Environment, Development and Sustainability (2025) 27:24799–24815
https://doi.org/10.1007/s10668-022-02585-z

Big data analytics-based traffic flow forecasting using inductive spatial-temporal network

Chunyang Hu[1] · Bin Ning[1] · Qiong Gu[1] · Junfeng Qu[1] · Seunggil Jeon[2] · Bowen Du[3]

Received: 27 January 2022 / Accepted: 26 May 2022 / Published online: 3 August 2022

Abstract
Traffic flow forecasting is crucial for urban traffic management, which alleviates traffic congestion. However, one inherent feature of urban traffic is it's instability, making it difficult to accurately forecast the future traffic flow. In this paper, we propose a model using Inductive Spatial-Temporal Network to predict the traffic flow speed of road networks. Specifically, we first utilize GraphSAGE(Graph SAmple and aggreGatE) to inductively extract the spatial features of road networks. Furthermore, we design a global temporal block to capture the temporal pattern. Then, we adopt the self-attention mechanism for evaluating the importance of nodes. Finally we introduced an autoregressive module to increase the robustness of the model. Experiments on real-world data demonstrate that considering spatial and temporal dependencies of the traffic data can achieves better performance than models without considering such relations.

Keywords Inductive spatial-temporal network · GraphSAGE · Global temporal block

1 Introduction

In recent years, with the process of urbanization has accelerated , the number of urban vehicles has been increasing rapidly. However, existing road networks are not able to accommodate such an increase, leading to severe congestion problem. Such problem causes further problems such as fuel consumption, air pollution, decline in travel quality and so on.

Effective traffic management can alleviate congestion problem and improve traffic condition, where it is rather essential to forecast the traffic flow. Many studies on traffic flow forecasting have been carried out and achieved better results in the past few decades. In the early years, researchers treated such prediction problem as time series problem. They applied statistical learning approaches to prediction, which cannot efficiently extract

✉ Chunyang Hu
huchunyang@hbuas.edu.cn

[1] School of Computer Engineering, Hubei University of Arts and Science, Xiangyang 441053, Hubei, China

[2] Samsung Electronics, 129, Samseong-ro, Yeongtong-gu, Suwon-si, Gyeonggi-do 16677, South Korea

[3] School of Computer Science and Engineering, Beihang University, Beijing 100191, China

temporal relations along the time axis due to the lack of representation ability. With the development of deep learning technologies, a growing number of researchers started to apply deep learning models such as RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory) to capture the complicated temporal pattern. However, these approaches cannot take full advantage of spatial features of road networks. In recent years, CNN (Convolutional Neural Networks), GNN (Graph Neural Networks) and GCN (Graph Convolutional Networks) are applied to capturing the spatial interactions among road segments. Many efforts have been done employing GNN and GCN directly on road networks, which has made great progresses in traffic flow prediction (Zhang et al., 2021). Although such methods make full use of inherent feature of road networks as graphs, they lack representation ability and flexibility. While using deep models to predict traffic flow, the sudden scale changing can hardly be captured due to the nonlinearity of deep models.

To tackle the aforementioned problems, we propose a model using ISTN to forecast traffic speed. Since GCN requires the whole graph structure to compute the Laplacian matrix, which is the main reason of its lack of flexibility. We use an inductive model, GraphSAGE, to embed graphs in a more flexible way. Considering that different nodes should contribute differently to the graph, we introduce self-attention to the embedded graph, which assigns different weights to the nodes. Then, we adopt one-dimensional CNN to simultaneously aggregate the spatial feature and the short-time temporal pattern, after which an LSTM module is applied to capturing the long-time temporal pattern. At last, we incorporate a traditional autoregressive linear module to enhance the robustness of the model. Overall, the main contribution of this paper includes:

- We propose spatial aggregator module to inductively extract the spatial interactions among the road segments, increasing the ability of generalization of the model, where the importance of different nodes is also considered.
- We design a global temporal block to extract the temporal pattern of the historical traffic flow.
- The autoregressive module is applied to enhance the robustness of the model.

The rest of the paper was organized as follows. Section 2 outlines the related work on traffic flow forecasting. In Sect. 3, we formalize the traffic flow prediction problem. Then Sect. 4 introduces our proposed inductive spatial-temporal network model. The experiment setting and results are shown in Sect. 5. Finally, we conclude our work and present the future directions in Sect. 6.

2 Related work

2.1 Traffic flow forecasting

Traffic forecasting has been a popular field in Intelligent Traffic System (ITS) for a long time. Early in 1970s, Ahmed and Cook (1979) proposed ARIMA (Autoregressive Integrated Moving Average) for short-term traffic flow prediction. Several variants of ARIMA such as Kohonen-ARIMA (Van Der Voort et al., 1996), seasonal ARIMA (Williams & Hoel, 2003) were presented subsequently. Those parametric models are of simple structure, which is suitable for single-sourced time-series data. However, such models are too simple to deal with complex and nonlinear traffic data. Thus, researches tried to utilize

nonparametric models to predict traffic flow. Davis and NiHan (1991) proposed a K-Nearest Neighbor regression method for short-term traffic forecasting. They used similar data series in historical data to improve the avoid the impact of fluctuations and achieved similar result as parametric models did. Jeong et al. (2013) presented an online learning method based on support vector regression (SVR), emphasizing the changes between adjacent time periods. The aforementioned methods lack of representation ability to extract the spatial-temporal features of traffic data. With the development of deep learning, it became possible to explore implicit feature of urban traffic data. Lv et al. (2014) utilized stacked autoencoder to deepen artificial neural network and avoided problems as gradient vanishing and gradient exploding. The experimental results showed its superiority against traditional machine learning methods. Fu et al. (2016) adopted LSTM to extract temporal correlation of input data. Furthermore, Spatial correlation of traffic data was also studied in traffic flow prediction. Wang et al. (2016) constructed a matrix containing historical and road network information, where CNN LeCun et al. (1995) was applied to simultaneously extracting spatial and temporal correlations, resulting better performance than other approaches. GNN (Scarselli et al., 2009) was also an effective tool for spatial interaction extracting. Different from CNN, GNN was able to extract the natural topological feature of road networks.

2.2 Graph neural network

CNN has been an effective method for handling grid-structure data. Huge number of applications have been made in image classification (Krizhevsky et al., 2017; He et al., 2016). However, it is not rational to formalize non-Euclidean data like road network to grid-structure since the rich spatial information could be lost. Thus, GNN was proposed to handle graph data in a more flexible way (Scarselli et al., 2009; Li et al., 2017; Beck et al., 2018). GNN feeds the features and states of neighbors, edges into neural network to update the state of the nodes in the graph, after which the final state is applied to regression or classification tasks. GCN is capable of extracting local feature on the graph, which generalizes convolution operation into graph data. Typically, there are two categories of GCN: Spatial-Based GCN and Spectral-Based GCN. Spatial-based GCN directly defines convolution operation among the nodes. Niepert et al. (2016) introduced spatial-based convolution on graph, which selected several most related neighbors to construct a two-dimensional data structure and applied traditional convolution operation in the constructed data. Atwood and Towsley (2016) proposed diffusion-convolutional neural networks (DCNNs) representing the neighbor nodes k hops away as a feature vector. Then the feature representation was used for node classification or graph classification. Spectral-based GCN is based on spectral graph theory, introducing Fourier transform into graph. Similar to Fourier transform in signal processing, it decompose a graph with the eigenvalue and eigenvector of its Laplacian matrix. Li et al. (2018) first introduced spectral-based GCN to action recognition. They constructed undirected attribute graph with human motion, with which they extracted spatial representation using GCN filter. Finally the motion sequence was fed into a RNN (Connor et al., 1991) for classification. Experiment showed that the proposed method was better than other approaches. Seo et al. (2018) combined GCN and RNN, which extended RNN into dynamic graph, extracting the spatial interaction and dynamic pattern of the input data. The model performed well on MNIST data and Penn Treebank data. Zhuang and Ma (2018) proposed DGCN to jointly consider local consistency and global consistency on graphs. The model outperfomed the state-of-the-art methods with both unsupervised and supervised loss function. Besides, GraphSAGE (Hamilton et al.,

2018) applied different kinds of aggregator on graphs, where a node could be represented by its neighbors. GraphSAGE solved the problem that GCN cannot be generalized into new graphs. Further, GAT (Veličković et al., 2018) introduced attention mechanism into this idea, assigning different weights to different neighbors.

2.3 Time series prediction

Traditional way to deal with time series prediction is applying ARIMA and its variants as introduced in previous subsection. Besides, time series prediction problem can be regarded as regression, thus machine learning methods are also solutions to such problem. Linear support vector regression (LSVR) (Gui et al., 2014; Cao & Tay, 2003) were introduced for financial prediction. Nevertheless those model can hardly capture the nonlinear correlation of the time series data. RNN was designed for time series prediction, which could handle history information as it took previous state and current data as input. However RNN has a few drawbacks as it could not capture long-term time pattern, meanwhile there is risk of gradient vanishing/exploding problem. LSTM (Hochreiter & Schmidhuber, 1997) was first proposed by Hochreater to solve the inherent drawback of RNN (e.g. gradient vanishing, gradient exploding), which introduced gating mechanism to maintain the long-term dependencies of input sequence. An LSTM cell contains three kinds of gate including input gate, forget gate and output gate, all of which controls how information flows through LSTM cells. Meanwhile, each cell also keeps its state during the feed-forward process. As deep learning developed rapidly, researchers proposed numbers of variants of LSTM. Gers and Schmidhuber (2000) introduced *peephole connection* into LSTM, where cell state was also a part of input passing into next cell. Cho et al. (2014) combined input gate and forget gate as one single update gate and mixed the hidden state and cell state of cells. The proposed model has simpler structure and fewer parameters than LSTM does, thus the training process would be faster. Apart from changes on the structure, some variants focuses on the input data structure. Shi et al. (2015) proposed ConvLSTM, combining convolution operation and LSTM. They replaced original matrix multiplication with convolution operation, so that the model can capture the spatial and temporal correlation simultaneously. Peng et al. (2017) did similar work as introducing graph convolution operation into LSTM cell. The presented graph LSTMs was used for relation extraction in NLP and got better performance than other approaches. In addition, CNN could also be applied to sequence problem. Kim (2014) proposed TextCNN for text classification, which treated sentences as matrices and applied CNN on the matrices. The convolution filter slid along the sentences to aggregate the context. Lai et al. (2018) applied this idea on multivariate time series prediction, where the convolution filter slid along time axis

3 Traffic forecasting formulation

In this section, we formalize the traffic flow prediction problem.

Definition 1 ***Road Network Structure*** We adopt an unweighted undirected graph $\mathscr{G}(\mathscr{V}, \mathscr{E}, \boldsymbol{A})$ to represent a road network structure in a city, where $\mathscr{V} = \{v_1, v_2, ..., v_N\}$ is the collection of nodes in the graph denoting road segments and $\mathscr{E}$ is a set of edges that indicates the pairwise connectivity between road segments. $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix

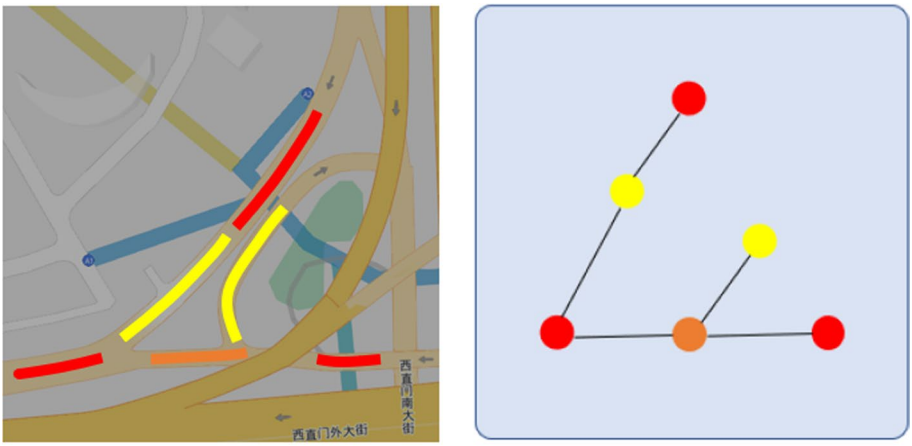
of the graph $\mathcal{G}$, where N is the number of nodes in the graph. $\boldsymbol{A}_{i,j} = 1$ if node v_i and node v_j are connected, otherwise $\boldsymbol{A}_{i,j} = 0$. Since $\mathcal{G}$ is an undirected graph, $\boldsymbol{A}$ is a symmetric matrix.

Definition 2 ***Traffic Flow*** In this paper, traffic flow at a specific time range is represented by the traffic speed of each road segment on road network. Given a graph $\mathcal{G}$, the traffic speed of $\mathcal{G}$ at time range t is denoted by $\boldsymbol{X}_v^t = \{x_v^{t,1}, x_v^{t,2}, ..., x_v^{t,N}\} \in \mathbb{R}^{N\times 1}$, where $x_v^{t,i} \in \mathbb{R}$ is the speed of the i-th road on the road network. Figure 1 illustrates the process of converting a road network to a graph.

The goal of the paper is to predict the traffic flow in the next period. The traffic speed of past T time periods is denoted by $\mathcal{X}_v^T = \{\boldsymbol{X}_v^{t-T+1}, \boldsymbol{X}_v^{t-1}, ..., \boldsymbol{X}_v^t\} \in \mathbb{R}^{T\times N\times 1}$. Besides the traffic speed of the roads, we also utilize the attributes of the roads as their spatial feature (e.g. width, length, etc.) which is similarly represented as $\mathcal{X}_r \in \mathbb{R}^{N\times d}$, where d is the dimension of the road feature. Since the road feature is static for a data instance, it is shared on all the time steps. The complete attributes description is shown in Table 1. In summary, a data instance can be expressed as $data_i = \{\mathcal{G}_i, \mathcal{X}_v^{i,T}, \mathcal{X}_r^i\}$, where $\mathcal{G}_i$ represents the local road network. We seek to use the aforementioned information to predict the traffic speed of the $\mathcal{G}_i$ after the accident, namely $\boldsymbol{X}_v^{t+1} \in \mathbb{R}^{N\times 1}$, as the follow equation shows:

$$\boldsymbol{X}_v^{t+1} = f(\mathcal{G}_i, \mathcal{X}_v^{i,T}, \mathcal{X}_r^i; \boldsymbol{W}), \tag{1}$$

where $\boldsymbol{W}$ is the trainable parameter. The training process aims to minimize error between the predicted result and the ground truth, during which $\boldsymbol{W}$ is updated by gradient decent.



(a) **(b)**

Fig. 1 Process of generating graph. **a** is a road network of Beijing with totally 6 road segments; **b** is the generated graph of (a), where each node represents a road segment in (a)

Table 1 Attributes of roads

Attribute	Comment
Road id	Unique ID of the road segment
Width	The width of the road
Direction	The direction of the road: 1 for two-way, 2 for start-to-end, 3 for end-to-start and 0 for unknown
Length	The length of the road(km)
Func class	The class of the road, totally 5 classes
Through	Whether it is permitted to cross the road, 0 for permitted and 1 for not permitted
Speed class	Class of limit speed, totally 8 classes
Lane num	Number of lanes,
Cross flag	The class of the cross on the end of the road
Light flag	Whether there is any traffic light on the road

4 Methodology

The whole overview of our model is illustrated in Fig. 2. Our model consists of five modules, where the spatial aggregator module (GraphSAGE module) and the self-attention module are used to extract spatial features, and the convolutional module, LSTM module and the autoregressive module are used to capture temporal hidden feature. We first use a 1D-convolutional

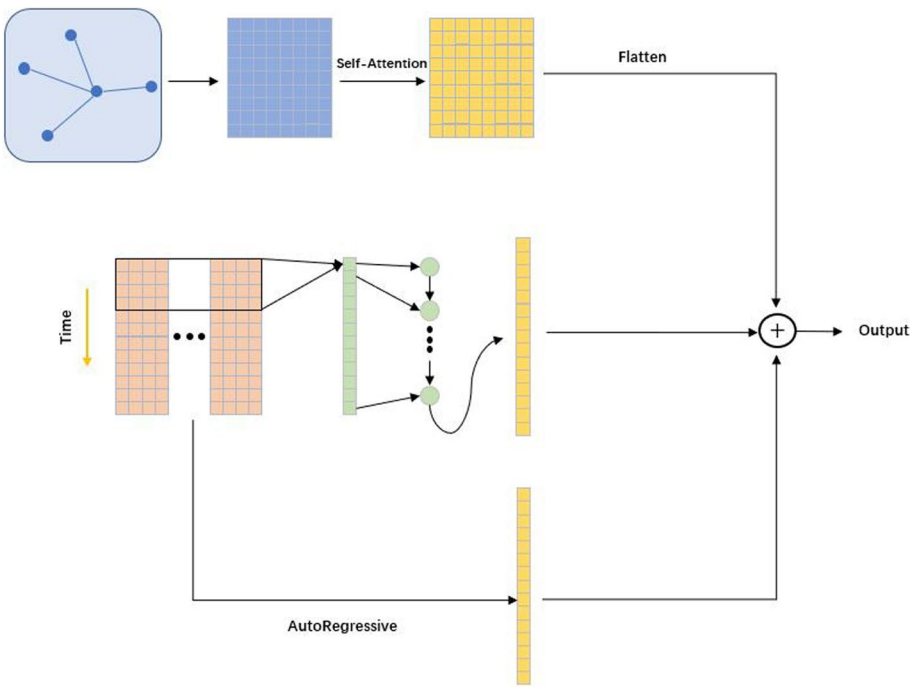


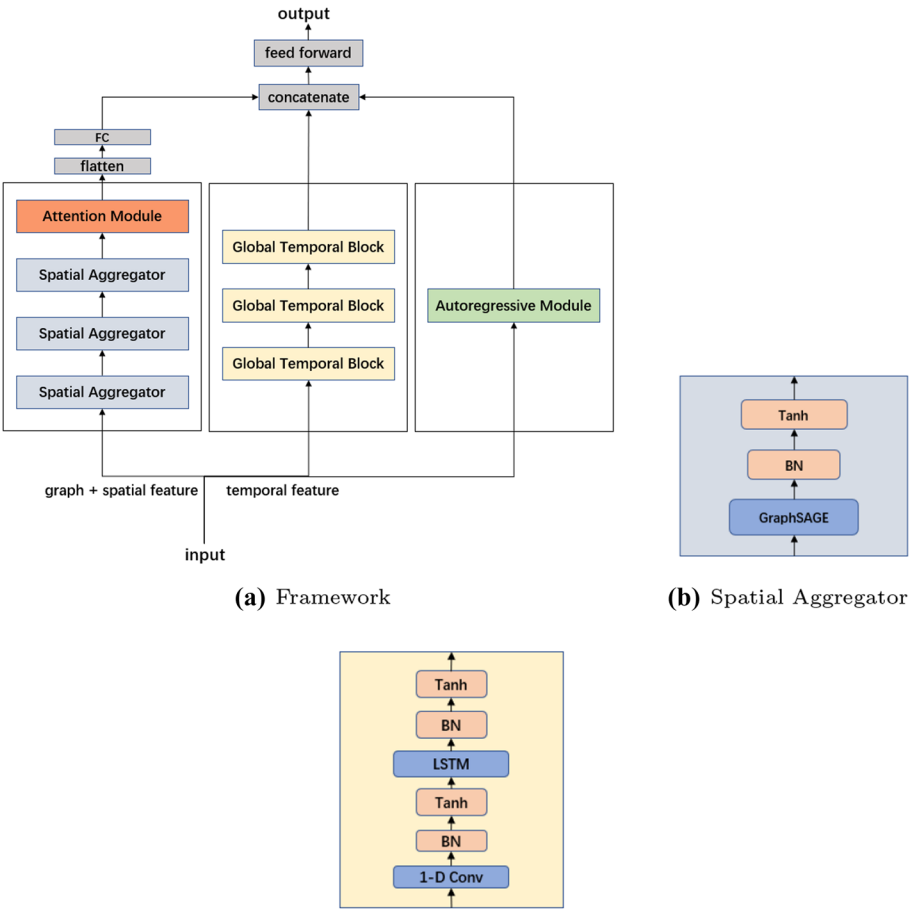
Fig. 2 The whole overview of Our ISTN model

module to aggregate temporal data of different nodes, the output of which is fed into an LSTM module to generate temporal hidden state. Additionally, we introduced an autoregressive module to improve the model robustness. In this section, we describe each of the modules aforementioned separately. Note that we only consider inference process (i.e. *batch size* = 1) for simplification. The overall framework of the proposed model is shown in Fig. 3.

4.1 Spatial feature aggregating

Given a connected graph $\mathcal{G}$, we describe the adjacency with an adjacency matrix $\boldsymbol{A}$. For each node $\mathcal{V}_i$ in the graph, we aim at aggregating its neighbor nodes as a vector:

$$h_{\text{nei}} = \text{aggregate}(\{h_n \| \ \forall n \in \mathcal{M}(\mathcal{V})\}) \tag{2}$$



(a) Framework **(b)** Spatial Aggregator

(c) Global Temporal Block

Fig. 3 **a** is the overall framework of the proposed model. **b** is the spatial aggregator. **c** is the global temporal block

Here h_n is the feature of node n, and $\mathcal{N}(\mathcal{V})$ is the set of current neighbor nodes. Using equation 2, we could get the hidden representation of the neighbor nodes of node $\mathcal{V}$. Then, we concatenate the neighbor representation and the feature of node $\mathcal{N}_i$, on which a nonlinear map is leveraged to obtain the new hidden representation of $\mathcal{N}_i$. The process is as follows:

$$h'_{\mathcal{N}} = \sigma(W \cdot [h_{\text{nei}}; h_{\mathcal{N}}]), \tag{3}$$

where W is the trainable parameter and σ is the activate function. By implementing Eqs. 2 and 3 once, we obtain the node representation with the neighbor information. Further, if k-hop neighbor information is needed, we could iteratively implement the aggregating and mapping process k times, which corresponds k spatial aggregators in the framework. After k times of iterations, each node is embedded into a vector with its neighbors' information. To additionally increase the ability of generalization and avoid problems as gradient vanishing/exploding, we add a batch-normalization (Ioffe & Szegedy, 2015) layer between every two layers. In our model, we choose *meanfunction* as aggregators. Finally we obtain an embedded matrix $H_S \in \mathbb{R}^{N \times d'}$ as the spatial feature of the graph.

4.2 Self-attention block

After the graph $\mathcal{G}$ is embedded into a matrix $H_S \in \mathbb{R}^{N \times d'}$, where each row represents the hidden representation of the corresponding node, we seek to provide different node with different importance. Self-attention was first proposed in transformer (Vaswani et al., 2017), which is able to generate weighted feature during decoding process. Here we use multi-head attention to describe the importance from different aspects, in other word, we perform self-attention k times with different parameter matrices. With the embedded graph matrix $\mathcal{Z}$, we first map it into key matrix K, query matrix Q and value matrix V:

$$K = \mathcal{Z} \cdot W_K, \tag{4}$$

$$Q = \mathcal{Z} \cdot W_Q, \tag{5}$$

$$V = \mathcal{Z} \cdot W_V, \tag{6}$$

where $W_K \in \mathbb{R}^{d' \times (h \times d_K)}$, $W_Q \in \mathbb{R}^{d' \times (h \times d_Q)}$ and $W_V \in \mathbb{R}^{d' \times (h \times d_V)}$ are trainable parameters, and h is the number of heads. For better understanding, here we only illustrate one head of the matrices, namely $K' \in \mathbb{R}^{N \times d_K}$, $Q' \in \mathbb{R}^{N \times d_Q}$ and $V' \in \mathbb{R}^{N \times d_V}$. We define the *Attn* operation as follows:

$$\text{Attn}(K, Q) = \text{softmax}(\frac{K \cdot W_A \cdot Q^T}{\sqrt{d_K}}). \tag{7}$$

With Eq. 7, we obtain the attention matrix $A \in \mathbb{R}^{N \times N}$, where $A_{i,j}$ represents the importance of node i to node j. Here W_A is a trainable matrix for more flexible similarity calculation. Then we could obtain the weighted feature with a simple multiplication:

$$H_S = A \cdot V, \tag{8}$$

where each row of $H_S \in \mathbb{R}^{N \times d_V}$ is the node embedding with other nodes' effect, assuming that if a node contributes little to the graph, the graph will mainly be represented by other nodes' feature.

4.3 Global temporal pattern capturing

When it comes to temporal feature, namely speed in our proposed model, we utilize CNN and LSTM simultaneously to aggregate the temporal feature and capture the pattern. First we use 1D-CNN to capture the short-term pattern as well as correlations between different nodes. Totally the CNN component consists of K filters, producing K channels as result. The output of k-th filter is

$$C_k = \tanh(F_k * \mathscr{X}_v + b), \tag{9}$$

where $F_k \in \mathbb{R}^{T' \times N}$ is the k-th filter and T' is the length of time steps to be aggregated. $C_k \in \mathbb{R}^{(T-T'+1)\times K}$ is the aggregated temporal feature, which is further fed into an LSTM component for long-term temporal pattern capturing. We stack all the channels to obtain the input of LSTM units.

In an LSTM unit at time t, the cell state and output are calculated as follows:

$$\begin{aligned}
i_t &= \sigma(\boldsymbol{X}_{t-1} \cdot W_{ii} + \boldsymbol{h}_{t-1} \cdot W_{ih} + b_i),\\
f_t &= \sigma(\boldsymbol{X}_{t-1} \cdot W_{fi} + \boldsymbol{h}_{t-1} \cdot W_{fh} + b_f),\\
o_t &= \sigma(\boldsymbol{X}_{t-1} \cdot W_{oi} + \boldsymbol{h}_{t-1} \cdot W_{oh} + b_o),\\
c'_t &= \tanh(\boldsymbol{X}_{t-1} \cdot W_{ci} + \boldsymbol{h}_{t-1} \cdot W_{ch} + b_c),\\
c_t &= f_t \odot c_{t-1} + i_t \odot c'_t,\\
h_t &= o_t \odot \tanh(c_t).
\end{aligned} \tag{10}$$

here c_t and h_t represent cell state and hidden state, respectively. After $T - T' + 1$ steps of LSTM feeding forward, we take the output of all the LSTM units as the input of next block. At last we obtain the hidden temporal embedding H_T.

4.4 Autoregressive module

Due to the inherent nonlinearity of deep models, these models are not sensitive to the scale changing of the input data. However, in the real-world scenario such as traffic speed prediction, the speed of the traffic flow is not stable. Assuming that if the speed reduces sharply before prediction, the influence of such changing may be weaken because of the nonlinearity. To address the problem, we additionally add a linear component to the model, namely autoregressive model, to maintain the influence of the scale changing. To reduce the number of parameter and increase the generalization performing, we adopt AR on each node with shared weights. For input speed data $\mathscr{X} \in \mathbb{R}^{T\times N}$, we define the linear output as:

$$H_L = \mathscr{X}_v^{\boldsymbol{T}} \cdot W_L, \tag{11}$$

where $W_L \in \mathbb{R}^{T\times 1}$ is the shared weights.

4.5 Output layer

In the previous subsections, we finally obtain spatial feature H_S, temporal feature H_T and linear feature H_L. With these features, we adopt a feed-forward neural network with batch-normalization for obtaining final result:

$$Y = \text{FeedForward}(\text{concat}(\text{flatten}(H_S), H_T, H_L)). \tag{12}$$

Here $Y \in \mathbb{R}^{1\times N}$ represents the speed of all nodes in the graph.

5 Experiment

In this section, we demonstrated the effectiveness of our model through conducting some experiments. In our experiment, the numbers of spatial aggregator and global temporal block were both set to 3. The quantity of the hidden units is 128. In global temporal blocks, the numbers of filters were set to 16, 32 and 64, respectively. We chose root-mean-square error (RMSE) as loss function and Adam as optimizer.

5.1 Dataset description

Our data consists of two parts: map of Beijing and the corresponding traffic speed information (Liao et al., 2018). The map of Beijing describes the features of over 85,000 road segments in Beijing, where each road segment had its unique ID. The detailed feature we chose has been described in Sect. 3. The traffic dataset contains the traffic speed of 15,073 road segments in Beijing, the ID of which matched the ID in the map data. The speed data was counted during April 1, 2017 - May 31, 2017 every 15 minutes. We randomly chose 479 road segments from the dataset as center nodes, with which we used Breadth First Search (BFS) algorithm to construct 479 graphs. Each graph contained 30 nodes with their static feature. As for the speed data, we set the input sequence length to 12, in other words, we used speed of last 3 hours to predict the speed in the next period. Finally we got 23950 instances, where 60% of them were used to train the model and the rest for validating the effectiveness of model.

5.2 Baseline & evaluation metric

To demonstrate the effectiveness of the proposed model, we selected the following machine learning and deep learning methods for comparison:

(1) *Historical Average (HA)* uses the average speed of each road segment on the road network in the past 2 hours to estimate the speed of next time period;
(2) *Linear Support Vector Regression (LSVR)* is the vector autoregression(VAR) model with Support Vector Regression;
(3) *Feed-forward neural network (FNN)* is a fully connected neural network with 3 hidden layers;
(4) *LSTM* uses cells with input, output, and forget-gate to control the flow of information;

(5) *Spectral-based GCN with LSTM (SGCN-LSTM)* uses spectral-based GCN to extract the spatial feature and uses LSTM to capture the temporal pattern;
(6) *Spatio-Temporal Graph Convolutional Networks (ST-GCN)* is the SOTA method in traffic forecasting.

For the evaluation, we chose Root-Mean-Square Error (RMSE), Mean-Absolute-Percentage Error (MAPE) and Mean-Absolute Error (MAE) to measure the performance of different methods. RMSE, MAPE and MAE are defined as follows:

$$\mathrm{RMSE}(y, \hat{y}) = \sqrt{\frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{m}}, \tag{13}$$

$$\mathrm{MAPE}(y, \hat{y}) = 100\% \times \frac{1}{m} \times \sum_{i=1}^{m} \| \frac{y_i - \hat{y}_i}{y_i} \| \tag{14}$$

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{m} \sum_{i=1}^{m} \|y_i - \hat{y}_i\| \tag{15}$$

5.3 Experiment result

Table 2 shows the experiment result of on the Beijing traffic data. We can see that our model outperforms the other models. The base models as HA, LSVR, FNN and LSTM can not efficiently extract the spatial feature of the road network, resulting in poor performance in traffic prediction. SGCN-LSTM and STGCN both use spectral-based GCN, which can hardly be generalized on unseen graphs. Meanwhile, the learned GCN parameter may have negative effect to the model when it meets unseen graphs, thus the performance of these two models are even worse than LSVR. Particularly the SGCN-LSTM has the worst performance because it can neither effectively capture the spatial nor the temporal feature of new graphs. The RMSE, MAPE and MAE decrease by 4.14%, 8.43% and 6.86%, respectively. Figure 4 shows that our proposed model is able to capture the temporal pattern and perform better on unseen graphs, while STGCN can hardly capture the temporal pattern due to the negative influence of the transductive GCN.

Table 2 Result summary of all the methods

Model	RMSE	MAPE(%)	MAE
HA	3.7987	9.94	2.9659
LSVR	3.3381	8.03	2.5168
FNN	2.7712	6.17	1.8987
LSTM	2.8497	6.16	1.9160
SGCN-LSTM	4.4338	10.74	3.2169
STGCN	3.5420	7.65	2.3515
ISTN	2.6566	5.65	1.7684

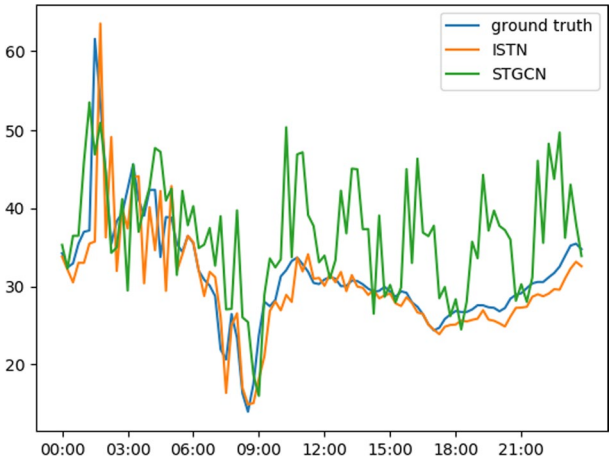


Fig. 4 Speed prediction during a day. The road network does not appear in the training data

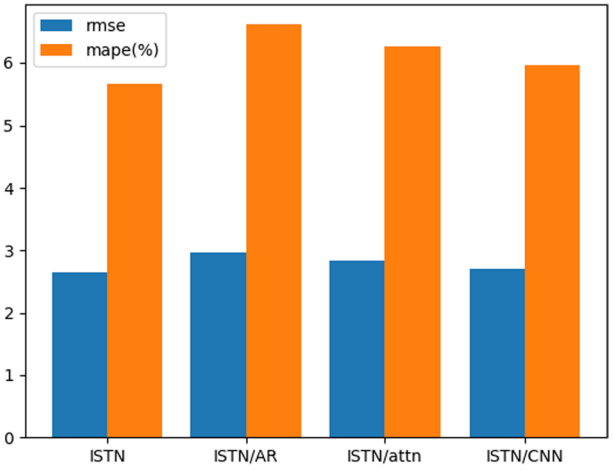


Fig. 5 Result of ablation experiment. ISTN/xx means the proposed model without xx module

5.4 Component effectiveness evaluation

To evaluate the effectiveness of attention block, AR module and CNN module, we conducted more experiments with or without these three modules. Figure 5 presents the result of ablation experiment. As we can see from the figure, all the three modules do improve the performance. The AR module contributes most to the improvement of the model performance. The RMSE and MAPE decrease by 10.39% and 14.52% respectively, with AR module in the model. The reason why the AR module influences the performance so much may be the instability of the traffic speed. The AR module can successfully capture the scale changing of the speed.

5.5 Case study

In order to further prove the effectiveness of our model, we sampled two instances from the dataset to conduct a real-world case experiment. Figures 6 and 7 show the traffic

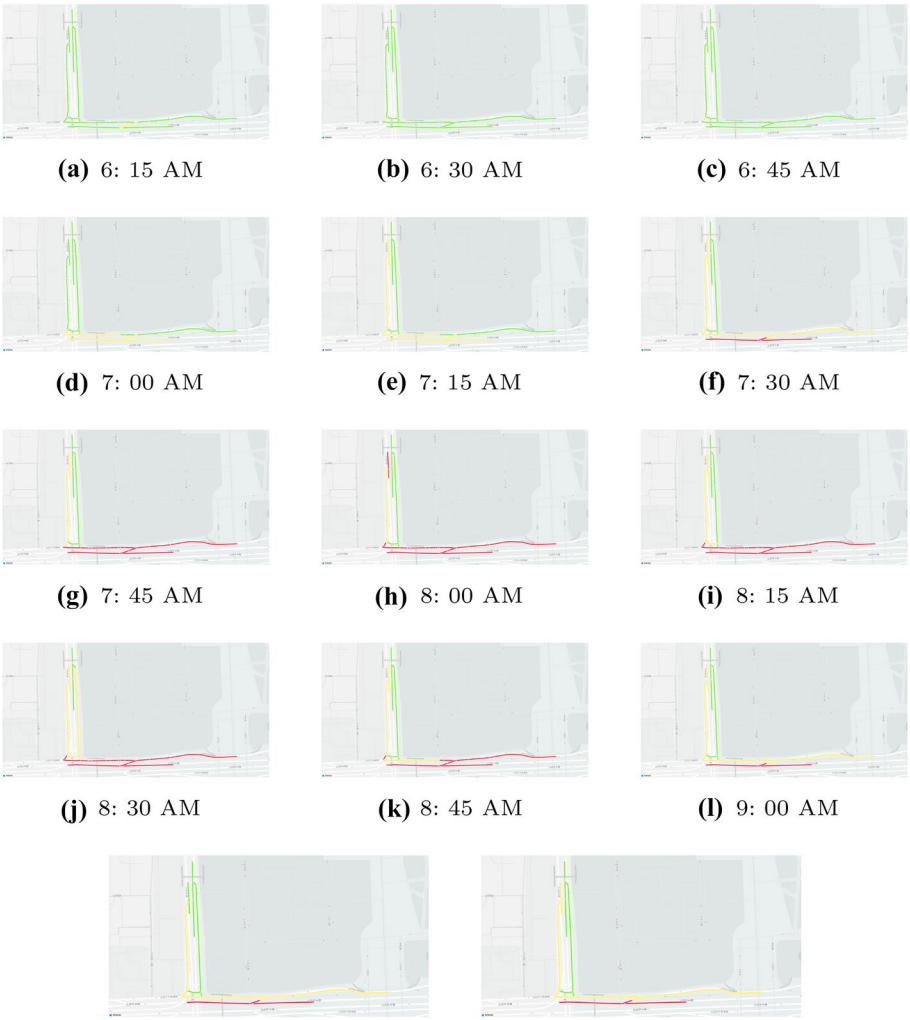


Fig. 6 The traffic conditions of a road network on the forth ring road of Beijing from 6: 15 AM to 9: 15 AM. The speed of road segment are demonstrated by lines of different colors, where red lines represent speed less than 20km/h, yellow lines represent speed between 20km/h and 40km/h, green lines represent speed over 40km/h. The traffic conditions between 6: 15 AM and 9: 00 AM are the input of the model which outputs a traffic condition as figure(n) shows

conditions of two road networks in Beijing. We demonstrate the speed of road segment by lines of different colors, where red lines represent speed less than 20km/h, yellow lines represent speed between 20km/h and 40km/h, green lines represent speed over 40km/h. Figure 6 demonstrates a typical case of Beijing traffic from 6:15 AM to 9:00 AM. As we can see from the figures, the traffic condition turns more and more congested as time flows especially after 7:00 AM, where the speed of several road segments are even less than 20km/h. As the peak hours ends, the traffic condition improves

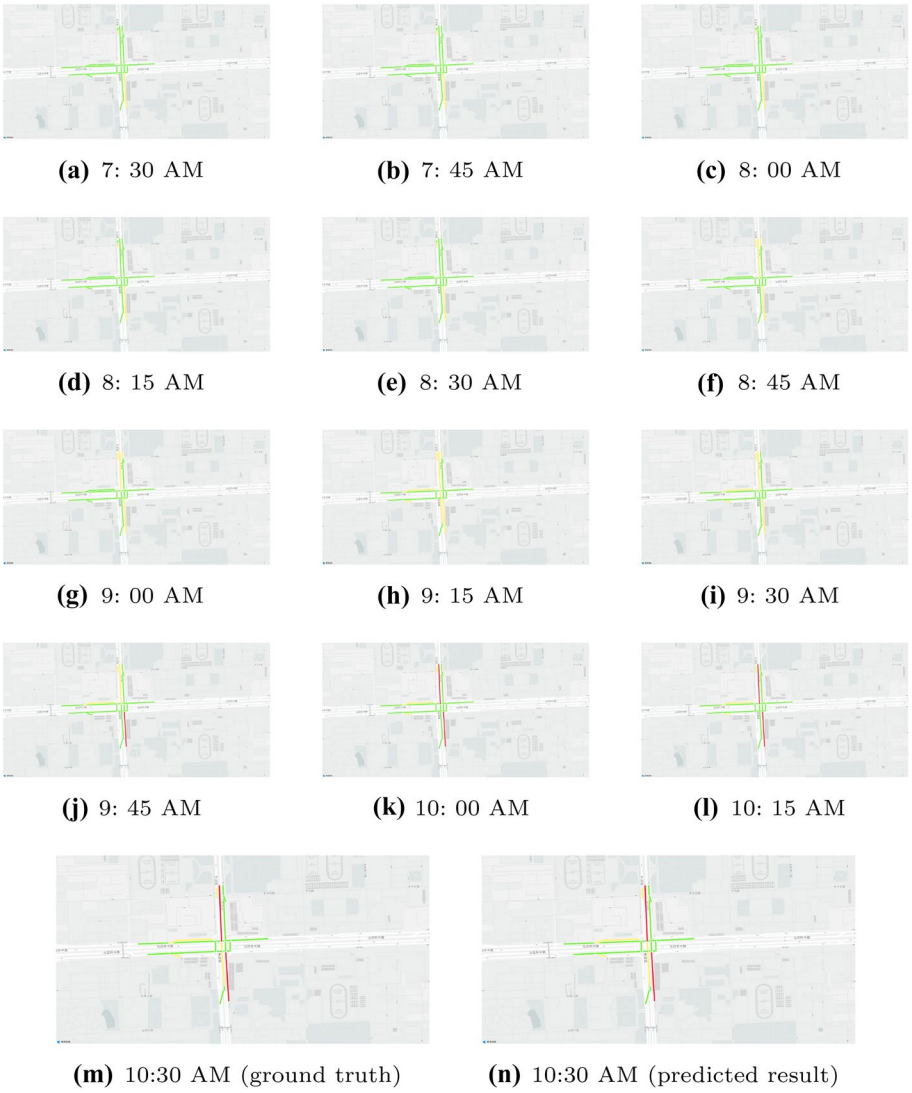


Fig. 7 The traffic conditions of a road network on the Xueyuan Street from 7: 30 AM to 10: 30 AM. The description of this figure is similar with figure 6

gradually. Figures 6(m) and 6(n) shows the actual and predicted traffic condition in 9:15 AM. Our model can successfully capture the temporal pattern during the morning peak hours, predicting the trend of both the main street and the side street on the road network. Further, we conducted a experiment on a road network with more complicated topology to show our model can capture the spatial relationship among the road segments. Figure 7 shows a crossroad in Xueyuan Street in Beijing. The road network contains both normal roads and flyovers, which makes it harder to predict the traffic flow. Our model can successfully predicts the traffic speed with history speed and road structure in such a complicated topology.

6 Conclusion and future work

In this paper, we propose a model using inductive spatial-temporal network for forecasting traffic speed. By integrating the strengths of GraphSAGE, CNN and RNN, we inductively capture the complex spatial-temporal features of traffic speed on road networks. With autoregressive module, we further increase the robustness of our model. Experiments present that our model surpass traditional machine learning approaches. Meanwhile, the efficiency of different component is also demonstrated with ablation experiment.

In future work, we will further optimize road network structure for periodicity extracting temporal feature. Moreover, we will explore the external features which could affect traffic flow such as weather and POI information.

Funding Funding was provided by the Guidance Programs of Science and Technology Funds of the Xiangyang city (2020ZD32), the Major Research Development Program of Hubei Province (No.2020BBB092).

Data availability statement Due to the nature of this research, participants of this study did not agree for their data to be shared publicly, so supporting data is not available.

References

Ahmed, M., & Cook, A. (1979). Analysis of freeway traffic time-series data by using Box-Jenkins techniques. *Transportation Research Board, 722*, 1–9.

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. In *30th Conference on Neural Information Processing Systems (NIPS)*.

Beck, D., Haffari, G., & Cohn, T. (2018). Graph-to-sequence learning using gated graph neural networks. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 273–283. Association for Computational Linguistics, Melbourne, Australia. https://doi.org/10.18653/v1/P18-1026. Retrieved 13 December 2019, from http://aclweb.org/anthology/P18-1026.

Cao, L. J., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks, 14*(6), 1506–1518. https://doi.org/10.1109/TNN.2003.820556.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. Retrieved 8 January 2020, from arXiv:1406.1078.

Connor, J., Atlas, L.E., & Martin, D.R. (1991). Recurrent networks and narma modeling. In: *Proceedings of the 4th International Conference on Neural Information Processing Systems*. NIPS'91, pp. 301–308. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Davis, G. A., & NiHan, N. L. (1991). Nonparametric regression and short-term freeway traffic forecasting. *Journal of Transportation Engineering, 117*(2), 178–188.

Fu, R., Zhang, Z., & Li, L. (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)*, pp. 324–328. IEEE. Wuhan, Hubei Province, China. https://doi.org/10.1109/YAC.2016.7804912. Retrieved 12 December 2020, from http://ieeexplore.ieee.org/document/7804912/.

Gers, F.A., & Schmidhuber, J. (2000). Recurrent nets that time and count. In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, pp. 189–1943. IEEE, Como, Italy. https://doi.org/10.1109/IJCNN.2000.861302. Retrieved 8 January 2020, from http://ieeexplore.ieee.org/document/861302/.

Gui, B., Wei, X., Shen, Q., Qi, J., & Guo, L. (2014). Financial time series forecasting using support vector machine. In: *2014 Tenth International Conference on Computational Intelligence and Security*, pp. 39–43. https://doi.org/10.1109/CIS.2014.22.

Hamilton, W.L., Ying, R., & Leskovec, J. (2018). Inductive representation learning on large graphs. arXiv:1706.02216.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. IEEE, Las Vegas, NV, USA (2016). https://doi.org/10.1109/CVPR.2016.90. Retrieved 12 Decemeber 2020, from http://ieeexplore.ieee.org/document/7780459/.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Retrieved 28 July 2020, from arXiv:1502.03167.

Jeong, Y.-S., Byon, Y.-J., Castro-Neto, M. M., & Easa, S. M. (2013). Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems, 14*(4), 1700–1707. https://doi.org/10.1109/TITS.2013.2267735.

Kim, Y. (2014). Convolutional neural networks for sentence classification. Retrieved 2 August 2020, from arXiv:1408.5882.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM, 60*(6), 84–90. https://doi.org/10.1145/3065386.

Lai, G., Chang, W.-C., Yang, Y., & Liu, H. (2018). Modeling long- and short-term temporal patterns with deep neural networks. Retrieved from 2 August 2020, from arXiv:1703.07015.

LeCun, Y., Bengio, Y., & Laboratories, T. B. (1995). Convolutional networks for images, speech, and time-series. In Arbib M. A. (Ed.), *The handbook of brain theory and neural networks*. MIT Press.

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2017). Gated graph sequence neural networks. Retrieved 13 December 2019, from arXiv:1511.05493.

Li, C., Cui, Z., Zheng, W., Xu, C., Ji, R., & Yang, J. (2018). Action-attending graphic neural network. *IEEE Transactions on Image Processing, 27*(7), 3657–3670. https://doi.org/10.1109/TIP.2018.2815744.

Liao, B., Zhang, J., Wu, C., McIlwraith, D., Chen, T., Yang, S., Guo, Y., & Wu, F. (2018). Deep sequence learning with auxiliary information for traffic prediction. Retrieved 28 July 2020, from arXiv:1806.07380.

Lv, Y., Duan, Y., Kang, W., Li, Z., & Wang, F.-Y. (2014). Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2014.2345663.

Niepert, M., Ahmed, M., & Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In: *International Conference on International Conference on Machine Learning*.

Peng, N., Poon, H., Quirk, C., Toutanova, K., & Yih, W.-T. (2017). Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics, 5,* 101–115. https://doi.org/10.1162/tacl_a_00049.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks, 20*(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605.

Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In L. Cheng, A. C. S. Leung, & S. Ozawa (Eds.), *Neural information processing* (pp. 362–373). Cham: Springer. https://doi.org/10.1007/978-3-030-04167-0_33.

Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., & Woo, W. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *29th Conference on Neural Information Processing Systems (NIPS)*.

Van Der Voort, M., Dougherty, M., & Watson, S. (1996). Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies, 4*(5), 307–318. https://doi.org/10.1016/S0968-090X(97)82903-8.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. Retrieved 24 July 2020, from arXiv:1706.03762.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lió, P., & Bengio, Y. (2018). Graph attention networks. arXiv:1710.10903.

Wang, J., Gu, Q., Wu, J., Liu, G., & Xiong, Z. (2016). Traffic speed prediction and congestion source exploration: A deep learning method. In *2016 IEEE 16th International Conference on Data*

Mining (ICDM), pp. 499–508. IEEE, Barcelona, Spain. https://doi.org/10.1109/ICDM.2016.0061. Retrieved 12 Decemeber 2020, from http://ieeexplore.ieee.org/document/7837874/.

Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering, 129*(6), 664–672. https://doi.org/10.1061/(ASCE)0733-947X(2003)129:6(664.

Zhang, Q., Yu, K., Guo, Z., Garg, S., Rodrigues, J., Hassan, M. M., & Guizani, M. (2021). Graph neural networks-driven traffic forecasting for connected internet of vehicles. *IEEE Transactions on Network Science and Engineering*. https://doi.org/10.1109/TNSE.2021.3126830.

Zhuang, C., & Ma, Q. (2018). Dual gaph convolutional networks for graph-based semi-supervised classification. In: *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*, pp. 499–508. ACM Press, Lyon, France. https://doi.org/10.1145/3178876.3186116. Retrieved 7 January 2020, from http://dl.acm.org/citation.cfm?doid=3178876.3186116.