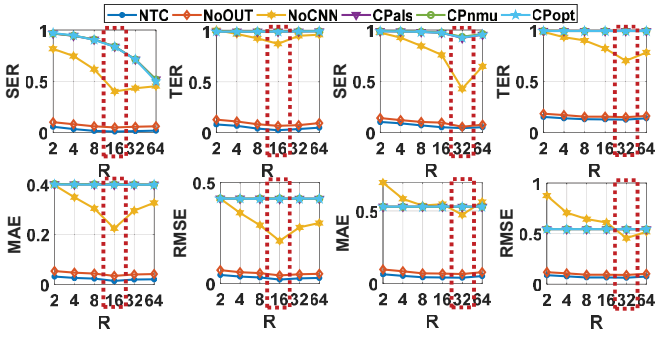
TABLE I
PERFORMANCE METRIC

Sampling Error Ratio (SER)	$\frac{\sqrt{\sum_{i,j,k\in\Omega}(x_{ijk}-\hat{x}_{ijk})^2}}{\sqrt{\sum_{i,j,k\in\Omega}x_{ijk}^2}}$
Testing Error Ratio (TER)	$\frac{\sqrt{\sum_{i,j,k\in\bar{\Omega}}(x_{ijk}-\hat{x}_{ijk})^2}}{\sqrt{\sum_{i,j,k\in\bar{\Omega}}x_{ijk}^2}}$
Mean Absolute Error (MAE)	$\frac{1}{m}(\sum_{i,j,k\in(\Omega+\bar{\Omega})}\lvert x_{ijk}-\hat{x}_{ijk}\rvert)$
Root Mean Square Error (RMSE)	$\sqrt{\frac{1}{m}\sum_{i,j,k\in(\Omega+\bar{\Omega})}(x_{ijk}-\hat{x}_{ijk})^2}$

Baselines. To justify the effectiveness of our proposed NTC, we implement 5 tensor completion algorithms. The first three (CP_{als} [44], CP_{opt} [45], CP_{nmu} [46]) complete the tensor based on the CP decomposition model. The last two (denoted as NoCNN and NoOUT) are the special cases of our NTC. NoCNN is a standard CP tensor completion model described in SectionVI-B. NoOUT follows all algorithms in our NTC with a modified 3D interaction map. Only the diagonals in the map are kept, while all other entries are zeros. In all the experiments, we set the default sample ratio to be 1%.

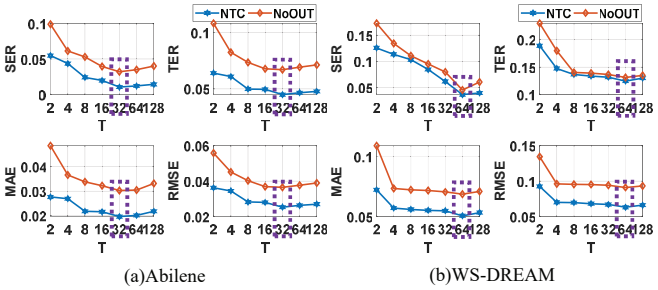
B. Model Analysis and Discussion

Impact of the Dimensions of the Latent Feature Space (R). R represents how many hidden features are captured. It directly impacts the size of the 3D interaction map. In Fig.8, our NTC outperforms all other baselines on all different R, which demonstrates that whatever R is, our proposed 3D interaction map and 3D CNN based feature extraction are very helpful in extracting signals for missing data recovery. A larger R does not necessarily lead to better model. With the increase of R thus the number of features, initially, the recovery performance improves with the decrease of SER, TER, MAE, and RMSE. After R reaches a certain point ($R = 16$ for Abilene and $R = 32$ for WS-DREAM), the increase of R degrades the recovery performance. Overfitting could be a possible reason. According to the results, we set $R = 16$ for Abilene and $R = 32$ for WS-DREAM in the rest of experiments.



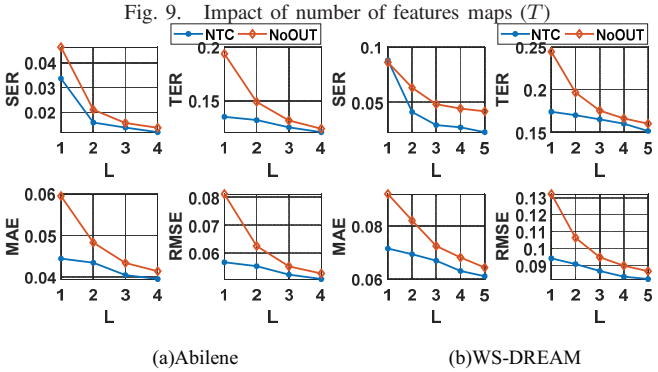
(a)Abilene (b)WS-DREAM

Fig. 8. Impact of the dimensions of the latent feature space (R)



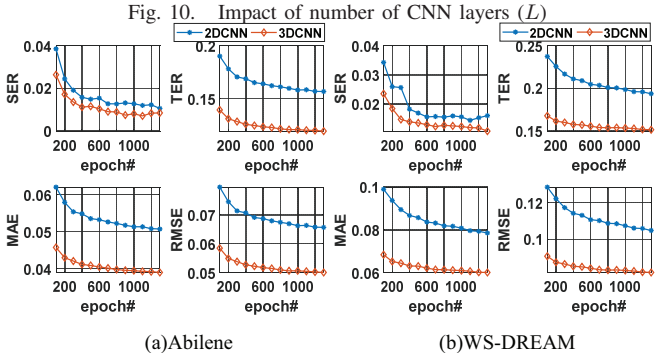
(a)Abilene (b)WS-DREAM

Fig. 9. Impact of number of features maps (T)



(a)Abilene (b)WS-DREAM

Fig. 10. Impact of number of CNN layers (L)



(a)Abilene (b)WS-DREAM

Fig. 11. Impact of CNN model

Impact of Number of Convolutional Kernels (T). It is an important parameter in 3D CNN. Among six tensor completion algorithms implemented, only NTC and NoOUT (a special case of NTC) exploit CNN to extract signals, and are shown in Fig.9. One convolutional kernel can only extract one type of features. With the increase of T, initially, the recovery performance of NTC and NoOUT improves, as extracting more types of features is more helpful for missing data recovery. When T reaches a certain point ($T = 32$ for Abilene and $T = 64$ for WS-DREAM), further increasing T degrades the recovery performance, because redundant features would bring negative impact for missing data recovery. According to the results, we set $T = 32$ for Abilene and $T = 64$ for WS-DREAM. Moreover, NTC always outperforms NoOUT under different T. This is because the 3D map used in NoOUT losses the information of correlations among different feature dimensions, while the 3D map of NTC calculated from the outer product can effectively capture such information.

Impact of Number of Convolutional Layers (L). In 3D CNN, the number of convolutional layers (L) directly affects the learning ability of 3D CNN and further affects the performance of the whole model and data recovery accuracy.

TABLE II
RECOVERY PERFORMANCE ON ABILENE AND WS-DREAM DATASETS

Model	Testing Error Ratio (TER) on Abilene						Mean Absolute Error (MAE) on Abilene					
	1%	3%	5%	7%	9%	11%	1%	3%	5%	7%	9%	11%
CPals	0.9943	0.9814	0.9691	0.9564	0.9438	0.9375	0.5596	0.5514	0.5433	0.5351	0.5270	0.5229
CPnmu	0.9961	0.9810	0.9675	0.9532	0.9386	0.9304	0.5609	0.5515	0.5432	0.5342	0.5252	0.5199
CPopt	0.9945	0.9815	0.9689	0.9565	0.9444	0.9384	0.5597	0.5514	0.5431	0.5350	0.5272	0.5232
NoCNN	1.9986	1.1260	0.1153	0.0796	0.0626	0.0533	0.8985	0.5101	0.0345	0.0255	0.0229	0.0227
NoOUT	0.0555	0.0424	0.0394	0.0386	0.0358	0.0346	0.0266	0.0211	0.0194	0.0193	0.0182	0.0175
NTC	**0.0485**	**0.0361**	**0.0317**	**0.0299**	**0.0287**	**0.0281**	**0.0213**	**0.0154**	**0.0136**	**0.0124**	**0.0121**	**0.0118**
Improve	**21times**	**27times**	**31times**	**32times**	**33times**	**33times**	**26times**	**36times**	**40times**	**43times**	**43times**	**44times**
Model	Testing Error Ratio (TER) on WS-DREAM						Mean Absolute Error (MAE) on WS-DREAM					
	1%	3%	5%	7%	9%	11%	1%	3%	5%	7%	9%	11%
CPals	0.9916	0.9732	0.9549	0.9362	0.9180	0.9086	0.5289	0.5188	0.5088	0.4985	0.4885	0.4834
CPnmu	0.9922	0.9730	0.9536	0.9343	0.9149	0.9055	0.5294	0.5189	0.5084	0.4979	0.4874	0.4822
CPopt	0.9916	0.9731	0.9551	0.9365	0.9183	0.9099	0.5290	0.5187	0.5088	0.4985	0.4884	0.4837
NoCNN	2.8502	0.9560	0.7276	0.6087	0.4071	0.2522	1.2011	0.8565	0.4432	0.3357	0.2345	0.2132
NoOUT	0.1564	0.1446	0.1408	0.1131	0.1130	0.1098	0.0698	0.0613	0.0587	0.0448	0.0437	0.0387
NTC	**0.1252**	**0.1179**	**0.1102**	**0.1090**	**0.1073**	**0.1051**	**0.0565**	**0.0456**	**0.0427**	**0.0410**	**0.0386**	**0.0334**
Improve	**8times**	**8times**	**9times**	**9times**	**9times**	**9times**	**9times**	**11times**	**12times**	**12times**	**13times**	**14times**

As expected, in Fig.10, with the increase of L, more learning ability of 3D CNN and better recovery performance can achieve under both NTC and NoOUT.

The 3D CNN in NTC is designed following the half-size tower structure. That is, the size of every convolutional layer is half the size of the previous layer. Therefore, the maximum number of CNN layers depends on the size of the interaction map, which further depends on the feature dimension R. As we set $R = 16$ for Abilene and $R = 32$ for WS-DREAM, the maximum numbers of convolutional layers for these two data sets are 4 (Abilene) and 5 (WS-DREAM), respectively.

Impact of CNN Model. Besides 3D CNN, we also implement 2D CNN to extract the features for missing data recovery. In Fig.11, as expected, the recovery performance under 3D CNN is much better than that under 2D CNN. The convolution in the depth direction of 3D CNN explicitly models high-order feature correlations while 2D CNN cannot achieve this in such an explicit manner.

C. Performance Comparison

Table II reports the experimental results of 6 tensor completion algorithms. Due to the limited space, Table II only lists the results of two metrics TER and MAE. As expected, with the increase of sampling ratio thus measurement samples, the recovery performance under all tensor completion algorithms increase with the reduction of TER and MAE. Under all sampling ratios, our NTC achieves the best recovery performance with the smallest TER and MAE. Even when the sampling ratio is 1%, a very low sampling ratio, TER under our NTC is around 0.05 (Abilene) and 0.13 (WS-DREAM), while TER under the best conventional CP-based tensor completion is 0.99 (Abilene) and 0.99 (WS-DREAM), which is 21 times and 8 times larger than our NTC. These results demonstrate that NTC is very effective in extracting complex hidden correlations in the monitoring data to achieve significantly better recovery performance.

VIII. CONCLUSION

In this paper, we propose a novel Neural Tensor Completion model (NTC) which enhances current tensor completion by extracting high-order and non-linear feature correlations among different feature dimensions through neural networks. In NTC, we propose a 3D interaction map to explicitly represent complex feature correlations among origin, destination, and time slot. To extract the hidden features for more accurate missing data recovery, we propose a novel framework to employ 3D CNN on the top of the 3D interaction map. We have demonstrated that our design can leverage 3D CNN to learn high-order correlations among feature dimensions. Through extensive experiments on two real world network monitoring datasets, we demonstrate that our NTC can achieve significantly better recovery accuracy even when the sampling ratio is very low.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61972144, Grant 61572184, Grant 61725206, and Grant 61976087, in part by the Hunan Provincial Natural Science Foundation of China under Grant 2017JJ1010, in part by U.S. NSF under Grant ECCS 78929, in part by NSF under Grant CNS 1526843, in part by the Open Project Funding (CARCH201809) of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, and in part by the Peng Cheng Laboratory Project of Guangdong Province under Grant PCL2018KP004.

REFERENCES

[1] P. Tune and M. Roughan, "Spatiotemporal traffic matrix synthesis," in *ACM SIGCOMM CCR*, vol. 45, pp. 579–592, ACM, 2015.

[2] I. Cunha, R. Teixeira, D. Veitch, and C. Diot, "Predicting and tracking internet path changes," in *ACM SIGCOMM CCR*, vol. 41, pp. 122–133, ACM, 2011.

[3] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, *et al.*, "Pingmesh: A large-scale system for data center network latency measurement and analysis," in *ACM SIGCOMM CCR*, vol. 45, pp. 139–152, ACM, 2015.

[4] A. Adams, P. Lapukhov, and J. H. Zeng, "Netno-rad: Troubleshooting networks via end-to-end probing," *Facebook White Paper, available online at: https://code. facebook. com/posts/1534350660228025/netnorad-troubleshooting-networks-via-end-to-endprobing*, 2017.

[5] Y. Peng, J. Yang, C. Wu, C. Guo, C. Hu, and Z. Li, "detector: a topology-aware monitoring system for data center networks," in *USENIX ATC 17*, pp. 55–68, 2017.

[6] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices (extended version)," *IEEE/ACM ToN*, vol. 20, no. 3, pp. 662 – 676, 2012.

[7] K. Xie, C. Peng, X. Wang, G. Xie, J. Wen, J. Cao, D. Zhang, and Z. Qin, "Accurate recovery of internet traffic data under variable rate measurements," *IEEE/ACM ToN*, vol. 26, no. 3, pp. 1137–1150, 2018.

[8] Bang Liu, Di Niu, Zongpeng Li, H. Vicky Zhao, "Network Latency Prediction for Personal Devices: Distance-Feature Decomposition from 3D Sampling," in *IEEE INFOCOM*, 2015.

[9] Rui Zhu, Bang Liu, Di Niu, Zongpeng Li, H. Vicky Zhao, "Network Latency Estimation for Personal Devices: a Matrix Completion Approach," *IEEE/ACM ToN*, vol. 25, no. 2, pp. 724–737, 2017.

[10] K. Xie, L. Wang, X. Wang, G. Xie, G. Zhang, D. Xie, and J. Wen, "Sequential and adaptive sampling for matrix completion in network monitoring systems," in *IEEE INFOCOM*, 2015.

[11] C. R. Kalmanek, S. Misra, and Y. R. Yang, "Guide to reliable internet services and applications," *Springer Science & Business Media*, 2010.

[12] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, J. Cao, and D. Zhang, "Fast tensor factorization for accurate internet anomaly detection," *IEEE/ACM ToN*, vol. 25, no. 6, pp. 3794–3807, 2017.

[13] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural analysis of network traffic flows," in *ACM SIGMETRICS*, 2003.

[14] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach," *IEEE/ACM ToN*, pp. 947–960, 2005.

[15] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," in *ACM IMW*, 2002.

[16] G. Gürsun and M. Crovella, "On traffic matrix completion in the internet," in *ACM IMC 2012*.

[17] Y.-C. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu, "Robust network compressive sensing," in *ACM MOBICOM*, 2014.

[18] M. Mardani and G. Giannakis, "Robust network traffic estimation via sparsity and low rank," in *IEEE ICASSP*, 2013.

[19] R. Du, C. Chen, B. Yang, and X. Guan, "Vanet based traffic estimation: A matrix completion approach," in *IEEE GLOBECOM*, 2013.

[20] Silvia Gandy, Benjamin Recht, and Isao Yamada. 2011. "Tensor completion and low-n-rank tensor recovery via convex optimization. " *Inverse Problems* 27, 2(2011), 025010.

[21] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2013. "Tensor completion for estimating missing values in visual data." *PAMI* 35, 1(2013), 208–220.

[22] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Graph based tensor recovery for accurate internet anomaly detection," in *IEEE INFOCOM*, 2018.

[23] K. Xie, X. Wang, X. Wang, Y. Chen, G. Xie, Y. Ouyang, J. Wen, J. Cao, and D. Zhang, "Accurate Recovery of Missing Network Measurement Data With Localized Tensor Completion," *IEEE/ACM ToN*, vol. 27, no. 6, pp. 2222–2235, 2019.

[24] K. Xie, C. Peng, X. Wang, G. Xie, and J. Wen, "Accurate recovery of internet traffic data under dynamic measurements," in *IEEE INFOCOM*, 2017.

[25] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, and G. Zhang, "Accurate recovery of internet traffic data: A tensor completion approach," in *IEEE INFOCOM*, 2016.

[26] J. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[27] R. A. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis," *UCLA WPP*, vol. 16, no. 1, p. 84, 1970.

[28] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE TIT*, vol. 52, no. 2, pp. 489–509, 2006.

[29] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J OPTIMIZ*, vol. 20, no. 4, pp. 1956–1982, 2010.

[30] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *FOUND COMPUT MATH*, vol. 9, no. 6, pp. 717–772, 2009.

[31] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE TIT*, vol. 56, no. 6, pp. 2980–2998, 2010.

[32] Y. Liu, Z. Long, H. Huang, and C. Zhu, "Low CP rank and Tucker rank tensor completion for estimating missing components in image data," *IEEE TCSVT*, DOI: 10.1109/TCSVT.2019.2901311, 2019.

[33] L. Zhang, L. Song, B. Du and Y. Zhang, "Nonlocal Low-Rank Tensor Completion for Visual Data," *IEEE TCYB*, doi: 10.1109/TCYB.2019.2910151.

[34] Z. Long, Y. Liu, L. Chen, and C. Zhu, "Low rank tensor completion for multiway visual data," *SP*, vol. 155, pp. 301–316, 2019.

[35] M. Ashraphijuo and X. Wang, "Fundamental conditions for low-cp-rank tensor completion," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2116–2145, 2017.

[36] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE TPAMI*, vol. 37, no. 9, pp. 1751–1763, 2015.

[37] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *NN*, 2(5):359-366, 1989.

[38] LeCun, Y., Bengio, Y. and Hinton, G. "Deep learning," *Nature*521,436-444(2015).

[39] Y. Bengio, A. Courville, and P. Vincent, "Representation Learning: A Review and New Perspectives," *IEEE TPAMI*, 35, 8 (2013), 1798–1828.

[40] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," *NN*, 4(2):251–257, 1991.

[41] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas,and H Sebastian Seung. "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit." *Nature*, 405(6789):947, 2000.

[42] *The Abilene Observatory Data Collections*. Accessed: Jul. 20, 2004. [Online]. Available: http://abilene.internet2.edu/observatory/datacollections.html.

[43] Zibin Zheng and M. R. Lyu, "WS-DREAM: A distributed reliability assessment Mechanism for Web Services," *IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, Anchorage, AK, 2008, pp. 392-397.

[44] B. W.Bader, T. G.Kolda et al., "Matlab tensor toolbox version 2.5," Available online, January 2012. [Online]. Available: http://www.sandia.gov/ tgkolda/TensorToolbox/.

[45] E. Acar and T. G. Dunlavy, Daniel M.and Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *J. Chemom.*, vol. 25, no. 2, p. 6786, 2011.

[46] Z. Wen, W. Yin, and Y. Zhang, "Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm," *MPC*, vol. 4, no. 4, pp. 333–361, 2012.