Traffic Flow Prediction via Spatial Temporal Graph Neural Network

Xiaoyang Wang
Beijing Jiaotong University
shawnwang@bjtu.edu.cn

Yao Ma
Michigan State University
mayao4@msu.edu

Yiqi Wang
Michigan State University
wangy206@msu.edu

Wei Jin
Michigan State University
jinwei2@msu.edu

Xin Wang
Changchun Institute of Technology
xinwangjlu@gmail.com

Jiliang Tang
Michigan State University
tangjili@msu.edu

Caiyan Jia*
Beijing Jiaotong University
cyjia@bjtu.edu.cn

Jian Yu
Beijing Jiaotong University
jianyu@bjtu.edu.cn

ABSTRACT

Traffic flow analysis, prediction and management are keystones for building smart cities in the new era. With the help of deep neural networks and big traffic data, we can better understand the latent patterns hidden in the complex transportation networks. The dynamic of the traffic flow on one road not only depends on the sequential patterns in the temporal dimension but also relies on other roads in the spatial dimension. Although there are existing works on predicting the future traffic flow, the majority of them have certain limitations on modeling spatial and temporal dependencies. In this paper, we propose a novel spatial temporal graph neural network for traffic flow prediction, which can comprehensively capture spatial and temporal patterns. In particular, the framework offers a learnable positional attention mechanism to effectively aggregate information from adjacent roads. Meanwhile, it provides a sequential component to model the traffic flow dynamics which can exploit both local and global temporal dependencies. Experimental results on various real traffic datasets demonstrate the effectiveness of the proposed framework.

CCS CONCEPTS

• **Theory of computation → Dynamic graph algorithms**; • **Information systems → Spatial-temporal systems**; • **Computing methodologies → Supervised learning**; **Artificial intelligence**.

KEYWORDS

Traffic Prediction, Graph Neural Networks, Spatial Temporal Model, Dynamic, Recurrent Neural Network, Transformer

*Corresponding author.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380186

ACM Reference Format:
Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic Flow Prediction via Spatial Temporal Graph Neural Network. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380186

1 INTRODUCTION

Traffic congestion is a tremendous challenge in current urban construction and management in various countries during the urbanization process. Analyzing and forecasting the dynamic traffic conditions are of great significance to the planning and construction of new roads and transportation management of smart cities in the new era. However, traffic flow forecasting has become increasingly challenging. First, it relies on the volatility and uncertainty of the vehicle flow in the temporal dimension. Vehicle traffic is often periodic in a long period so that it is relatively easier to be summarized and predicted in a long-period case. However, there exist many uncertainties in the cases of short periods, where traditional methods are not applicable. In the case of sudden burstiness such as traffic accidents and special events, it is extremely difficult for the predictive model to quickly adapt to such rapid changes in the traffic flow. Second, the complex relations between roads and vehicles in the spatial dimension also play important roles. For example, the temporal evolution of traffic flow caused by incidents on one road can affect distant roads in the near future. Furthermore, due to the complexities of road crossings or lanes, the interplay between roads is not easily discernible, leading to the difficulty of predicting where the traffic flow will go.

Extensive research has been conducted to solve these aforementioned challenges. Existing methods can be majorly divided into traditional knowledge-driven methods which apply queuing theory and behavior simulators [6], and machine learning methods such as Vector Auto-Regressions (VAR) [25, 28], Support Vector Regression (SVR) [35], Auto-Regressive Integrated Moving Average (ARIMA) model [21, 42, 47] and Kalman filtering [20, 26]. However, these methods rely on the ideal stationary assumption [40] that often do not hold in complicated real traffic dynamics. Recently, Deep Recurrent Neural Networks (RNNs) [9] and its successors such as Long-Short-Term-Memory (LSTM) networks[16] and Gated

Recurrent Unit (GRU) networks [8] have been extensively applied in learning complicated schemes from a huge number of sequence information [11, 29, 38, 45]. However, RNN models treat traffic sequences from different roads as independent data streams. They are not capable of utilizing spatial information from traffic data. The traffic data can be represented as a time series on a road network, where the connections between roads are built according to the spatial proximity between roads. Thus, graph neural network (GNN) models such as T-GCN [49], DCRNN [24], and GaAN [48] have been proposed to capture both the spatial and temporal information. These models typically combine the recurrent neural networks and graph neural networks to model the spatial-temporal relations in traffic data.

Although existing GNN and RNN based models have remarkably improved the prediction performance, they still have limitations. All these methods utilize GNNs to capture the spatial relations between roads, which can be viewed as transforming and aggregating information through the edges in the road network. Hence, the relations between the roads in the network play a crucial role. However, the influence of the traffic on one road to others is dynamic that cannot be simply modeled by the static spatial proximity between them. These relations are much more complicated and dependent on various factors such as the number of lanes, road conditions, vehicle density, population density, and relevant emergent events. Though the attention mechanism is adopted to capture the influence between roads [48], the attention score only depends on the speed information while ignoring other aforementioned factors. Meanwhile, these models adopt recurrent neural networks to capture sequential information on the traffic network. RNNs model the temporal dependency indirectly where the forward and backward signals have to traverse along a long recurrent path in the network. The longer these paths between input and output components, the harder it is to learn long-range dependencies efficiently [15, 39]. Moreover, the traffic information may not be only sequentially dependent in the temporal dimension. For example, road maintenance can be periodical and the impact of an accident can be long-lasting. Hence, it is also important to directly extract the global temporal dependencies.

To address the aforementioned challenges, in this paper, we propose a novel spatial temporal graph neural network for traffic flow prediction. It combines positional graph neural layer, recurrent neural network layer and transformer layer to better capture the complex relations between roads from both spatial and temporal aspects. In summary, the main contributions of this paper are as follows:

- We propose a new Graph Neural Network layer with a position-wise attention mechanism to better aggregate information of traffic flows from adjacent roads;
- We combine a recurrent network and a Transformer layer to capture the local and global temporal dependence;
- We propose a new spatial temporal GNN framework STGNN which is particularly designed for modeling series data with complex topological and temporal dependency; and
- We validate the feasibility and advantages of the proposed framework on real traffic datasets, especially for the short period traffic speed prediction in terms of minutes which is more challenging compared to longer period prediction in terms of days or weeks. Experiments show that our model outperforms state-of-the-art methods significantly.

The rest of the paper is organized as follows. In Section 2, we introduce some preliminary concepts and formalize the traffic prediction problem. In Section 3, we first introduce the overall framework of the Spatial Temporal Graph Neural Network (STGNN) model and demonstrate the intuitions of the model. Then we discuss in detail about the three model components to deal with the spatial and temporal dependencies, separately. Experiments and further analysis are presented in Section 4. In Section 5, we review related work. We conclude the work in Section 6 with future work.

2 PROBLEM STATEMENT

In the problem of traffic flow forecasting, we intend to predict the traffic flow in the future by leveraging historical traffic flow data. Specifically, the historical traffic flow data can be represented as a time series on the traffic network. The traffic network can be denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of N traffic sensor nodes (each road may have multiple sensors at different sections) and $\mathcal{E}$ is a set of edges connecting these nodes. These connections between nodes can also be described by a symmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, with i, j-th element $\mathbf{A}[i, j]$ representing the strength of the relation between nodes v_i and v_j, which is usually measured by the geographical proximity of the two sensors. Note that $A[i, j] = 0$ if there is no close relation between the two nodes in geography. The historical traffic information can be represented as a sequence $\mathcal{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_\tau)$ on the traffic network, where each $\mathbf{Y}_t \in \mathbb{R}^{N \times 1}$ represents the traffic flow information of N nodes at time t. With the aforementioned notations, the problem can be formally stated as:

Given the traffic network $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ and the historical traffic information $\mathcal{Y} = (\mathbf{Y}_1, \ldots, \mathbf{Y}_\tau)$, we aim to build a model f, which can take a new sequence $\mathcal{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_T)$ of length T as input and predict the traffic information for the next T' time steps $\mathcal{X}_{pred} = (\mathbf{X}_{T+1}, \ldots, \mathbf{X}_{T+T'})$.

Specifically, during the training stage, we slide a $T + T'$ window over the historical traffic sequence $\mathcal{Y}$ to generate the training samples to train the model f.

3 THE PROPOSED FRAMEWORK

The proposed spatial temporal graph neural network framework is shown in Figure 1. It mainly consists of three components: 1) the spatial graph neural network (S-GNN) layers, which aim to capture the spatial relations between the roads through the traffic network; 2) the GRU layer, which is to capture the temporal relation sequentially (or local temporal dependency); and 3) the transformer layer, which aims to directly capture the long-range temporal dependence in the sequence (or global temporal dependence). Note that the S-GNN layer is utilized to model the spatial relation between the nodes and it is applied to both the input and the hidden representations of the GRU unit as shown in Figure 1. Both the GRU layer and the transformer layer are used to capture the temporal dependency for each node individually, while they capture the dependency from different perspectives. Next, we first discuss modeling spatial dependency with the S-GNN and then

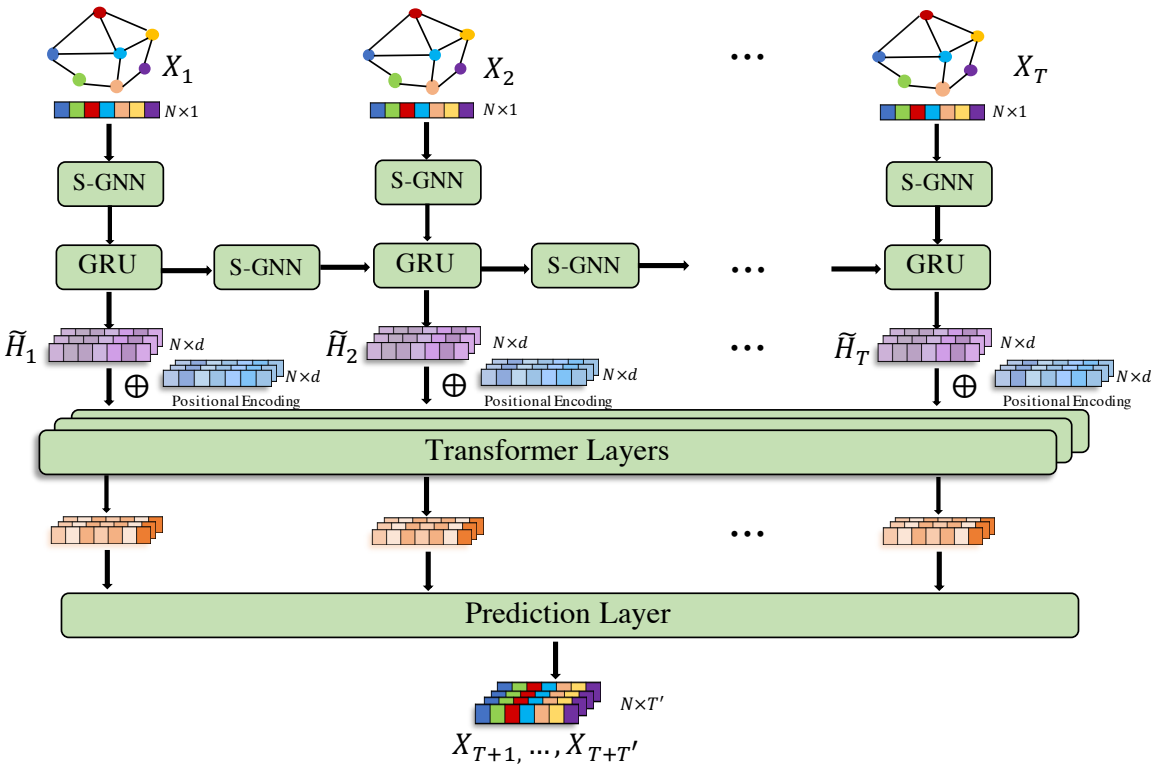


Figure 1: The Proposed Spatial Temporal Graph Neural Network Framework

describe how the GRU layer and transformer layer can be used to capture the temporal dependency. We will briefly summarize the entire framework after introducing these components.

3.1 Modeling the Spatial Dependency

The traffic network $\mathcal{G}$ encodes the relations between the roads. The connected roads in the traffic network are more likely to share similar attributes. Specific to the traffic flow prediction problem, if two roads are located closely, traffic conditions on the roads are more likely to influence each other. Hence, to capture the spatial relations, we adopt the Graph Neural Networks model proposed in [19] to transform and propagate information through the network. Specifically, given an input information on $\mathbf{X}_{in} \in \mathbb{R}^{N \times d_{in}}$ on the network, the output $\mathbf{X}_{out} \in \mathbb{R}^{N \times d_{out}}$ can be generated as follows:

$$\mathbf{X}_{out} = \sigma(\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}_{in}\mathbf{W}) \tag{1}$$

where σ is a non-linear activation function and we adopt $ReLU(\cdot)$ in this work. $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ is the refined adjacency matrix with I_N the N-dimensional identity matrix, and $\tilde{\mathbf{D}}$ is the refined degree matrix $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$. $\mathbf{W}$ are the parameters to be learned. For convenience, we summarize the operation in eq. (1) as follows:

$$\mathbf{X}_{out} = f_g(\mathbf{A}, \mathbf{X}_{in}). \tag{2}$$

Note that this operation will be frequently used in different parts of the proposed framework, which we will discuss in a more detailed way later.

In the above formulation, the operation is purely based on the network information, which is built upon the geographical proximity between sensors on roads. However, the influence between roads can be much more complex. There are many other factors such as the number of lanes connecting different roads, road conditions, vehicle density, population density and emergent events that can affect the traffic flow on the roads. Hence, the information of neighboring nodes should not be evenly aggregated to a given central node when performing the aggregation in eq. (1). Recently, GaAN [48] tries to utilize the attention mechanism to model the complex relations between the roads. However, when calculating the attention scores, GaAN only utilizes the speed information. Ideally, we could use the aforementioned factors to calculate the attention score. However, these factors are not always available. Furthermore, there could be some other factors affecting the relations between nodes that we are not aware of. Hence, in this paper, we propose to learn a positional representation to capture these factors for each node. Specifically, for each node v_i, we try to learn a latent positional representation $\mathbf{p}_i$. We then model the pair-wise relations between any road nodes as:

$$\mathbf{R}[i,j] = \frac{exp(\phi(Score(\mathbf{p}_i, \mathbf{p}_j)))}{\sum_{k=1}^{N} exp(\phi(Score(\mathbf{p}_i, \mathbf{p}_k)))} \tag{3}$$

where $Score()$ is a relation score function modeled with dot product as follows:

$$Score(\mathbf{p}_i, \mathbf{p}_j) = \mathbf{p}_i^T \mathbf{p}_j \tag{4}$$

where $\mathbf{W_1}$ and $\mathbf{W}_2$ are two transformation matrices to be learned. We further use a mask to sparsify the relation matrix $\mathbf{R}$ to reduce the computational complexity:

$$mask(\mathbf{R}) = \begin{cases} \mathbf{R}_{ij}, & \text{if } \tilde{\mathbf{A}}_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Then the GCN operation can be performed over the newly learned relation matrix $mask(\mathbf{R})$:

$$\mathbf{X}_{out} = \sigma(\tilde{\mathbf{D}}_R^{-1/2}\tilde{\mathbf{R}}\tilde{\mathbf{D}}_R^{-1/2}\mathbf{X}_{in}\mathbf{W}^{(l)}). \tag{6}$$

where $\tilde{\mathbf{R}} = mask(\mathbf{R}) + \mathbf{I}_N$ and $\tilde{\mathbf{D}}_R$ is the degree matrix for $\tilde{\mathbf{R}}$. For convenience, we summarize the operation in eq. (6) as:

$$\mathbf{X}_{out} = f_a(\mathbf{A}, \mathbf{X}_{in}). \tag{7}$$

Instead of the operation in (1), we use the operation in (7) to capture the spatial relations.

3.2 Modeling the Temporal Dependency

To capture the temporal dependency, we adopt the gated recurrent unit (GRU) [8] to process the sequence information. We keep a hidden representation for each time step, which will be used to control the information flow to the next time step and serve as the output of the current time step. Note that the GRU operation is applied to each node individually, and the parameters of the GRUs for all the nodes are shared with each other. To incorporate the spatial relations while processing the sequence, we apply the modified GCN operation in eq. (7) to both the input and the hidden representation of the GRU. Specifically, at time step t, given the input $\mathbf{X}_t$ and the hidden representations from the previous step $\mathbf{H}_{t-1}$, we apply the modified GCN operation to both of them as follows:

$$\begin{aligned} \tilde{\mathbf{X}}_t &= f_a(\mathbf{A}, \mathbf{X}_t), \\ \tilde{\mathbf{H}}_{t-1} &= f_a(\mathbf{A}, \mathbf{H}_{t-1}). \end{aligned} \tag{8}$$

Then, for each node v_i at time step t, the operation of the GRU can be expressed as follows:

$$\begin{aligned} z_t &= \sigma_z(\mathbf{W}_z\tilde{\mathbf{X}}_t[i,:] + \mathbf{U}_z\tilde{\mathbf{H}}_{t-1}[i,:] + \mathbf{b}_z), \\ r_t &= \sigma_r(\mathbf{W}_r\tilde{\mathbf{X}}_t[i,:] + \mathbf{U}_r\tilde{\mathbf{H}}_{t-1}[i,:] + \mathbf{b}_r), \\ \hat{\mathbf{H}}_t[i,:] &= tanh\left(\mathbf{W_h}\tilde{\mathbf{X}}_\mathbf{t}[\mathbf{i},:] + \mathbf{U_h}(\mathbf{r_t} \odot \mathbf{U_h}\tilde{\mathbf{H}}_\mathbf{t-1}[\mathbf{i},:]) + \mathbf{b_h}\right), \\ \mathbf{H}_t[i,:] &= (1 - z_t)\tilde{\mathbf{H}}_{t-1}[i,:] + z_t \odot \tilde{\mathbf{H}}_t[i,:]. \end{aligned} \tag{9}$$

where $\odot$ denotes the element-wise multiplication, $\mathbf{W}_z, \mathbf{W}_r, \mathbf{W}_h$, $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_h$ are the parameters to be learned and $\mathbf{H}_t[i,:]$ is the output of the current time step, which also serves as the input to the next time step.

GRU can help capture the local temporal information. However, in the traffic forecasting problem, the temporal information may not only be sequentially dependent. Hence, it is important to capture the global temporal information for traffic speed forecasting. Thus, after the GRU layer, we adopt a transformer layer [39] to directly capture the global dependency.

Similar to the GRU layer, the transformer layer is also applied to each node individually. Specifically, for node v_i, we take the output sequence $(\mathbf{H}_1[i,:], \ldots, \mathbf{H}_T[i,:])$ from GRU as the input for the transformer. As shown in Figure 2, a transformer layer consists

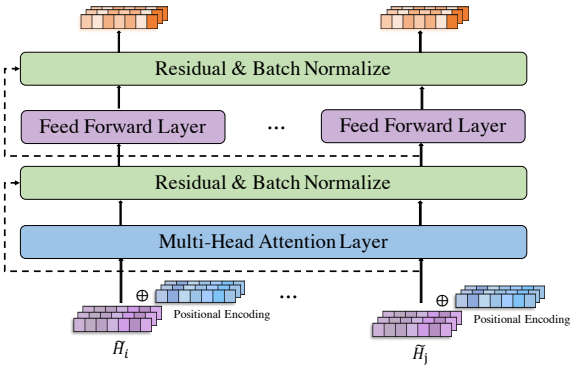


Figure 2: The Transformer Layer

of a multi-head attention layer, a shared feed-forward neural work layer and batch normalization layers between them. The multi-head attention layer [39] is built upon the dot product attention mechanism. Next, we first introduce single head attention and then describe how we can extend it to multi-head attention. In the transformer, the element in the position i of the sequence attends to all the elements in the sequence. Specifically, the inputs of the attention function consist of queries, keys with dimension d_k and values with dimension d_v of all the positions in the sequence. We compute the dot products of a given query with all keys, divide each by $\sqrt{d_k}$ and then apply a softmax function to obtain the attention scores for each position [39]. These attention scores are then served as the weights to aggregate information from the corresponding values. In practice, we compute the attentions for the queries of all positions simultaneously as follows:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\mathbf{V}), \tag{10}$$

where $\mathbf{Q}, \mathbf{K} \in \mathbb{R}^{T \times d_k}$ and $\mathbf{V} \in \mathbb{R}^{T \times d_v}$ denote the queries, keys and values for all the nodes. Specifically, the i-th row of $\mathbf{Q}$ denotes the query corresponding to the position i in the sequence. Here, we obtain $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ all from the output $(\mathbf{H}_1[i,:], \ldots, \mathbf{H}_T[i,:])$ of the GRU layer. Specifically, we first organize the output sequence from the GRU layer in a matrix form by stacking $(\mathbf{H}_1[i,:], \ldots, \mathbf{H}_T[i,:])$ row-wisely according to the sequential order. We denote the obtained matrix as $\mathbf{H}^{v_i} \in \mathbb{R}^{T \times d}$, where the superscript v_i indicates that it is corresponding to the node v_i. We then linearly project the matrix $\mathbf{H}^{v_i}$ into queries, keys and values as follows:

$$\mathbf{Q}^{v_i} = \mathbf{H}^{v_i}\mathbf{W}^Q, \ \mathbf{K}^{v_i} = \mathbf{H}^{v_i}\mathbf{W}^K, \ \mathbf{V} = \mathbf{H}^{v_i}\mathbf{W}^V, \tag{11}$$

where $\mathbf{W}^Q \in \mathbb{R}^{d \times d_k}$, $\mathbf{W}^K \in \mathbb{R}^{d \times d_k}$ and $\mathbf{W}^V \in \mathbb{R}^{d \times d_v}$ are the projection matrices to be learned, which are shared by all the nodes. The attention function can be rewritten as:

$$Attention(\mathbf{H}^{v_i}) = \text{softmax}\left(\frac{(\mathbf{H}^{v_i}\mathbf{W}^Q)(\mathbf{H}^{v_i}\mathbf{W}^K)^T}{\sqrt{d_k}}\mathbf{H}^{v_i}\mathbf{W}^V\right). \tag{12}$$

Instead of performing a single attention function, multi-head attention is preferred as it can jointly aggregate information from

different representation sub-spaces thus to enhance the presentation capabilities of the model [39]. In the multi-head attention, a total K sets of projection matrices are utilized to project $\mathbf{H}^{v_i}$ to different K sets of queries, keys and values. The result of the multi-head attention is a concatenation of the output of each individual attention function. Specifically, it can be expressed as follows:

$$\begin{aligned} Multihead(\mathbf{H}^{v_i}) &= Concat(head_1, \dots, head_S)\mathbf{W}^O; \\ \text{where } head_s &= attention_s(\mathbf{H}^{v_i}) \\ &= \text{softmax}\left(\frac{(\mathbf{H}^{v_i}\mathbf{W}_s^Q)(\mathbf{H}^{v_i}\mathbf{W}_s^K)^T}{\sqrt{d_k}}\mathbf{H}^{v_i}\mathbf{W}_s^V\right). \end{aligned} \tag{13}$$

$\mathbf{W}_s^Q, \mathbf{W}_s^K$ and $\mathbf{W}_s^V$ are the projection matrices for the s-th attention head and $\mathbf{W}^O$ is another linear output projection.

Note that the attention mechanism in the transformer layer ignores the relative position in the sequence as it treats different positions equally while calculating the attention function. To ensure the transformer layer to be aware of the relative position of $\mathbf{H}_t[i,:]$ in the entire sequence, we adopt a position encoding e_t for each position as in [39]. Specifically, we generate new representations $\mathbf{H}'_t[i,:]$ by combining $\mathbf{H}_t[i,:]$ together with the position encoding as follows:

$$\mathbf{H'}_t[i,:] = \mathbf{H}_t[i,:] + \mathbf{e_t}, \tag{14}$$

where e_t is defined as

$$\mathbf{e_t} = \begin{cases} sin(t/10000^{2i/d_{model}}), & \text{if } t = 0, 2, 4\dots \\ cos(t/10000^{2i/d_{model}}), & \text{otherwise.} \end{cases} \tag{15}$$

In practice, instead of the sequence $(\mathbf{H}_1[i,:], \dots, \mathbf{H}_T[i,:])$ directly from the GRU layer, we use the one with positional encoding $(\mathbf{H}'_1[i,:], \dots, \mathbf{H}'_T[i,:])$ as input for the transformer layer. After the multi-head attention layer, the output states will be passed to a point-wise feed-forward neural network layer. As shown in Figure 2, after each sub-layer, there is a residual connection [14] and layer normalization [2]. Finally, we arrive at the output of the transformer layer, which can be denote as $\mathbf{H}_{out}^{v_i} \in \mathbb{R}^{T\times d}$.

3.3 The prediction layer

After the transformer layer, we take the $\{\mathbf{H}_{out}^{v_i} | v_i \in \mathcal{V}\}$ as input and use a multi-layer feed-forward network to predict the traffic speed of future periods.

During the training, we first split the historical data $\mathcal{Y}$ into training part $\mathcal{Y}_{train}$, test part $\mathcal{Y}_{test}$ and validation part $\mathcal{Y}_{valid}$ (We will discuss the details of the split in the experimental setting section). We then slide a window of length $T + T'$ over the training data $\mathcal{Y}_{train}$ to generate the training sequences. For each generated training sequence, the first T elements are used as input and the rest T' elements are used as the ground truth. Then we take the mean absolute error as the loss, which can be expressed as follows:

$$\mathcal{L} = \sum_{t=1}^{|\mathcal{Y}_{train}|-T'} d(f(\mathbf{Y}_{t+1}, \dots \mathbf{Y}_{t+T}), (\mathbf{Y}_{t+T+1}, \dots, \mathbf{Y}_{t+T+T'})) \tag{16}$$

where

$$\begin{aligned} &d(f(\mathbf{Y}_{t+1}, \dots \mathbf{Y}_{t+T}), (\mathbf{Y}_{t+T+1}, \dots, \mathbf{Y}_{t+T+T'})) \\ &= \sum_{i=1}^{T'} \|\hat{\mathbf{Y}}_{t+T+i} - \mathbf{Y}_{t+T+i}\|_1, \end{aligned} \tag{17}$$

with

$$(\hat{\mathbf{Y}}_{t+T+1}, \dots, \hat{\mathbf{Y}}_{t+T+T'}) = f(\mathbf{Y}_{t+1}, \dots \mathbf{Y}_{t+T})). \tag{18}$$

Here $f()$ is the predictive model, which can be trained by minimizing the loss eq. (16).

3.4 The framework

We now briefly summarize the entire framework. As shown in Figure 1, we utilize the GRU layer to capture the sequential temporal relations where both the input and the hidden states are filtered using the S-GNN layers. The S-GNN layer helps capture the spatial relations between the roads by transforming and propagating information through the traffic network. A transformer layer is followed with the GRU layer to capture the global temporal dependency. Finally, the output of the transformer layer is used to make the predictions for future time steps.

4 EXPERIMENT

In this section, we conduct experiments to demonstrate the effectiveness of the proposed framework. We first introduce the datasets used for the experiments and the metrics to evaluate the performance. We then introduce experimental settings, following which, the experiment results with discussion are presented. Finally, we conduct a case study to further understand the framework.

4.1 Datasets

We perform the experiments on two real-world traffic datasets. We briefly describe them as follows:

- **METR-LA** This traffic dataset contains high-resolution spatio-temporal transportation data, which includes traffic speed or volume from loop-detectors located on the LA County road network [17]. Specifically, the traffic information is recorded at the rate of every 5 minutes. We use data from 207 sensors over a period of 4 months from Mar 1st 2012 to Jun 30th 2012 [24]. The total number of 5 minute time slices for each sensor is 34,272.
- **PEMS-BAY** This dataset comes from the California Department of Transportation (Caltrans) Performance Measurement System (PeMS) [7]. We use data from 325 sensors in the Bay Area over a period of 6 months from Jan 1st 2017 to May 31st 2017 [24]. It contains 52,116 of 5-minute slices for each sensor.

Some of the key statistics of the two datasets are summarized in Table 1.

4.2 Baseline Methods

To demonstrate the effectiveness of the proposed model, we compare it with some traditional methods and representative methods based on GNNs and RNNs. The baselines are briefly summarized as follows:

Table 1: Dataset Statistics

Dataset	Sensors	Length	Unit	Size
META-LA	207	4 month	5 min	34,272
PEMS-BAY	325	6 month	5 min	52,116

- **HA:** Historical Average models the traffic flow as a periodic process (in days or weeks) and uses the weighted average of previous periods as the prediction for future periods.
- **ARIMA:** Auto-regressive Integrated Moving Average model [21, 42, 47] with Kalman filter is widely used in time series analysis. It fits time series data to predict future points in the series.
- **VAR:** Vector Auto-Regression [25, 28] assumes that the passed time series is stationary and estimates relations between the time series and their lagged values.
- **SVR:** Support Vector Regression [35] uses a support vector machine to do regression on the traffic sequence. All the above traditional statistical methods either are limited by stability assumptions or lack of capacity for modeling spatial or long-term dependence.
- **FNN and FL-LSTM:** Feed Forward Neural network and Fully Connected Long Short Term Memory network [36] can be directly used to deal with traffic sequences, but they can only model temporal dependencies.
- **T-GCN:** Temporal GCN [49] simply combines the graph convolutional network and the gated recurrent unit to do the traffic forecasting.
- **DCRNN:** Diffusion Convolutional Recurrent Neural Network [24] models the traffic flow as a diffusion process. It captures the spatial dependency using bidirectional random walks on the graph, and the temporal dependency using the encoder-decoder architecture. But it still relies on the road network to propagate information which limits the model's flexibility.
- **GaAN:** Gated Attention Networks [48] use a new multi-head attention-based aggregator with additional gates on the attention heads, which only uses the traffic speed information. For the temporal information they also directly apply GRU.
- **STGNN w/o GRU:** STGNN w/o GRU is a variant of our proposed model, where the GRU layers are removed and the output of the S-GNN layers is directly input into the transformer layer.
- **STGNN w/o Transformer:** STGNN w/o GRU is a variant of our proposed model, where the Transformer layer is removed and the output of the GRU layers is directly used for the prediction.

4.3 Experimental Settings and Evaluation Metrics

We follow the experimental settings in DCRNN [24], where the dataset is split into three parts. Specifically, 70% of the data is used for training, 20% is used for test and the remaining 10% for validation. After splitting the dataset into the three parts, we generate sequence samples by sliding a window of width $T + T'$. Specifically, each sample sequence consists of 24 time steps with intervals of 5-minutes, where the first 12 time steps are treated as the input while the rest 12 time steps are regarded as the ground truth. All the baseline methods follow the best parameters and results reported in their papers. For our models, we set all the hidden dimensions to 64, batch size to 64, and attention heads to 4. The number of recurrent and transformer layers are both set to 1. Adam optimizer with learning rate decay is employed to train the model. The maximum training iteration is set to 1,000. We use grid search and early stopping to tune the learning rate, weight decay, and training epochs.

In order to measure the prediction performance of different methods, we adopt RMSE, MAE and MAPE as metrics. As the traffic conditions of different areas may be diverse, applying absolute error might indicate that the model is overfitting to relatively simple samples, while squared error gives more punishment to difficult and unpredictable ones which can better show the performance under complex situations. Next, we briefly describe these metrics as follows:

- Root Mean Squared Error (RMSE) is defined as the rooted average squared difference between the predicted values and the ground truth.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}, \tag{19}$$

  where N denotes the number of samples, and Y_i and $\hat{Y}_i$ denote the ground truth and the prediction of the i-th sample, respectively.
- Mean absolute error (MAE) is defined as the average of the absolute errors.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i|. \tag{20}$$

- Mean absolute percentage error (MAPE) is defined as:

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right|, \tag{21}$$

  which considers the percentage of errors to ground truth values.

For the MAE, RMSE and MAPE metrics, smaller values indicate the better prediction performance.

4.4 Experiment Results

The results are shown in Tables 2 and 3 for **METR-LA** and **PEMS-BAY**, respectively. We averaged the prediction results of 15 minutes, 30 minutes and 60 minutes ahead forecasting and the best result for each setting is highlighted. We can make the following observations from tables.

- HA uses the summary of historical daily periodic patterns as the prediction without considering the traffic conditions of the immediate past. As the data sequence is generated by sliding a window from original traffic data, the result of HA is invariant to the increases in the forecasting horizon. It relatively performs well when we analyze long term traffic

Table 2: Traffic Flow Prediction on the METR-LA Dataset

Methods	15 minutes			30 minutes			60 minutes		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	4.16	7.80	13.0%	4.16	7.80	13.0%	4.16	7.80	13.0%
ARIMA	3.99	8.21	9.6%	5.15	10.45	12.7%	6.90	13.23	17.4%
VAR	4.42	7.89	10.2%	5.41	9.13	12.7%	6.52	10.11	15.8%
SVR	3.99	8.45	9.3%	5.06	10.87	12.1%	6.72	13.76	16.7%
FNN	3.99	7.94	9.9%	4.23	8.17	12.9%	4.49	8.69	14.0%
FC-LSTM	3.44	6.30	9.6%	3.77	7.23	10.9%	4.37	8.96	13.2%
T-GCN	3.03	5.26	7.81%	3.52	6.12	9.45%	4.30	7.31	11.8
DCRNN	2.77	5.38	7.3%	3.15	6.45	8.8%	3.60	7.59	10.5%
GaAN	2.71	5.24	6.99%	3.12	6.36	8.56%	3.64	7.65	10.62%
STGNN w/o GRU	2.69	5.18	6.79%	3.06	6.12	8.07%	3.65	7.43	10.21%
STGNN w/o Transformer	2.65	5.14	6.74%	3.07	6.17	8.13%	3.63	7.32	10.33%
Our STGNN	**2.62**	**4.99**	**6.55%**	**2.98**	**5.88**	**7.77%**	**3.49**	**6.94**	**9.69%**

Table 3: Traffic Flow Prediction on the PEMS-BAY Dataset

Methods	15 minutes			30 minutes			60 minutes		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HA	2.88	5.59	6.8%	2.88	5.59	6.8%	2.88	5.59	6.8%
ARIMA	1.62	3.3	3.5%	2.33	4.76	5.4%	3.38	6.5	8.3%
VAR	1.74	3.16	3.6%	2.32	4.25	5.0%	2.93	5.44	6.5%
SVR	1.85	3.59	3.8%	2.48	5.1	5.5%	3.28	7.08	8.0%
FNN	2.2	4.42	5.19%	2.3	4.63	5.43%	2.46	4.98	5.89%
FC-LSTM	2.05	4.19	4.8%	2.20	4.55	5.2%	2.37	4.96	5.7%
T-GCN	1.50	2.83	3.14	1.73	3.40	3.76%	2.18	4.35	4.94%
DCRNN	1.38	2.95	2.9%	1.74	3.97	3.9%	2.07	4.74	4.9%
GaAN	-	-	-	-	-	-	-	-	-
STGNN w/o GRU	1.27	2.57	2.70%	1.56	3.46	3.43%	1.98	4.60	4.53%
STGNN w/o Transformer	1.23	2.59	2.46%	1.55	3.53	3.25%	2.03	4.61	4.55%
Our STGNN	**1.17**	**2.43**	**2.34%**	**1.46**	**3.27**	**3.09%**	**1.83**	**4.20**	**4.15%**

condition. But it is not sufficient to handle the dynamic changes or sudden burstiness of traffic flow.

- Traditional statistical methods such as ARIMA, VAR and SVR perform well on PEMS-BAY for short term prediction. Especially, they are even better than the simple fully connected neural network model in shorter horizons. However, they perform worse than neural networks based models when predicting long-term traffic because they are weak in handling non-stationary time series data. On METR-LA, for both short-term and long-term predictions, statistical methods perform worse than neural networks based methods. This is likely because METR-LA contains some missing data which is more difficult for traditional methods to tackle.
- Recently proposed spatial-temporal models can overcome the above difficulties to some extent. They achieve noticeable improvements compared with the traditional methods and simple neural networks based method. Comparing GaAN with DCRNN, on METR-LA, GaAN can perform slightly better than DCRNN for 15 and 30 minutes prediction, while for 60 minutes prediction, GaAN cannot get better results than DCRNN. This may indicate that simply considering traffic information does not offer sufficient capability to determine complex road relations, although GaAN uses more complicated gated multi-head attention mechanism. Note that due to the unavailability of codes, we do not provide the results of GaAN on PEMS-BAY.
- The proposed framework outperforms all the baseline models in all settings. Compared with best baseline results for each setting on METR-LA, the proposed model achieves approximately 4.5% higher performance in terms of MAE, 5.1% higher in terms of RMSE and 9.2% higher in terms of MAPE. On PEMS-BAY, the improvements are even more significant. In terms of MAE, RMSE and MAPE, we get 15.6%, 14.1%, 19.3% improvements respectively compared with the best baseline results. These results demonstrate the effectiveness of our proposed model.
- Both variants of the proposed model do not perform as well as STGNN, which indicates that all the components

of STGNN are important to the framework. While these variants do not perform as well as STGNN, they still outperform most of the baselines, which further indicates the significance of the proposed components.

4.5 Case study

In this subsection, we carry out a case study to show how the proposed model can handle complex traffic situations. Specifically, we randomly select a traffic sensor node from each dataset. We then visualize the 15 minutes ahead prediction results of these two nodes (one from METR-LA and the other from PEMS-BAY) in Figure 3 and Figure 4, respectively. Specifically, Figure 3(a)-Figure 3(d) show how the prediction results get improved as the number of training epochs increases for the node sampled from METR-LA. Figure 4(a)-Figure 4(d) show the prediction results for the node sampled from PEMS-BAY.

As shown in Figure 3, the model does not perform well at the early training stage (after 1 epoch). After 5 epochs, the model can perform quite well but still cannot fit some abrupt changes well (the one around 175 time intervals). When performing even more training epochs, the model keeps improving its predictive ability and it can almost perfectly fit the ground truth after 100 epochs. Furthermore, there are some missing values (denoted as zeros in the dataset) around the 100-th time interval. After 100 training epochs, the model is able to handle these missing values and bridge the gap of missing values with reasonable predictions.

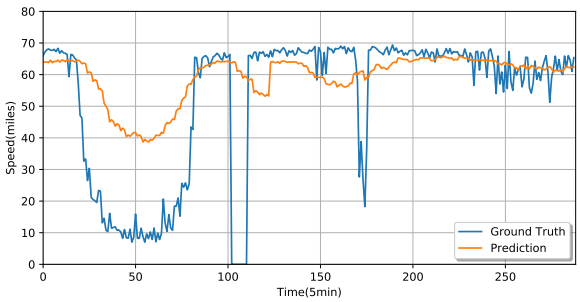
For the node sampled from PEMS-BAY as shown in Figure 4, we can observe that the ground truth curve is quite complicated especially at the end. Again, the model can quickly learn to perform reasonable predictions in these complicated situations, which indicates that the proposed model can handle complex traffic flow dynamics.

5 RELATED WORK

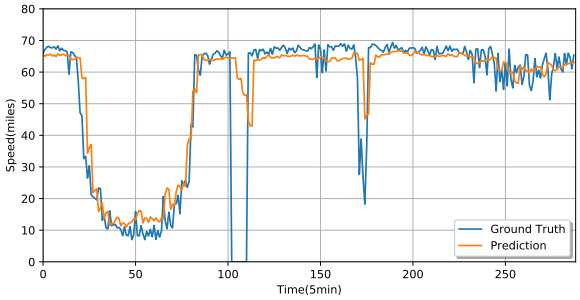
Traffic prediction [30] is more challenging since the number of vehicles is continuously growing with the rapid urbanization process all over the world.

Classic methods in the literature can be categorized as knowledge-driven methods and data-driven methods. Knowledge-driven methods are based on queuing theory and behavior simulators [6], but they often require a wealth of prior knowledge and can not handle unpredictability or complex factors. The data-driven methods can mainly be divided into traditional machine learning methods and recent deep neural network learning methods. However, traditional autoregressive models like Vector Auto-Regressions (VAR) [25, 28], Support Vector Regression (SVR) [35] are not adequate to handle complicated time series because they require the time series to be stationary. Auto-Regressive Integrated Moving Average (ARIMA) model [21, 42, 47] is generalized to handle non-stationary series but still can not perform well on series data with turning points or long-term dependencies.

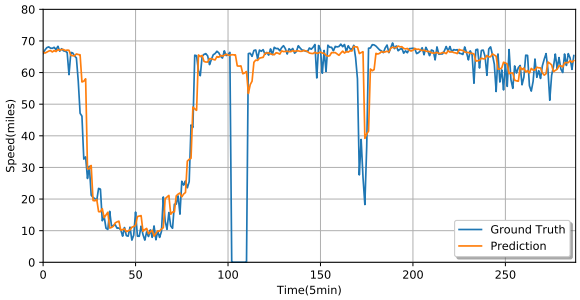
Deep Recurrent Neural Network [9] and its successors such as Long-Short-Term-Memory (LSTM) networks [16] and Gated Recurrent Unit (GRU) networks [8] are proposed to solve language modeling problems. They can be directly used on traffic datasets and dramatically improve the model ability to learn complicated schemes



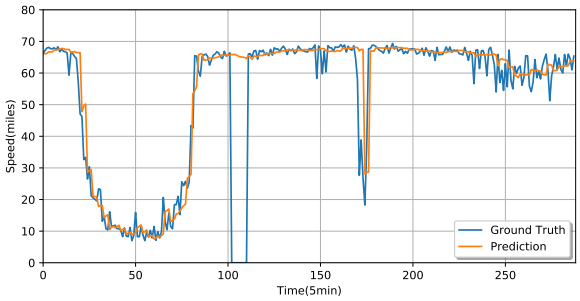
(a) 1 Epoch



(b) 5 Epochs

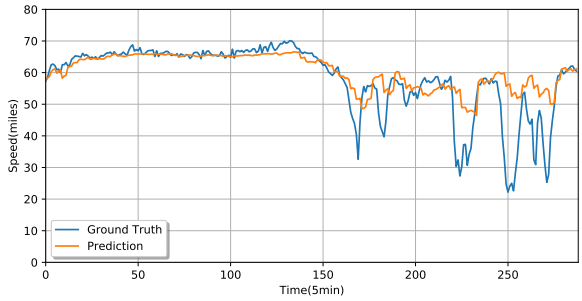


(c) 10 Epochs

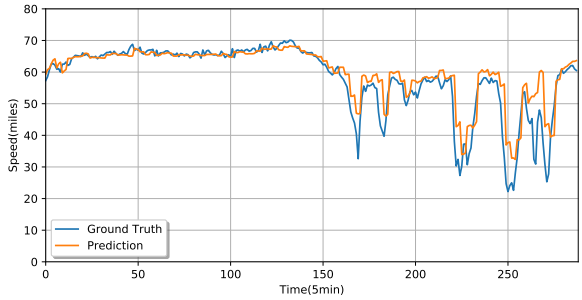


(d) 100 Epochs

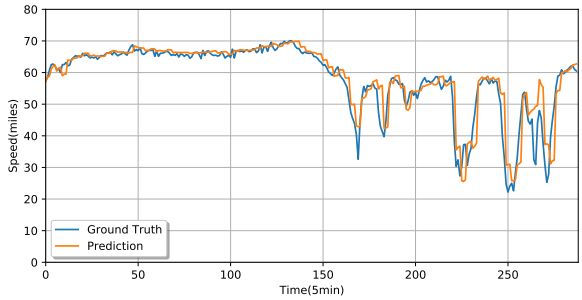
Figure 3: Visualization of Prediction Results on METR-LA



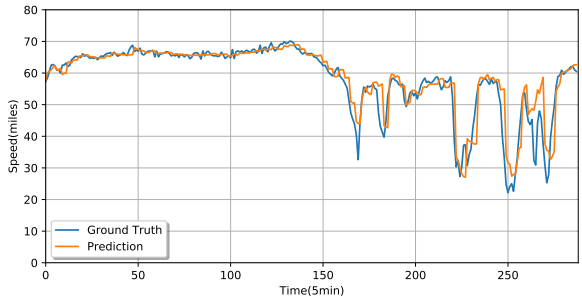
(a) 1 Epoch



(b) 5 Epochs



(c) 10 Epochs



(d) 100 Epochs

Figure 4: Visualization of Prediction Results on PEMS-BAY

from a huge number of sequence information [11, 29, 38, 45]. However, simple RNN models treat traffic sequences of different roads as independent data streams. They are not capable to utilize spatial information of traffic data. Transformer with self-attention mechanism is proved to be more effective than recurrent networks in various areas [27, 32, 50]. It applies multi-head attention mechanism to directly learn dependencies between each pair of input and output positions without any latency or loss.

Graph Neural Network is widely used in modeling the underlying relationships of non-Euclidean data [43]. It can be generally categorized into recurrent graph neural networks (RecGNNs) [34], convolutional graph neural networks (ConvGNNs) [4], graph autoencoders (GAEs) [5], and spatial-temporal graph neural networks. On the other hand, Convolutional Neural Networks [22] and generalized Graph Convolution Neural Networks (GCN) [19] provide new insights to capture spatial information but they can not directly deal with time series problems. Many spatial-temporal models based on graph neural networks are proposed to deal with different spatial-temporal problems such as action recognition [44, 44] and traffic prediction [24, 48, 49]. Specifically, T-GCN [49] combines graph convolutional network [19] and gated recurrent unit to learn information based on topological structures and traffic dynamic patterns. But in the GCN layer, as it aggregates node features based on input adjacent matrix, the relation between road nodes is fixed and it may limit the model's ability to adapt to realistic road conditions. DCRNN [24] proposes diffusion convolution to model the spatial dependency as a diffusion process on a bidirectional graph. But diffusion operation still relies on the road network structure to propagate information, which can't truly represent the actual road connections. GaAN [48] uses gated multi-head attention as a graph aggregator to explore multiple representation sub-spaces between the center node and its neighbors. But it only focuses on the traffic flow information which is changing over time. More importantly, it is only partially related to road structure, not considering the complex realities of the road.

Network Representation Learning draws a lot of attention in recent years [46]. It aims to learn low-dimensional distributed representations for network nodes which then can be used in a wide variety of applications, such as node classification [13], community detection [23], link prediction [37], recommendation [18], knowledge graph [41], and so on. The idea of random walk [33] and diffusion process [12] also inspire researchers to learn better representations for entities by capturing spatial dependencies [1, 24] in the field of traffic forecasting. But traffic networks are different from other real-life networks like social networks [3] and citation networks [10]. Those networks can often be interpreted by six degrees of separation theory [31], i.e., any node is connected to any other node through a path of length no greater than six. On the contrary, in transportation networks, two distant road nodes can never be directly connected, but indirectly connected by other road nodes between them. Traffic networks are flat in space, and their average shortest distance between any two nodes is very large. These differences may shed some light on research in related fields.

6 CONCLUSION

In this paper, we propose a new spatial-temporal graph neural network framework to address the traffic flow prediction problem. We use graph neural networks with a position-wise attention mechanism to capture the spatial dependencies of traffic nodes in the urban road network. At the same time, gated recurrent neural network with transformer layer is used to capture both local and global information of traffic flow sequences in the temporal dimension. Experimental results demonstrate the effectiveness and superiority of the proposed framework in solving traffic prediction problems. As we mentioned above, traffic networks and other kinds of networks such as social networks and biological networks, have many differences in structural and dynamic characteristics. It would be ill-considered to directly apply this kind of model to other network structures or application scenarios. In the future, we plan to further analyze the characteristics and dynamics of different networks, and try to retain the advantages of this spatial-temporal model to social network analysis and prediction tasks.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (61876016, 61632004) and National Key R&D Program of China (2018AAA0100302), the National Science Foundation of United States under IIS1928278, IIS1715940, IIS1845081 and IIS1907704, and a fellowship from China Scholarship Council.

REFERENCES

[1] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1993–2001.
[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).
[3] Albert-Laszlo Barabasi. 2003. *Linked: How everything is connected to everything else and what it means*. Plume.
[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).
[5] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*.
[6] Ennio Cascetta. 2013. *Transportation systems engineering: theory and methods*. Vol. 49. Springer Science & Business Media.
[7] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.
[8] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[9] Jerome T Connor, R Douglas Martin, and Les E Atlas. 1994. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks* 5, 2 (1994), 240–254.
[10] Ergin Elmacioglu and Dongwon Lee. 2005. On six degrees of separation in DBLP-DB and more. *ACM SIGMOD Record* 34, 2 (2005), 33–40.
[11] Rui Fu, Zuo Zhang, and Li Li. 2016. Using LSTM and GRU neural network methods for traffic flow prediction. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, 324–328.
[12] Adrien Guille, Hakim Hacid, Cecile Favre, and Djamel A Zighed. 2013. Information diffusion in online social networks: A survey. *ACM Sigmod Record* 42, 2 (2013), 17–28.
[13] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).
[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.
[15] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.
[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[17] HV Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big data and its technical challenges. *Commun. ACM* 57, 7 (2014), 86–94.
[18] Meng Jiang, Peng Cui, Fei Wang, Qiang Yang, Wenwu Zhu, and Shiqiang Yang. 2012. Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 1422–1431.
[19] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[20] Selvaraj Vasantha Kumar. 2017. Traffic flow prediction using Kalman filtering technique. *Procedia Engineering* 187 (2017), 582–587.
[21] S Vasantha Kumar and Lelitha Vanajakshi. 2015. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *European Transport Research Review* 7, 3 (2015), 21.
[22] Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.
[23] Ye Li, Chaofeng Sha, Xin Huang, and Yanchun Zhang. 2018. Community detection in attributed graphs: an embedding approach. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
[24] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926* (2017).
[25] Marco Lippi, Matteo Bertini, and Paolo Frasconi. 2013. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems* 14, 2 (2013), 871–882.
[26] Hui Liu, Hong-qi Tian, and Yan-fei Li. 2012. Comparison of two new ARIMA-ANN and ARIMA-Kalman hybrid methods for wind speed prediction. *Applied Energy* 98 (2012), 415–424.
[27] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198* (2018).
[28] Helmut Lütkepohl. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.
[29] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2014. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2014), 865–873.
[30] Attila M Nagy and Vilmos Simon. 2018. Survey on traffic prediction in smart cities. *Pervasive and Mobile Computing* 50 (2018), 148–163.
[31] Mark Ed Newman, Albert-László Ed Barabási, and Duncan J Watts. 2006. *The structure and dynamics of networks*. Princeton university press.
[32] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. *arXiv preprint arXiv:1802.05751* (2018).
[33] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.
[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.
[35] Alex J Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing* 14, 3 (2004), 199–222.
[36] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.
[37] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*. 2071–2080.
[38] Claudia Ulbricht. 1994. Multi-recurrent networks for traffic forecasting. In *AAAI*. 883–888.
[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[40] Wan Kamarul Ariffin Wan Ahmad and Sabri Ahmad. 2013. Arima model and exponential smoothing method: A comparison. In *AIP Conference Proceedings*, Vol. 1522. AIP, 1312–1321.
[41] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.
[42] Yibing Wang and Markos Papageorgiou. 2005. Real-time freeway traffic state estimation based on extended Kalman filter: a general approach. *Transportation Research Part B: Methodological* 39, 2 (2005), 141–167.
[43] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).
[44] Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI*

Conference on Artificial Intelligence.

[45] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. 2017. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 777–785.

[46] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE transactions on Big Data* (2018).

[47] G Peter Zhang. 2003. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* 50 (2003), 159–175.

[48] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294* (2018).

[49] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. 2019. T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems* (2019).

[50] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. 2018. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8739–8748.