Spatial–Temporal Tensor Graph Convolutional Network for Traffic Speed Prediction

Xuran Xu, Tong Zhang, Chunyan Xu, Zhen Cui, *Member, IEEE*, and Jian Yang

***Abstract*— Accurate traffic speed prediction is crucial for the guidance and management of urban traffic, which at the same time requires a model with a satisfactory computational burden and memory space in applications. In this paper, we propose a factorized Spatial-Temporal Tensor Graph Convolutional Network for traffic speed prediction. Traffic networks are modeled and unified into a graph tensor that integrates spatial and temporal information simultaneously. We extend graph convolution into tensor space and propose a tensor graph convolution network to extract more discriminating features from spatial-temporal graph data. We further introduce Tucker decomposition and derive a factorized tensor convolution to reduce the computational burden, which performs separate filtering in small-scale space, time, and feature modes. Besides, we can benefit from noise suppression of traffic data when discarding those trivial components in the process of tensor decomposition. Extensive experiments on the three real-world datasets demonstrate that our method is more effective than traditional prediction methods, and achieves state-of-the-art performance.**

***Index Terms*— Traffic speed prediction, tensor decomposition, spatial-temporal graph convolutional network, higher-order principal components analysis.**

I. Introduction

ACCURATE traffic prediction plays an important role in promoting the development of the Intelligent Transportation System (ITS). For example, Baidu maps uses a contextual spatial-temporal graph attention network for travel time estimation in [11], and an area-wide traffic speed-flow model was developed for the Singapore CBD in [26]. The content of traffic prediction includes flow, speed, demand, space occupancy, etc. Among them, traffic speed is one of the crucial factors to describe the traffic situation. Traffic speed prediction targets at predicting future speed at those specified traffic locations based on historical traffic information, including but not limited to traffic speed. With the future traffic speed provided, the prediction can be of great significance to traffic signal control systems to facilitate urban traffic planning, traffic management, and traffic control.

Since traffic speed prediction has a wide range of applications, numerous methods have been proposed to tackle this issue. The early works mainly focused on statistical models, e.g., Historical Average (HA), Auto-Regressive Integrated Moving Average (ARIMA) [38], Vector Auto-Regressive [45], Hidden Markov Model [29], Gaussian Process [40], etc. However, traffic data does not always satisfy the assumption of linearity and stationarity [42]. Besides these statistical models, plenty of traditional machine learning methods are applied to traffic speed prediction, such as Support Vector Machine [12], K-Nearest Neighbors [23], Random Forest Regression [15], but the ability of their feature learning is limited due to the shallow architecture. Recently, deep learning-based techniques are flourishing in the task of traffic prediction. For example, convolution neural network (CNN) is used to extract correlations of the spatial domain and recurrent neural network (RNN) is leveraged to learn time-series patterns in [39]. However, CNN fails to capture the irregular topology of traffic networks. To solve this problem, the layout of spatial locations is viewed as a structured graph and a graph convolutional network (GCN) is employed to extract structured spatial dependency in [44]. Moreover, the attention mechanism is introduced to model more discriminating spatial-temporal dependencies in [1].

Although promising success has been achieved by the aforementioned methods, there are still several challenging issues to be tackled, which include three main points. First, the complex spatial-temporal dependencies inside traffic data are still not sophisticatedly modeled. Dependency may exist among spatial locations which form an irregular topology with the traffic evolving. For example, congestion in one position tends to cause congestion in surrounding places. Some previous works, e.g. the spatial-temporal graph neural network method [44], have attempted to capture the spatial-temporal dependencies by combining GCN (modeling spatial information) and Gated Recurrent Unit (GRU) (modeling temporal information). However, hybrid architecture may not be powerful enough to well jointly/simultaneously model the spatial-temporal dependencies as it divides the spatial-temporal learning process into two separate phases. Second, redundant components/noise may exist in the collected traffic data, which may degrade the speed prediction performance. However, few existing algorithms consider alleviating the influence caused by noise/redundancy. Third, the speed prediction for large traffic networks

Manuscript received 10 January 2021; revised 22 May 2021, 25 September 2021, 15 April 2022, and 20 September 2022; accepted 22 September 2022. Date of publication 27 October 2022; date of current version 26 January 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61906094 and Grant 62072244, in part by the Natural Science Foundation of Shandong Province under Grant ZR2020LZH008, and in part by the Innovation Foundation of Maritime Defense Technologies Innovation Center. The Associate Editor for this article was L. Hajibabai. *(Xuran Xu and Tong Zhang contributed equally to this work.) (Corresponding author: Zhen Cui.)*

The authors are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: xuxuran@njust.edu.cn; tong.zhang@njust.edu.cn; zhen.cui@njust.edu.cn; cyx@njust.edu.cn; csjyang@njust.edu.cn).

Digital Object Identifier 10.1109/TITS.2022.3215613

confronts an enormous memory space and computational burden.

Considering the above challenges and difficulties, in this work, we propose a Spatial-Temporal Tensor Graph Convolutional Network (ST-TGCN) for traffic speed prediction tasks. To facilitate the simultaneous modeling of both spatial and temporal dependencies, a graph tensor is first constructed to model the invariant spatial architecture between different locations. Then, a joint graph convolution operation on tensors, aka tensor graph convolution (TG-convolution), is proposed to infer on the constructed graph tensor to extract higher-dimensional features. Based on the TG-convolution learning, spatial-temporal dependencies could be well modeled to promote the traffic speed prediction. Furthermore, considering the redundant components/noise in the traffic data as well as the high memory overhead and computational burden of the graph inference on tensors, we derive a factorized TG-convolution operation to well approximate the original TG-convolution. The constructed graph tensor is first factorized into three separate modes (w.r.t space, time, feature) through tensor decomposition, then the factorized TG-convolution is conducted by applying classic graph convolution correspondingly on each separate mode. Hence, the factorized TG-convolution benefits the ST-TGCN framework in not only suppressing noises through the tensor decomposition, which discards redundant components, but also reducing the memory requirements and computational cost. We verify the proposed factorized ST-TGCN framework on the three traffic speed prediction datasets, and experimental results verify its effectiveness.

In summary, our contributions are three folds:

(1) We propose a novel ST-TGCN framework to simultaneously model the spatial and temporal dependencies which are crucial for traffic speed prediction. To the best of our knowledge, this is the first work to construct a graph tensor on traffic data and conduct the TG-convolution to learn high-level feature representation for the traffic speed prediction task.
(2) We propose the factorized TG-convolution to effectively and efficiently infer on a graph tensor. The factorized TG-convolution not only suppresses the noise by discarding redundant components but also reduces the memory requirement and computational cost.
(3) We evaluate the proposed factorized ST-TGCN framework on the three traffic speed prediction datasets and report the state-of-the-art performances.

II. Related Work

In this section, we first introduce some algorithms about graph convolutional network, then review the previous methods targeting at traffic speed prediction.

A. Graph Convolutional Network

Graph Convolutional Network is a natural extension of CNN on graph-structured data and has shown promising performance in various applications. Previous works adopted GCN to cope with static graph data, which can be categorized into spectral-based and spatial-based models. The former (e.g., [3], [9], and [18]) usually derives from Graph Signals Processing [32], and the latter (e.g., [25]) tends to form neighbor nodes' features into regularization.

Graph-structured data is ubiquitous in the real world, such as recommender systems, social networks, computational biology, and traffic systems, where some of them possess spatial-temporal architecture. To deal with this specific part of data, recent works proposed to utilize various models to model the spatial-temporal dependencies. As a representative, Nguyen et al. [24] proposed a continuous-time dynamic network that is capable of learning time-respecting embeddings from the temporal ordered random walk path. To better capture temporal dependency between graph data, some works integrate GCN with RNN or its variants. For example, Seo et al. [31] proposed a graph convolutional recurrent neural network that utilizes GCN to identify spatial structures and introduces RNN to find dynamic patterns. Pareja et al. [27] utilized RNN to evolve GCN parameters to capture the dynamism of the input graph sequence. Recently, the attention mechanism has been embedded into GCN to further discover the crucial temporal patterns, e.g., Bai et al. [1] and Guo et al. [14] adopted attention mechanism to assign different weights to spatial and temporal dependencies. However, these methods leverage combined two-phase/hybrid ways instead of a unified spatial-temporal framework.

B. Traffic Speed Prediction

Early approaches to cope with the traffic speed prediction problem are generally based on data statistic, among which the Historical Average (HA) and the Auto-Regressive Integrated Moving Average (ARIMA) [38] are two representative models. The HA model computes an average historical traffic speed as a prediction value. Hence, it fails to capture spatial dependency and complex temporal dependency. Although ARIMA linearly combines historical traffic data, it is merely applicable to stationary data. Then traditional machine learning algorithms are introduced to deal with the traffic speed prediction problem, e.g., Support Vector Regression (SVR) is used to predict traffic speed in [13]. Qi and Ishak [29] proposed a Hidden Markov Model to model traffic behavior as a stochastic process and predict traffic speed with state transition probabilities. Chen et al. [5] used the k-nearest neighbor algorithm to select the most informative data for traffic speed prediction.

Recently, deep learning-based approaches have been widely adopted to cope with traffic speed prediction problems. Inspired by the traditional prediction model, traffic speed data is modeled as time series data and deep sequential models are employed. For example, Ma et al. [22] utilized Long Short-Term Neural Networks (LSTM) to capture long temporal dependency of traffic speed data. Moreover, considering that traffic speed states are also influenced by spatial topology structure, Lv et al. [21] proposed to construct adjacent roads feature matrix with nodes' neighbors, where CNN can be used to extract spatial dependency by padding each node's adjacent road matrix to the same size. However, in the application, the regular adjacent grid is inconsistent with the actual layout of traffic networks and additional computation for looking

up adjacent nodes is required. More recently, the GCN-based models are introduced to model traffic data. Generally, these deep graph models can be divided into three types, i.e. the random walk algorithm (e.g., [20]), the hierarchical combination of GCN and RNN (e.g., [2] and [44]), and the pure convolutional architecture (e.g. [43]). To model a non-Euclidean structure traffic networks, Li et al. [20] characterized a directed traffic networks with a bidirectional random walk. However, the random walk algorithm is inefficient for the modeling of the large graph structure. Afterwards, a series of hybrid architectures that combine RNN and GCN are proposed. For example, Zhao et al. [44] combined GCN and GRU to model spatial-temporal patterns of traffic speed data. To constrain the number of traffic sensors to avoid the limitation of complexity and data dimension, Bogaerts et al. [2] introduced max-pooling before convolution to reduce the dimension of the input data. On the other hand, rather than using RNN to capture temporal dependency, some methods tend to utilize purely convolutional structures. Yu et al. [43] proposed spatial-temporal graph convolutional networks (STGCN) to capture complex and long traffic networks dependencies. Specifically, STGCN introduces GCN based on the Chebyshev polynomial approximation to capture spatial dependency and applies a fully convolutional structure on the time axis to capture sequential dependency for each node. Accordingly, the long dependencies are modeled by stacking GCN and temporal convolution layers. Moreover, with the purely convolutional structures, STGCN complies with fast and parallel training procedures. Besides, considering the decent performance in the natural language processing, the attention mechanism is applied to traffic speed prediction framework. For example, Bai et al. [1] and Park et al. [28] embedded attention mechanism into spatial-temporal convolution network to capture more adaptive spatial and temporal dependencies. Besides, traffic data noise should also be taken into consideration for traffic prediction. Accordingly, a few models [4] employee the Kalman filter to reduce data noise.

Different from all previous works above, we propose a novel factorized spatial-temporal tensor graph convolutional network. Rather than using a hierarchical spatial and temporal processing framework, we employ tensor graph convolution to simultaneously model spatial-temporal dependencies. Considering the computational complexity and memory requirement, the factorized TG-convolution is further derived by first extracting the principal mode (aka component) through tensor decomposition, then conducting spectral filtering to each mode. Based on this process, the factorized TG-convolution well approximates the tensor graph convolution and also possesses the advantage of suppressing noise by discarding redundant components.

III. Preliminaries

A. Tensor Operation

We will briefly introduce the standard tensor conception according to the literature [16], [19]. To distinguish tensor, matrix, and vector, we respectively denote them with boldface Euler script, bold uppercase, and bold lowercase, e.g. $\mathcal{X}$, $\mathbf{X}$, and $\mathbf{x}$. Tensor is defined as a multi-dimensional array, a M-dimensional tensor can be represented as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$, where M is the number of dimensions and I_M indicates the size of the M-th dimension. Similar to matrix notation, the element at the index position $(i_1, i_2, \cdots, i_M)$ is denoted as $\mathcal{X}_{i_1 i_2 \cdots i_M}$, and we use the colon (e.g. $\mathcal{X}_{i_1 : i_3 \cdots i_M}$) to denote the full range of a given index. Furthermore, a tensor can be reordered into a matrix by unfolding or flattening, the mode-n matricization of a tensor $\mathcal{X}$ is denoted as $\mathbf{X}_{(n)}$ with the size $I_n \times \left(\prod_{m=1, m \neq n}^{N} I_m\right)$. The n-mode product of a tensor is used to calculate a multiplication with a matrix in the mode n. The n-mode product of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is denoted as $\mathcal{X} \times_n \mathbf{U}$. Formally, the tensor product is calculated as

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_M} = \sum_{i_n=1}^{I_n} \mathcal{X}_{i_1 i_2 \ldots i_M} \mathbf{U}_{j i_n}, \tag{1}$$

where $(\mathcal{X} \times_n \mathbf{U}) \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_M}$.

In addition, the tensor-matrix multiplication satisfies the commutative law and the associative law,

- Given matrices $\mathbf{U} \in \mathbb{R}^{J \times I_n}$,$\mathbf{V} \in \mathbb{R}^{K \times I_m}$ and $n \neq m$, then we have

$$\mathcal{X} \times_m \mathbf{U} \times_n \mathbf{V} = \mathcal{X} \times_n \mathbf{V} \times_m \mathbf{U}. \tag{2}$$

- Given matrices $\mathbf{U} \in \mathbb{R}^{J \times I_n}$,$\mathbf{V} \in \mathbb{R}^{K \times J}$, then

$$\mathcal{X} \times_n \mathbf{U} \times_n \mathbf{V} = \mathcal{X} \times_n (\mathbf{V}\mathbf{U}). \tag{3}$$

Tensor decomposition is widely used for signal processing [33]. As a representative, Tucker decomposition [35], [36], [37] is regarded as a higher-order generalization of matrix singular value decomposition (SVD). It factorizes a tensor into a much smaller core tensor and factor matrices along with each mode. Concretely, the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ can be decomposed into

$$\mathcal{X} = \mathcal{C} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \ldots \times_N \mathbf{A}^{(M)}, \tag{4}$$

where $\mathcal{C} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_M}$ is a core tensor, d_i donates the numbers of components used to summarize the entities in the mode i and $\mathbf{A}^{(i)} \in \mathbb{R}^{I_i \times d_i}$ is the factor matrix of the i-th mode [17]. In particular, we consider the case that the order of $\mathcal{X}$ is equal to 3, which can be written as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. But our method could be easily generalized into higher-order tensors without any extra limitations/constraints.

B. Graph Convolution

From the perspective of graph signal processing, spectral convolution on graphs was introduced in the literature [3]. A graph is defined as $\mathtt{G} = (\mathtt{V}, \mathbf{A}, \mathbf{x})$, where $\mathtt{V}$ is a set of N nodes, $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix, and $\mathbf{x} \in \mathbb{R}^N$ is the graph signal. Each node is associated with a signal.

Spectral convolution [6] is defined as $\mathbf{y} = \mathbf{U} g(\Lambda) \mathbf{U}^\top \mathbf{x}$, where $g(\cdot)$ is a filter function on spectrum matrix Λ, $\mathbf{U}$ is a transformer from space domain to frequency domain. We use normalized graph laplacian matrix $\mathbf{L} = \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{\frac{1}{2}}$ or normalized adjacency matrix $\widetilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{\frac{1}{2}}$ to obtain $\mathbf{U}$, Λ through singular value decomposition. Here $\mathbf{I}_N \in \mathbb{R}^{N \times N}$ is an identity matrix and $\mathbf{D} \in \mathbb{R}^{N \times N}$ is a degree matrix,

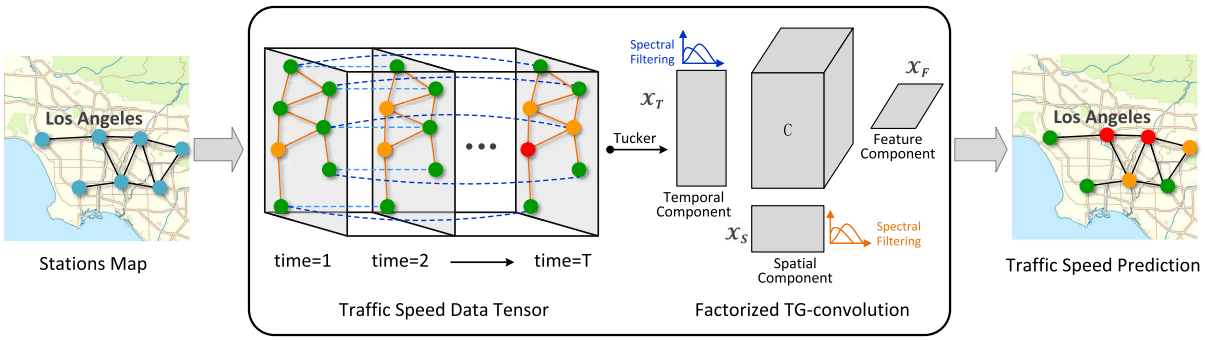

Fig. 1. The architecture of the proposed factorized ST-TGCN framework. For the input traffic networks, the corresponding spatial-temporal graph tensor is first constructed with each traffic location as a node. Then, the factorized TG-convolution operation is conducted to model the complex spatial-temporal correlations among the traffic locations. In this process, two critical operations named Tucker decomposition and factorized spectral filtering (along with spatial and temporal domains, respectively) are involved. More details could be found in the main body. Specifically, in the constructed graph tensor, green, orange, and red nodes represent high, intermediate, and low speed, respectively. The blue dotted edges represent the temporal connection while the orange edges denote spatial correlation.

$\mathbf{D}_{ii} = \sum_{j=0}^{N} \mathbf{A}_{ij}$. Formally, $\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^{\top}$, Λ is a diagonal matrix.

Due to filters are not localized and expensive computation of eigenvalue decomposition, a polynomial filter function [9] is introduced as $g(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$. A graph filters can be approximated by truncated Chebyshev polynomial as $g(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\Lambda})$, where $\widetilde{\Lambda}$ represents the normalized form of Λ and $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$, $T_1 = x$. Accordingly, the convolution on the graph can be rewritten as

$$\mathbf{y} = \sum_{k=0}^{K-1} \theta_k T_k(\widetilde{\mathbf{L}})\mathbf{x}, \tag{5}$$

where $\theta_k | k = 0, \cdots, K-1$ is the filter parameters to be learned.

IV. Problem Description

It is a natural process to view traffic networks as spatial-temporal graph-structure data. For traffic networks, we can use the adjacency matrix $\mathbf{A}$ to describe connection relations between different traffic locations. At each timestamp, we construct a graph G'' to express the states of the traffic networks, denoted as $\mathbb{G} = (\mathbb{V}, \mathbf{A}, \mathbf{X})$, where $\mathbb{V}$ is a set of N nodes w.r.t traffic locations. The state matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ records traffic speeds of all nodes, where each row corresponds to one node and d is the feature dimensionality of nodes, here we use historical T time slice speed as the nodes' initial feature. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ describes connections between different nodes (i.e., locations).

In the task of traffic speed prediction, at the time stamp t, given the historical observations of the whole traffic networks speed data from previous T time stamps, we expect to accurately estimate the node states of the next T' time slice. Formally,

$$(\widehat{\mathbf{y}}^{(t+1)}, \widehat{\mathbf{y}}^{(t+2)}, \cdots, \widehat{\mathbf{y}}^{(t+T')}) = \varphi(\mathbb{G}^{(t)}, \mathbb{G}^{(t-1)}, \cdots, \mathbb{G}^{(t-T+1)}), \tag{6}$$

where φ is the prediction function, and $\mathbb{G}^{(t)} = (\mathbb{V}, \mathbf{A}, \mathbf{X}^{(t)})$. As the traffic locations are usually invariant, commonly, we need to only predict the future traffic speeds $(\widehat{\mathbf{y}}^{(t+1)}, \widehat{\mathbf{y}}^{(t+2)}, \ldots, \widehat{\mathbf{y}}^{(t+T')})$ such that the traffic condition could be coordinated in time. In the following section, we reduce this problem into the graph scheme and propose a spatial-temporal tensor graph convolution method to estimate the states of future time slices.

V. Spatial-Temporal Tensor Graph Convolution

Given the observed historical traffic speed of the previous T time slice, we can use one structured graph to model the traffic information at each timestamp. Thus, we can get the sequential graph data $\{\mathbb{G}^{(t)}, \mathbb{G}^{(t-1)}, \cdots, \mathbb{G}^{(t-T+1)}\}$ and aggregate them in a graph tensor, based on which our aim is to estimate future traffic speeds. As shown in Fig. 1, to model the complex spatial-temporal correlations, we conduct TG-convolution on the constructed graph tensor where the spectral filtering is applied along spatial and temporal axes, respectively.

Concretely, we collect the features/embeddings of all nodes, $\{\mathbf{X}^{(t)}, \mathbf{X}^{(t-1)}, \ldots, \mathbf{X}^{(t-T+1)}\}$ w.r.t previous T time slices, and stack them into a 3-D tensor $\mathcal{X} \in \mathbb{R}^{N \times D \times T}$, where N is the node (i.e. location) number and T is the temporal length. The frontal slices $\mathcal{X}_{::t} = \mathbf{X}^{(t)}$ represents nodes' embeddings at time t. Further, we formulate spatial-temporal graph convolution into the tensor product,

$$\widetilde{\mathcal{X}} = \sum_{k_S, k_T=0}^{p} \mathcal{X} \times_1 \mathbf{A}_S^{k_S} \,\widetilde{\times}_3\, \mathcal{A}_T^{k_T} \times_2 \Theta_{k_S k_T}, \tag{7}$$

where

- Batch tensor product $\widetilde{\times}_3$: The operator performs batch processing in the first dimension. Given two tensors, $\mathcal{A} \in \mathbb{R}^{N \times D \times T}$ and $\mathcal{B} \in \mathbb{R}^{N \times T \times T}$, we define the calculation as follows

$$[\mathcal{A} \,\widetilde{\times}_3\, \mathcal{B}]_{n::} = \mathcal{A}_{n::} \times \mathcal{B}_{n::}. \tag{8}$$

- Spatial adjacency matrix $\mathbf{A}_S \in \mathbb{R}^{N \times N}$: It expresses the relations between different spatial traffic locations. In practice, due to the invariance of traffic locations, we set the shared spatial adjacent relations, i.e., a common adjacency matrix $\mathbf{A}_S$ for all time slices.

- Temporal adjacency tensor $\mathcal{A}_{\mathrm{T}} \in \mathbb{R}^{N\times T\times T}$: It describes different temporal connection patterns of different traffic locations. Each $T \times T$ slice represents the traffic speed connection among different time horizons of one node. Specifically, the temporal connection means the speed correlation of different time horizons. For example, the element in the i-th row and j-th column of the $T\times T$ slice represents the correlation of the traffic speed between the i-th and j-th time horizons.
- Filtering parameters $\Theta \in \mathbb{R}^{D'\times D}$: The original feature space of D-dimension would be embedded into a new D'-dimension space through a tensor product with Θ, which is required to be learned during training.
- The order number $k_{\mathrm{S}}, k_{\mathrm{T}} \in N''$: The power calculation in the spatial domain, i.e., $\mathbf{A}_{\mathrm{S}}^{k_{\mathrm{S}}}$, follows the general matrix power operation. The power operation on tensor takes the way of batch processing at the first dimension, i.e., $[\mathcal{A}_{\mathrm{T}}^{k_{\mathrm{T}}}]_{n::} = [\mathcal{A}_{\mathrm{T}}]_{n::}^{k_{\mathrm{T}}}$. For the zero-order calculation, we have $\mathbf{A}^0 = \mathbf{I}$. The high-order power matrix $\mathbf{A}^k$ indicates the path reachability within k hops, which reflects connection relations of long-distant scope.

In the above formula, we successively conduct three operations on an input feature tensor: spatial aggregation, temporal aggregation, and feature transformation, and finally produce the encoded feature tensor $\widetilde{\mathcal{X}} \in \mathbb{R}^{N\times D'\times T}$. It is worth noting that sorting on three operations could be randomly arranged, and cannot yet alter the final calculation result because of the satisfactory property of the commutative law.

The aim is to estimate traffic speeds of the next T' moments at different locations. To this end, we define a transformation from encoded features to expected results as

$$\widehat{\mathbf{Y}} = \sigma(\widetilde{\mathbf{X}}_{(1)} \times \mathbf{W} + \mathbf{b}), \tag{9}$$

where the matrix $\mathbf{W} \in \mathbb{R}^{D'T\times T'}$ is the projection operation to be learned, $\mathbf{b}$ is a bias term, σ is a non-linear activation function, and $\widetilde{\mathbf{X}}_{(1)} \in \mathbb{R}^{N\times D'T}$ takes a widely-used tensor notation to flatten other dimensions except the specified dimension (here the first dimension). The estimated result $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}^{t+1}, \widehat{\mathbf{y}}^{t+2}, \cdots, \widehat{\mathbf{y}}^{t+T'}] \in \mathbb{R}^{N\times T'}$ are expected to be equal to real values $\mathbf{Y}$ during training. Therefore, we have the objective function as

$$\min_{\Theta,\mathbf{W},\mathbf{b}} \quad \|\mathbf{Y} - \widehat{\mathbf{Y}}\|_F^2 + \lambda \times \varsigma(\Theta, \mathbf{W}, \mathbf{b}), \tag{10}$$

where ς is the regularization term with L_2-norm on the parameters therein, and the balance parameter λ is usually set as a small fraction (e.g., 1.0E-5).

VI. Factorized TG-Convolution

To reduce the computational burden, we take the tensor-factorized way to perform spatial-temporal graph convolution. The advantages are four folds: i) reduce computational complexity, ii) save memory resources, iii) full parallelizability in three operations (i.e., spatial aggregation, temporal aggregation, and feature transformation), and iv) reduce the effects of input noises. Please see a more detailed analysis, which is discussed in the following section.

To simplify the derivation, we consider the case when $k_{\mathrm{S}} = 1$ and $k_{\mathrm{T}} = 1$ in (7), and omit those unnecessary superscripts and subscripts. Then we can have the tensor-matrix multiplication as,

$$\widetilde{\mathcal{X}} = \mathcal{X} \times_1 \mathbf{A}_{\mathrm{S}} \;\widetilde{\times}_3\; \mathcal{A}_{\mathrm{T}} \times_2 \Theta, \tag{11}$$

where $\mathcal{X} \in \mathbb{R}^{N\times D\times T}$ is the input feature tensor, $\mathbf{A}_{\mathrm{S}} \in \mathbb{R}^{N\times N}$ is the adjacency matrix of nodes, $\mathcal{A}_{\mathrm{T}} \in \mathbb{R}^{N\times T\times T}$ is the temporal relation tensor of all nodes and $\Theta \in \mathbb{R}^{D'\times D}$ is the feature transformation matrix.

According to Tucker decomposition, we decompose the input tensor $\mathcal{X}$ into several components, i.e.,

$$\mathcal{X} \approx \mathcal{C} \times_1 \mathbf{X}_{\mathrm{S}} \times_2 \mathbf{X}_{\mathrm{F}} \times_3 \mathbf{X}_{\mathrm{T}}, \tag{12}$$

where the core tensor is $\mathcal{C} \in \mathbb{R}^{n\times d\times t}$, the spatial component is matrix $\mathbf{X}_S \in \mathbb{R}^{N\times n}$, the feature component is matrix $\mathbf{X}_F \in \mathbb{R}^{D\times d}$, and the temporal component is matrix $\mathbf{X}_T \in \mathbb{R}^{T\times t}$. The three components are independent of each other. Moreover, we often have the constraints, $n \ll N, t \ll T, d \ll D$, which largely reduce the computational burden on time and memory requirements. In the case of tensor decomposition, we can derive the following proposition about the computation of encoded features.

Proposition 1: Given the tensor decomposition of $\mathcal{X}$ in (12), we can compute the spatial-temporal encoded feature $\widetilde{\mathcal{X}}$ in (11) by the following formulas

$$[\widetilde{\mathcal{X}}]_{k::} \approx \; \mathcal{C} \times_1 [\widetilde{\mathbf{X}}_S]_{k:} \times_3 [\widetilde{\mathcal{X}}_T]_{k::} \times_2 \widetilde{\mathbf{X}}_F, \tag{13}$$

$$\widetilde{\mathbf{X}}_S = \mathbf{A}_S \times \mathbf{X}_S, \tag{14}$$

$$[\widetilde{\mathcal{X}}_T]_{k::} = [\mathcal{A}_T]_{k::} \times \mathbf{X}_T, \tag{15}$$

$$\widetilde{\mathbf{X}}_F = \Theta \times \mathbf{X}_F, \tag{16}$$

where $\widetilde{\mathbf{X}}_S \in \mathbb{R}^{N\times n}, \widetilde{\mathbf{X}}_F \in \mathbb{R}^{D'\times d}$, and $\widetilde{\mathcal{X}}_T \in \mathbb{R}^{N\times T\times t}$. $[\widetilde{\mathbf{X}}_S]_{k:}$ and $[\mathcal{X}_T]_{k::}$ correspond to spatial and temporal correlation information for node k, respectively. Moreover, the three components $\widetilde{\mathbf{X}}_S, \widetilde{\mathbf{X}}_F, \widetilde{\mathcal{X}}_T$ are irrelevant to each other and so may be calculated in a parallel way.

Proof: We substitute (12) into (11), and then have

$$\begin{aligned}\widetilde{\mathcal{X}} &= \mathcal{X} \times_1 \mathbf{A}_{\mathrm{S}} \;\widetilde{\times}_3\; \mathcal{A}_{\mathrm{T}} \times_2 \Theta \\ &\approx (\mathcal{C} \times_1 \mathbf{X}_{\mathrm{S}} \times_2 \mathbf{X}_{\mathrm{F}} \times_3 \mathbf{X}_{\mathrm{T}}) \times_1 \mathbf{A}_{\mathrm{S}} \;\widetilde{\times}_3\; \mathcal{A}_{\mathrm{T}} \times_2 \Theta.\end{aligned} \tag{17}$$

According to the associative law and the commutative law for tensor-matrix multiplication, we can rewrite $\widetilde{\mathcal{X}}$ as

$$\begin{aligned}\widetilde{\mathcal{X}} &\approx (\mathcal{C} \times_1 (\mathbf{A}_{\mathrm{S}} \times \mathbf{X}_{\mathrm{S}}) \times_2 (\Theta \times \mathbf{X}_{\mathrm{F}}) \;\widetilde{\times}_3\; (\mathcal{A}_{\mathrm{T}} \times_3 \mathbf{X}_{\mathrm{T}}) \\ &= (\mathcal{C} \times_1 \widetilde{\mathbf{X}}_{\mathrm{S}} \times_2 \widetilde{\mathbf{X}}_{\mathrm{F}}) \;\widetilde{\times}_3\; \widetilde{\mathcal{X}}_{\mathrm{T}},\end{aligned} \tag{18}$$

where $\widetilde{\mathbf{X}}_{\mathrm{S}}$, $\widetilde{\mathbf{X}}_{\mathrm{F}}$ and $\widetilde{\mathcal{X}}_{\mathrm{T}}$ are defined in the proposition. Based on the batched tensor multiplication in (8), we take the k-th slice of the first dimension w.r.t nodes, and rewrite it as

$$\begin{aligned}[\widetilde{\mathcal{X}}]_{k::} &\approx [\mathcal{C} \times_1 \widetilde{\mathbf{X}}_{\mathrm{S}} \times_2 \widetilde{\mathbf{X}}_{\mathrm{F}}]_{k::} \;\widetilde{\times}_3\; [\widetilde{\mathcal{X}}_{\mathrm{T}}]_{k::} \\ &= \mathcal{C} \times_1 [\widetilde{\mathbf{X}}_{\mathrm{S}}]_{k:} \times_3 [\widetilde{\mathcal{X}}_{\mathrm{T}}]_{k::} \times_2 \widetilde{\mathbf{X}}_{\mathrm{F}}.\end{aligned} \tag{19}$$

It can be easily inferred that operation $\times_3$ equals $\widetilde{\times}_3$ when $[\widetilde{\mathcal{X}}_{\mathrm{T}}]_{k::} \in \mathbb{R}^{1\times T\times t}$. In the above derivation process, we sometimes preserve the original dimension when taking the slice operation, e.g., $[\widetilde{\mathbf{X}}_{\mathrm{S}}]_{k:} \in \mathbb{R}^{1\times n}$, which is clear in the context.

Further, we can extend the above tensor calculation to the high-order cases, where k_S and $k_T >= 1$. At this time, we only need to update the corresponding adjacency matrices with different orders. Finally, we can aggregate multi-scale receptive field responses as formulated in (7).

Advantages of Factorization In contrast to the original joint spatial-temporal TG-convolution, the case of tensor decomposition have four aspects of advantages. Below we provide a detailed analysis.

- Computation complexity: As the crucial step is the computation of feature encoding in (11), we mainly analyze its computational complexity. In the general convolution of tensor multiplication, the complexity is about $\mathcal{O}(N^2DT + NDT^2 + NDD'T)$.
  In the factorized version, we need to compute the formulas (12)∼(16). According to the analysis in the literature [7], [10],[1] the cost for Tucker decomposition in (12) with higher-order singular value decomposition (HOSVD) and higher order orthogonal iteration (HOOI) [8] is respectively about $\mathcal{O}(2NDT(N+D+T) - \frac{2}{3}(N^3 + D^3 + T^3) + 5(n^2DT + Nd^2T + NDt^2))$ and $\mathcal{O}(NTD(n+t+d)+DT(7n^2+n)+NT(7d^2+d)+ND(7t^2+t))$. The complexity of calculations from (14) to (16) is about $\mathcal{O}(N^2n+DD'd+NT^2t)$, which contains the computation in spatial domain, temporal domain and feature domain. The decoding process in (18) is about $\mathcal{O}(Nndt + NTdt + ND'Td)$. As there exist $n \ll N$, $d \ll D, t \ll T$ and $D' \sim D$ in practice, and $T \leq N$, $D \leq N$ or $D \sim N$ at most for large traffic networks, so the cost of factorized ST-TGCN is dominated by tensor decomposition term: $\mathcal{O}(2NDT(N + D + T))$, which holds on the same level of computation complexity as the original.
- Memory requirement: After tensor decomposition, we only need the memory space of $nN+tT+dD+ndt$, which is far less than the original memory space NDT. Subsequently, for the factorized version, the memory spaces in the next calculations would be decreased compared with the original version.
- Parallel computation: After tensor decomposition, the computation in spatial domain, temporal domain, and feature domain, i.e., (14)∼(16), can be conducted in a fully parallel way. In contrast, these three operations in (7) cannot be performed for parallelization.
- Noise suppression: In tensor decomposition, we preserve those most crucial components by discarding those trivial components. Commonly, those trivial components contain more noise information. This phenomenon is validated for many subspace methods such as principal component analysis (PCA). Hence, our method can benefit from tensor decomposition, which can suppress noises to some extent. In the following experiment, we also verify this point by comparing the factorized version with the original ST-TGCN.

[1]Of course, some fast algorithms may be also employed to speed up tensor decomposition.

VII. Experiments

In this section, we present three traffic speed datasets to evaluate the methods, then compare the proposed factorized ST-TGCN with existing state-of-the-art works, and finally analyze factorized ST-TGCN by conducting ablation studies.

A. Datasets

1) SZ-Taxi: This dataset is collected from a taxi trajectory of Shenzhen, China. It defines 156 major roads as nodes, which are characterized by the speed at which taxis pass through the roads. And the adjacency matrix represents the positional connections between roads. These data are collected every 15 minutes, ranging from 1/1/2015 to 1/31/2015.

2) Los-Loop: This dataset is collected by loop detectors deployed on the highway of Los Angeles in America. It contains 207 nodes characterized by the speed collected by loop detectors. And adjacency matrix represents the positional connections between loop detectors. These data are collected every 5 minutes, ranging from 3/1/2012 to 3/7/2012.

3) PeMSD7(L): This dataset is collected from the California Transportation Agency Performance Measurement System[2](PeMS). The processing of the dataset follows the literature [43]. In detail, we randomly select 1026 stations among District 7 of California and construct stations adjacent matrix according to their latitude and longitude. These data are collected every 5 minutes, ranging from the weekdays of 5/1/2012 to 7/1/2012.

For the experiments, we strictly follow the widely adopted protocol in previous works [44]. We split 80% of data for training and 20% of data for testing. The proposed model predicts traffic speed for prediction horizons from 15 to 60 minutes by using previous data of 12 time slices.

B. Implementation Details

All experiments in this work are conducted on a single NVIDIA RTX2080Ti. In the experiments, We firstly utilize two fully connected layers to map nodes into a 128-dimension feature vector. Then we construct two-layer graph convolutional neural networks with 128 and 64 filters respectively. Specifically, a layer of tensor graph convolutional network of D' filters consists of spatial filters with the shape of $D \times D'$ and temporal filters with the shape of $N \times T \times T$, where D is the dimension of input data and D' is the dimension of the output data. In GCN layers, we use $relu(\cdot)$ as an activation function and order numbers k_S, k_T are set to 2. Finally, we use Adam optimizer with a learning rate of 0.001 and batch size of 32 to train our proposed model for 500 epochs.

C. Metrics

Following [1] and [44], we adopt the five metrics to comprehensively evaluate factorized ST-TGCN,

- Root Mean Squared Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}^i - y^i)^2}, \tag{20}$$

[2]http://pems.dot.ca.gov/

TABLE I
COMPARISONS WITH STATE-OF-THE-ART METHODS (* MEANS THE VALUE IS PRETTY SMALL)

Model	Time	SZ-taxi					Los-loop				
		$RMSE$	MAE	$Accuracy$	R^2	var	$RMSE$	MAE	$Accuracy$	R^2	var
HA	15min	4.2951	2.7815	0.7008	0.8307	0.8307	7.4427	4.0145	0.8733	0.7121	0.7121
ARIMA	15min	7.2406	4.9824	0.4463	*	0.0035	10.0439	7.6832	0.8275	0.0025	*
	30min	6.7899	4.6765	0.3845	*	0.0081	9.3450	0.6891	0.8275	0.0031	*
	45min	6.7852	4.6734	0.3847	*	0.0087	10.0508	7.6924	0.8273	*	0.0035
	60min	6.7708	4.6655	0.3851	*	0.0111	10.0538	7.6952	0.8273	*	0.0036
SVR	15min	4.1455	2.6233	0.7112	0.8423	0.8424	6.0084	3.7285	0.8977	0.8123	0.8146
	30min	4.1628	2.6875	0.7100	0.8410	0.8413	6.9588	3.7248	0.8815	0.7492	0.7523
	45min	4.1885	2.7359	0.7082	0.8391	0.8397	7.7504	4.1288	0.8680	0.6899	0.6947
	60min	4.2156	2.7751	0.7063	0.8370	0.8379	8.4388	4.5036	0.8562	0.6336	0.5593
GCN	15min	5.6596	4.2367	0.6107	0.6654	0.6655	7.7922	5.3525	0.8673	0.6843	0.6844
	30min	5.6918	4.2647	0.6085	0.6616	0.6617	8.3353	5.6118	0.8581	0.6402	0.6404
	45min	5.7142	4.2844	0.6069	0.6589	0.6564	8.8036	5.9534	0.8500	0.5999	0.6001
	60min	5.7361	4.3034	0.6054	0.6590	0.6564	9.2657	6.2892	0.8421	0.5583	0.5593
GRU	15min	3.9994	2.5955	0.7249	0.8329	0.8329	5.2182	3.0602	0.9109	0.8576	0.8577
	30min	4.0942	2.6906	0.7184	0.8249	0.8250	6.2802	3.6505	0.8931	0.7957	0.7958
	45min	4.1534	2.7743	0.7143	0.8198	0.8199	7.0343	4.5186	0.8801	0.7446	0.7451
	60min	4.0747	2.7712	0.7197	0.8266	0.8267	7.6621	0.7957	0.8694	0.6980	0.6984
STGCN	15min	3.9941	2.6608	0.7211	0.8525	0.8529	6.0844	3.3577	0.8958	0.8155	0.8162
	30min	4.0336	2.6937	0.7186	0.8498	0.8503	7.6831	4.1249	0.8686	0.7066	0.7098
	45min	4.0553	2.7205	0.7173	0.8482	0.8491	8.6429	4.6632	0.8523	0.6295	0.6358
	60min	4.0666	2.7334	0.7169	0.8475	0.8484	9.4822	5.1523	0.8380	0.5547	0.5649
T-GCN	15min	3.9162	2.7061	0.7306	0.8541	0.8626	5.1264	3.1802	0.9172	0.8634	0.8634
	30min	3.9617	2.7452	0.7275	0.8523	0.8523	6.0598	3.7466	0.8968	0.8098	0.8100
	45min	3.9950	2.7666	0.7252	0.8509	0.8509	6.7065	4.1158	0.8857	0.7679	0.7684
	60min	4.0141	2.7889	0.7238	0.8503	0.8504	7.2677	4.6021	0.8762	0.7283	0.7290
A3T-GCN	15min	3.8989	2.6840	0.7318	0.8512	0.8512	5.0904	3.1365	0.9133	0.8653	0.8653
	30min	3.9228	2.7038	0.7302	0.8493	0.8493	5.9974	3.6610	0.8979	0.8137	0.8137
	45min	3.9461	2.7261	0.7286	0.8474	0.8474	6.6840	4.1712	0.8861	0.7694	0.7705
	60min	3.9707	2.7391	0.7269	0.8454	0.8454	7.0990	4.2343	0.8790	0.7407	0.7415
factorized ST-TGCN	15min	**3.1080**	**2.0198**	**0.7835**	**0.9114**	**0.9114**	**3.5969**	**2.2265**	**0.9387**	**0.9330**	**0.9330**
	30min	**3.5181**	**2.2951**	**0.7548**	**0.8865**	**0.8865**	**4.9283**	**2.8690**	**0.9159**	**0.8749**	**0.8749**
	45min	**3.6039**	**2.3689**	**0.7488**	**0.8809**	**0.8809**	**5.5733**	**3.3600**	**0.9049**	**0.8402**	**0.8409**
	60min	**3.7358**	**2.4476**	**0.7396**	**0.8720**	**0.8720**	**5.8225**	**3.4772**	**0.9006**	**0.8258**	**0.8263**

- Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}^i - y^i|, \tag{21}$$

- Accuracy

$$Accuracy = 1 - \frac{\sqrt{\sum_{i=1}^{N}|\hat{y}^i - y^i|^2}}{\sqrt{\sum_{i=1}^{N}|y^i|^2}}, \tag{22}$$

- R-squared

$$R^2 = 1 - \frac{\sum_{i=1}^{N}|\hat{y}^i - y^i|^2}{\sum_{i=1}^{N}|y^i - \frac{1}{N}\sum_{k=1}^{N}y^k|^2}, \tag{23}$$

- Variance

$$var = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N}|(y^i - \hat{y}^i) - \frac{1}{N}\sum_{k=1}^{N}(y^k - \hat{y}^k)|^2}{\frac{1}{N}\sum_{i=1}^{N}|y^i - \frac{1}{N}\sum_{k=1}^{N}y^k|^2}, \tag{24}$$

where y^i, $\hat{y}^i$ represent ground-truth and prediction, N is the number of nodes. As the square root operation of RMSE amplifies the gaps between ground-truth and prediction, the MAE is introduced. Considering RMSE and MAE merely calculate the distance between the real data and the predicted data, three additional metrics named Accuracy, R^2, var are introduced to normalize value, where R^2 is the coefficient of determination and var explains the variance score.

D. Comparisons With State-of-the-Art Methods

To comprehensively evaluate our proposed factorized ST-TGCN, we compare it with traditional methods (HA, ARIMA [38], and SVR [34]) and deep learning-based methods (GCN [18], GRU [44], STGCN [43], T-GCN [44], and A3T-GCN [1]). Since the HA model simply computes an average of past traffic speed and uses the identical value as prediction value for horizons from 15 minutes to 60 minutes, we report its performance for the prediction horizons of 15 minutes. According to Table I, overall, deep learning-based methods generally outperform the traditional ones except for the GCN which doesn't consider the temporal dynamics. In this section, we concentrate on analyzing experimental results for the prediction horizons of 15 minutes and other prediction horizons results have similar conclusions. Based on the above experimental results, while SVR achieves satisfying performance among traditional models, our model still

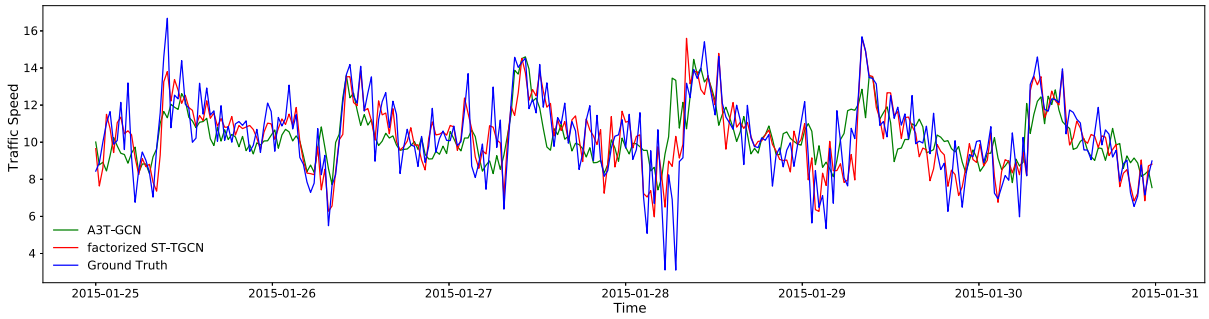

Fig. 2. Coarse results of the speed prediction on the SZ-taxi for prediction horizons of 15 minutes.

TABLE II
EXPERIMENTAL RESULTS ON THE PEMSD7(L) DATASET

Model	Time	$RMSE$	MAE	$Accuracy$	R^2	var
ST-GCN	15min	3.7498	1.9514	0.9388	0.9107	0.9107
	30min	5.5416	2.7631	0.9092	0.8083	0.8086
	45min	6.6775	3.2568	0.8906	0.7221	0.7224
	60min	7.5160	3.6827	0.8769	0.6484	0.6488
A3T-GCN	15min	4.3329	2.4846	0.9296	0.8772	0.8773
	30min	5.2441	2.9788	0.9147	0.8203	0.8211
	45min	6.3629	3.6664	0.8965	0.7356	0.7356
	60min	6.4278	3.8883	0.8954	0.7311	0.7332
factorized ST-TGCN	15min	**2.1151**	**1.2483**	**0.9657**	**0.9708**	**0.9710**
	30min	**3.0045**	**1.6365**	**0.9512**	**0.9409**	**0.9409**
	45min	**3.8756**	**2.1569**	**0.9370**	**0.9016**	**0.9041**
	60min	**4.8583**	**2.6013**	**0.9210**	**0.8454**	**0.8454**

TABLE III
PERFORMANCE GAIN OF THE PROPOSED FACTORIZED ST-TGCN IN R^2 TAKING THE GRU MODEL AS THE BASE

Model	SZ-taxi				Los-loop			
	15min	30min	45min	60min	15min	30min	45min	60min
GRU	base	base	base	base	base	base	base	base
A3T-GCN	+1.8%	+2.4%	+2.8%	+1.9%	+0.8%	+1.8%	+2.5%	+4.3%
factorized ST-TGCN	+7.9%	+6.2%	+6.1%	+4.5%	+7.5%	+7.9%	+9.6%	+12.8%

TABLE IV
COMPARISONS OF INFERENCE TIME (S)

Dataset	ARIMA	SVR	GRU	STGCN	T-GCN	A3T-TGCN	factorized ST-TGCN
SZ-taxi	0.6075	0.0014	0.0097	0.0469	0.0166	0.0173	0.0683
Los-loop	0.5188	0.0010	0.0125	0.0672	0.0180	0.0231	0.0833
PeMSD7(L)	2.1282	0.0047	0.0221	0.3239	0.0348	0.0926	0.6574

obtains 7% and 12% improvement in R^2 on the datasets SZ-taxi and Los-loop respectively. Such improvement can be verified on RMSE, MAE, Accuracy, and var metrics as well. Traditional methods predict speed merely by modeling temporal dependency of traffic data, but irregular spatial and non-linear temporal dependencies are also necessary. Then we compare the proposed method to deep learning-based models. In Table I, GCN fails to show great performance for it absolutely ignores the temporal dependence. GRU only considers temporal dependency and factorized ST-TGCN increases R^2 by about 8% on the SZ-taxi. Therefore, taking one type of dependency into account is far from sufficient for deep learning models. Finally, we observe that the graph-based deep learning model including STGCN, T-GCN, and A3T-GCN have a comparable performance. Compared to A3T-GCN which obtains the best results so far, factorized ST-TGCN decreases the RMSE and MAE by about 20% and 25% on the SZ-taxi. Besides, it also increases Accuracy, R^2, and var by about 5%, 6%, and 6% respectively. To make it more clear how much performance improvement is achieved by our factorized ST-TGCN, we specifically present the performance gain by using one of the most crucial metrics named R^2 while taking the GRU as the base. The result is shown in Table III. Additionally, in Table II, we evaluate our proposed method on the larger traffic speed dataset PeMSD7(L). With the sufficient training data provided, we observed that the proposed method significantly improves the performance by decreasing RMSE to 2.1151. Compared to the existing state-of-the-art method A3T-GCN with about 9% improvement in R^2 for horizons of 15 minutes.

Besides, we show the inference time of those models (at the horizon of 15 minutes) in Table IV. Compared with A3T-GCN, our factorized ST-TGCN consumes more time due to the tucker decomposition operation. However, our factorized ST-TGCN still well satisfies the application requirements as it only costs less than 0.1s to predict the traffic speed more than one hour later. Therefore, our factorized ST-TGCN generally outperforms A3T-GCN with a significant performance gain while costing the acceptable inference time.

We visualize the results of traffic prediction in Fig. 2, Fig. 3 (for the coarse time granularity), and Fig. 4 (for the fine time granularity). It can be observed that factorized ST-TGCN shows different performance on the two datasets, we infer that this is caused by speed changes in the SZ-taxi is more dramatic than those in the Los-loop.

E. Ablation Study

1) Component Analysis of Factorized ST-TGCN: To confirm that both spatial and temporal dependencies are crucial to traffic speed prediction, we devise a spatial tensor graph convolutional neural network (factorized S-TGCN) and

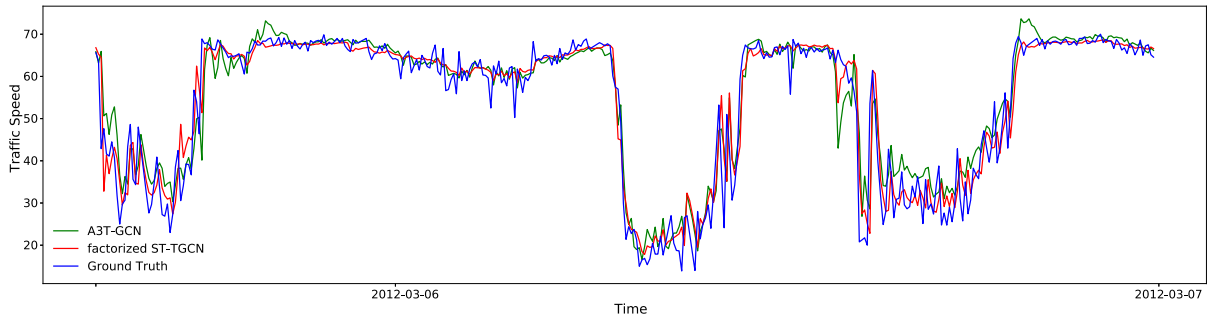

Fig. 3. Coarse results of the speed prediction on the Los-loop for prediction horizons of 15 minutes.

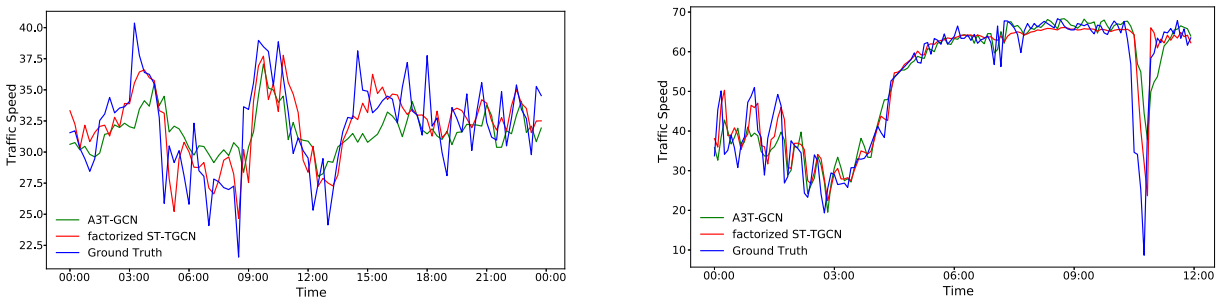

Fig. 4. Fine results of one day speed prediction on the SZ-taxi (left) and Los-loop (right) for prediction horizons of 15 minutes.

TABLE V
COMPONENT ANALYSIS OF FACTORIZED ST-TGCN

Dataset	Model	$RMSE$	MAE	$Accuracy$	R^2	var
SZ-taxi	factorized S-TGCN	3.6849	2.4511	0.7433	0.8754	0.8754
	factorized T-TGCN	3.2894	2.1537	0.7709	0.9007	0.9007
	factorized ST-TGCN	**3.1080**	**2.0198**	**0.7835**	**0.9114**	**0.9114**
Los-loop	factorized S-TGCN	4.2940	2.5598	0.9268	0.9045	0.9053
	factorized T-TGCN	4.5326	2.7062	0.9227	0.8937	0.8937
	factorized ST-TGCN	**3.5969**	**2.2265**	**0.9387**	**0.9330**	**0.9330**

TABLE VI
TUCKER DECOMPOSITION ON THE SZ-TAXI

Model	Time	$RMSE$	MAE	$Accuracy$	R^2	var
ST-TGCN	15min	3.8976	2.5799	0.7285	0.8606	0.8607
	30min	3.9261	2.5927	0.7264	0.8586	0.8586
	45min	3.9362	2.5966	0.7256	0.8579	0.8579
	60min	3.9541	2.6128	0.7244	0.8565	0.8565
factorized ST-TGCN	15min	**3.1080**	**2.0198**	**0.7835**	**0.9114**	**0.9114**
	30min	**3.5181**	**2.2951**	**0.7548**	**0.8865**	**0.8865**
	45min	**3.6039**	**2.3689**	**0.7488**	**0.8809**	**0.8809**
	60min	**3.7358**	**2.4476**	**0.7396**	**0.8720**	**0.8720**

a temporal tensor graph convolutional neural network (factorized T-TGCN). Factorized S-TGCN merely regards nodes that are jointed in the spatial domain as neighbor nodes, factorized T-TGCN considers nodes are just connected in the temporal pattern and our proposed factorized ST-TGCN integrates them. Table V shows the performance of three models on the two datasets for prediction horizons of 15 minutes. Compared with factorized S-TGCN, approximately 4% improvement in R^2 is attained by factorized ST-TGCN on the SZ-taxi. In addition to this, factorized ST-TGCN obtains about 1% improvement compared with factorized T-TGCN. These demonstrate that spatial-temporal dependencies are important. We consider that the better performance of factorized ST-TGCN is not only attributed to comprehensively considering spatial-temporal dependencies but also due to tensor decomposition operation. Therefore, we further explore the Tucker decomposition operation in the following parts.

2) The Effectiveness of Tucker Decomposition: To explore the effectiveness of Tucker decomposition, we conduct comparative experiments between ST-TGCN and factorized ST-TGCN. ST-TGCN directly conducts a multi-graph convolutional neural network in the spatial-temporal domains without tensor decomposition operation, which defined as (7). As shown in Table VI, the factorized ST-TGCN is capable of showing better performance than the ST-TGCN model. For the prediction horizons of 15 minutes on the dataset SZ-taxi, factorized ST-TGCN obtains about 5% improvement in R^2. Moreover, as shown in Table VII, factorized ST-TGCN is capable of achieving roughly equivalent improvement on the Los-loop dataset for the prediction horizons of 15 minutes. We also observe that factorized ST-TGCN still outperforms ST-TGCN with prediction horizons from 15 to 60 minutes, so factorized ST-TGCN successfully maintains effectiveness and robustness for the long-term prediction. Such improvement demonstrates that Tucker decomposition as one of the higher-order generalizations of the matrix SVD is useful for

TABLE VII
TUCKER DECOMPOSITION ON THE LOS-LOOP

Model	Time	$RMSE$	MAE	$Accuracy$	R^2	var
ST-TGCN	15min	4.5068	2.6846	0.9232	0.8948	0.8948
	30min	5.2875	3.0651	0.9098	0.8560	0.8561
	45min	5.8455	3.4912	0.9002	0.8243	0.8247
	60min	6.2267	3.6849	0.8937	0.8007	0.8008
factorized ST-TGCN	15min	**3.5969**	**2.2265**	**0.9387**	**0.9330**	**0.9330**
	30min	**4.9283**	**2.8690**	**0.9159**	**0.8749**	**0.8749**
	45min	**5.5733**	**3.3600**	**0.9049**	**0.8402**	**0.8409**
	60min	**5.8225**	**3.4772**	**0.9006**	**0.8258**	**0.8263**

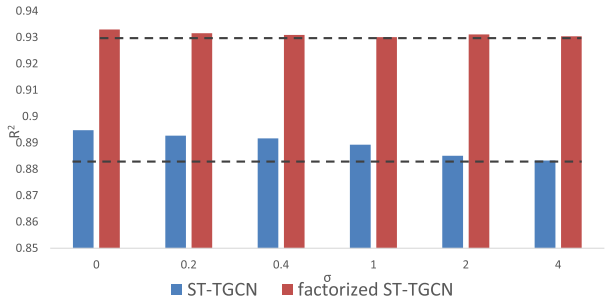

Fig. 5. Perturbation analysis on the Los-loop for prediction horizons of 15 minutes.

compression and can remove irrelevant components to improve performance.

3) Perturbation Analysis: The traffic data collected by sensors is inevitably noisy, therefore, the ability to resist noise perturbation reflects the effectiveness and robustness of the model. Here, we conduct a perturbation analysis to validate the property of our proposed model. We perform perturbations to traffic speed data by adding random Gaussian noise [30]. The probability density function (PDF) of Gaussian distribution $N \in (0, \sigma^2)$ is formulated as

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2}), \tag{25}$$

where μ is the mean value and σ is the standard deviation. We compare factorized ST-TGCN with ST-TGCN to validate the ability to resist noise perturbation. We set $\sigma = \{0, 0.2, 0.4, 1, 2, 4\}$, where $\sigma = 0$ means there is no Gaussian noise added to traffic data.

As shown in Fig. 5, ST-TGCN falls much more than factorized ST-TGCN in the metric of R^2. Therefore, we conclude that i) ST-TGCN introduces the idea of multi-graph to deal with traffic prediction, so ST-TGCN is capable of resisting perturbation to some extent. However, ST-TGCN fails to maintain robustness when σ increases. ii) Factorized ST-TGCN outperforms ST-TGCN and shows great robustness with perturbation growing.

4) Stability Analysis: In literature [41], the convergence of HOOI algorithm has been proved, therefore, the solution of tucker decomposition can be guaranteed. To further analyze the stability of our proposed deep learning model, we run the proposed model for ten runs and visualize the training loss curve on the dataset SZ-taxi for prediction horizons

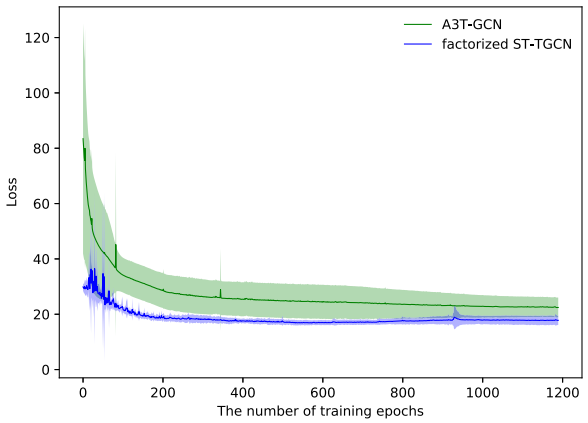

Fig. 6. The loss curves with different number of epochs. We removed the first ten epochs, whose loss error fluctuates too much as an adaptation stage and was not informative.

TABLE VIII
PERFORMANCE OF DIFFERENT NUMBERS COMPONENTS

Dataset	n	$RMSE$	MAE	$Accuracy$	R^2	var
SZ-taxi	$n = N$	3.8980	2.5828	0.7285	0.8606	0.8606
	$n = N^{\frac{1}{2}}$	**3.1080**	**2.0198**	**0.7835**	**0.9114**	**0.9114**
	$n = N^{\frac{1}{3}}$	3.6997	2.4670	0.7423	0.8744	0.8746
Los-loop	$n = N$	4.9783	2.9430	0.9151	0.8716	0.8758
	$n = N^{\frac{1}{2}}$	**3.5969**	**2.2265**	**0.9387**	**0.9330**	**0.9330**
	$n = N^{\frac{1}{3}}$	3.7119	2.2918	0.9367	0.9286	0.9287

of 15 minutes. In Fig. 6, the darker line and the shaded bars (i.e. light color fill area) represent the mean and the standard deviation of multiple runs, respectively. Compared to the existing state-of-the-art method A3T-GCN, our proposed method further decreases the training loss error, moreover, it can be observed that the standard deviation of the proposed method is smaller, so the proposed method can achieve more stable optimization.

5) The Components Number of Factorized ST-TGCN: In the Tucker decomposition, choosing how many components to use for describing the original tensor is related to noise compression and computational complexity. Based on three criteria proposed by Kiers and Mechelen [17], for our tensor data $\mathcal{X} \in \mathbb{R}^{N \times D \times T}$, we have i) N denotes the number of nodes, the number of N tends to be larger, which is Tucker decomposition focus on. ii) T and D are the constant value that is not affected by the scale of datasets and T tends to be small. As Table VIII shows, we compare the performance of factorized ST-TGCN with a varying number of components in mode N, where core tensor $\mathcal{C} \in \mathbb{R}^{n \times d \times t}$. Considering d and t can be omitted, we vary them with n. When n equals $N^{\frac{1}{2}}$, t equals $T^{\frac{1}{2}}$ and d equals $D^{\frac{1}{2}}$. Specifically, we have component matrices $\mathbf{X}_S \in \mathbb{R}^{N \times n}$, $\mathbf{X}_F \in \mathbb{R}^{D \times d}$, and $\mathbf{X}_T \in \mathbb{R}^{T \times t}$. In Table VIII, factorized ST-TGCN shows the best performance when n is equal to $N^{\frac{1}{2}}$. Moreover, we compare computational time for one epoch and memory space of the training process in Table IX, results suggest that an appropriate number of

TABLE IX
COMPARISONS ON THE NUMBER OF COMPONENTS

Dataset	n	Time(s)	Memory Space
SZ-taxi	$n = N$	31.49	NDT
	$n = N^{\frac{1}{2}}$	0.35	$N^{\frac{3}{2}} + T^{\frac{3}{2}} + D^{\frac{3}{2}} + (NTD)^{\frac{1}{2}}$
	$n = N^{\frac{1}{3}}$	**0.25**	$\boldsymbol{N^{\frac{4}{3}} + T^{\frac{4}{3}} + D^{\frac{4}{3}} + (NTD)^{\frac{1}{3}}}$
Los-loop	$n = N$	43.80	NDT
	$n = N^{\frac{1}{2}}$	0.65	$N^{\frac{3}{2}} + T^{\frac{3}{2}} + D^{\frac{3}{2}} + (NTD)^{\frac{1}{2}}$
	$n = N^{\frac{1}{3}}$	**0.54**	$\boldsymbol{N^{\frac{4}{3}} + T^{\frac{4}{3}} + D^{\frac{4}{3}} + (NTD)^{\frac{1}{3}}}$

components is capable of improving performance as well as reducing computational time and memory space.

VIII. CONCLUSION

In this paper, we proposed a factorized spatial-temporal tensor graph convolutional network (factorized ST-TGCN) which originates from high-dimensional tensor data for traffic speed prediction. A basic multi-scale spatial-temporal tensor graph convolution (TG-convolution) was first constructed to model spatial-temporal dependencies simultaneously. Based on this, we further introduced Tucker decomposition into tensor convolution and accordingly derived the factorized TG-convolution with effectiveness in terms of computational cost, memory requirements, and noise suppression. Experimental results on three real-world traffic datasets showed that the factorized ST-TGCN outperforms existing state-of-the-art methods.

REFERENCES

[1] J. Bai et al., "A3T-GCN: Attention temporal graph convolutional network for traffic forecasting," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 7, p. 485, 2021.

[2] T. Bogaerts, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data," *Transp. Res. C, Emerg. Technol.*, vol. 112, pp. 62–77, Mar. 2020.

[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.

[4] F. Chen, Z. Chen, S. Biswas, S. Lei, N. Ramakrishnan, and C.-T. Lu, "Graph convolutional networks with Kalman filtering for traffic prediction," in *Proc. 28th Int. Conf. Adv. Geograph. Inf. Syst.*, Nov. 2020, pp. 135–138.

[5] X.-Y. Chen, H.-K. Pao, and Y.-J. Lee, "Efficient traffic speed forecasting based on massive heterogenous historical data," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2014, pp. 10–17.

[6] F. R. Chung and F. C. Graham, *Spectral Graph Theory*, vol. 92. Providence, RI, USA: AMS, 1997, p. 207.

[7] P. Comon, X. Luciani, and A. L. F. de Almeida, "Tensor decompositions, alternating least squares and other tales," *J. Chemometrics*, vol. 23, nos. 7–8, pp. 393–405, Jul. 2009.

[8] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "On the best rank-1 and rank-(R_1, R_2, ..., R_N) approximation of higher-order tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1324–1342, 2000.

[9] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[10] L. Eldén and B. Savas, "A Newton–Grassmann method for computing the best multilinear rank-(r_1, r_2, r_3) approximation of a tensor," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 2, pp. 248–271, 2009.

[11] X. Fang, J. Huang, F. Wang, L. Zeng, H. Liang, and H. Wang, "ConSTGAT: Contextual spatial–temporal graph attention network for travel time estimation at Baidu maps," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2697–2705.

[12] H. Fu, H. Ma, Y. Liu, and D. Lu, "A vehicle classification system based on hierarchical multi-SVMs in crowded traffic scenes," *Neurocomputing*, vol. 211, pp. 182–190, Oct. 2016.

[13] G. Gopi et al., "Bayesian support vector regression for traffic speed prediction with error bars," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 136–141.

[14] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial–temporal graph convolutional networks for traffic flow forecasting," in *Proc. AAAI*, vol. 33, 2019, pp. 922–929.

[15] U. Johansson, H. Boström, T. Löfström, and H. Linusson, "Regression conformal prediction with random forests," *Mach. Learn.*, vol. 97, nos. 1–2, pp. 155–176, 2014.

[16] H. A. L. Kiers, "Towards a standardized notation and terminology in multiway analysis," *J. Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.

[17] H. A. L. Kiers and I. V. Mechelen, "Three-way component analysis: Principles and illustrative application," *Psychol. Methods*, vol. 6, no. 1, pp. 84–110, 2001.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

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

[20] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2017, *arXiv:1707.01926*.

[21] Z. Lv, J. Xu, K. Zheng, H. Yin, P. Zhao, and X. Zhou, "LC-RNN: A deep learning model for traffic speed prediction," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 3470–3476.

[22] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[23] M. May, D. Hecker, C. Körner, S. Scheider, and D. Schulz, "A vector-geometry based spatial kNN-algorithm for traffic frequency predictions," in *Proc. IEEE Int. Conf. Data Mining Workshops*, Dec. 2008, pp. 442–447.

[24] G. H. Nguyen, J. B. Lee, R. A. Rossi, N. K. Ahmed, E. Koh, and S. Kim, "Continuous-time dynamic network embeddings," in *Proc. Companion Web Conf.*, 2018, pp. 969–976.

[25] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2014–2023.

[26] P. Olszewski, H. S. L. Fan, and Y.-W. Tan, "Area-wide traffic speed-flow model for the Singapore CBD," *Transp. Res. A, Policy Pract.*, vol. 29, no. 4, pp. 273–281, Jul. 1995.

[27] A. Pareja et al., "EvolveGCN: Evolving graph convolutional networks for dynamic graphs," in *Proc. AAAI*, 2020, pp. 5363–5370.

[28] C. Park et al., "STGRAT: A spatio–temporal graph attention network for traffic forecasting," 2019, *arXiv:1911.13181*.

[29] Y. Qi and S. Ishak, "A hidden Markov model for short term prediction of traffic conditions on freeways," *Transp. Res. C, Emerg. Technol.*, vol. 43, pp. 95–111, Jun. 2014.

[30] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Berlin, Germany: Springer, 2003, pp. 63–71.

[31] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2018, pp. 362–373.

[32] D. I. Shuman, M. J. Faraji, and P. Vandergheynst, "A multiscale pyramid transform for graph signals," *IEEE Trans. Signal Process.*, vol. 64, no. 8, pp. 2119–2134, Apr. 2016.

[33] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017.

[34] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[35] L. R. Tucker, "A method for synthesis of factor analysis studies," U.S. Dept. Army, Washington, DC, USA, Personnel Res. Sect. Rep. 984, 1951.

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

[37] L. Tucker, N. Frederiksen, and H. Gulliksen, *Contributions to Mathematical Psychology*. New York, NY, USA: Holt, Rinehardt & Winston, 1964, pp. 109–127.

[38] H. Wang, L. Liu, Z. Qian, H. Wei, and S. Dong, "Empirical mode decomposition–autoregressive integrated moving average: Hybrid short-term traffic speed prediction model," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2460, no. 1, pp. 66–76, Jan. 2014.
[39] Y. Wu and H. Tan, "Short-term traffic flow forecasting with spatial–temporal correlation in a hybrid deep learning framework," 2016, *arXiv:1612.01022*.
[40] Y. Xie, K. Zhao, Y. Sun, and D. Chen, "Gaussian processes for short-term traffic volume forecasting," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 2165, no. 1, pp. 69–78, Jan. 2010.
[41] Y. Xu, "On the convergence of higher-order orthogonal iteration," *Linear Multilinear Algebra*, vol. 66, no. 11, pp. 2247–2265, Nov. 2018.
[42] J. Ye, J. Zhao, K. Ye, and C. Xu, "How to build a graph-based deep learning architecture in traffic domain: A survey," 2020, *arXiv:2005.11691*.
[43] B. Yu, H. Yin, and Z. Zhu, "Spatio–temporal graph convolutional networks: A deep learning framework for traffic forecasting," 2017, *arXiv:1709.04875*.
[44] L. Zhao et al., "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2019.
[45] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," in *Modeling Financial Time Series With S-PLUS*. New York, NY, USA: Springer, 2006, pp. 385–429 and 1020.

Xuran Xu received the B.S. degree in software engineering from the Jiangsu University of Science and Technology, Zhenjiang, China, in 2019. She is currently pursuing the M.S. degree with the School of Computer Science and Engineering, Nanjing University of Technology and Science, Nanjing, China. Her current research interests include graphical model, tensor graph convolutional networks, intelligent transportation systems, and deep learning.

Tong Zhang received the B.S. degree in information science and technology from Southeast University, Nanjing, China, in 2011, the M.S. degree from the Research Center for Learning Science, Southeast University, in 2014, and the Ph.D. degree from the School of Information Science and Engineering, Southeast University, in 2018. He is currently working with the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include pattern recognition, affective computing, and computer vision.

Chunyan Xu received the Ph.D. degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, in 2015. From 2013 to 2015, she was a Visiting Scholar with the Department of Electrical and Computer Engineering, National University of Singapore. She currently works with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. Her research interests include computer vision, manifold learning, and deep learning.

Zhen Cui (Member, IEEE) received the B.S. degree from Shandong Normal University in 2004, the M.S. degree from Sun Yat-sen University in 2006, and the Ph.D. degree from the Institute of Computing Technology (ICT), Chinese Academy of Sciences, in 2014. He was a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), from 2014 to 2015. He also spent half a year as a Research Assistant with Nanyang Technological University (NTU) from June 2012 to December 2012. He is currently a Professor with the Nanjing University of Science and Technology, China. His research interests mainly include deep learning, computer vision, and pattern recognition.

Jian Yang received the Ph.D. degree from the Nanjing University of Science and Technology (NUST) on the subject of pattern recognition and intelligence systems in 2002. In 2003, he was a Post-Doctoral Researcher at the University of Zaragoza. From 2004 to 2006, he was a Post-Doctoral Fellow at the Biometrics Centre, Hong Kong Polytechnic University. From 2006 to 2007, he was a Post-Doctoral Fellow at the Department of Computer Science, New Jersey Institute of Technology. He is currently a Chang-Jiang Professor with the School of Computer Science and Engineering, NUST. He is the author of more than 100 scientific papers in pattern recognition and computer vision. His journal articles have been cited more than 4000 times in the ISI Web of Science, and 9000 times in the Web of Scholar Google. His research interests include pattern recognition, computer vision, and machine learning. Currently, he is/was an Associate Editor of *Pattern Recognition Letters*, IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and *Neurocomputing*. He is a fellow of IAPR.