Received March 18, 2019, accepted May 3, 2019, date of publication May 7, 2019, date of current version April 15, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2915364

Sequential Graph Neural Network for Urban Road Traffic Speed Prediction

ZHIPU XIE[1], WEIFENG LV[1,2], SHANGFO HUANG[1], ZHILONG LU[1], BOWEN DU[1,2], AND RUNHE HUANG[3], (Fellow, IEEE)

[1]Key State Laboratory of Software Development Environment, Beihang University, Beijing 100091, China
[2]Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing 100091, China
[3]Hosei University, Tokyo 102-8160, Japan

Corresponding author: Bowen Du (dubowen@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 51778033 and Grant 51822802, and in part by the Beijing Science and Technology Project under Grant Z171100005117001.

ABSTRACT Accurate speed predictions for urban roads are highly important for traffic monitoring and route planning, and also help relieve the pressure of traffic congestion. Many existing studies on traffic speed prediction are based on convolutional neural networks, and these have primarily focused on capturing the spatial proximity among different road segments. However, the real cause of the spread of traffic congestion is the connectivity of these road segments, rather than their spatial proximity. This makes it very challenging to improve prediction accuracy. Using graph neural networks (GNNs), the connectivity of these road segments can be modeled as a graph in which the properties of road segments and the connections between them are embedded as the properties of the nodes and edges, respectively. This paper describes a novel approach that combines the advantages of sequence-to-sequence (Seq2Seq) models and GNNs. Specifically, the evolution of traffic conditions on road networks is modeled as a sequence of graphs. Thus, the proposed SeqGNN model represents both the inputs and outputs as graph sequences. Finally, the extensive experiments using real-world datasets demonstrate the effectiveness of our approach and its advantages over the state-of-the-art methods.

INDEX TERMS Graph neural network, Seq2Seq, traffic speed prediction.

I. INTRODUCTION

Predictions of travel speeds on urban roads are an important component of intelligent transportation systems. Popular transportation applications such as traffic monitoring, navigation, and congestion avoidance [1] are heavily reliant on high-quality speed predictions. However, such predictions are complicated by the dynamic and complex traffic environment and irregular flow patterns. Thus, an increasing number of studies have attempted to capture these patterns so as to improve the prediction accuracy.

In recent years, deep learning models have achieved excellent performance in many fields [2]. Transportation researchers are interested in applying deep neural networks (DNNs) to traffic information prediction problems. Most of the current research is based on convolutional neural networks (CNNs) [3], recurrent neural networks (RNNs) [4], or a combination of the two [5]. CNN-based methods usually model the road network as an image [6] so as to capture the spatial dependency of network-wide traffic flows. In contrast, RNN-based methods are suitable for learning temporal dynamics. Long short-term memory (LSTM) neural networks, which are capable of learning both long- and short-term time series, have achieved remarkable traffic prediction results.

However, traditional CNNs and RNNs do not explore important prior knowledge that can be represented as a graph, such as the topology of the road network, the properties of road segments, and the connections between them. Although spatial proximity seems to have an impact on the correlation of traffic conditions, the real cause of the spread of traffic congestion is the connectivity of these road segments within the road network topology.

For example, as shown in Fig. 1, segments A, B, C, D, and E are located on main roads connecting the city to the suburbs. A, B, D, and E run from downtown to the suburbs, whereas C runs in the opposite direction. During the evening peak, congestion starts at A in Fig. 1(a). After some time,

The associate editor coordinating the review of this manuscript and approving it for publication was Alexandros Iosifidis.

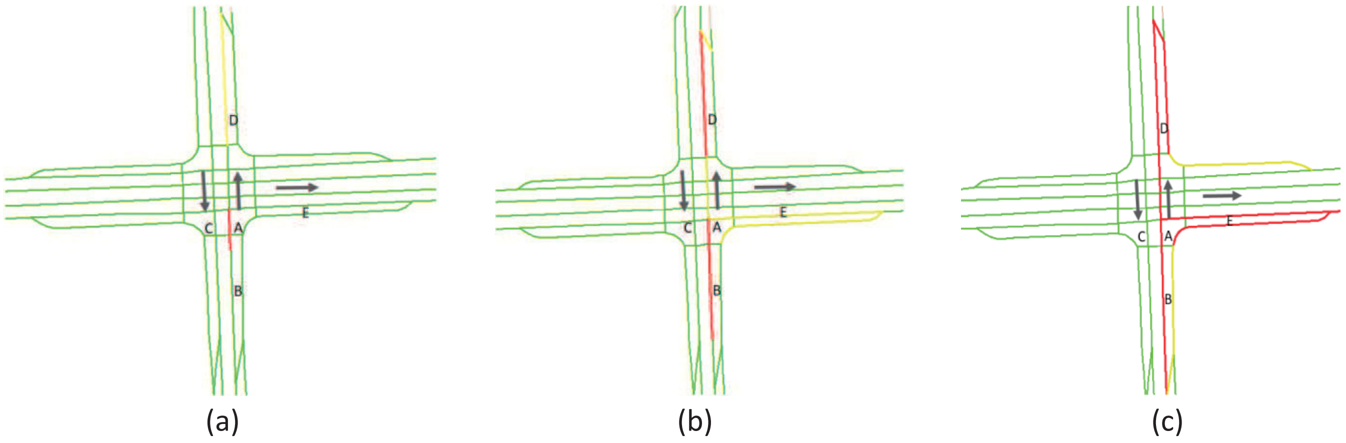

FIGURE 1. Traffic condition evolution in a complex road network structure.

segments *B*, *D*, and *E* become affected (Fig. 1(b)). Finally, all segments connected with *A* are heavily congested, whereas *C* remains congestion-free (Fig. 1(c)). This indicates that the connectivity of segments in the road network topology plays a more important role than spatial proximity in the spread of a traffic state.

Studies on graph neural networks (GNNs) [7], [8] and graph CNNs (GCNNs) [9] have introduced new ideas to the whole road network in terms of traffic speed prediction. Motivated by this, we formulate both the input and output of our problem as sequential graphs. However, this leaves three challenges. The first is how to identify useful features from complex road networks so as to capture patterns in the evolution of traffic speeds. Second, existing works on GNNs simply feed the network with sequential inputs, but only output a single value rather than a sequence of graphs. Third, the encoder and decoder mechanisms of traditional sequence-to-sequence (Seq2Seq) models are unsatisfactory for handling graph-structure data directly.

Thus, in this paper, a novel traffic speed prediction model called SeqGNN is proposed. The main contributions of this paper are summarized as follows:

- Based on GNNs, the property and structure of road networks are considered. Our proposed model feeds the whole road network into the neural network and learns to forecast the traffic flow.
- Our method is an extension of a GNN, modifying the input and output to sequential graphs using GRNN blocks.
- We conduct extensive experiments on real-life datasets to verify the prediction accuracy of the proposed SeqGNN. In our experiments, state-of-the-art techniques and strong baselines are compared with our method in terms of traffic flow forecasting.

The remainder of this paper is organized as follows. Section II discusses some related studies. Section III introduces some preliminary definitions that will be applied in this paper. Section IV discusses the proposed method, before Section V evaluates the performance of SeqGNN in terms of traffic flow prediction and presents the experimental results. Finally, Section VI concludes the paper with some ideas for future research.

II. RELATED WORK

A. URBAN ROAD TRAFFIC SPEED PREDICTION

The problem of predicting traffic speeds on urban roads has attracted many researchers, and various approaches have been proposed. Early studies [10] defined road traffic speed prediction as a time series sequence problem that can be resolved using autoregressive moving average methods. Other studies used statistical learning methods such as k-nearest neighbor regression [11], support vector regression [12], and extreme gradient boosting [13]. The advantage of statistical learning methods over time series methods is their ability to combine relevant information, which enhances the prediction performance. However, the potential for improving the performance of these models is very limited because of their insufficient modeling of complex scenarios.

Recently, there have been a number of breakthroughs in the theoretical basis of DNNs, as well as successful applications in multiple areas. Of the wide variety of DNN models, CNNs and RNNs have received considerable attention. Initially, CNNs were mainly used to process image data, though they have since been applied to other data that can be expressed in two-dimensional matrix form. The convolutional layers of a CNN consist of a number of convolution kernels with shared weights. A convolution kernel is only connected to variables in a certain region, which greatly reduces the number of hyperparameters in the neural network. In addition, pooling is commonly used in CNNs. For example, maximum pooling refers to finding the largest-value element in a matrix. RNNs contain a loop structure that allows them to process data of a corresponding form, including various types of sequential data such as articles and speech. LSTM, a kind of RNN, is characterized by solving the gradient vanishing problem in the traditional RNN with gated structures.

Prediction methods based on DNNs further improve performance by obtaining more meaningful spatiotemporal information. Tian and Pan [4] used an LSTM-RNN method to capture the nonlinearity and randomness of traffic flow more effectively, resulting in higher prediction accuracy. Chen et al. [3] proposed a CNN for modeling periodic traffic data in which the time series is folded into a two-dimensional matrix for input to the CNN. In addition, this CNN models multiscale traffic patterns, whereby the general trend is determined by the overall situation and more detailed changes need to be obtained locally. Jiang and Zhang [6] transformed geospatial data into images, and then used a CNN and residual network as a deep-learning framework for traffic prediction. Yu et al. [5] proposed a novel spatiotemporal recurrent convolutional network for traffic forecasting, thus inheriting the advantages of both CNNs and LSTM. However, one limitation of traditional CNNs and RNNs is that complex road networks cannot be modeled efficiently.

B. GRAPH NEURAL NETWORKS

Neural networks have been employed to represent graph-structured data, e.g., social networks and knowledge bases. The classical CNNs and RNNs have also been deployed on graph-structured data. For instance, [14] introduces a CNN that operates directly on graphs of arbitrary size and shape, whereas [15] describes a scalable approach with a convolutional architecture based on a localized approximation of spectral graph convolutions, an efficient variant that can operate directly on graphs. These methods can only be implemented on undirected graphs.

GNNs (neural networks that model graphs) were first defined in 2005 [16], and their scope and popularity have grown rapidly in recent years. The entire GNN family covers a wide range of topics including supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning. For example, the GNN model proposed by Scarselli et al. [17] can map a graph G and all its nodes $n \in N$ into an m-dimensional Euclidean space, while Li et al. [18] extended the work of [17] using gated recurrent units to form GNN output sequences. In addition to a number of different GNNs, Bruna et al. [19] proposed the GCNN, in which a CNN is used to process graph data. Monti et al. [20] proposed a unified CNN-based architecture to deal with non-Euclidean structures including graphs and 3D surfaces. Later, Gilmer et al. [21] unified multiple GNNs and GCNNs into one form: message-passing neural networks (MPNNs). Similarly, the non-local neural network (NLNN) proposed by Wang et al. [22] unifies various "self-attention models."

Although the concept of GNNs was proposed some time ago, its development is still based on the improvement of the theoretical background provided by research on DNNs. The basic idea in constructing a GNN is based on the theory of RNNs. Specifically, in GNNs, certain neural network modules are defined and then recursively processed on each node and each edge. In this way, the implementation of the GNN usually has more complicated operations, and so the recent rapid development of GNN theory has benefited from research on accelerating neural network training speeds with graphic processing units (GPUs).

Recently, Battaglia et al. [8] proposed a new type of neural network module: the graph network (GN), which summarizes and extends various types of GNNs, including MPNNs and NLNNs. The GN defines unified neural network units called GN blocks, each of which is a "graph-to-graph" module that takes a graph as input, performs some computations over the structure, and returns a graph as output. The construction of complex architectures is supported by organizing GN blocks in various ways. Knowledge in the form of a graph structure has recently been used for traffic predictions, such as a GCNN + LSTM model for real-time traffic prediction [23], but its performance has only been tested with experiments on simulation data rather than real-world data.

Although graph structure modeling methods based on GNNs or GCNNs are developing rapidly, modeling the evolution of traffic conditions requires an approach for handling graph sequences.

III. PROBLEM FORMULATION

Modeling real-world road networks as linkage graphs would enable them to be the inputs to a GNN. In this section, we first define both real-world road networks and linkage graphs, and attach road attributes to the vertexes and edges of the graph. The problem of urban road traffic speed prediction is then introduced.

Definition 1 (Road Network): We use an unweighted directed graph $RNG(RNV, RNE)$ to describe the topological structure of a real-world road network. Each road intersection is a node, the vertexes V form a set of nodes, $RNV = \{rnv_1, rnv_2, \cdots, rnv_N\}$, and N is the number of road intersections. E is a set of directional edges denoting the linkages between contiguous intersections, i.e., road segments. $RNE = \{rne_1, rne_2, \cdots, rne_M\}$, where M is the number of road segments. A directional edge from intersection rnv_i to rnv_j is established if and only if there is a road segment linking intersections rnv_i and rnv_j. We use a subgraph of the whole traffic system, as shown in Fig. 2, as an example.

The real-world road network shown in Fig. 2(a) consists of vertexes A–L, representing the intersections, and directed edges a_1–a_8, representing directional road segments between intersections.

Definition 2 (Linkage Network): To handle simple graphs and combine the useful information contained in vertexes, as well as to simplify the propagation pattern in the graph, we transform the road network to a linkage graph-structured network [7]. The topological structure of the connected graph-structured network is described as an unweighted directed graph $LG(LV, LE)$, where the vertexes LV are formed by a set of nodes, and each node represents a road segment. Thus, $LV = \{lv_1, lv_2, \cdots, lv_O\}$, and O is the number of road segments. LE is a set of edges, each of which denotes a direct connection between adjacent road segments. Thus, $LE = \{le_1, le_2, \cdots, le_P\}$, where P is the number of linkages. A directional edge from segment lv_i to lv_j will be established

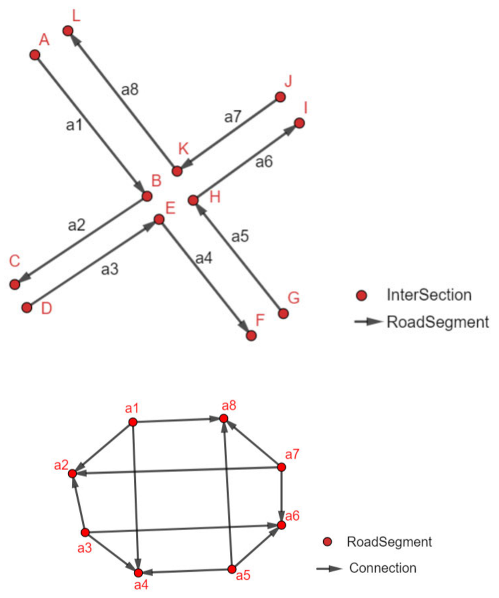

FIGURE 2. Differences between the real-world network and the linkage network.

if and only if the final intersection of segment lv_i corresponds to the first intersection of segment lv_j.

Fig. 2(b) illustrates the graph structure of the linkage network transformed from the real-world road network shown in Fig. 2(a). Each node in the graph represents a road segment and each edge represents the connection between road segments.

Definition 3 (Linkage Attribute Graph): According to the linkage network defined above, we can describe the topology of the road network with graphs. However, to feed more comprehensive information of the road network into the GNN, corresponding attributes of the nodes and edges in the linkage network must be assigned. A Linkage Attribute Graph (LAG) is defined as a linkage network combined with attribute information of both nodes and edges. The attributes of nodes, which represent different road segments, include the road segment width, length, and direction. The attributes of edges, which are the intersections between road segments, include the type of intersection and whether there are traffic lights. More details of these attributes are listed in Tables 1 and 2.

Definition 4 (Traffic Speed Graph): The Traffic Speed Graph (TSG) of a linkage network in time interval T, represented as TSG_T, uses the travel speed values on corresponding road segments as the node attributes. The speed values are calculated as the average speed of vehicles along certain road segments.

In this study, the goal is to predict the traffic speed over a certain period of time based on the historical traffic speeds on the roads in the network. We define the traffic congestion prediction problem as follows:

Definition 5 (Road Traffic Speed Prediction) The speed prediction problem is to estimate the travel speed on road segments over the s time intervals from t_{r+1} to t_{r+s}, given

TABLE 1. Attributes of road segments in the road network.

Field	Description
RS_id	Road segment id
RS_name	Road segment name
RS_length	Road segment length
RS_width	Road segment width
RS_kind	An integer indicates the type of the road segment, ranging from 1 to 7.
Direction	An integer indicates the direction of the road segment, ranging from 1 to 8.
RS_freeflow	Vehicle speed values when traffic volume is low, usually equals to the maximum allowed speed of the road segment.
RS_funclass	The function class of the road segment.
Lane_number	The number of lanes

TABLE 2. Attributes of intersections in the road network.

Field	Description
I_id	Intersection id
I_type	An integer indicates the Intersection type, such as "turn left", "turn right", "turn around" and so on.
Has_light	Whether there is a traffic light in the intersection or not
Has_tollgate	Whether there is a tollgate in the intersection or not

the data during the r time intervals from t_1 to t_r. For each road segment v_i of a specific road network $G(V, E)$, given the sequential traffic speeds $RS_{t_j}^{v_i}$ over the r time intervals $t_1, t_2, \cdots, t_r$, $i = 1, \cdots, N$, we aim to predict the stochastic traffic speed of the next s time intervals $t_{r+1}, t_{r+2}, \cdots, t_{r+s}$.

In addition to historical speed data, we can incorporate contextual information such as temporal features. We denote the contextual features for a segment v_i and a time interval t_j as a vector $d^{v_i} \in \mathcal{R}$. Therefore, our final goal is to predict

$$\begin{aligned}&\left(RS_{t_{r+1}}^{v_i}, RS_{t_{r+2}}^{v_i}, \cdots, RS_{t_{r+s}}^{v_i}\right)\\&\quad = \mathcal{F}\left(RS_{t_1}^{v_i}, RS_{t_2}^{v_i}, \cdots, RS_{t_r}^{v_i}; d^{v_i}\right)(i \in 1, 2, \cdots, N). \qquad (1)\end{aligned}$$

In our approach, we predict all N segments simultaneously.

IV. METHODOLOGY

We now introduce the proposed SeqGNN by applying GNNs to the prediction of traffic congestion. We first show how to construct the graph, and then describe SeqGNN in detail.

A. GN BLOCK FOR GRAPH TO GRAPH PROCESSING

A GN block is a "graph-to-graph" module, which means it takes a graph as input and outputs another graph. Within a GN block, a graph is defined as a triple $\mathcal{G} = (\mathrm{u}, \mathcal{V}, \mathcal{E})$. u is a global attribute of $\mathcal{G}$, representing a system-level property of the graph. $\mathcal{V} = \{(vid_i, \mathbf{v}_i)\}$ denotes a set of nodes, where each vid_i is the index of a node and $\mathbf{v}_i$ is the vector containing its attributes. $\mathcal{E} = \{(eid_k, rid_k, sid_k, \mathbf{e}_k)\}$ is the set of edges, where each eid_k is the index of an edge, rid_k is the vid of the receiver node, sid_k is the vid of the sender node, and $\mathbf{e}_k$ denotes the vector of its attributes. Herein, the GN blocks are formulated as a function:

$$\mathbf{GN}: (\mathrm{u}', \mathcal{V}', \mathcal{E}') \mapsto (\mathrm{u}, \mathcal{V}, \mathcal{E}). \qquad (2)$$

The computations are performed over the following three steps.

1. Update $\mathbf{e}_k$ of each edge to $\mathbf{e}'_k$ by applying

$$\mathbf{e}'_k = W^e_{out} \cdot tanh(W^e_{hid} \cdot [\mathbf{e}_k; \mathbf{v}_{rk}; \mathbf{v}_{sk}; \mathrm{u}] + b^e_{hid}) + b^e_{out}, \quad (3)$$

where $\mathbf{v}_{rk}$ denotes the attributes of the receiver and $\mathbf{v}_{sk}$ denotes the attributes of the sender. $[\mathbf{x_1}; \mathbf{x_2}; \ldots; \mathbf{x_k}]$ indicates vector/tensor concatenation, and $W^e_{out} \cdot tanh(W^e_{hid} \cdot (*) + b^e_{hid}) + b^e_{out}$ represents a two-layer fully connected network for updating the edges.

2. Compute the updated attributes of node i, denoted as $\mathbf{v}'_i$, by applying

$$\mathbf{v}'_i = W^v_{out} \cdot tanh(\mathcal{N}^v_{hid}) + b^v_{out}, \quad (4)$$

$$\mathcal{N}^v_{hid} = W^v_{hid} \cdot ([aggr_{e\to v}(\mathcal{E}'_i); \mathbf{v}_i; \mathrm{u}]) + b^v_{hid}, \quad (5)$$

where $W^v_{out} \cdot tanh(W^v_{hid} \cdot (*) + b^v_{hid}) + b^v_{out}$ represents a two-layer fully connected network for updating the nodes, $\mathcal{E}'_i$ is a subset of $\mathcal{E}$ and $\mathcal{E}'_i = \{e'_i \in \mathcal{E} \mid e'_i.rid_k = i\}$, i.e., edges whose receivers are node i, and $aggr_{e\to v}(*)$ is an aggressive function that takes variable numbers of vectors as input and outputs a single vector; here, we choose $aggr_{e\to v}(*)$ to denote the summation operator.

3. Update u to u' by

$$\mathrm{u}' = W^u_{out} \cdot tanh(\mathcal{N}^u_{hid}) + b^u_{out}, \quad (6)$$

$$\mathcal{N}^u_{hid} = W^u_{hid} \cdot ([aggr_{e\to u}(\mathcal{E}'); aggr_{v\to u}(\mathcal{V}'); \mathrm{u}]) + b^u_{hid}, \quad (7)$$

where $W^u_{out} \cdot tanh(W^u_{hid} \cdot (*) + b^u_{hid}) + b^u_{out}$ represents a two-layer fully connected network for updating u, and $aggr_{e\to u}(*)$, $aggr_{v\to u}(*)$ are aggressive functions acting as summation operators.

In the GN block, the input and output graphs are isomorphic, as defined in Definition 6.

Definition 6 (Isomorphic Graphs): The graphs $\mathcal{G}_1 = (\mathrm{u}_1, \mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathrm{u}_2, \mathcal{V}_2, \mathcal{E}_2)$ are said to be isomorphic if and only if there exist bijections $\phi(*)$ of $\mathcal{V}_1$ and $\mathcal{V}_2$ and $\psi(*)$ of $\mathcal{E}_1$ and $\mathcal{E}_2$ and, for each $e = (eid, rid, sid, ef) \in \mathcal{E}_1$ and $\tilde{e} = \psi(e) = (\tilde{eid}, \tilde{rid}, \tilde{sid}, \tilde{ef}) \in \mathcal{E}_2$, $\tilde{rid} = \phi(rid)$ and $\tilde{sid} = \phi(sid)$. In other words, if two graphs are isomorphic, their topologies are the same but the features of the elements are always different.

B. GRAPH RECURRENT NEURAL NETWORK BLOCK

Our model uses a structure called a graph RNN (GRNN) block. This structure is a combination of the GN block, which can process data related to the graph structure, and the RNN cell, which can process sequential data. A basic RNN cell, as shown in Fig. 3(a), is abstracted as a function that takes two vectors as inputs and outputs two different vectors:

$$\textbf{Basic RNN Cell}: (x, h) \mapsto (y, h'), \quad (8)$$

where x is the input of the current loop, h represents the current hidden state, h' is the hidden state of the next loop, and y is the output. As an analogy, the GRNN block shown in Fig. 3(b) is denoted as a function that takes two graphs as inputs and outputs two different graphs:

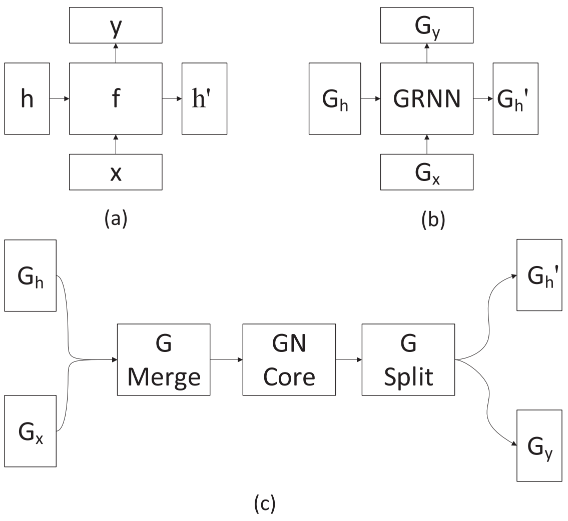

FIGURE 3. Inner architecture of GRNN block.

$$\textbf{GRNN block}: (\mathcal{G}_x, \mathcal{G}_h) \mapsto (\mathcal{G}_y, \mathcal{G}'_h), \quad (9)$$

where $\mathcal{G}_x$ is the input graph of the current loop, $\mathcal{G}_h$ is the current hidden state, which is represented as a graph, $\mathcal{G}'_h$ is the hidden state of the next loop, and $\mathcal{G}_y$ is the output.

The inner structure of the GRNN block is shown in Fig. 3(c), where **GMerge** is a module that merges $\mathcal{G}_x = (\mathrm{u}_x, \mathcal{V}_x, \mathcal{E}_x)$ and $\mathcal{G}_h = (\mathrm{u}_h, \mathcal{V}_h, \mathcal{E}_h)$, denoted as $\mathcal{G}_M(\mathrm{u}_M, \mathcal{V}_M, \mathcal{E}_M)$:

$$\mathcal{G}_M = \textbf{GMerge}(\mathcal{G}_x, \mathcal{G}_h), \quad (10)$$

where $u_M = concat(\mathrm{u}_x, \mathrm{u}_h)$, $\mathcal{V}_M = \{(vid^x_i, concat(\mathbf{v}^x_i, \mathbf{v}^h_i))\}$, and $\mathcal{E}_M = \{(eid^x_k, rid^x_k, sid^x_k, concat(\mathbf{e}^x_k, \mathbf{e}^h_k))\}$. $\mathcal{G}_x$, $\mathcal{G}_h$, and $\mathcal{G}_M$ are isomorphic graphs that have different attributes. By convention, $concat(\mathbf{x_1}, \mathbf{x_2})$ denotes the concatenation of two vectors. Thus, in (10), the attributes of $\mathcal{G}_M$ are calculated by concatenating the attributes of $\mathcal{G}_x$ and $\mathcal{G}_h$.

$\mathcal{G}_M$ is then processed by **GNCore**, which is a series of nested GN blocks, to generate a new graph $\mathcal{G}_N$:

$$\mathcal{G}_N = \textbf{GNCore}(\mathcal{G}_M) = \prod_{i=1}^{k} \textbf{GN}_i(\mathcal{G}_M), \quad (11)$$

where $\prod_{i=1}^{k} \textbf{GN}_i(*) = \textbf{GN}_k(\prod_{i=1}^{k-1} \textbf{GN}_i(*)) = \textbf{GN}_k(\textbf{GN}_{k-1}(\prod_{i=1}^{k-2} \textbf{GN}_i(*))) = \ldots$ is the composition of k different **GN** blocks.

Finally, to output $\mathcal{G}'_h$ and $\mathcal{G}_y$, $\mathcal{G}_N$ is split by **GSplit**:

$$\begin{aligned}(\mathcal{G}'_h, \mathcal{G}_y) &= \textbf{GSplit}(\mathcal{G}_N)\\ \mathcal{G}'_h &= (\mathrm{u}_N[0:len(u)/2],\\ &\quad (vid^N_i, \mathbf{v}^N_i[0:len(u)/2]),\\ &\quad (eid^N_k, rid^N_k, sid^N_k, \mathbf{e}^N_k[0:len(u)/2]))\\ \mathcal{G}_y &= (\mathrm{u}_N[len(u)/2::], \quad (vid^N_i, \mathbf{v}^N_i[len(u)/2::]),\\ &\quad (eid^N_k, rid^N_k, sid^N_k, \mathbf{e}^N_k[len(u)/2::])), \quad (12)\end{aligned}$$

where $\mathbf{x}[0 : len(x)/2]$ denotes a vector containing elements from $\mathbf{x}$ with indexing from 0 to $len(x)/2$, and $\mathbf{x}[len(x)/2 ::]$

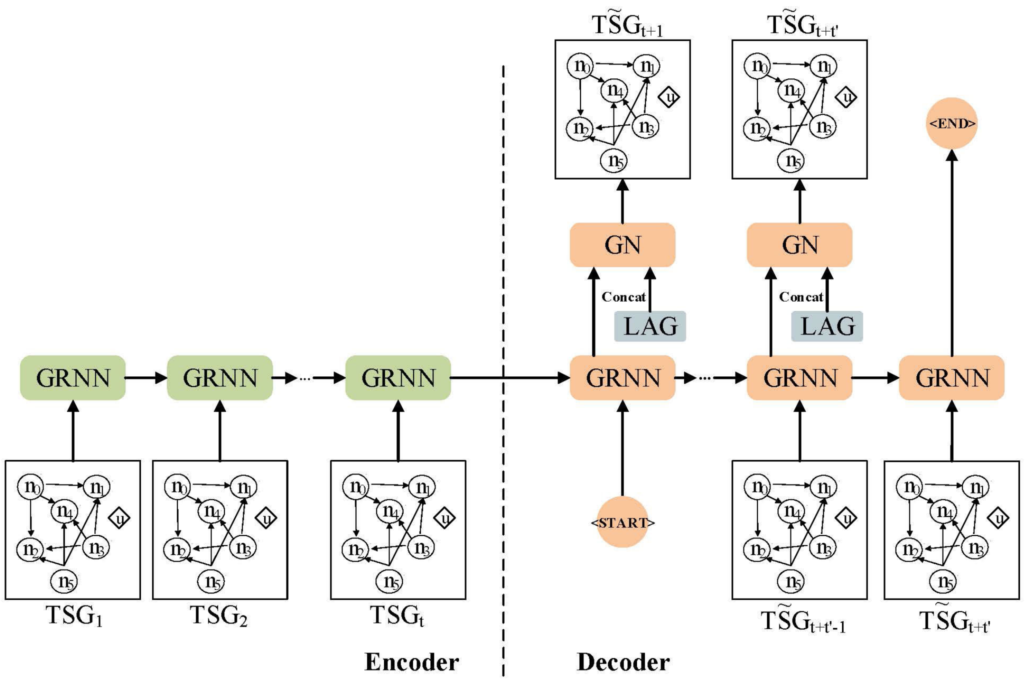

FIGURE 4. SeqGNN: The model incorporates offline geographical and temporal attributes (*AG*) into the decoder of the Seq2Seq model.

denotes a vector containing elements from **x** with indexing from $len(x)/2$ to $len(x) - 1$. **GSplit** is the reverse process of **GMerge**.

C. GRAPH SEQUENCE TO GRAPH SEQUENCE

As introduced in Section III, TSG_T is used to describe the traffic speed values of road segments in a linkage network during a certain time interval T. Based on this, the problem of predicting future traffic speeds is formulated as the prediction of a sequence of TSGs belonging to a series of time intervals. The historic speeds are also represented as TSGs and form another graph sequence. Therefore, the basic traffic speed prediction problem is to predict a future graph sequence from a historic graph sequence.

To handle this "graph sequence to graph sequence" problem, the GRNN block mentioned above is used to construct a SeqGNN network. The structure of SeqGNN is shown in Fig. 4. Similar to traditional Seq2Seq models [24], [25], the SeqGNN model is composed of an encoder and a decoder.

In the encoder phase, t historic TSGs belong to the latest t time intervals, i.e., $TSG_i, i = 1, 2, \ldots, t$, are fed as inputs. In the decoder phase, t' future TSGs belonging to time intervals $t + 1$ to $t + t'$ are predicted. For example, when decoding TSG_{t+k}, the prediction of TSG_{t+k-1} and the hidden state graph HSG_{t+k-1}, which were generated in the previous process, are needed. The GRNN block takes these two graphs as inputs and outputs the decoding result graph DRG_{t+k}, as well as the new hidden state graph HSG_{t+k}. Combining DRG_{t+k} with the LAG information, the prediction of HSG_{t+k} is generated as the outputs of a GN block.

In the GRNN model, it is beneficial to introduce both geographical and temporal attributes into traffic sequence learning, rather than merely using speed information [26]. 1) Geographical attributes include road segment attributes such as the road width, road segment direction, free flow speed, and the number of lanes as well as intersection attributes such as the intersection type, whether there are traffic lights, and whether there is a toll gate. 2) Temporal attributes include information on public holidays, workdays, peak hours, and off-peak hours. Inspired by [27], we use a wide and deep network that combines the benefits of deep learning and feature engineering. The LAGs are concatenated directly into the decoder network of the Seq2Seq model to offer more significant information.

V. EXPERIMENTAL RESULTS

To demonstrate the efficiency of our model for traffic flow prediction, we conduct a comprehensive suite of experiments on large-scale, real-world datasets. We first describe the details of our datasets and the parameter settings, and then introduce several baseline methods. We also discuss the results given by different models.

A. EXPERIMENTAL SETTINGS

1) DATA DESCRIPTION

In this study, road network data and road segment speed data from Beilin District, Xi'an City, are used to evaluate the model. The speed data are calculated from taxi trajectory data and the ST-Matching [28] algorithm was applied to match the road network with the low-sampling-rate GPS trajectories. Specifically, the data cover 1102 road segments, the

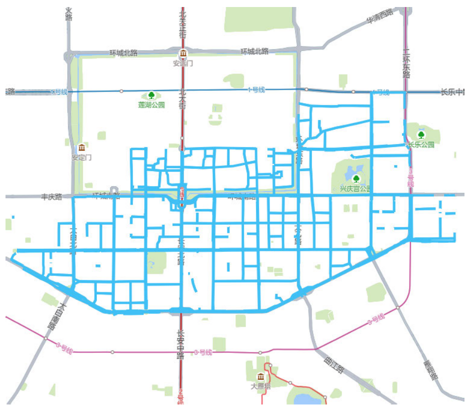

FIGURE 5. Distribution of the road segments.

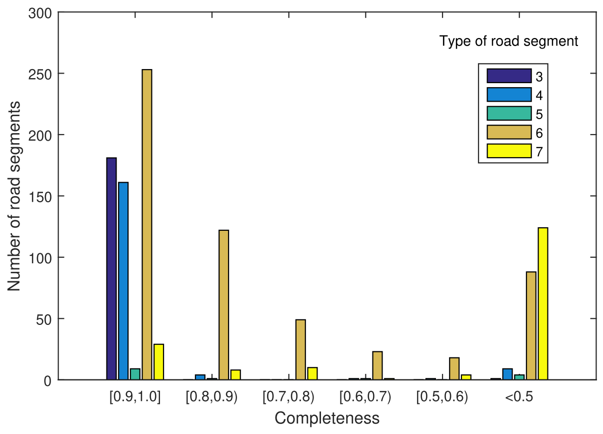

FIGURE 6. Data completeness of different road segment types.

distribution of which is illustrated in Fig. 5, from September 1 to November 30 (91 days).

The road network dataset includes basic information such as the width, length, and type of the road segments. Road segment types are represented by an integer ranging from 1 to 7. For example, a Type 1 road is an important state-owned highway, whereas a Type 7 road is generally an unimportant narrow street.

As the traffic speeds on road segments are calculated from taxi trajectory data, if none of the taxis passed by a certain road segment during time interval T, the traffic speed value of this road segment in time interval T will be missing. Thus, the traffic speed dataset is incomplete.

In Fig. 6, we illustrate statistical results showing the data completeness of different road segment types. The statistics indicate the following: 1) There are no Type 1 or 2 road segments in our dataset; 2) Type 6 is the most common road segment type; 3) The problem of data incompleteness is more prominent in Type 6 or 7 road segments.

The traffic speeds of road networks are calculated within certain time intervals. In existing research, different length time intervals have been applied, such as 2-min [5], 5-min [3], [4], and 10-min [7]. In our experiments, we aggregated the traffic speed readings into 5-min intervals, which is the most common interval in both research and industrial applications. In each instance, traffic speed values from 06:00–08:00 (i.e., 24 time intervals) on weekdays compose the input sequence, and traffic speed values for 08:00–09:00 (12 time intervals) are given as the output sequence. As the data are sparse or missing at certain times on many road segments, we select those segments with sufficient data as our research objects. This leaves us with 132 road segments. Data from the last 14 days are used for testing, and the remaining 77 days are used for training. In this way, our training set contains $77 \times 132 = 10164$ samples and our testing set contains $14 \times 132 = 1848$ samples. The input and output sequences have lengths of 24 and 12, respectively.

2) EVALUATION METRICS

The mean absolute error (MAE), mean absolute percentage error (MAPE), and root mean square error (RMSE) are used to evaluate the performance of the proposed method and comparative approaches. These metrics are defined as

$$MAE = \frac{1}{T}\sum_{i=1}^{T}|v_t - \tilde{v}_t| \tag{13}$$

$$MAPE = \frac{1}{T}\sum_{i=1}^{T}|\frac{v_t - \tilde{v}_t}{v_t}| \tag{14}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(v_t - \tilde{v}_t)^2} \tag{15}$$

where v_t and $\tilde{v}_t$ are the actual and predicted traffic speeds at time t, respectively.

3) PARAMETER SETTINGS

All selected road segments were placed into different groups, with the segments in each group passing through the same intersection. Thus, a linkage network was formed according to the connection relationships among these road segments. In the training and testing processes of SeqGNN, each instance contains all segments in a group. Obviously, there are fewer instances for models like SeqGNN, which predict all segments in one group at the same time, unlike traditional models that predict different segments one by one.

A graph-to-graph model requires a vector u that reflects the features of the whole graph. This is very helpful in cases like graph embedding, which involve transferring a whole graph to a feature vector. In our model, however, there is no clear meaning for the features of a whole graph, but we still assign global features to the graphs. These were initialized as vectors of length 3 in which all elements are equal to 0. These global features were updated in the process of model training such that the final values indicate a hidden feature of the graph. In the construction of SeqGNN, the number of hidden graphs was set to 8. The early stopping mechanism was exploited to terminate the training process if the error on the validation set increased.

We implemented our model in Python using the PyTorch and TensorFlow packages. The model was executed on a

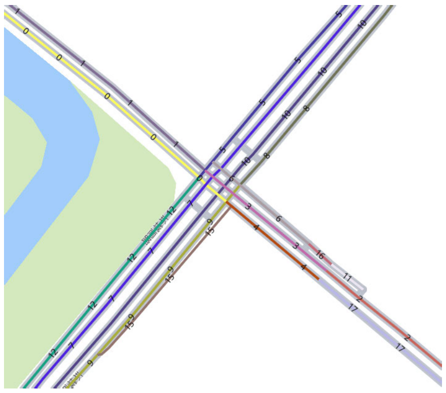
FIGURE 7. Road segments around an intersection and their IDs.

desktop with an Intel Xeon 3.2 GHz 32 CPU system and four NVIDIA TITAN Xp GPUs.

B. BASELINES

The proposed model is compared with four baselines:

- **Multilayer perceptron (MLP)**: In our experiments, the MLP had four fully connected layers consisting of 128, 128, 64, and 34 hidden units, respectively. Rectified Linear Unit (ReLU) functions were applied as activation functions and the dropout mechanism was used to prevent overfitting.
- **NLSTM [29]**: Nested LSTM is a useful method for predicting future values in a time series. The lengths of vectors produced by NLSTM cells were set to 10. The outputs of the NLSTM cells were connected with two full connection layers before the final predictions were generated.
- **Seq2Seq [25]**: This model uses an RNN to encode the input sequences into a feature representation and another RNN to make predictions. In the experiments, 3-layer LSTM (deep LSTM) cells were applied in both the encoder and the decoder. The vectors output by the deep LSTM cells contained 10 elements.
- **DA-RNN [30]**: A dual-stage attention model for time series predictions, DA-RNN has achieved state-of-the-art performance. Again, 3-layer LSTM cells were applied with output vectors of length 10.

All neural network-based approaches were implemented using PyTorch and trained using the Adam optimizer with learning rate annealing. The optimal hyperparameters were chosen based on the validation dataset.

C. RESULTS AND DISCUSSION

The convergence of the losses during the training of SeqGNN is shown in Fig. 8. The MAE loss function on both the training and validation sets over 10,000 epochs is illustrated, with the blue line representing training losses and the red line representing verification losses. As illustrated, SeqGNN converges after about 2,000 epochs, that is, both the training loss and the verification loss converge to relatively small values, and the difference between them is quite small.

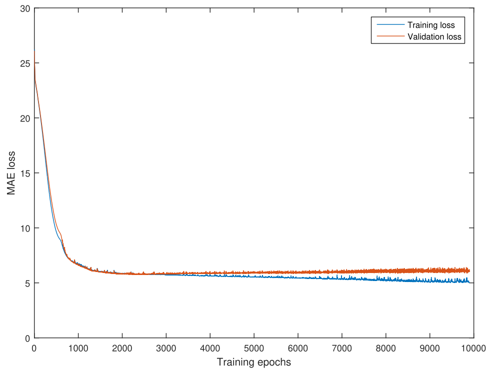

FIGURE 8. Convergence plots for MAE loss of SeqGNN.

After about 3,000 epochs, the training loss is further reduced, and the verification loss becomes unstable, indicating model overfitting. Therefore, to train a reasonable model, the number of training epochs should be controlled between 2,000 and 3,000; thus, the early stopping mechanism was applied to stop the training at a suitable time.

To illustrate the SeqGNN results on a subgraph of the real-world road network, the road segments around an intersection (Fig. 7) are chosen as an example. Small integers marked on the road segments indicate their ID numbers within the group. The traffic conditions of these road segments changed with time, and the results form a sequence of graphs whose nodes represent traffic speed values of corresponding road segments.

In Fig. 9, the first two rows, i.e., Fig. 9(a)–(f), illustrate the traffic speed values of these road segments from 07:45 to 08:15 (six time intervals). Among them, the first three time intervals belong to the input sequences and the subsequent three time intervals belong to the output sequences. Additionally, the third row, i.e., Fig. 9 (g)–(i), illustrates the prediction results of the second row given by SeqGNN. Specifically, Fig. 9(g) is the estimation/prediction of Fig. 9(d); Fig. 9(h) is the estimation/prediction of Fig. 9(e); and Fig. 9(i) is the estimation/prediction of Fig. 9(f). The values on the lines represent the traffic speed of the corresponding road segment, and the line colors indicate their traffic conditions. Green denotes smooth traffic in which the real speed is more than 80% of the free flow speed; yellow indicates slow traffic where the real speed is between 50% and 80% of the free flow speed; orange denotes congested traffic in which the real speed is between 25% and 50% of the free flow speed; and red denotes heavily congested traffic where the real speed is less than 25% of the free flow speed.

From the illustration of the traffic speed sequence in Fig. 9, many patterns of the traffic condition evolution can be inferred. For example, after the traffic condition of a road segment becomes worse, congestion is likely to spread to road sections connected to this road segment in the road network topology, rather than to road segments with a shorter geographical distance to the congested segment. In addition, the attributes of road segments influence the road condition. Traffic travel speeds on elevated roads, such as those with IDs

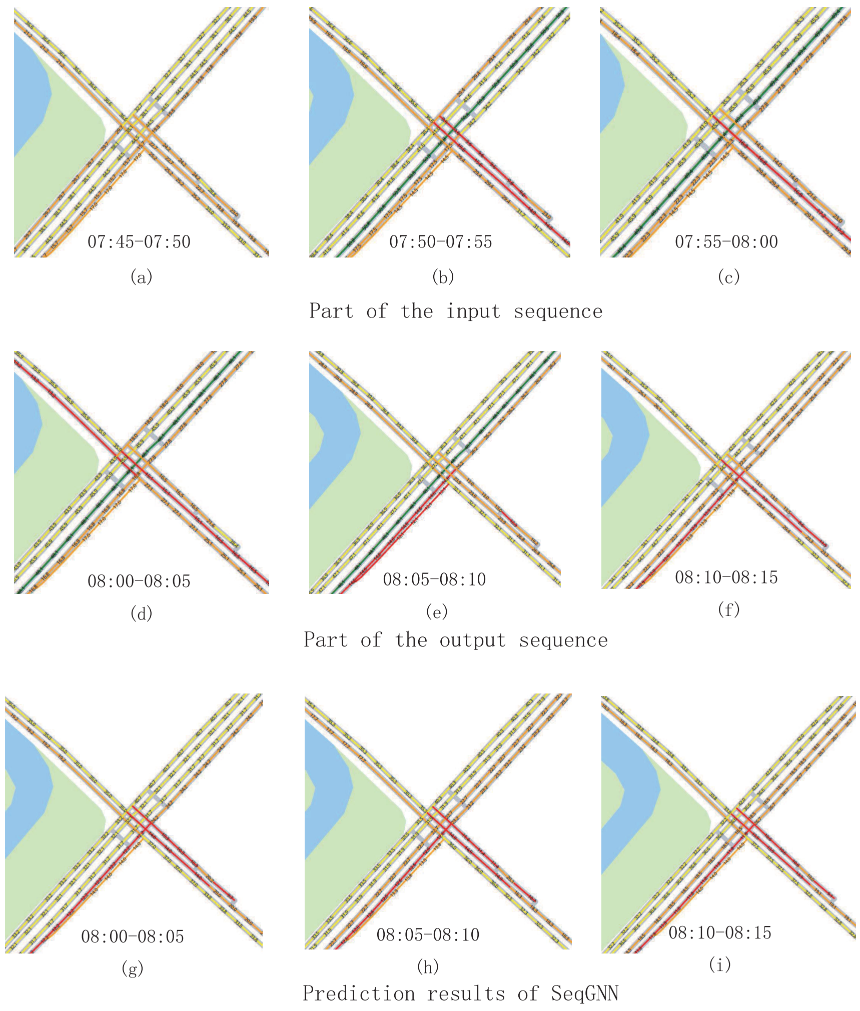

FIGURE 9. Evolution of traffic speed values on different road segments in a subgraph, ground truths, and predictions.

TABLE 3. Comparison with different baselines.

Model name	MAE (km/h)	MAPE	RMSE (km/h)
MLP	10.429152	0.485869	12.851951
NLSTM	9.187403	0.325004	11.733179
Seq2Seq	9.356720	0.326953	12.014731
DARNN	8.327549	0.293416	10.136378
SeqGNN	**7.190303**	**0.275605**	**9.607012**

from 7–10, are always faster, because there is less interruption from traffic lights.

The prediction results in the third row of Fig. 9 illustrate the advantages of the SeqGNN model. By applying the GNN block, the prediction model is able to use knowledge contained in the road network topology. Patterns of congestion spreading along connecting road segments are learned and, moreover, different attributes of road segments and their intersections are input to the prediction model.

Experiments on the whole test set were conducted, and the results of a comparison with the baselines is presented in Table 3. From this table, we can see that our model outperforms the others on all three metrics. Specifically, SeqGNN achieves an MAE of 7.19, RMSE of 9.607, and

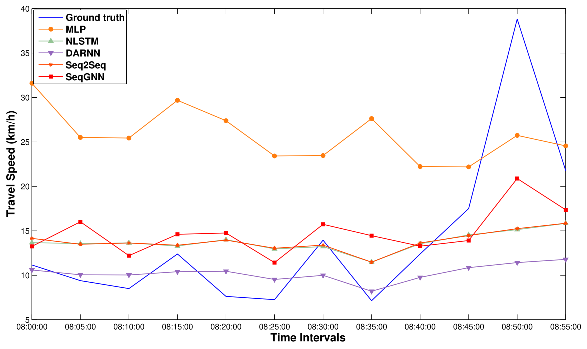

FIGURE 10. Performance of SeqGNN and the baselines.

MAPE of 0.275. Compared with the baseline methods, this represents an improvement of at least 13.6% in MAE, 5.2% in RMSE, and 6.07% in MAPE.

Additionally, we chose a random road network to highlight aspects of the results by comparing the proposed SeqGNN with the baselines. Predictions from 08:00 to 09:00 on Nov. 25, 2017, are shown in Fig. 10. As well as the higher accuracy discussed above, SeqGNN tracks the ground truth more closely. These findings confirm the advantages of SeqGNN in using and unlocking knowledge hidden in the road network topologies and road segment attributes as well as the temporal information in the time series.

VI. CONCLUSIONS AND FUTURE WORK

This paper has described a novel traffic speed prediction model called SeqGNN. Inspired by GNNs, the properties and structure of a road network were considered in developing a model that feeds the whole road topology graph into a neural network that can be trained to forecast traffic speeds. Moreover, we have improved the conventional GNN methods using GRNN blocks, thus transforming the input and output vectors into sequential graphs. Our SeqGNN model reduces the training loss faster than other baselines. We conducted extensive experiments on a real-world, large-scale dataset obtained from Xi'an, China. The results show that our model simultaneously achieves the best performance against four baselines in terms of MAE, MAPE, and RMSE.

In future work, we will investigate other external influential factors such as weather and points of interest to improve the prediction accuracy. The proposed model will also be applied to other spatiotemporal forecasting tasks involving graph structure information, such as bus and subway passenger flow predictions.

REFERENCES

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, "Data-driven intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 4, pp. 1624–1639, Dec. 2011.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," presented at the Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

[3] M. Chen, X. Yu, and Y. Liu, "PCNN: Deep convolutional networks for short-term traffic congestion prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 11, pp. 3550–3559, Nov. 2018.

[4] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," presented at the IEEE Int. Conf. Smart City/SocialCom/SustainCom (SmartCity), Dec. 2015, pp. 153–158.

[5] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, "Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks," *Sensors*, vol. 17, no. 7, p. 1501, 2017.

[6] W. Jiang and L. Zhang, "Geospatial data to images: A deep-learning framework for traffic forecasting," *Tsinghua Sci. Technol.*, vol. 24, no. 1, pp. 52–64, Feb. 2019.

[7] X. Wang, C. Chen, Y. Min, J. He, B. Yang, and Y. Zhang. (2018). "Efficient metropolitan traffic prediction based on graph recurrent neural network." [Online]. Available: https://arxiv.org/abs/1811.00740

[8] P. W. Battaglia *et al.* (2018). "Relational inductive biases, deep learning, and graph networks." [Online]. Available: https://arxiv.org/abs/1806.01261

[9] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," presented at the Int. Conf. Learn. Represent., 2018, pp. 1–16.

[10] N. L. Nihan and K. O. Holmesland, "Use of the box and jenkins time series technique in traffic forecasting," *Transportation*, vol. 9, no. 2, pp. 125–143, 1980.

[11] D. Xia, B. Wang, H. Li, Y. Li, and Z. Zhang, "A distributed spatial–temporal weighted model on mapreduce for short-term traffic flow forecasting," *Neurocomputing*, vol. 179, pp. 246–263, Feb. 2016.

[12] X. Feng, X. Ling, H. Zheng, Z. Chen, and Y. Xu, "Adaptive multi-kernel SVM with spatial-temporal correlation for short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[13] X. Dong, T. Lei, S. Jin, and Z. Hou, "Short-term traffic flow prediction based on XGBoost," presented at the IEEE 7th Data Driven Control Learn. Syst. Conf. (DDCLS), May 2018.

[14] D. K. Duvenaud *et al.*, "Convolutional networks on graphs for learning molecular fingerprints," presented at the Adv. Neural Inf. Process. Syst., 2015.

[15] T. N. Kipf and M. Welling. (2016). "Semi-supervised classification with graph convolutional networks." [Online]. Available: https://arxiv.org/abs/1609.02907

[16] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," presented at the IEEE Int. Joint Conf. Neural Netw. (IJCNN), Jul./Aug. 2005.

[17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.

[18] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. (2015). "Gated graph sequence neural networks." [Online]. Available: https://arxiv.org/abs/1511.05493

[19] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. (2013). "Spectral networks and locally connected networks on graphs." [Online]. Available: https://arxiv.org/abs/1312.6203

[20] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," presented at the CVPR, 2017.

[21] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. (2017). "Neural message passing for quantum chemistry." [Online]. Available: https://arxiv.org/abs/1704.01212

[22] X. Wang, R. Girshick, A. Gupta, and K. He. (2017). "Non-local neural networks." [Online]. Available: https://arxiv.org/abs/1711.07971

[23] S. Mohanty and A. Pozdnukhov, "Graph CNN+ LSTM framework for dynamic macroscopic traffic congestion prediction," presented at the 14th Int. Workshop Mining Learn. Graphs (MLG), 2018.

[24] K. Cho *et al.* (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." [Online]. Available: https://arxiv.org/abs/1406.1078

[25] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," presented at the Adv. Neural Inf. Process. Syst., 2014.

[26] B. Liao *et al.*, "Deep sequence learning with auxiliary information for traffic prediction," presented at the 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2018.

[27] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," presented at the 1st Workshop Deep Learn. Recommender Syst., 2016.

[28] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Map-matching for low-sampling-rate GPS trajectories," presented at the 17th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., 2009.

[29] J. R. A. Moniz and D. Krueger. (2018). "Nested LSTMs." [Online]. Available: https://arxiv.org/abs/1801.10308

[30] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell. (2017). "A dual-stage attention-based recurrent neural network for time series prediction." [Online]. Available: https://arxiv.org/abs/1704.02971

[31] Z. Cui, K. Henrickson, R. Ke, and Y. Wang. (2018). "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting." [Online]. Available: https://arxiv.org/abs/1802.07007

[32] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Graph convolutional recurrent neural network: Data-driven traffic forecasting," Tech. Rep., 2017.

[33] X. Zhou, Y. Shen, Y. Zhu, and L. Huang, "Predicting multi-step city-wide passenger demands using attention-based neural networks," presented at the 11th ACM Int. Conf. Web Search Data Mining, 2018, pp. 736–744.

[34] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," presented at the Int. Conf. Mach. Learn., 2016.

[35] J. Zhang, Y. Zheng, D. Qi, R. Li, and X. Yi, "DNN-based prediction model for spatio-temporal data," presented at the 24th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst., 2016.