applied sciences

MDPI

Article

Time-Evolving Graph Convolutional Recurrent Network for Traffic Prediction

Weimin Mai, Junxin Chen and Xiang Chen *

School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou 510006, China; maiwm3@mail2.sysu.edu.cn (W.M.); chenjx248@mail2.sysu.edu.cn (J.C.)
* Correspondence: chenxiang@mail.sysu.edu.cn

Abstract: Accurate traffic prediction is crucial to the construction of intelligent transportation systems. This task remains challenging because of the complicated and dynamic spatiotemporal dependency in traffic networks. While various graph-based spatiotemporal networks have been proposed for traffic prediction, most of them rely on predefined graphs from different views or static adaptive matrices. Some implicit dynamics of inter-node dependency may be neglected, which limits the performance of prediction. To address this problem and make more accurate predictions, we propose a traffic prediction model named Time-Evolving Graph Convolution Recurrent Network (TEGCRN), which takes advantage of time-evolving graph convolution to capture the dynamic inter-node dependency adaptively at different time slots. Specifically, we first propose a tensor-composing method to generate adaptive time-evolving adjacency graphs. Based on these time-evolving graphs and a predefined distance-based graph, a graph convolution module with mix-hop operation is applied to extract comprehensive inter-node information. Then the resulting graph convolution module is integrated into the Recurrent Neural Network structure to form an general predicting model. Experiments on two real-world traffic datasets demonstrate the superiority of TEGCRN over multiple competitive baseline models, especially in short-term prediction, which also verifies the effectiveness of time-evolving graph convolution in capturing more comprehensive inter-node dependency.

Keywords: traffic prediction; spatiotemporal network; graph convolutional network; time-evolving graphs; graph generation

Citation: Mai, W.; Chen, J.; Chen, X. Time-Evolving Graph Convolutional Recurrent Network for Traffic Prediction. *Appl. Sci.* **2022**, *12*, 2842. https://doi.org/10.3390/app12062842

Academic Editors: Xinqiang Chen, Zhuang Dai and Xiaolei Ma

Received: 30 January 2022
Accepted: 8 March 2022
Published: 10 March 2022

1. Introduction

In recent years, the rapidly growing number of vehicles in urban transportation systems has raised huge challenges to society. Traffic congestion in the road network increases the commuting time of residents. In addition, it brings additional loss to the freight supply chain, which reduces the efficiency of economic activities and produces more pollution [1]. To cope with these problems, constructing an efficient and convenient transportation system is of great necessity. Traffic prediction, which aims to predict future traffic states (e.g., traffic speed, traffic volume and traffic congestion index) in the road networks based on historical observations, acts a crucial part in the intelligent transportation system. Accurate prediction of traffic is the foundation of a wide range of real-world applications, such as travel time estimation, route planning, traffic light management, autonomous vehicle control and logistics distribution [2].

However, the tasks of traffic prediction remain challenging due to the complicated spatial and temporal dependency in the road networks. On the one hand, there are complex non-linear patterns in the time domain. The traffic conditions of a traffic node may follow a similar periodic trend over different days, but may see anomalous fluctuation in a short period. For example, as shown in Figure 1, the traffic speed of a main road segment decreases during rush hours in the morning and evening, and increases back to a normal level gradually. However, as indicated by the red dotted box, there is a

dramatic drop in the afternoon of a day, which increases the difficulty of making accurate prediction. On the other hand, traffic shows complicated spatial correlations. For example, as shown in Figure 2, a traffic node usually has similar conditions to its neighboring nodes. The traffic speed of a road segment may become slower because of the congestion at its downstream road segment. Moreover, the traffic nodes that are not neighboring but with similar road topology or in similar functional areas tend to share similar patterns. Making accurate predictions requires comprehensive modeling of inter-node dependency. Therefore, with the spatiotemporal relations mentioned above, the tasks of traffic prediction are still challenging.

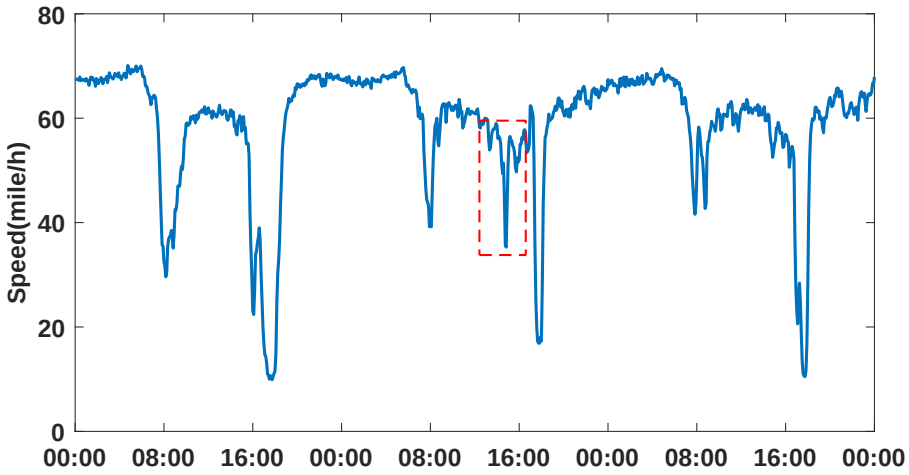

Figure 1. Traffic speed of a traffic node on three consecutive days. The value of speed follows a similar periodic trend on these days, but may see anomalous fluctuation in the short period indicated by the red dotted box.

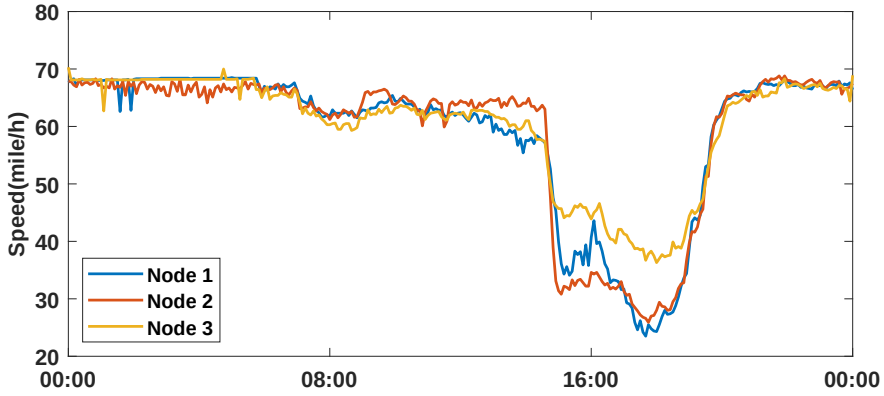

Figure 2. Traffic speed of three neighboring traffic nodes on a day. The values of these three nodes show strong correlations.

Extensive studies have been conducted on the tasks of traffic prediction. Early studies [3–5] employ statistical methods or shallow machine-learning methods, which are limited in representation capability and thus have lower performance. With the emergence of deep learning, methods based on Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) are applied in traffic prediction to extract more complex spatial and temporal features [6–10]. However, since CNNs can only process spatial traffic dependency in a grid-based manner, Graph Convolutional Networks (GCN) [11] are introduced to handle traffic in irregular road networks with non-Euclidean structures. This kind of method aggregates information from selected neighbors according to a certain graph structure. More recently, integrating GCNs with other temporal models has become a paradigm for traffic-related tasks, which is followed by a number of studies [12–17].

Although existing approaches have achieved promising results, we find that the dynamics of spatial dependency have been rarely considered. Most of the existing graph-based works utilize one or more adjacency matrices to represent the connections among traffic nodes according to predefined similarity relations from different perspectives, such as road network distance [12,13], time-series similarity [18], edge-wise line graph [19] or point of interest (POI) similarity [20]. Some works also use adaptive adjacency matrices with trainable parameters [14,21]. However, these inter-node relations are all static, which means that the spatial adjacency does not change along with time. Therefore, some implicit dynamic patterns of spatial adjacency may be neglected, limiting the performance of the spatiotemporal models.

In view of the limitations mentioned above, we aim to introduce dynamic representations into the adjacency graphs to achieve better prediction performance. In this paper, we propose a general traffic prediction framework named Time-Evolving Graph Convolutional Recurrent Network (TEGCRN), which takes advantage of time-evolving graph convolution to capture the dynamic inter-node dependency adaptively at different time slots. The contributions of our method can be summarized as follows:

- We construct time-evolving adjacency graphs at different time slots with self-adaptive time embeddings and node embeddings based on a tensor-composing method. This method makes full use of information shared in the time domain and is parameter-efficient compared to defining an adaptive graph in each time slot.
- To model the inter-node patterns in traffic networks, we apply a kind of mix-hop graph convolution that utilizes both the adaptive time-evolving graphs and a predefined distance-based graph. This kind of graph convolution module is verified as being effective in capturing more comprehensive inter-node dependency than those with static graphs.
- We integrate the aforementioned graph convolution module with RNN encoder–decoder structure to form a general traffic prediction framework, which allows it to learn the dynamics of inter-node dependency when modeling traffic sequential features. Experiments on two real-world traffic datasets demonstrate the superiority of the proposed model over multiple competitive baselines, especially in short-term prediction.

The rest of the paper is organized as follows. In Section 2, we review the related works systematically and explain their relations with our work. Then we formulate the problem and introduce the details of the proposed traffic prediction model in Section 3. After that, we conduct experiments on real-world traffic datasets to evaluate the performance of the proposed method in Section 4. In Section 5 we conclude this paper.

2. Related Work

2.1. Traffic Prediction Methods

The methods of traffic prediction have been extensively researched. Traditional methods view this task as a univariant time-series regression and forecasting problem, using statistical signal processing models or classic machine-learning models, such as Auto-Regressive Integrated Moving Average (ARIMA) [3,4], Kalman filtering [22,23], Vector Auto-Regression (VAR) [24] and Support Vector Regression (SVR) [5]. However, these methods heavily rely on the stationary assumption, so they have limited representation space and can hardly capture the dynamics of real traffic. Meanwhile, they treat the sequences of different traffic nodes separately and therefore are incapable of utilizing the information of spatial dependency.

With the emergence of deep-learning methods, some researchers employ CNN-based and RNN-based models in the tasks of traffic prediction. Zhang et al. [8] proposed a CNN-based residual network to model the citywide spatial correlations and predict the region crowd traffic. Wu et al. [7] build a feature-level regression model with spatial and temporal features captured by a 1-D CNN and two LSTMs, respectively, in different time

scales. These works show the advantages of deep-learning methods in modeling nonlinear spatiotemporal relationships.

More recently, to tackle traffic data in irregular road topologies with non-Euclidean structures, graph neural networks have been widely used. Integrating graph networks such as GCN with other temporal models to construct spatiotemporal graph networks has become a new paradigm. We compare some typical works with our model and list the key components in Table 1 to show the gaps. Li et al. [12] model the spatial dependency of traffic with a diffusion process on the road distanced graph and replace the linear layer in GRU with diffusion convolution operation. Yu et al. [13] propose the STGCN model, which stacks temporal 1-D CNN with ChebyNet [25] alternately and conducts prediction in a complete convolutional manner. Based on [13], Guo et al. [15] also add spatial and temporal attention mechanisms to refine the correlations. Considering that the aforementioned works limit the inter-node adjacency to predefined distance graphs, some works attempt to extend the inter-node dependency to more views. Song et al. [16] design an augmented graph with adjacency along both the spatial and temporal dimension, and capture these two types of features with graph convolution synchronously. Following this idea, Li et al. [18] further add the time-series similarity subgraphs into the spatiotemporal augmented graph. Chen et al. [19] propose an attentive bicomponent GCN based on both the road distance graph and its corresponding edge-wise line graph. Wu et al. [14,21] further equip graph convolution with static adaptive adjacency graphs and apply dilated convolution in the temporal dimension.

However, in the works mentioned above, the adjacency graphs of traffic nodes are all static, and the graph convolutions are used to capture information in a fixed topology. In this way, some implicit dynamics of the inter-node dependency may be neglected. In contrast, our model takes advantage of evolving adjacency graphs at different time slots, and learns the dynamic spatial dependencies together with sequential signal features.

Table 1. The comparison table of graph-based traffic prediction models.

Models	Graph Construction for Inter-Node Dependency	Inter-Node Dependency Evolves with Time?	Spatial Modeling	Temporal Modeling
DCRNN [12]	road distance graph	✗	diffusion convolution [1]	GRU
STGCN [13]	road distance graph	✗	ChebyNet	CNN
ASTGCN [15]	road binary graph	✗	ChebyNet + attention	CNN + attention
Graph-WaveNet [14]	road distance graph + adaptive graph	✗	mix-hop GCN	dilated CNN
STSGCN [16]	spatio and temporal augmented graph	✗	JK-Net	– [2]
STFGNN [18]	spatio and temporal and time-series similarity augmented graph	✗	JK-Net	–
MRA-BGCN [19]	road distance graph + edge-wise line graph	✗	GCN + attention	GRU
MTGNN [21]	adaptive graph	✗	mix-hop GCN + residual	dilated CNN + inception
TEGCRN (ours)	road distance graph + adaptive graph	✓	mix-hop GCN + residual	GRU

[1] Diffusion convolution [12], ChebyNet [25], GCN [11], JK-Net [26] are different variants of graph neural/convolutional networks. [2] "–" in "Temporal Modeling" means the model has no separate temporal module and process the temporal information along with the spatial module in "Spatial Modeling".

2.2. Graph Convolutional Networks

Graph convolutional networks have received extensive attention in recent years because of their capability for handling graph-structure data. Bruna et al. [27] generalize the Fourier transform to graph signals and first apply trainable graph convolutions in the spectral domain. Defferrard et al. [25] propose ChebyNet, which restricts the kernels to Chebyshev polynomials and localizes the convolution operations. Based on this, Kipf and Welling [11] further employ first-order approximation and propose the most popular form of GCN, which simply conducts weighted aggregation directly in the neighborhood defined by the graph. Veličković et al. [28] introduce an attention mechanism to learn the weights between nodes. Li et al. [12] propose bidirectional diffusion convolution, which can be regarded as the extended version of the model in [25].

It is demonstrated by some works that GCNs act as low-frequency filters and smooth the signals on the graphs [29,30]. Too many layers and non-linear operations tend to lead to the over-smoothing phenomenon and degrade performance [31]. Since spatiotemporal networks are usually stacked by lots of layers or blocks, it is quite common that the inner GCNs are shallow and the mix-hop operations are applied to aggregate information in multi-scale receptive fields [21,26].

3. Proposed Method

3.1. Problem Preliminaries

A traffic network can be regarded as a weighted directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A})$. Here, $\mathcal{V}$ is the set of N traffic nodes (i.e., sensors or road segments), $\mathcal{E}$ is the set of edges (i.e., directed connections between traffic nodes), $\mathbf{A} \in \mathbb{R}^{N \times N}$ is a certain type of weighted adjacency matrix representing the proximity between each nodes pair, e.g., road distance graph.

The traffic features observed at time slot t are represented as the graph signals $\mathbf{X}_t \in \mathbb{R}^{N \times F}$, where F is the number of traffic features of each node. For example, the traffic features can be the traffic speed, traffic volume and congestion index of the road segments or other traffic related parameters. Given a traffic graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A})$, the purpose of traffic prediction is learning a model f to map the historical graph signals observations of P time steps to the signals of the next Q time steps, which is denoted as:

$$[\mathbf{X}_{t-P+1}, \cdots, \mathbf{X}_t] \xrightarrow{f} [\mathbf{X}_{t+1}, \cdots, \mathbf{X}_{t+Q}]. \tag{1}$$

3.2. Method Overview

Figure 3 illustrates the overview of our proposed method, which is referred to as Time-Evolving Graph Convolutional Recurrent Network (TEGCRN). The model mainly consists of three parts. First, the time-evolving adjacency graphs are generated through a tensor-composing method using the adaptive embeddings of traffic nodes and time slots. Then, the resulting adaptive graphs are combined with predefined static distance-based graphs and employed in graph convolution module to capture the inter-node information. Finally, the graph convolution modules with time-evolving graphs are used to replace the fully-connected layers in GRU to form an integrated RNN encoder–decoder predicting model. In each time step of the encoder–decoder, a different adaptive graph is selected depending on the time of day. The key thought of this method is to learn the implicit dynamics of spatial adjacency together with the traffic sequential features so as to capture more detailed spatiotemporal information. In the following subsections, we will describe each part in detail.

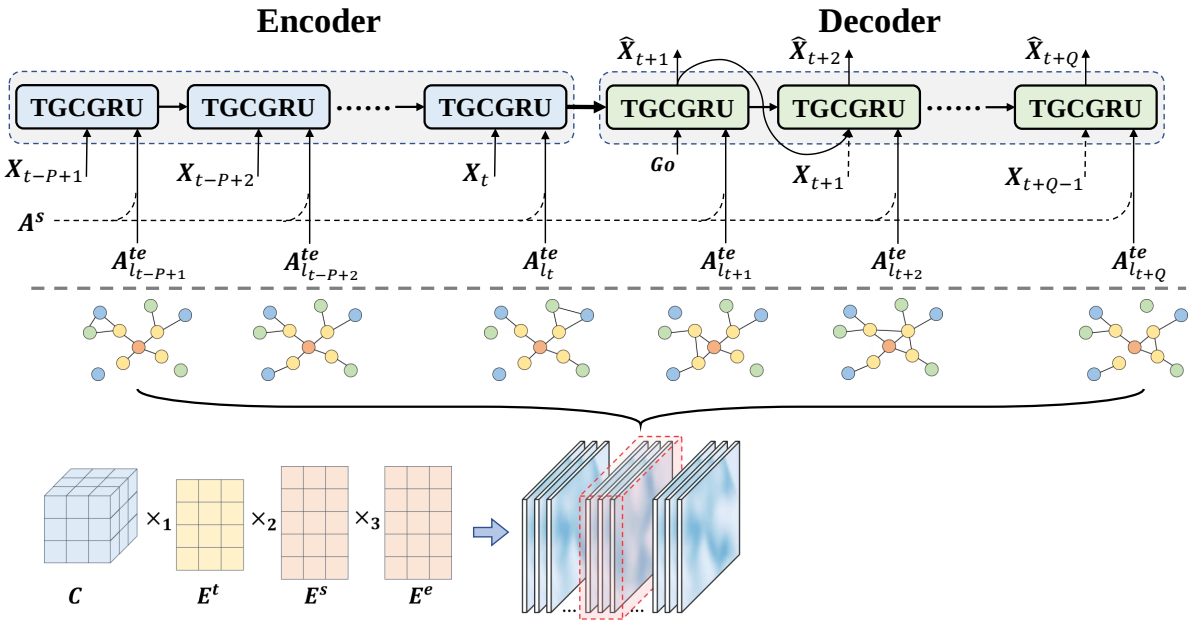

Figure 3. Overview of the proposed model TEGCRN.

3.3. Generation of Time-Evolving Adaptive Graphs

As mentioned above, the spatial relationships among traffic nodes vary along with time, so our aim is to model these dynamics adaptively using time-evolving adjacency graphs. Moreover, we assume that the dynamics of the spatial adjacency also follow the periodicity in a day. Thus, the same time slot on different days can share a graph. Previous works [14,21] adopt matrix multiplication of trainable node embeddings to construct static adjacency graphs. The obtained adaptive graphs are evaluated to be effective in capturing global spatial information. To extend the adaptive graph to a time-evolving manner, a simple idea is to directly assign a distinct set of node-embedding parameters to each time slot. However, such a method may lead to a large number of trainable parameters, making it hard to converge, especially when the number of traffic nodes N is large. To reduce the number of model parameters, we employ a tensor-composing method similar to [32] to generate the time-evolving adjacency graphs.

Suppose that a day is divided into N_t time slots, and the traffic network contains N traffic nodes. We first construct three embedding matrices $\mathbf{E^t} \in \mathbb{R}^{N_t \times d}$, $\mathbf{E^s} \in \mathbb{R}^{N \times d}$, $\mathbf{E^e} \in \mathbb{R}^{N \times d}$ and a core tensor $\mathbf{C} \in \mathbb{R}^{d \times d \times d}$, where d is the embedding dimension, $d \ll N$. Specifically, $\mathbf{E^t}$ represents the embedding matrix of different time slots in a day, $\mathbf{E^s}$ represents the embedding matrix of source nodes, $\mathbf{E^e}$ represents the embedding matrix of target nodes. Tensor $\mathbf{C}$ builds connections among these embeddings and models the implicit factors shared across time and space. All these embeddings and the core tensor are trainable and randomly initialized. Next, a spatiotemoral tensor $\mathbf{A}' \in \mathbb{R}^{N_t \times N \times N}$ is calculated as the following formula:

$$\mathbf{A}' = \mathbf{C} \times_1 \mathbf{E^t} \times_2 \mathbf{E^s} \times_3 \mathbf{E^e}, \tag{2}$$

where $\times_i$ denotes the tensor mode-i product [33]. Specifically, the elements in $\mathbf{A}'$ can be denoted as

$$\mathbf{A}'_{i,j,k} = \sum_{w=1}^{d} \sum_{v=1}^{d} \sum_{u=1}^{d} \mathbf{C}_{u,v,w} \mathbf{E^t}_{i,u} \mathbf{E^s}_{j,v} \mathbf{E^e}_{k,w}. \tag{3}$$

Then, non-linear transformation and normalization are applied to $\mathbf{A}'$ to obtain the tensor of time-evolving graphs $\mathbf{A^{te}} \in \mathbb{R}^{N_t \times N \times N}$:

$$\mathbf{A^{te}} = softmax\big(LeakyReLU(\mathbf{A}')\big), \tag{4}$$

where the softmax normalization is operated in the last dimension. The l-th slice of $\mathbf{A^{te}}$ along its first dimension $\mathbf{A}_l^{\mathbf{te}}$ is the adaptive spatial adjacency graph of the l-th time slot in a day.

With the approach above, we build connections between the representations of traffic nodes and the representations of time slots through the calculations of tensor composing. The resulting adaptive time-evolving graphs actually denote how much importance should be attached to different node pairs in different time slot of a day when aggregating spatial information in the road network. Later, these graphs will be used in the graph convolution module to extract detailed spatial information.

3.4. Graph Convolution Module

A graph convolution module is used to capture spatial patterns by aggregating information based on the adjacency graphs. In our proposed model, we conduct graph convolution with both the aforementioned adaptive time-evolving graphs and predefined distance-based static graph. In this way, the static graph captures local patterns, and the time-evolving graphs provide dynamic inter-node dependency from a global view as a complement.

Following [12], we build a distance-based static graph $\mathbf{A^s}$ using thresholded Gaussian kernel [34] as:

$$\mathbf{A}^{\mathbf{s}}_{i,j} = \begin{cases} \exp\left(-\frac{d^2_{v_i,v_j}}{\sigma^2}\right), & d_{v_i,v_j} \leq \kappa \\ 0, & others \end{cases}, \tag{5}$$

where d_{v_i,v_j} denotes the shortest directed distance from node v_i to node v_j in the road network, σ denotes the standard deviation of these distances, and κ is the threshold to control the sparsity of the graph.

As mentioned in the Section 3.2, in our proposed model the graph convolution operations are integrated with an RNN structure. The recurrent manner in RNN inherently expands the depth of the model. In fact, GCNs may suffer from performance degradation due to the over-smoothing problem when the layers go deeper [31]. It is also argued that the entanglement of information propagation and feature transformation in GCN may limit the model performance [35]. By decoupling feature transformation with propagation and adding residual of input representation [36], or combining the representations of multiple layers [26], these problems can be relieved to some extent. Motivated by these ideas, we employ graph convolution operations in two steps: information propagation and weighted mix-hop operation, similar to the method in [21]. First, for a certain type of adjacency graph, the node information is propagated according to the graph topology and the node representations of K hops are generated. Denote the origin node representation as $\mathbf{H_{in}} \in \mathbb{R}^{N \times d_{in}}$, and the representation of the k-th hop $\mathbf{H}^{(k)}$ is calculated as:

$$\mathbf{H}^{(k)} = (1-\alpha)\widetilde{\mathbf{A}}\mathbf{H}^{(k-1)} + \alpha\mathbf{H_{in}}, \tag{6}$$

where α is a hyper-parameter to control the ratio of residual original information, $\widetilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$ is the normalized form of the adjacency graph. Then, weighted mix-hop operation is employed to obtain $\mathbf{H_{mix}} \in \mathbb{R}^{N \times d_{out}}$ using the propagated representations of all hops as per the following formula:

$$\mathbf{H_{mix}} = \sum_{k=0}^{K} \mathbf{H}^{(k)}\mathbf{W}^{(k)}, \tag{7}$$

where $\mathbf{H}^{(0)} = \mathbf{H_{in}}$, $\mathbf{W}^{(k)} \in \mathbb{R}^{d_{in} \times d_{out}}$ is the learnable weighted transformation matrix.

For a time slot t, the corresponding index in a day is l_t, graph convolution operations are applied using three graphs: the time-evolving graph in this time slot $\mathbf{A}_{l_t}^{\mathbf{te}}$, the distance-based static graph $\mathbf{A^s}$ and its transposed one $\mathbf{A^s}^T$. For $\mathbf{A^s}$ and $\mathbf{A^s}^T$, asymmetric normalization is applied with $\widetilde{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1}(\mathbf{A} + \mathbf{I}_N)$ and $\widetilde{\mathbf{D}}_{ii} = 1 + \sum_j \mathbf{A}_{ij}$. The time-evolving

graphs have been normalized during generation. We further aggregate the mix-hop representations of these three graphs to obtain the final output of the entire graph convolution module $\mathbf{H_{out}}$. For convenience, we briefly denote the calculations in this module as $\mathbf{H_{out}} = \Theta *_G (\mathbf{H_{in}}, \mathbf{A}_{l_t}^{\mathbf{te}})$, where $*_G$ denotes the graph convolution operation, Θ represents all the trainable parameters in a module. The static graph remains the same in all modules so we omit its notation in the formula.

3.5. Temporal Recurrent Module

RNN is devised as a recurrent structure which processes input signals and updates hidden states step-by-step. This kind of structure is suitable for combining with time-evolving graphs to model the dynamics of spatial dependency. As is shown in Figure 3, in each time step, a different adaptive graph is selected according to the time index in a day. Then, following [12,19], we replace the matrix multiplication in GRU with the graph convolution module introduced in the previous subsection. The structure of the resulting recurrent unit is shown in Figure 4, which is referred to as Time-evolving Graph Convolutional GRU (TGCGRU).

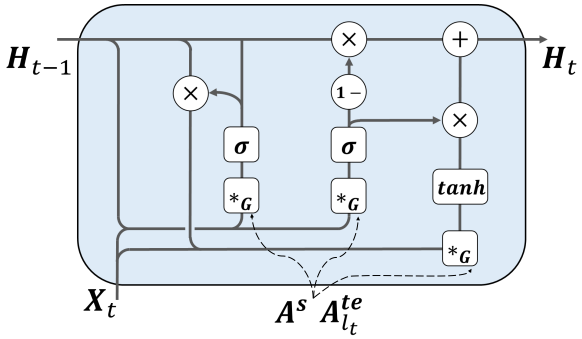

Figure 4. The structure of time-evolving graph convolutional gate recurrent unit.

For each time slot t, given the corresponding time-evolving graph $\mathbf{A}_{l_t}^{\mathbf{te}}$, the calculations in TGCGRU are fomulated as follows:

$$\begin{aligned}
\mathbf{r}_t &= \sigma\left(\Theta_r *_G \left(\mathbf{X}_t \| \mathbf{H}_{t-1}, \mathbf{A}_{l_t}^{\mathbf{te}}\right) + \mathbf{b}_r\right), \\
\mathbf{u}_t &= \sigma\left(\Theta_u *_G \left(\mathbf{X}_t \| \mathbf{H}_{t-1}, \mathbf{A}_{l_t}^{\mathbf{te}}\right) + \mathbf{b}_u\right), \\
\mathbf{C}_t &= \tanh\left(\Theta_c *_G \left(\mathbf{X}_t \| (\mathbf{r}_t \odot \mathbf{H}_{t-1}), \mathbf{A}_{l_t}^{\mathbf{te}}\right) + \mathbf{b}_c\right), \\
\mathbf{H}_t &= \mathbf{u}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{u}_t) \odot \mathbf{C}_t,
\end{aligned} \tag{8}$$

where $\mathbf{X}_t \in \mathbb{R}^{N \times F}$, $\mathbf{H}_t \in \mathbb{R}^{N \times h}$ denote the input traffic features and output hidden state at time step t, respectively, h is the hidden size of TGCGRU. The notation $\|$ denotes concatenation operation, $\sigma(\cdot)$ is the sigmoid function, and $\odot$ denotes the Hadamard product. $\mathbf{r}_t \in \mathbb{R}^{N \times h}$ and $\mathbf{u}_t \in \mathbb{R}^{N \times h}$ represent the reset gate and update gate, respectively. Θ_r, Θ_u, Θ_C are parameters for the corresponding graph convolution modules and $\mathbf{b}_r \in \mathbb{R}^N$, $\mathbf{b}_u \in \mathbb{R}^N$, $\mathbf{b}_C \in \mathbb{R}^N$ are trainable bias vectors. These parameters share all the time steps.

We stack multiple TGCGRUs to form an RNN sequence-to-sequence architecture [37] for multi-step traffic prediction, as is shown at the top of Figure 3. On the decoder side, an additional fully-connected layer is used to map the output hidden state of TGCGRU to the output features. The output features of all time steps in the decoder are the predictions we need. Both the embedding parameters for time-evolving graph generation and the parameters in TGCGRU are trained together in an end-to-end manner. To relieve the discrepancy in input distribution between the training and testing stage, the scheduled sampling [38] strategy is applied. During training, the decoder takes as input either the

ground truth with probability ε or the prediction of the previous step with probability $1-\varepsilon$. The probability of the i-th iteration ε_i is calculated as

$$\varepsilon_i = \frac{\tau}{\tau + \exp\left(\frac{i}{\tau}\right)}, \tag{9}$$

where τ is a constant to control the speed of probability decay.

3.6. Example of the Prediction Process of TEGCRN

In this section, we will explain the entire process of TEGCRN and take the prediction for traffic speed as an example. Given the historical speed observations of P steps $[\mathbf{X}_{t-P+1}, \cdots, \mathbf{X}_t]$, the prediction targets are the speeds of the next Q steps. In this case, $\mathbf{X}_t \in \mathbb{R}^{N \times F}$ represents the speed of all the N nodes at time slot t and $F = 1$. As shown in Figure 3, we first employ Equations (2) and (4) to generate tensor $\mathbf{A^{te}}$ and obtain the time-evolving graphs of the corresponding time slots $[\mathbf{A}^{\mathbf{te}}_{l_{t-P+1}}, \cdots, \mathbf{A}^{\mathbf{te}}_{l_t}]$ and $[\mathbf{A}^{\mathbf{te}}_{l_{t+1}}, \cdots, \mathbf{A}^{\mathbf{te}}_{l_{t+Q}}]$ for the encoder and decoder, respectively. Next, for each historical time slot, the speed snapshot $\mathbf{X}_t$ and the corresponding time-evolving graph $\mathbf{A}^{\mathbf{te}}_{l_t}$ are input to the TGCGRU encoder step-by-step, as the calculations show in Equation (8). The graph convolution operations in Equation (8) are formulated by Equations (6) and (7). The output in the last step of the encoder is used to initialize the hidden state of the decoder. Then, following the calculations similar to the encoder, in each step, the decoder takes the prediction of the previous step as input and generates the speed predictions $[\hat{\mathbf{X}}_{t+1}, \cdots, \hat{\mathbf{X}}_{t+Q}]$ step-by-step.

4. Experiments and Discussion

4.1. Datasets

We evaluate the proposed model TEGRCN on two public real-world traffic datasets, METR-LA and PEMS-BAY, which were released by Li et al. in [12]. METR-LA records the traffic speed data collected from 207 sensors on the highways of Los Angeles County, ranging from 1 March 2012 to 30 June 2012. PEMS-BAY records the speed data collected from 325 sensors in the bay area by California Transportation Agencies, ranging from 1 January 2017 to 30 June 2017. The units of the speed data are both miles per hour. Some basic statistics of these two datasets are listed in Table 2. Each sensor in the road network is viewed as a traffic node. The numbers of the edges are counted according to the distance-based static adjacency graphs. We can see that there is a relatively high proportion of missing values in METR-LA while the counterpart in PEMS-BAY is almost negligible. Following the data pre-processing procedures in [12], we set the interval between two time steps to 5 min. Then, each dataset is split in chronological order, with 70% for training, 10% for validation and 20% for testing. Z-score normalization is applied to the inputs.

In these two datasets, only the traffic speed in the road network is provided, so the number of traffic features is $F = 1$. Take METR-LA as an example, the graph of traffic network contains $N = 205$ nodes, so the traffic features (also called as graph signals) at a specific time slot t can be denoted as $\mathbf{X}_t \in \mathbb{R}^{205 \times 1}$. To make multiple-steps-ahead predictions with the historical observations, the series of these traffic features are input to the encoder–decoder of the proposed model step-by-step.

Table 2. The statistics of of datasets.

Dataset	Nodes	Edges	Size	Sample Rate	Missing Proportion
METR-LA	207	1515	34272	5 min	8.109%
PEMS-BAY	325	2369	52116	5 min	0.003%

4.2. Baseline Methods

To verify the predicting performance of TEGCRN, we compare it with typical time-series models and some representative graph-based spatiotemporal networks. The selected baselines are introduced as follows:

- HA : Historical Average, which views the traffic flow as seasonal signals and predicts future speed with the average of the values at the same time slots of previous weeks.
- ARIMA: Auto-Regressive Integrated Moving Average model with Kalman filter, a traditional time-series predicting method.
- FC-LSTM [37]: RNN sequence-to-sequence model with LSTM units, using a fully-connected layer when calculating the inner hidden vectors.
- DCRNN [12]: Diffusion Convolutional Recurrent Neural Network, which combines the bidirectional graph convolution with GRU encoder–decoder model. Only the distance-based graph is utilized and the adjacency in different time steps remains static.
- STGCN [13]: Spatiotemporal Graph Convolutional Network, which alternately stacks temporal 1-D CNN with spectral graph convolution ChebyNet to form a predicting framework in a complete convolutional manner. Likewise, it relies only on the distance-based static graph.
- Graph-WaveNet [14]: Based on the alternate convolutional architecture in [13], this model applies dilated convolution in the time dimension and incorporates an adaptive adjacency graph in the graph convolution module.
- MTGNN [21]: This model introduces a uni-directional adaptive graph and mix-hop propagation into graph convolutions, and introduces dilated inception layers on the time dimension to capture features in receptive fields of multiple sizes.

4.3. Experiments Settings and Metrics

As described in Section 3, the proposed model aims to make a multi-step prediction of future traffic states according to historical observations. Following the typical settings in the baseline methods using the same datasets, we set both the length of historical observations and future predictions to one hour, i.e., $P = Q = 12$, to make fair comparison. When generating the time-evolving graphs, the embedding dimension d is set to 30 for METR-LA and 40 for PEMS-BAY. Since the length of each time slot is 5 min, the number of time slots in a day $N_t = 288$. As for the graph convolution module, the maximum hop of K is set to 2 for both datasets and α is set to 0.01. The number of TGCGRU layers is set to two and the size of the hidden state in TGCGRU is set to 40 for METR-LA and 50 for PEMS-BAY. These hyper-parameters that are related to the proposed model are all selected via grid search. During the training process, the value τ for scheduled sampling is set to 2000. The model is trained by Adam optimizer and the learning rate is initialized to 0.01 with a decay rate of 0.1. Mean absolute error (MAE) is selected as the loss function. The batch size of input data is set to 64 for METR-LA and 32 for PEMS-BAY. Early stopping strategy is employed. For each dataset, the experiment is repeated five times to obtain the average metrics. To evaluate the performance of traffic prediction, three types of metrics are used, including mean absolute error (MAE), root-mean-squared error (RMSE) and mean absolute percentage error (MAPE).

4.4. Prediction Performance

The prediction results of our proposed method TEGCRN and the baseline methods on METR-LA and PEMS-BAY are shown in Tables 3 and 4, respectively. We report the results for 15 min (3 steps)-, 30 min (6 steps)-, 60 min (12 steps)-ahead predictions to show the performance in the short term, middle term and long term.

As shown in Tables 3 and 4, TEGCRN achieves higher performance on both datasets. It almost outperforms these baseline methods in all the time steps except for the long-term prediction on METR-LA. We can also see that the performances of GCN-based methods significantly outperform traditional series models such as ARIMA and FC-LSTM. This emphasizes the importance of introducing spatial information into traffic prediction. In

contrast to the models using only predefined distance-based graphs such as DCRNN and STGCN, Graph-WaveNet and MTGNN achieve better performance. This is because more inter-node dependency can be learned by the inner adaptive adjacency graph from a global view. Moreover, the inception layers in MTGNN may capture more temporal patterns in different time scales, which helps MTGNN surpass Graph-WaveNet in long-term prediction. By using time-evolving graph convolutions and integrating them with recurrent structure, our proposed model TEGCRN can capture more implicit dynamics of inter-node dependency, so it can achieve better performance compared with the models with static graphs. Since the recurrent structure in RNN usually suffers from accumulative errors when the number of output time steps increases, TEGCRN may be more sensitive to missing values in input data than the CNN-based models. However, the proportion of missing values of METR-LA is much higher than that of PEMS-BAY. This may lead to the small performance margin between TEGCRN and MTGNN when making long-term predictions on METR-LA. To sum up, our proposed model TEGCRN shows its superiority over multiple competitive traffic-predicting baseline models especially in short-term prediction.

Table 3. Prediction Results of METR-LA Dataset.

	15 min (3 Steps)			30 min (6 Steps)			60 min (12 Steps)		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
HA	4.16	7.80	13.00%	4.16	7.80	13.00%	4.16	7.80	13.00%
ARIMA	3.99	8.21	9.60%	5.15	10.45	12.70%	6.90	13.23	17.40%
FC-LSTM	3.44	6.30	9.60%	3.77	7.23	10.90%	4.37	8.69	13.20%
DCRNN	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
STGCN	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
Graph-WaveNet	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
MTGNN	2.69	5.18	6.86%	3.05	6.17	8.19%	3.49	7.23	9.87%
TEGCRN	2.65	5.10	6.74%	3.04	6.16	8.22%	3.49	7.32	10.03%

Table 4. Prediction Results of PEMS-BAY Dataset.

	15 min (3 Steps)			30 min (6 Steps)			60 min (12 Steps)		
	MAE	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
HA	2.88	5.59	6.80%	2.88	5.59	6.80%	2.88	5.59	6.80%
ARIMA	1.62	3.30	3.50%	2.33	4.76	5.4%	3.38	6.50	8.30%
FC-LSTM	2.05	4.19	4.80%	2.20	4.55	5.20%	2.37	4.96	5.70%
DCRNN	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
STGCN	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
Graph-WaveNet	1.30	2.74	2.73%	1.63	3.70	3.67%	1.95	4.52	4.63%
MTGNN	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	4.53%
TEGCRN	1.29	2.72	2.69%	1.60	3.67	3.59%	1.88	4.39	4.43%

4.5. Ablation Study

We conduct an ablation study on the METR-LA dataset to validate the effectiveness of the graph convolution module with adaptive time-evolving graphs. We first build a model that replaces the time-evolving graphs in TEGCRN with a static adaptive graph as the manner in [14]. Additionally, we further remove the time-evolving graph generation module and use only static predefined distance-based graph, as per the manner in [12]. For these models, we calculate the average metrics of all the output time steps. The results are reported in Table 5. We can see that, with learned global dependencies, the convolution with adaptive graph achieves higher performance than that with only predefined distance-based graph. Furthermore, the proposed time-evolving graph convolution in TEGCRN further improves the prediction performance by capturing more implicit inter-node dynamics. Figure 5 shows the heatmaps of the normalized distance-based static adjacency graph and the learned time-evolving adjacency graphs in some selected time slots. A darker color means a higher value of weight between the corresponding nodes. When the static graph

assigns high weights to the neighboring node pairs in a local manner, the time-evolving graphs can learn some other sparse connections among the nodes from a global view. In different time slots, the trained time-evolving graphs attach different importance to the traffic node pairs and form dynamic topologies. These learned implicit dynamics of inter-node dependency can act as the complement of the static distance-based graph. This helps the graph convolution to capture more urban semantics, which contributes to the better prediction performance. In sum, the proposed graph convolution with adaptive time-evolving graphs is effective to model more comprehensive inter-node dependencies than those with only static graphs.

Table 5. The results of ablation study on METR-LA.

Method	Overall MAE	Overall RMSE	Overall MAPE
TEGCRN	2.908	5.705	7.866%
Static Adpative	2.969	5.792	8.072%
Static Predefined	3.015	5.887	8.258%

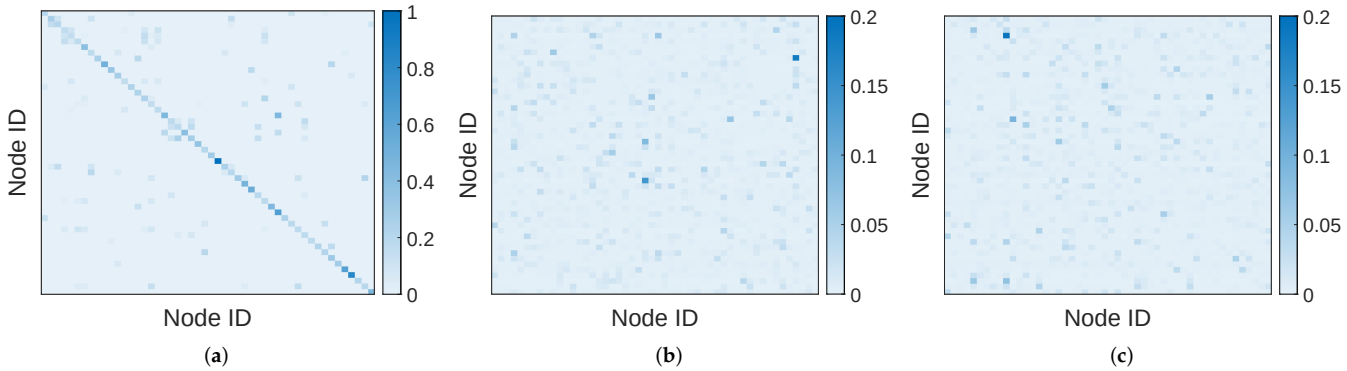

Figure 5. Heatmaps of predefined distance-based graph and the learned time-evolving graphs on METR-LA. (**a**) Distance-based graph. (**b**) Learned time-evolving graph at 08:00. (**c**) Learned time-evolving graph at 17:30.

4.6. Parameters Study

We studied the model performance with different values of two important hyper-parameters: the embedding dimension d in the time-evolving graph generation module and the hidden size h of GRU units. The latter is also the output dimension of the graph convolution module. For each experiment, we only change the selected hyper-parameter while fixing other settings. The average MAE loss of different settings on METR-LA and PEMS-BAY are shown in Figures 6 and 7, respectively. The optimal values of both embedding dimension and hidden size on PEMS-BAY are larger because of the larger amount of traffic nodes. For METR-LA, we set the embedding dimension at 30 and the hidden size at 40. The counterparts of PEMS-BAY are 40 and 50. The performance on METR-LA is more sensitive to the values of hyper-parameters.

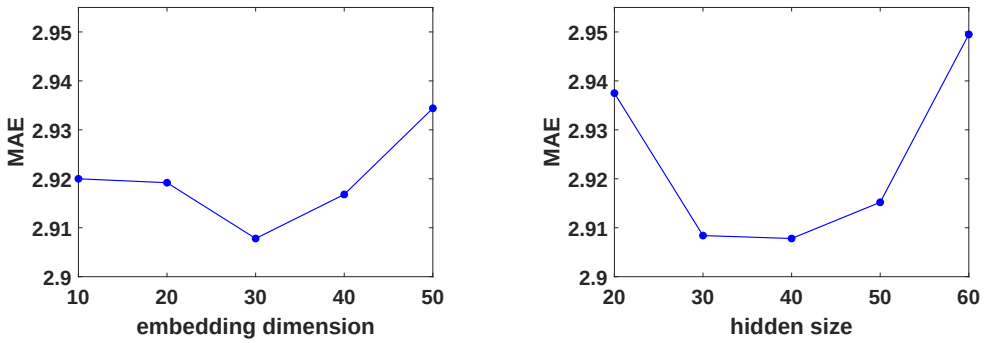

Figure 6. Parameters study on METR-LA. (**a**) Performance with different embedding dimensions. (**b**) Performance with different hidden sizes.

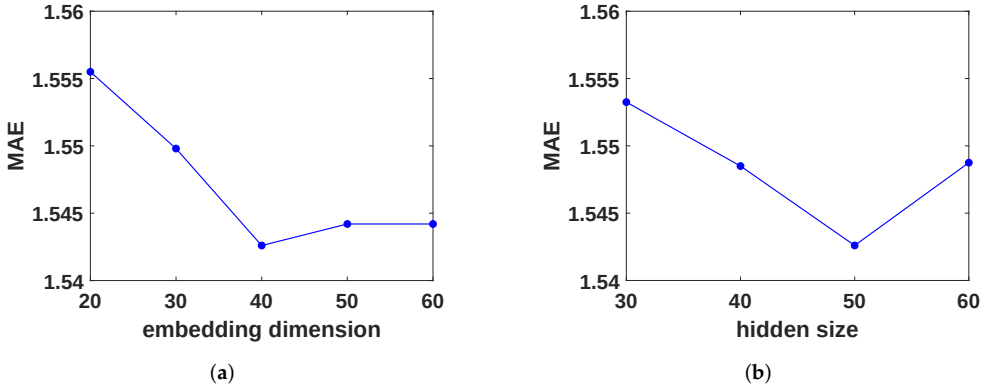

Figure 7. Parameters study on PEMS-BAY. (**a**) Performance with different embedding dimensions. (**b**) Performance with different hidden sizes.

4.7. Discussion

The proposed model TEGCRN may need more costs when training the time-evolving adjacency graphs. However, once the model is trained and the weights in the time-evolving graphs are frozen, the time-evolving graphs can be stored offline and the calculations of graph generation are no longer needed. So, the computational complexity of the graph convolution modules is the same as those with static graphs when making predictions. In the experiments described in the previous subsections, we take the prediction task for the traffic speed as an example and set the number of traffic features $F = 1$. However, it is worth mentioning that the proposed model is devised as a general traffic prediction framework which can be applied to the predictions of other parameters in the traffic network (e.g., traffic volume and congestion index) and take multiple features as input. Therefore, the proposed model can be employed as a building block in different traffic support systems and provide more accurate predictions for multiple applications such as route planning and logistics distribution.

5. Conclusions

In this paper, we propose a traffic prediction model named Time-Evolving Graph Convolutional Recurrent Network. Our method uses a tensor-composing method to generate adaptive time-evolving graphs, then integrates the graph convolution module with these time-evolving graphs into the RNN structure to a form general prediction framework. Experiments on two real-world traffic datasets demonstrate the superiority of TEGCRN over multiple competitive baseline models especially in short-term prediction. We verify the effectiveness of time-evolving graph convolution in capturing more comprehensive inter-node dependency than the models with static graph convolution. The proposed model

can be employed as a support block in traffic-related systems and provide more accurate predictions for multiple applications. In the future, we plan to work on the heterogeneous relationships in traffic networks and build a predicting model with more explainability.

Author Contributions: Conceptualization, W.M. and J.C.; methodology, W.M.; software, W.M.; validation, W.M., J.C. and X.C.; formal analysis, W.M. and X.C.; investigation, W.M.; writing—original draft preparation, W.M.; writing—review and editing, W.M., J.C. and X.C.; visualization, W.M.; supervision, X.C.; project administration, X.C.; funding acquisition, X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key Research and Development Program of China (No. 2019YFE0114000), Guangdong R&D Project in Key Areas under Grant (2019B010158001), Industry-University-Research Cooperation Project in Zhuhai (No. ZH22017001200072PWC).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used to support the findings of this study are included in the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
GCN	Graph Convolutional Network
POI	Point of Interest
TEGCRN	Time-Evolving Graph Convolution Recurrent Network
ARIMA	Auto-Regressive Integrated Moving Average
VAR	Vector Auto-Regression
SVR	Support Vector Regression
LSTM	Long Short-Term Memory
GRU	Gated Recurrent Unit

References

1. Kechagias, E.P.; Gayialis, S.P.; Konstantakopoulos, G.D.; Papadopoulos, G.A. An application of an urban freight transportation system for reduced environmental emissions. *Systems* **2020**, *8*, 49. [CrossRef]
2. Kechagias, E.P.; Gayialis, S.P.; Konstantakopoulos, G.D.; Papadopoulos, G.A. Traffic flow forecasting for city logistics: A literature review and evaluation. *Int. J. Decis. Support Syst.* **2019**, *4*, 159–176. [CrossRef]
3. Kumar, S.V.; Vanajakshi, L. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *Eur. Transp. Res. Rev.* **2015**, *7*, 1–9. [CrossRef]
4. Zhang, G.P. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* **2003**, *50*, 159–175. [CrossRef]
5. Lippi, M.; Bertini, M.; Frasconi, P. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 871–882. [CrossRef]
6. Ma, X.; Dai, Z.; He, Z.; Ma, J.; Wang, Y.; Wang, Y. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* **2017**, *17*, 818. [CrossRef] [PubMed]
7. Wu, Y.; Tan, H. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv* **2016**, arXiv:1612.01022.
8. Zhang, J.; Zheng, Y.; Qi, D. Deep spatio-temporal residual networks for citywide crowd flows prediction. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
9. Yao, H.; Wu, F.; Ke, J.; Tang, X.; Jia, Y.; Lu, S.; Gong, P.; Ye, J.; Li, Z. Deep multi-view spatial-temporal network for taxi demand prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
10. Lin, Z.; Feng, J.; Lu, Z.; Li, Y.; Jin, D. Deepstn+: Context-aware spatial-temporal neural network for crowd flow prediction in metropolis. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 1020–1027.

11. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
12. Li, Y.; Yu, R.; Shahabi, C.; Liu, Y. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
13. Yu, B.; Yin, H.; Zhu, Z. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 3634–3640.
14. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Zhang, C. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 1907–1913.
15. Guo, S.; Lin, Y.; Feng, N.; Song, C.; Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 922–929.
16. Song, C.; Lin, Y.; Guo, S.; Wan, H. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 914–921.
17. Kong, X.; Xing, W.; Wei, X.; Bao, P.; Zhang, J.; Lu, W. STGAT: Spatial-temporal graph attention networks for traffic flow forecasting. *IEEE Access* **2020**, *8*, 134363–134372. [CrossRef]
18. Li, M.; Zhu, Z. Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; Volume 35, pp. 4189–4196.
19. Chen, W.; Chen, L.; Xie, Y.; Cao, W.; Gao, Y.; Feng, X. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3529–3536.
20. Geng, X.; Li, Y.; Wang, L.; Zhang, L.; Yang, Q.; Ye, J.; Liu, Y. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 3656–3663.
21. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Online, 6–10 July 2020; pp. 753–763.
22. Wang, Y.; Papageorgiou, M. Real-time freeway traffic state estimation based on extended Kalman filter: A general approach. *Transp. Res. Part Methodol.* **2005**, *39*, 141–167. [CrossRef]
23. Kumar, S.V. Traffic flow prediction using Kalman filtering technique. *Procedia Eng.* **2017**, *187*, 582–587. [CrossRef]
24. Zhang, Y.; Zhang, Y. A comparative study of three multivariate short-term freeway traffic flow forecasting methods with missing data. *J. Intell. Transp. Syst.* **2016**, *20*, 205–218. [CrossRef]
25. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3844–3852.
26. Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.I.; Jegelka, S. Representation learning on graphs with jumping knowledge networks. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 5449–5458.
27. Bruna, J.; Zaremba, W.; Szlam, A.; LeCun, Y. Spectral networks and locally connected networks on graphs. In Proceedings of the 2nd International Conference on Learning Representations, Banff, AB, Canada, 14–16 April 2014.
28. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph Attention Networks. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
29. Chen, D.; Lin, Y.; Li, W.; Li, P.; Zhou, J.; Sun, X. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3438–3445.
30. Xu, B.; Shen, H.; Cao, Q.; Cen, K.; Cheng, X. Graph convolutional networks using heat kernel for semi-supervised learning. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 1928–1934.
31. Li, Q.; Han, Z.; Wu, X.M. Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
32. Han, L.; Du, B.; Sun, L.; Fu, Y.; Lv, Y.; Xiong, H. Dynamic and Multi-faceted Spatio-temporal Deep Learning for Traffic Speed Forecasting. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, Singapore, 14–18 August 2021; pp. 547–555.
33. Tucker, L.R. Some mathematical notes on three-mode factor analysis. *Psychometrika* **1966**, *31*, 279–311. [CrossRef] [PubMed]
34. Shuman, D.I.; Narang, S.K.; Frossard, P.; Ortega, A.; Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Process. Mag.* **2013**, *30*, 83–98. [CrossRef]
35. Liu, M.; Gao, H.; Ji, S. Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Online, 6–10 July 2020; pp. 338–348.

36. Klicpera, J.; Bojchevski, A.; Günnemann, S. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In Proceedings of the 7th International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
37. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 3104–3112.
38. Bengio, S.; Vinyals, O.; Jaitly, N.; Shazeer, N. Scheduled sampling for sequence prediction with recurrent Neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1, Montreal, QC, Canada, 7–12 December 2015; pp. 1171–1179.