Information Sciences 611 (2022) 522–539

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Spatial–temporal short-term traffic flow prediction model based on dynamical-learning graph convolution mechanism ☆

Zhijun Chen [a], Zhe Lu [b], Qiushi Chen [c], Hongliang Zhong [c], Yishi Zhang [d,e,*], Jie Xue [f], Chaozhong Wu [a,*]

[a] Intelligent Transportation Systems Research Center, Wuhan University of Technology, Wuhan 430063, China
[b] School of Transportation and Logistics Engineering, Wuhan University of Technology, Wuhan 430063, China
[c] School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China
[d] School of Management, Wuhan University of Technology, Wuhan 430070, China
[e] Research Institute of Digital Governance and Management Decision Innovation, Wuhan University of Technology, Wuhan 430070, China
[f] Faculty of Technology, Policy and Management, Safety and Security Science Group (S3G), Delft University of Technology, Delft 2628BX, The Netherlands

ARTICLE INFO

Article history:
Received 29 March 2021
Received in revised form 17 August 2022
Accepted 24 August 2022
Available online 27 August 2022

Keywords:
Short-term traffic flow prediction
Graph convolution
Intelligent connected transportation
Deep learning
Group vehicle movement prediction

ABSTRACT

Short-term traffic flow prediction is a core branch of intelligent traffic systems (ITS) and plays an important role in traffic management. The graph convolution network (GCN) is widely used in traffic prediction models to efficiently handle the graphical structural data of road networks. However, the influence weights among different road sections are usually distinct in real life and are difficult to analyze manually. The traditional GCN mechanism, which relies on a manually set adjacency matrix, is unable to dynamically learn such spatial patterns during training. To address this drawback, this study proposes a novel location graph convolutional network (location-GCN). The location-GCN solves this problem by adding a new learnable matrix to the GCN mechanism, using the absolute value of this matrix to represent the distinct influence levels among different nodes. Subsequently, long short-term memory (LSTM) is employed in the proposed traffic prediction model. Moreover, trigonometric function encoding was used in this study to enable the short-term input sequence to convey long-term periodic information. Finally, the proposed model was compared with the baseline models and evaluated on two real-world traffic flow datasets. The results show that our model is more accurate and robust than the other representative traffic prediction models.

1. Introduction

In intelligent traffic systems (ITS) research, short-term traffic flow prediction has drawn increasing attention in recent years as a crucial task in intelligent traffic system research. A more accurate traffic prediction usually results in a better allo-

☆ This study was supported in part by the National Key R&D Program of China [Grant No. 2018YFB1600600]; in part by the National Natural Science Foundation of China [Grant Nos. 52072288, 71702066]; in part by the Institute of Distribution Research, Dongbei University of Finance and Economics [Grant No. IDR2021YB004]; and in part by the Hubei Province Science and Technology Major Project [Grant No. 2020AAA001]. Yishi Zhang is the principal corresponding author of this study.

* Corresponding author at: School of Management, Wuhan University of Technology, Wuhan 430070, China, and Intelligent Transportation Systems Research Center, Wuhan University of Technology, Wuhan 430063, China.
E-mail addresses: yszhang@whut.edu.cn (Y. Zhang), wucz@whut.edu.cn (C. Wu).

https://doi.org/10.1016/j.ins.2022.08.080
0020-0255/

cation of limited public transportation resources [48,17,3]. Precise traffic prediction can also help city administrators efficiently manage urban traffic, avoiding potential accidents, risk situations, or traffic paralysis beforehand. The purpose of traffic flow prediction is to predict the future traffic flow in several road nodes in an area based on historical traffic data. Traffic flow in the same position usually has a steady temporal changing pattern, which indicates a strong correlation between historical and future flows. Thus, traditional traffic prediction models, such as autoregressive integrated moving average (ARIMA) and linear regression (LR), conduct predictions based on an analysis of the temporal pattern of traffic flow [35,7]. Molnár et al. [26] considered traffic flow as an evolution of variables and proposed a Lagrangian continuum traffic model, bridging microscopic and macroscopic approaches. However, these traditional mechanisms cannot efficiently handle multi-feature sequence input, which is commonly observed in traffic prediction tasks, limiting their prediction accuracy.

In recent years, artificial neural networks (ANN) have demonstrated excellent performance in traffic prediction tasks owing to their outstanding ability for pattern fitting and data processing. Inspired by the simple architecture and good performance in nonlinear relationship fitting of ANN, Ma et al. [23] used a multi-layer perceptron (MLP) and ARIMA to conduct traffic state prediction. Recurrent neural network (RNN), long short-term memory (LSTM), and gated recurrent unit (GRU) models are all widely used ANN models that can analyze temporal patterns hidden in traffic data [32,50,43]. Because the spatial pattern is also an important factor that needs to be carefully considered during traffic prediction, many widely used spatial pattern analysis models exist. Convolutional neural networks (CNNs) and graph convolutional networks (GCNs) are typical examples of such methods [42,27]. The graph wavelet gated recurrent (GWGR) neural network [5] uses wavelet transform to detect sudden changes and peaks in the temporal signal, considering that the traffic status on a road segment is highly influenced by the upstream/downstream segments and nearby bottlenecks. All these models consider the mutual influences of nearby areas.

However, changes in traffic flow are affected by both temporal and spatial factors. Considering this, the most recent studies are attempting to develop spatial–temporal prediction methods. Among these studies, RNN and GCN models are usually employed to compose a hybrid model, such as the diffusion convolutional recurrent neural network (DCRNN) proposed by Li et al. [19]. RNN can process the temporal traffic pattern properly owing to its good memory ability; Moreover, GCN can be more effectively applied in real-life tasks than CNN because it can handle graphical road network data. The classical GCN mechanism relies on a manually set adjacency matrix to convey information about the influence weights among different nodes. The elements of the adjacency matrix are usually set to the connectivity or distance of the road nodes to represent the different influence weights of different road nodes. However, the distinct degrees of influence are usually difficult to analyze manually without training. When the connectivity is set as the element of the adjacency matrix, the adjacency matrix becomes a 0–1 matrix, which means that all the upstream nodes of a downstream node contribute the same influence weights in the convolution layer. This is completely contrary to real-life scenarios and will reduce the prediction accuracy.

To efficiently handle these shortcomings, a spatial–temporal traffic flow prediction method called location graph convolution long short-term memory (Loc-GCLSTM) is proposed in this study. It uses the absolute value of a newly added trainable matrix of the GCN to dynamically learn the different influence weights among the nodes during the training process. Moreover, Loc-GCLSTM attempts to place the periodic pattern of traffic data into model inputs using trigonometric function encoding. The main contributions of this study are as follows:

(a) The location-GCN is proposed to enable the convolution operation to dynamically learn the distinct influence weights among different road section nodes, which allows the GCN mechanism to be more consistent with the real-life scenario and improves the performance of the GCN model.
(b) A spatial–temporal traffic prediction model (Loc-GCLSTM) is proposed, which combines the proposed location-GCN and LSTM to more efficiently capture the spatial–temporal information contained in real-life traffic data.
(c) Experiments based on two real-world traffic network datasets are conducted to verify the effectiveness of the proposed method.

The remainder of this paper is organized as follows. Section 2 describes the related work and research status in the traffic prediction domain. Section 3 illustrates a specified description of the application scenario and the technical details of the proposed model. Then, in Section 4, several experiments based on our OpenITS and METR-LA datasets are designed and conducted to comprehensively evaluate the proposed model. Finally, Section 5 summarizes the work and raises potential points that warrant further study for improvement.

2. Literature Review

2.1. Classical Traffic Prediction Methods

Early models of traffic prediction, such as auto-regression (AR), autoregressive moving average (ARMA), and ARIMA, conducted tasks based on AR and data stationary assumptions [38,35]. However, these models can only perform temporal analysis on long-time-span data and can only be fit and tested on a single road, rendering them difficult to apply to dynamic and volatile real-life traffic prediction. With the development of machine-intelligence techniques, some studies have attempted to apply machine learning to traffic prediction. For example, some classical machine learning methods,

such as the *k*-nearest neighbor (*k*NN), support vector regression (SVR), and extreme gradient boosting (XGBoost), have been applied to traffic flow prediction[21,37,9]. To achieve better robustness in prediction, researchers have used a combination of different machine learning models. Zhang et al. [45] combined XGBoost with the light gradient boosting machine (LightGBM) [44] to conduct the traffic prediction task. Min and Wynter [25] built a model based on spatio-temporal correlations using a multivariate spatial–temporal autoregressive (MSTAR) model. Rajabzadeh et al. [31] used the stochastic differential equations (SDEs) to optimize a two-step prediction based on the Hull-White (HW) model and the Vasicek model (EV).

2.2. Traffic Prediction using Single Deep Learning Model

In recent years, a series of studies have attempted to use ANN, such as MLP and AutoEncoder, to predict traffic flows [16,4]. The RNN was first used to analyze the changing patterns of historical traffic data for temporal models. However, RNNs are not able to analyze long-term temporal dependencies, which usually continue for several moments. To this end, LSTM and GRU have recently been proposed. To better analyze the bidirectional temporal pattern in flow changing, Cui et al. [6] applied Bi-LSTM to generate the traffic pattern from both original and reversed order data using bidirectional LSTM, achieving higher accuracy in prediction tasks. To better handle lane-level traffic data, Gu et al. [11] used entropy-based grey relation analysis (EGRA) to analyze the influence among different lanes and presented a lane-level traffic prediction model combining GRU and LSTM. Nested LSTM puts an LSTM unit inside an LSTM to improve the stability of the learning process, with the goal of improving the learning of long-term traffic patterns in historical traffic data [24]. Inspired by natural language processing (NLP) tasks, many researchers have attempted to utilize sequence-to-sequence (Seq2Seq) and attention mechanisms in traffic prediction, with the goal of obtaining more accurate prediction results [20,29].

Spatial information is also of significant value for traffic flow prediction. For spatial models, the convolution operation is typically applied to generalize and analyze the spatial correlations among nearby locations or road sections. CNN was first used in traffic flow prediction to analyze several mutual influences of several nearby locations in a transportation network [42]. Because CNN cannot efficiently analyze non-Euclidean structure data, GCN is introduced into traffic prediction to handle graph-structured data, such as a transportation network composed of various road section nodes [2]. Because the road network in a GCN cannot dynamically change over time, graph attention (GAT) uses the attention mechanism to describe the correlations among different road section nodes and analyze graph structure traffic data [13,34]. Geng et al. [10] developed a multi-GCN model which introduces many other spatial factors among nodes, such as a distant map into the normal GCN mechanism for increased awareness of various traffic information.

2.3. Spatial–temporal Traffic Prediction

However, real traffic flows are influenced by both spatial and temporal factors. Recent research has focused on analyzing traffic patterns from both spatial and temporal perspectives [19,33,49]. DCRNN [19], temporal graph convolutional network (T-GCN) [47], and sequence-to-sequence model based on graph convolution (GC-Seq2Seq) [12] are three representative methods that directly combine spatial layers with temporal layers. To effectively use the mutual influence between spatial and temporal factors, Xiao et al. [39] utilized spatial–temporal blocks for data analysis from both spatial and temporal dimensions and proposed a spatial–temporal graph convolutional network (STGCN) model. In the STGCN model, the temporal convolutional layers are realized by the CNN, and the graph convolutional layers are realized by the GCN, attempting to better analyze the correlations between nearby moments and locations. To more flexibly handle the dynamically changing traffic network, Wu et al. [36] proposed a graph attention LSTM network (GAT-LSTM) to predict traffic flow, which uses GAT to update spatial information and LSTM to generalize the temporal patterns hidden in historical traffic data to conduct spatial–temporal analysis dynamically. The attention graph convolutional sequence-to-sequence model (AGC-Seq2Seq) was proposed by Zhang et al. [46] to train the Seq2Seq model in traffic prediction, using GCN and an attention mechanism to promote accuracy. Peng et al. [28] similarly considered the spatio-temporal features of traffic data and proposed a dynamic graph recurrent convolutional neural network for traffic flow prediction.

In summary, spatiotemporal deep learning models have been the main trend in recent short-term prediction research. Although these studies can achieve a higher prediction accuracy ratio than traditional regression models, most of them still have some shortcomings. First, the GCN cannot analyze and distinguish the different influence weights of various upstream nodes toward the same downstream node as the GAT, despite outperforming the GAT owing to its superior learning ability. Furthermore, because ANN models usually consider a short sequence as input, it is difficult to convey periodic information across weeks or even months in short-term prediction models. In addition, owing to the better performance and stronger generalization ability of the GCN and LSTM, this study attempted to overcome the aforementioned drawbacks based on an improved GCN-LSTM hybrid model.

3. Methodology

3.1. Problem Description

The proposed ANN model aims to efficiently handle short-term traffic flow prediction tasks. In such tasks, historical traffic data of several positions in an area are used to predict the future traffic data of the same positions.

In this study, the digraph $G = (V, E, A)$was used to describe the urban road network area where our traffic prediction task was conducted, where $V$is the set of road section nodes, Edenotes the set of edges among road section nodes, and Arepresents the adjacency matrix of the section nodes in the target area. Furthermore, $A_{i,j}$indicates whether the ith section is connected to the jth section ($A_{i,j}$=1 if they are connected and zero otherwise). The diagonal values in the adjacency matrix were zero. For an area with Nroad section nodes, the corresponding adjacency matrix is denoted as $A \in R^{N\times N}$. In this study, different road directions were considered for two different road sections.

3.2. Spatial–Temporal Analysis

Traditional traffic flow prediction models, including GRU, LSTM, GCN, or GAT, focus on analyzing the traffic pattern from a single perspective. However, real-life traffic flow changes are affected by spatial and temporal factors. To better analyze historical traffic data and achieve a more accurate prediction performance, historical traffic data must be investigated from both spatial and temporal perspectives.

Based on a thorough consideration, LSTM and GCN were chosen as the basic methods in our model to conduct spatial and temporal analyses.

3.2.1. Long Short-Term Memory

Among all the temporal analysis mechanisms in traffic flow prediction, LSTM is the most efficient, widely used, and representative. Therefore, it was used in our model for temporal pattern processing. LSTM uses the cell state and three gated structures: the input gate i, forget gate f, and output gate o, to realize the analysis, memory, and output of the long-term dependencies of temporal sequences, respectively. Fig. 1 shows the structure of the LSTM unit.

$$f_t = \sigma(V_f h_{t-1} + W_f X_t + b_f) \tag{1}$$
$$i_t = \sigma(V_i h_{t-1} + W_i X_t + b_i) \tag{2}$$
$$\widetilde{C}_t = \tanh(V_C h_{t-1} + W_C X_t + b_C) \tag{3}$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t \tag{4}$$
$$o_t = \sigma(V_o h_{t-1} + W_o X_t + b_o) \tag{5}$$
$$h_t = o_t \cdot \tanh C_t \tag{6}$$

The aforementioned equations describe the whole processing flow of LSTM, where $\sigma$and $tanh$are activation functions, Eq. 1 indicates the process of forget gate, Eqs. 2 and 3 indicate the process of input gate, Eq. 4 represents the fusion process of information from the aforementioned two processes, as well as Eqs. 5 and 6 represent the process of output gate.

3.2.2. Graph Convolution Network

A GCN can efficiently handle graph-structured data, such as urban traffic network data. This was used in our model for the spatial analysis. When processing graph-structured data with Nnodes, the traditional GCN mechanism in a road network can be expressed as follows:

$$H_l = \begin{cases} X, & l = 0 \\ (A + E)H_{l-1}W_l & l \geqslant 1 \end{cases} \tag{7}$$

$A \in R^{N\times N}$is the adjacency matrix, $X \in R^{N\times F}$denotes the Ffeature' input data of Nnodes, $H_l \in R^{N\times F_l}(l \geqslant 1)$denotes the convolution result of $F_l$features after $l$thconvolution, $E$is the identity matrix, and $W$is the trainable matrix. Fig. 2 shows the physical meaning of graph convolution. The convolution operation is a weighted summation of several nearby data points in an area. When convoluting the red target nodes in the image, if $l = 1$, the algorithm performs a weighted summation of the traffic data of the target nodes and their upstream nodes. Subsequently, it updates the target nodes' data, integrating upstream flow information into it. If $l = 2$, the algorithm performs a weighted summation of the green points whose distance is two, then updates the target nodes' data, spreading layer by layer until reaching the manually set maximum distance limit.

The traditional GCN mechanism uses $A + E$to describe graph information, where adjacency matrix $A$and identity matrix Econsist of only zero and one, respectively. The target node with more than one upstream node's data increases in size during the graph convolution. Therefore, the inverse matrix of the digraph's degree matrix is typically used to multiply the matrix $A + E$, normalizing the data after every convolution operation. The equation can be written as follows:

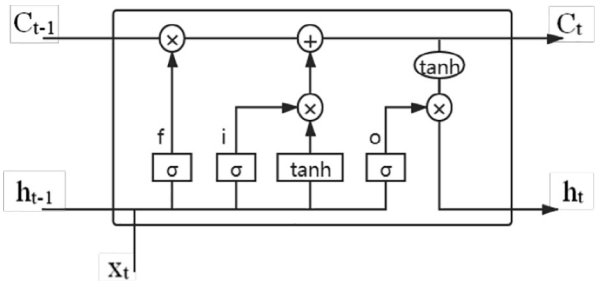

Fig. 1. Unit structure of LSTM.

$$H_l = \begin{cases} X, & l = 0 \\ D^{-1}(A+E)H_{l-1}W_l & l \geqslant 1 \end{cases} \tag{8}$$

3.3. Location Graph Convolution Network

3.3.1. Trainable Matrix

However, the classical graph convolution mechanism has certain drawbacks. As shown in Fig. 3, every row in the matrix H_{l-1}represents the traffic data of a single node in the road network. During its multiplication with $D^{-1}(A+E)$, elements in a different row of $H_{l-1}$are weighted and summed to form a new row, which means that the traffic data in different upstream nodes are conveyed into a single downstream node, updating the data of the target nodes. However, in the multiplication between this weighted sum result and the original trainable matrix W_l, no interaction occurs between the two rows or nodes. Therefore, in addition to the node traffic data matrix H_{l-1}, the only matrix involved in the weighted sum between upstream and downstream nodes is $D^{-1}(A+E)$. However, all elements in $D^{-1}(A+E)$, which represent the influence weights between the two different nodes, were manually set before training. In most cases, the different influence relationships among various road section nodes are difficult to analyze manually. It is usually preferable to acquire this knowledge through model training. Moreover, in most cases, the values of the adjacency matrix are set to zero and one, implying that the mechanism implicitly proves that every upstream node has the same influence on the target downstream node, which is completely contrary to reality. Both of the aforementioned points are detrimental to the model's ability to effectively analyze the traffic patterns and achieve high prediction accuracy.

To overcome this drawback, because the original trainable matrix $W_l$cannot be used for dynamical learning, a trainable matrix $W_{mask}$is proposed to learn the different weights of the two road nodes. We also hope that by using W_{mask}, the improved GCN will be able to automatically adjust the influence weights during the training process.

$W_{mask}$is an $N \times N$matrix, where $N$is the number of road-section nodes. It has the same shape as the adjacency matrix A. The value at position $(i,j)$is the weight of $i$thsection's influence on the $j$thsection. In contrast to the traditional GCN mechanism, which uses $A+E$to acquire the graph information between nodes in the analysis, this matrix can independently change different node pairs' influences via the training, rather than merely being set as zero or one. The initial weight of this matrix can be randomly set, although it is not recommended to set it as all zero because it does not correspond with real life. Through this, information on the differences in the influence levels among different nodes can be learned.

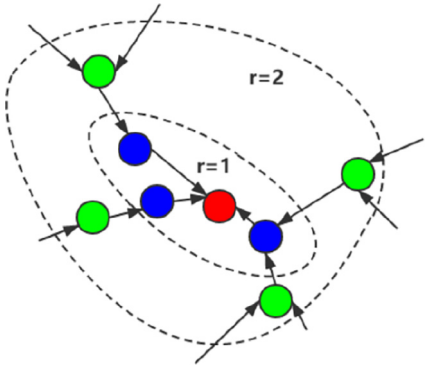

Fig. 2. Layer by layer spreading by graph convolutional algorithm.

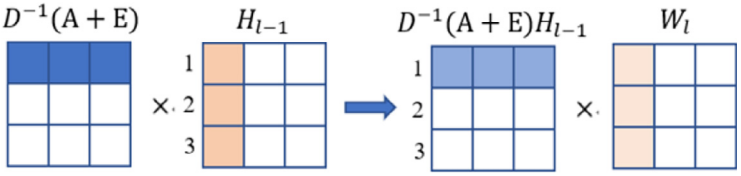

Fig. 3. Drawbacks of traditional graph convolution.

3.3.2. Calculating Improvement

To enable the proposed GCN mechanism operate, the calculation method of the GCN must be changed. After the pattern learning of W_{mask}, the absolute value of each element in $W_{mask}$is used to create a new matrix W_{abs}because the influence of upstream roads on downstream roads is usually positive. It is then used to multiply the $A+E$matrix to use the different influence levels among various nodes learned from real-life training data to adjust the values of the adjacency matrix. Thus, the model can learn different influence levels for various road nodes. The improved mechanism can be described using the following equation:

$$W_{abs} = \mid W_{mask}[i,j] \mid \quad (i \in [0, N-1], j \in [0, N-1]) \tag{9}$$

$$H_l = \begin{cases} X, & l = 0 \\ D^{-1}(W_{abs} \cdot (A+E))H_{l-1}W_l & l \geqslant 1 \end{cases} \tag{10}$$

By using $W_{mask}$and W_{abs}, the location-GCN can analyze a more detailed spatial pattern in traffic flow. A comparison sample between traditional GCN and the proposed method is shown in Figure 4.

In a road network with four nodes, Nodes 2 and 3 are the upstream nodes of Node 0. Because the traditional GCN can only represent the influence levels between point pairs with a value of zero or one, the weight values at positions $[2,0]$and $[3,0]$in $A+E$are both 0.5 after normalizing D^{-1}, leading to the mechanism assuming that the influence weights of nodes 2 and 3 toward node 0 are equal. However, our mechanism can enable the model to learn the difference in influences during training and allow the convolution operation to obtain this knowledge via dot multiplication with W_{abs}.

3.4. Trigonometric Function Encoding

For short-term traffic flow prediction, each input sample usually varies over a short time span, such as one or two hours. The periodic changing pattern among days or weeks cannot be directly analyzed using the neural network model in each sample processing. Thus, the periodic data must be encoded into the input samples of the network model.

As periodic functions, the sine and cosine functions can describe periodic information in traffic data. Thus, we used these trigonometric functions to encode the hourly and moment information in the period. The time interval of the task was set to 5 min, allowing the time point data to be first transferred to (i,j), implying that the *ith*5 min in the day and the *jth*h in the week. Then, trigonometric function encoding encodes (i,j)into four columns *moment_sin*, *moment_cos*, *hour_sin*, and *hour_cos*, using the following equations:

$$moment\_sin = \sin\frac{2\pi i}{moment\_num} \tag{11}$$

$$moment\_cos = \cos\frac{2\pi i}{moment\_num} \tag{12}$$

$$hour\_sin = \sin\frac{2\pi j}{hour\_num} \tag{13}$$

$$hour\_cos = \cos\frac{2\pi j}{hour\_num}, \tag{14}$$

where *moment_num* is the total number of 5-min-interval moments in a day and *hour_num*is the total number of hours per week. Using this method, our prediction method can analyze the long-term periodic patterns in short-term traffic flow prediction tasks.

3.5. The proposed model Loc-GCLSTM

To better handle the periodic pattern in historical traffic data, Loc-GCLSTM first applies a trigonometric function to encode moment data, using the periodic property of the sine function and the cosine function to describe periodic information in the traffic pattern. Subsequently, the data is put into a location-GCN layer, where a trainable matrix is utilized to learn the different influence levels of various upstream road sections toward each downstream section. Finally, the spatial-analyzing location-GCN is combined with the temporal-analyzing LSTM to conduct the prediction. After the spatial analysis conducted by location-GCN and temporal analysis conducted by LSTM, the last step output of LSTM is chosen to be put into a fully con-

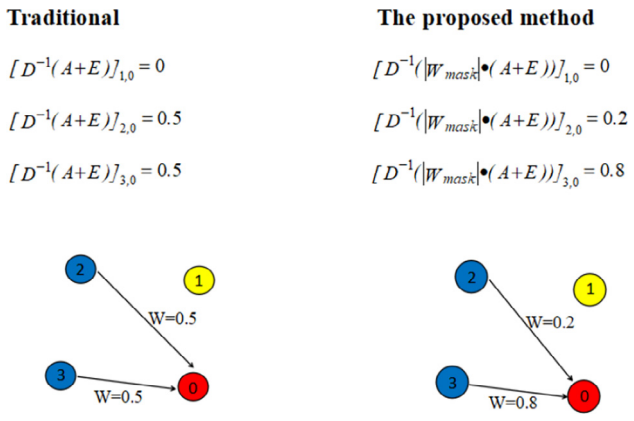

Fig. 4. Drawbacks of traditional graph convolution.

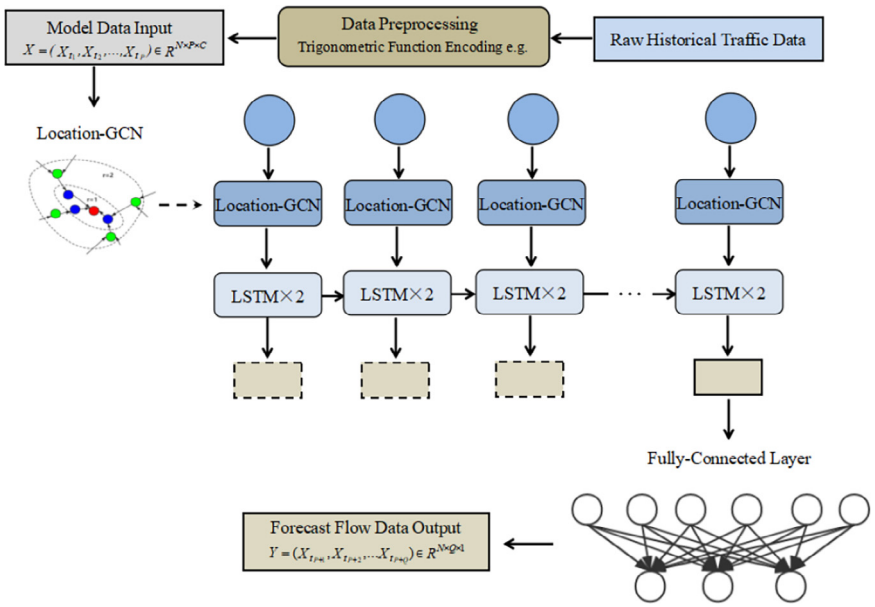

Fig. 5. Structure of Loc-GCLSTM.

nected layer to generate the output forecast flow sequences, considering that the last step output of LSTM is the semantic vector that generalizes the pattern of the sample flow. The structure of the proposed model is shown in Fig. 5. The computational complexity of the spatial analysis is $\mathcal{O}(n^2)$, where n denotes input data size [41]. For the temporal analysis, its computational complexity per time step is $\mathcal{O}(W)$, where $W = KH + KCS + CSI + HI$ denotes the number of weights; K, C, S, H, and I denote the number of output units, number of memory cell blocks, size of the memory cell blocks, number of hidden units, and (maximal) number of units connected to memory cells, gate units, and hidden units, respectively, [14]. Hence, the computational complexity of Loc-GCLSTM per time step was $\mathcal{O}(n^2 + W)$.

4. Experiments and Results

4.1. Datasets

We conducted experiments on two real-world datasets: OpenITS and METR-LA[1]. The **OpenITS** dataset was collected from the OpenITS platform, with data recorded between 3 o'clock and 24 o'clock each day from July 1st, 2019 to July 31st, 2019 (July 2nd was not included) in Xuancheng, Anhui Province, PRC. Thirteen road section observation nodes were selected for the exper-

[1] The raw dataset is available at https://github.com/liyaguang/DCRNN/tree/master/data

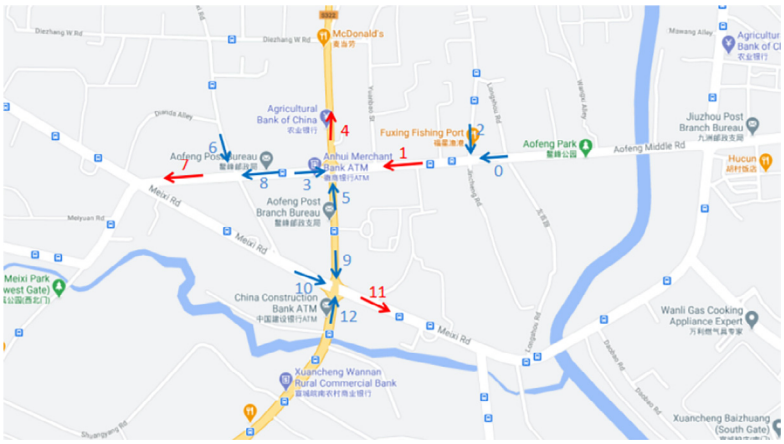

Fig. 6. Locations of observation points in OpenITS dataset.

iments, and the time interval was set to 5 min. The specified locations of the observation nodes are shown in Fig. 6. Each observation section is denoted by an arrow, and the direction of the arrow indicates the flow direction of the observation node. The red nodes represent selected test nodes. The obtained data were saved in the form of CSV files, and a new weather column was manually added to the data. The test nodes' 06:30–09:30, 10:30–13:30, as well as 17:00–22:00 data on the 3rd, 5th, 7th, 15th, 16th, 18th, and 27th of July was chosen as the test dataset. All observation points' 03:00–24:00 data on days other than the seven test days were used as the training dataset. In this study, the K-nearest mechanism was applied to fill in the missing data.

METR-LA contains the all-day traffic data collected in 207 positions on the highways of Los Angeles County, from March 2012 to June 2012. We selected data from March and April as our experimental dataset. With more road sections and a much more complicated traffic situation in Los Angeles compared to XuanCheng in the OpenITS dataset, METR-LA can more clearly show the effectiveness of the promotions used in Loc-GCLSTM. The METR-LA dataset was randomly divided into training and test datasets in a proportion of four to one. All the data for every road section and moment are included in this dataset, rendering the experiment more challenging. The K-nearest mechanism was also applied to fill in the missing data of the METR-LA dataset. Fig. 7 shows the observation-point settings for the METR-LA dataset.

4.2. Data Preprocess

The two datasets were preprocessed using the same method. A sliding window was used to process the datasets: starting from the first five minutes of each day, samples were generated every two consecutive hours (24-timesteps with a 5-min interval), with the 24-timesteps not being interrupted. Moreover, 5440 samples for the OpenITS dataset and 17520 samples for the METR-LA dataset were collected. We applied fivefold cross-validation to conduct the experiments, that is, we randomly divided the dataset into five folds, using the first (second, third, fourth, and fifth) fold as the test set and the remaining folds as the training set. The average result was reported as the performance indicator of each method used in our experiments.

The training and test samples were arranged into two NumPy arrays in the shape of $[sample\_num, node\_num, time\_lags, feature\_nums]$, where $sample_num$denotes the total number of samples, $node_num$represents the number of nodes in the road network, $time_lags$denotes the time steps (12 in our experiments) of each sample, and $feature_nums$represents the number of features. The traffic flow, average speed, flow density, proportion of large and regular vehicles, moment (5-min interval, 1-day period), hour (1-h interval, 1-week period), lane number, and weather of every observation point were chosen as the input features in the OpenITS dataset. Traffic flow, moment (5-min interval, 1-day period), and hour (1-h interval, 1-week period) were selected in the METR-LA dataset. The time point and hour data were encoded by a trigonometric function, whereas the weather was encoded by a number. These data were input into the model after z-score standardization.

4.3. Evaluation Criteria and Comparison Models

The root mean square error (RMSE), mean absolute percentage error (MAPE), mean absolute error (MAE), median absolute percentage error (MdAPE), and median absolute error (MdAE) were chosen as the evaluation criteria for our experiments. These five criteria are widely used in traffic-prediction tasks. Notably, MdAPE and MdAE use the median of the errors instead of their mean (in contrast to MAPE and MAE).

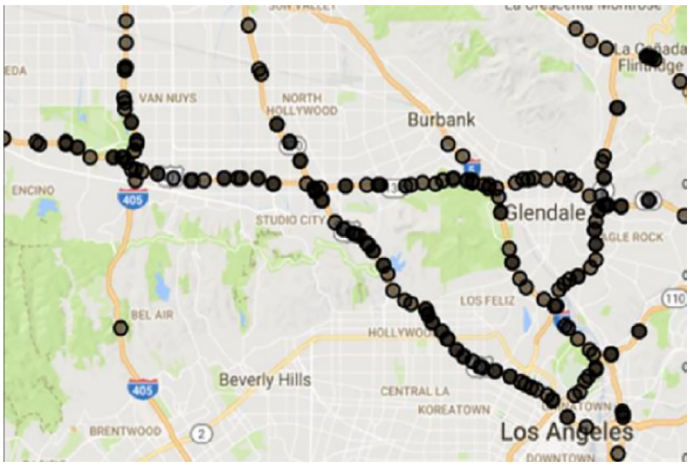

Fig. 7. Locations of observation points in METR-LA dataset [19].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=T+1}^{T+n}\left(\widehat{Y}_i - Y_i\right)^2} \tag{15}$$

$$MAE = \frac{1}{n}\sum_{i=T+1}^{T+n} |\ \widehat{Y}_i - Y_i\ | \tag{16}$$

$$MAPE = \frac{1}{n}\sum_{i=T+1}^{T+n} |\ \frac{\widehat{Y}_i - Y_i}{Y_i} \times 100\ | \tag{17}$$

$$MdAE = median\left(|\ \widehat{Y}_1 - Y_1\ |, \cdots, |\ \widehat{Y}_n - Y_n\ |\right) \tag{18}$$

$$MdAPE = median\left(|\ \frac{\widehat{Y}_1 - Y_1}{Y_1} \times 100\ |, \cdots, |\ \frac{\widehat{Y}_n - Y_n}{Y_n} \times 100\ |\right) \tag{19}$$

In our experiments, three conventional machine learning methods (LR, XGBoost, and SVR) and three deep learning methods (LSTM, STGCN, and DCRNN) were selected for comparison. LR, XGBoost, SVR, and LSTM are temporal analysis models, whereas STGCN and DCRNN are spatial–temporal prediction models. Detailed information on these models is as follows:

- LR [7] is the most basic short-term traffic prediction method, which uses linear parameters to simply fit the training samples' patterns. LR usually uses ordinary least squares or linear least squares to update its parameters. After the data fitting, the LR model can be described as the following equation:

$$Y = w_1x_1 + w_2x_2 + \cdots + w_nx_n \tag{20}$$

  - XGBoost [9] is an improvement based on gradient boosting decision tree (GBDT) [40], which is the best performing and widely used boosting model in recent traffic prediction competitions. XGBoost is an ensemble model based on several single-tree models. Thus, it can reduce the sensitivity to the abnormal samples of the model, enhancing stability.
- SVR [37] is an outstanding machine learning mechanism focusing on vital samples, avoiding local data perturbation. It uses a soft margin mechanism to allow the samples to fluctuate in a small error range, improving the model's generalization ability. In addition, it selects vital samples as support vectors to conduct the data fitting, obtaining a better view of the global data samples, which results in a comparable high accuracy.
- LSTM [50], as mentioned previously, is the most classical neural network model used in the temporal analysis. With the use of cell state and gated mechanism, LSTM can efficiently learn the long-term dependency pattern compared with other methods; thus, it is commonly used in temporal analysis tasks. However, a single LSTM network cannot handle the spatial factors, which are also crucial in traffic prediction.
- STGCN [39] combines temporal analysis using CNN and spatial analysis using GCN and introduces gated linear units (GLU) to optimize its output ability. CNN is used to generalize the temporal changing pattern of traffic data layer by layer. GLU is used to optimize the CNN output by enabling it to fit a more complex pattern. A GCN is always put between every two CNNs, aiming at analyzing the spatial pattern of traffic flow. Using such a structure, STGCN is a spatial–temporal traffic prediction model that can consider the mutual influence between two perspectives. However, because STGCN does not use the RNN-based mechanism for spatial analysis, it cannot thoroughly investigate the flow changing details.

- DCRNN [19] is a spatial–temporal model, which consists of bidirectional GCN and GRU. It adds a pair of trainable weights to learn the influence of one node's upstream and downstream positions separately. Moreover, the weighted sum of the upstream and downstream convolution is put into a multi-layer GRU network to further analyze the temporal changing pattern of the historical traffic data.

4.4. Experimental Setting

All experiments were compiled and tested on a Windows platform (CPU: Intel(R) Core(TM) i7-8700 K CPU @ 3.70 GHz, GPU: NVIDIA GeForce RTX 2080 Ti). Tensorflow was used for the implementation of all neural-network-based models[1]. During the training of the neural network, the mean square error (MSE) was used as the loss function, whereas the root mean square propagation (RMSProp) [22,18,8] was used as the optimizer. The neural network model was trained for 600 epochs in the OpenITS experiment and for 300 epochs in the METR-LA experiment. In the proposed model, cosine annealing learning rate adjustment (CALRA) [15] was used to improve the training performance. The period cycle number of CALRA was four. The maximum initial learning rate and minimum initial learning rate were 2.4e-5 and 1.5e-5, respectively. In addition, we used a grid search to configure an optimal combination [30] and obtained performance with different parameters; the effects of different parameters are shown in Fig. 8. Batch size refers to the number of samples traversed before calculating the loss function. Although a smaller batch size may lead to better model performance, it will increase the computational cost. A similar effect was observed when the number of units increased. However, no absolute negative correlation exists among batch size, units, and model performance, as shown in Fig. 9. Therefore, we set the batch size and number of units to 64 and 256, respectively, after the grid search. The setting of parameters in each layer is shown in Table 1. Further analysis of the grid settings for each parameter and the margin effect on the prediction performance of the proposed model are described in detail in the Appendix.

A grid search was conducted to obtain the best parameters for the baseline models. In the LSTM model, the batch size was set to 32, the number of units was set to 256 for both LSTM layers, and a fully connected layer of 12 units was connected to yield the output. In the STGCN model, number of channels in the three layers in the ST-Conv block were set to 256, 64, and 256, and the graph convolution kernel size and temporal convolution kernel size were both set to three. In addition, the learning rate was set to the same value as that of the proposed model, and the batch size was set to 32. In the DCRNN model, the maximum steps of random walks, that is, K, was set to one, the numbers of units in the convolution layer and in the two GRU layers were set to 64 and 128, respectively, the batch size was set to 32, and the decaying learning rate degrades the performance of this model. Therefore, we employed a fixed learning rate with a value of $3e-5$. In XGBoost, the best results were achieved when the maximum depth was set to six and the subsample was set to 0.9. The radial basis function (RBF) kernel and penalty term $C=1$ were applied in SVR. Default parameters were used for the LR. Only a limited number of parameter values can be examined using the grid search strategy; however, it is clear from these results that the effect of the parameters on the performance is much less than the improvement provided by the model structure. Therefore, we applied these parameters to obtain the results of the comparison. The next subsection presents the results of the comparison.

4.5. Results

4.5.1. Prediction results

Table 2 shows the accuracy comparison among the models in the five evaluation criteria for the OpenITS dataset. Loc-GCLSTM achieved the best results according to all criteria. Notably, the STGCN spatial–temporal model performs worse than LSTM because of the lack of LSTM for long-term dependency analysis. When compared with LSTM, Loc-GCLSTM improved the RMSE by 17.91 %, MAE by 17.14 %, and MAPE by 1.51 %. When compared with the DCRNN, Loc-GCLSTM improved the RMSE by 4.99 %, MAE by 4.89 %, and MAPE by 5.52 %.

Table 3 shows the accuracy comparison between the models for the five evaluation criteria on the METR-LA dataset. Furthermore, Loc-GCLSTM outperformed the other methods according to all the criteria. Because the task is to predict the traffic flow for the next 12 five-minute time intervals, the LSTM model also performs well on this dataset, which fully demonstrates the excellence of LSTM in performing multiple sequence prediction tasks. Loc-GCLSTM improved by 7.54% in RMSE, 6.08% in MAE, and 8.67% in MAPE compared with LSTM, and improved by 8.54% in RMSE, 10.86% in MAE, and 13.02% in MAPE compared with DCRNN. The improvement of the proposed model over the other selected models in each evaluation criterion can also be observed in the table.

Fig. 10 shows the distribution of the prediction errors of LSTM, DCRNN, and Loc-GCLSTM on the OpenITS and METR-LA datasets, where the x-axis represents the error between the predicted result and the ground truth, and the y-axis represents the distribution of the error. It is clear that the distribution of Loc-GCLSTM is denser near 0, indicating that Loc-GCLSTM obtains more accurate results in most cases. It is also denser along the x-axis, indicating that Loc-GCLSTM is more robust in terms of prediction accuracy.

4.5.2. Convergence Rate

The predicted results of LSTM, STGCN, DCRNN, and Loc-GCLSTM for the test dataset were recorded after each epoch during the training process. The RMSE was used to evaluate the convergence rate. Fig. 11 shows the evaluation results, where the X-axis records the number of epochs and the Y-axis records the RMSE score, the yellow line corresponds to LSTM, the

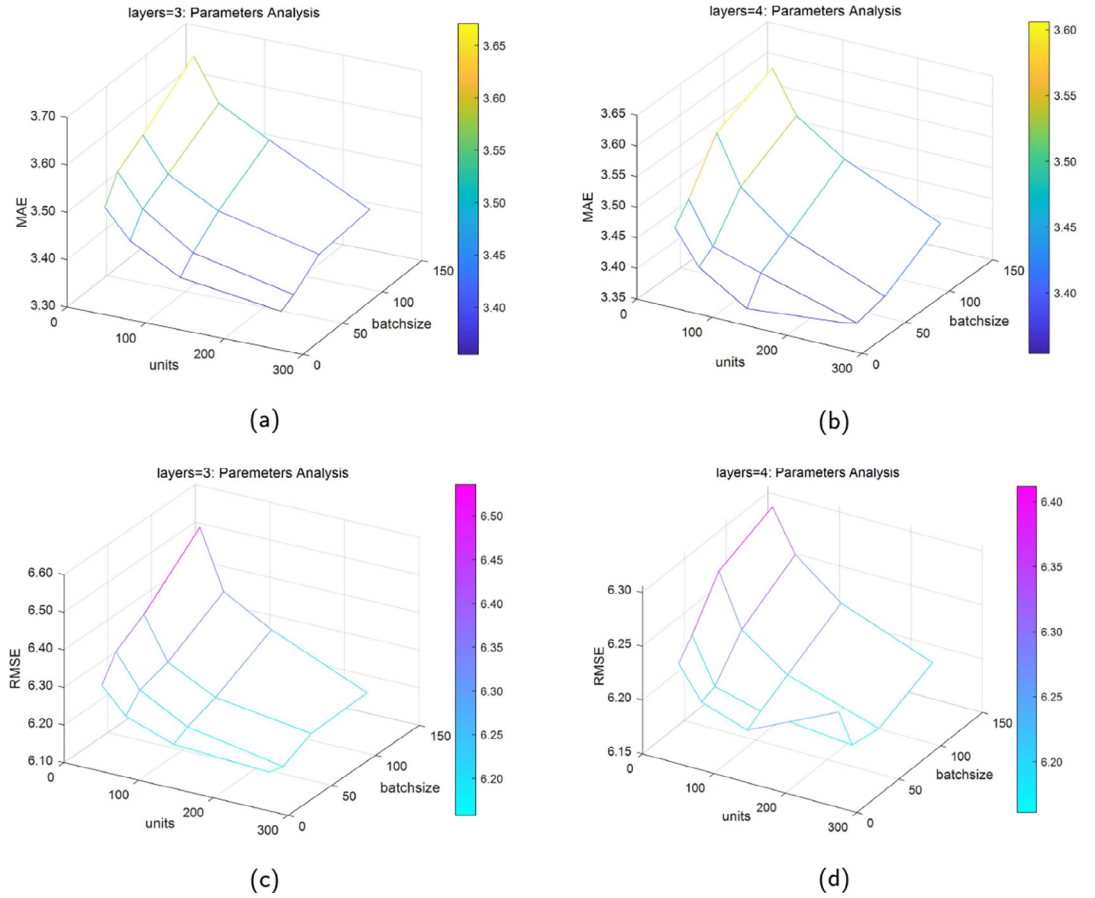

Fig. 8. Effects of different parameters (a) MAE value of layers = 3; (b) MAE value of layers = 4; (c) RMSE value of layers = 3; (d) RMSE value of layers = 4.

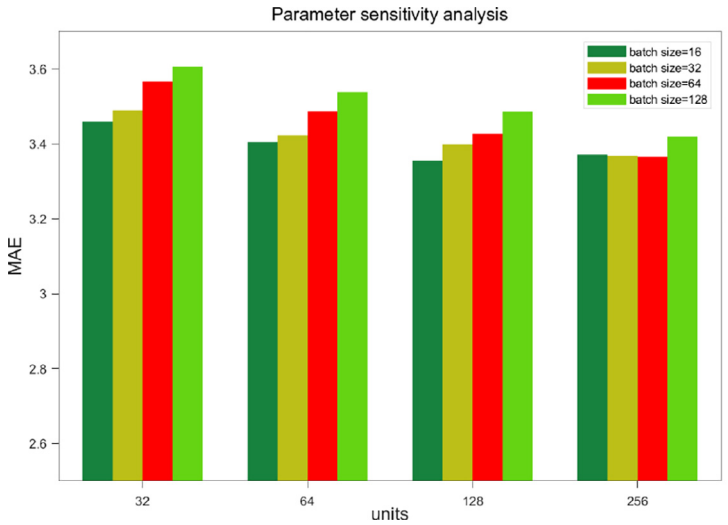

Fig. 9. Parameter sensitivity analysis of layers = 4.

blue line corresponds to STGCN, the purple line corresponds to DCRNN, and the red line corresponds to Loc-GCLSTM. It is evident that Loc-GCLSTM has the fastest convergence rate and the most stable and highest prediction accuracy on both the OpenITS and METR-LA datasets.

Table 1
Settings of each layer of the model.

Number	Name	Settings
1	graph_convolution_1	units = 128,steps = 1
2	lstm_1	units = 256,bias_regularizer = l2(0.01)
3	lstm_2	units = 256,bias_regularizer = l2(0.01)
4	fully-connected_1	units = 12

Table 2
Performance comparison between the proposed model and the selected baseline models on OpenITS dataset.

	LR	XGBoost	SVR	LSTM	STGCN	DCRNN	Loc-GCLSTM
RMSE	12.325	11.547	11.287	9.463	9.982	8.176	**7.768**
MAE	**9.173**	**8.626**	**8.389**	**7.095**	**7.497**	**6.181**	**5.879**
MAPE(%)	**28.195**	**24.252**	**23.405**	**20.275**	**21.985**	**18.214**	**17.208**
MdAE	**7.150**	**6.794**	**6.507**	**5.452**	**5.808**	**4.814**	**4.598**
MdAPE(%)	**28.205**	**24.133**	**23.323**	**20.239**	**22.083**	**18.177**	**17.198**

Table 3
Performance comparison between the proposed model and the selected baseline models on the METR-LA dataset.

	LR	XGBoost	SVR	LSTM	STGCN	DCRNN	Loc-GCLSTM
RMSE	7.073	7.006	7.138	6.664	7.392	6.736	**6.161**
MAE	**3.830**	**3.585**	**3.685**	**3.589**	**4.297**	**3.775**	**3.365**
MAPE(%)	**10.215**	**9.548**	**9.666**	**9.968**	**12.087**	**10.467**	**9.104**
MdAE	**2.149**	**1.834**	**2.074**	**1.947**	**2.526**	**2.151**	**1.898**
MdAPE(%)	**8.471**	**8.086**	**8.031**	**7.925**	**10.120**	**8.607**	**7.548**

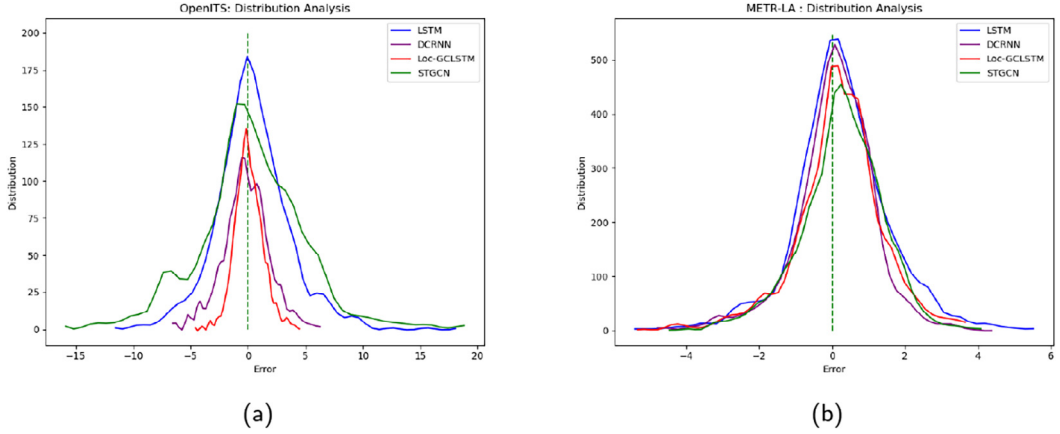

Fig. 10. Comparison between the distribution of prediction errors on (a) OpenITS dataset; (b) METR-LA dataset.

4.5.3. Prediction-Truth Comparison

Fig. 12 shows a comparison between the prediction result and the ground truth for both the OpenITS and METR-LA datasets, where the X-axis is the value of the ground truth and the Y-axis is the value of the prediction result. In each image, 100 randomly sampled points of Loc-GCLSTM and the comparison model exist. It can be observed that the sample points of Loc-GCLSTM, compared with the other models, are denser around the line $y = x$, indicating higher accuracies.

4.5.4. Predicted Results and Ground Truth

Under the evaluation of the OpenITS dataset, the first and fourth observation nodes at 07:30–09:30, as well as 11:30–13:30 on July 5th, July 7th, and July 16th, a total of four time slot prediction results of LSTM and Loc-GCLSTM were chosen for comparison with the ground truth. In Fig. 13, the green lines represent the ground truth, the blue lines represent the predicted results of LSTM, and the red lines represent the prediction results of Loc-GCLSTM. It can be observed that the prediction results of Loc-GCLSTM are closer to the ground truth.

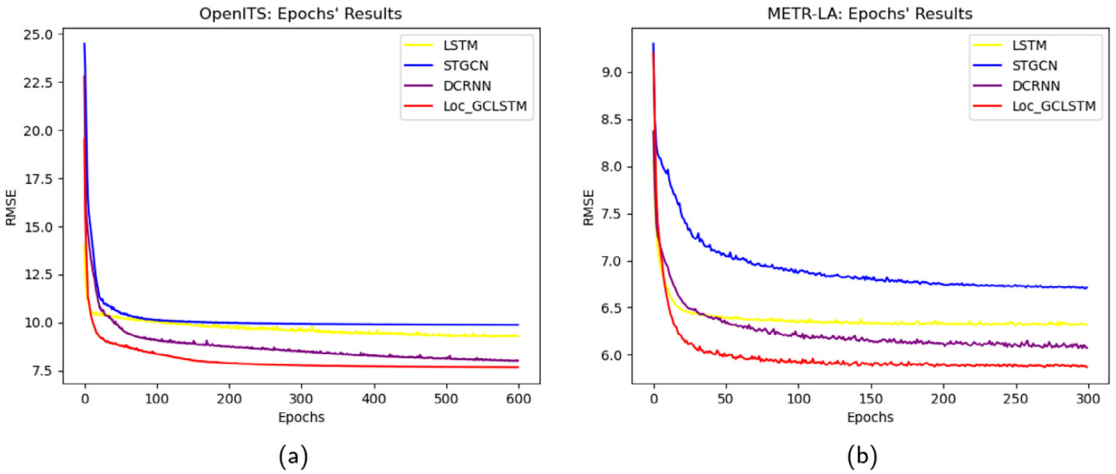

Fig. 11. Comparison of convergence rates on (a) OpenITS dataset and (b) METR-LA dataset.

4.5.5. Predicted Accuracy on Each Road

As shown in Fig. 14, Loc-GCLSTM was compared with normal LSTM, DCRNN, and STGCN on each road's prediction accuracy using RMSE, MAPE, and MAE, under the evaluation of the OpenITS dataset. On every selected test road, Loc-GCLSTM outperformed LSTM and DCRNN. The accuracy improvements did not vary significantly between each road section, indicating that Loc-GCLSTM can usually conduct a more accurate prediction on most real-life roads.

4.5.6. Location Module Analysis

To further analyze the effect of the location module on the performance of the proposed model in traffic flow prediction, we constructed a comparison model without a location-based learning component for validation, in which all parameters were the same as the proposed model except for the GCN layer. As shown in Fig. 15, the location module did not have a significant impact on model's performance on the OpenITS dataset. However, on the META-LA dataset, the proposed location-based learning component had a significant impact on the model performance in traffic flow prediction. This is possibly because the key to improving the performance of our model is to capture the different impact weights of roads. However, because the OpenITS dataset contains a small number of roads and a small amount of data, the proposed component does not fully demonstrate its performance. For the META-LA dataset, which has enough roads and sample size, the location module can completely exploit the potential information between road nodes, thus significantly improving the accuracy of the traffic flow prediction.

5. Conclusion

In this study, a new short-term traffic flow prediction model named Loc-GCLSTM is proposed. The model combines a location-GCN with LSTM to conduct predictions from both temporal and spatial perspectives. Location-GCN enables the graph convolution mechanism to learn different influence weights among nodes dynamically during training using the absolute value of an added trainable matrix. Moreover, trigonometric function encoding is used to preprocess the time point data, enabling the short-term input sequence to convey periodic patterns to the network model.

The experimental results show that the proposed Loc-GCLSTM model can achieve the best prediction performance compared with various models on both the OpenITS and METR-LA datasets. It also has good robustness compared to other models, which indicates a better application potential in real life. The convergence rate of our model is also faster than that of the other compared models, indicating a lower requirement for computing resources. Moreover, our main contributions, such as location-GCN, do not conflict with advanced models such as DCRNN and GC-Seq2Seq; they can also be used in those models.

However, the location-GCN used in our study only considers the differences among the influences of different road sections. It is still not possible to learn how the influence weights of different nodes change over time. Because location-GCN is still based on the traditional GCN mechanism, the node sections of the road network cannot be changed in both the training and testing processes, limiting the convenience and flexibility of the model in real-life usage. All these drawbacks merit further study.

CRediT authorship contribution statement

Zhijun Chen: Data curation, Formal analysis, Methodology, Resources, Writing - original draft, Funding acquisition. **Zhe Lu:** Data curation, Writing - review & editing, Methodology, Software. **Qiushi Chen:** Data curation, Writing - review & edit-

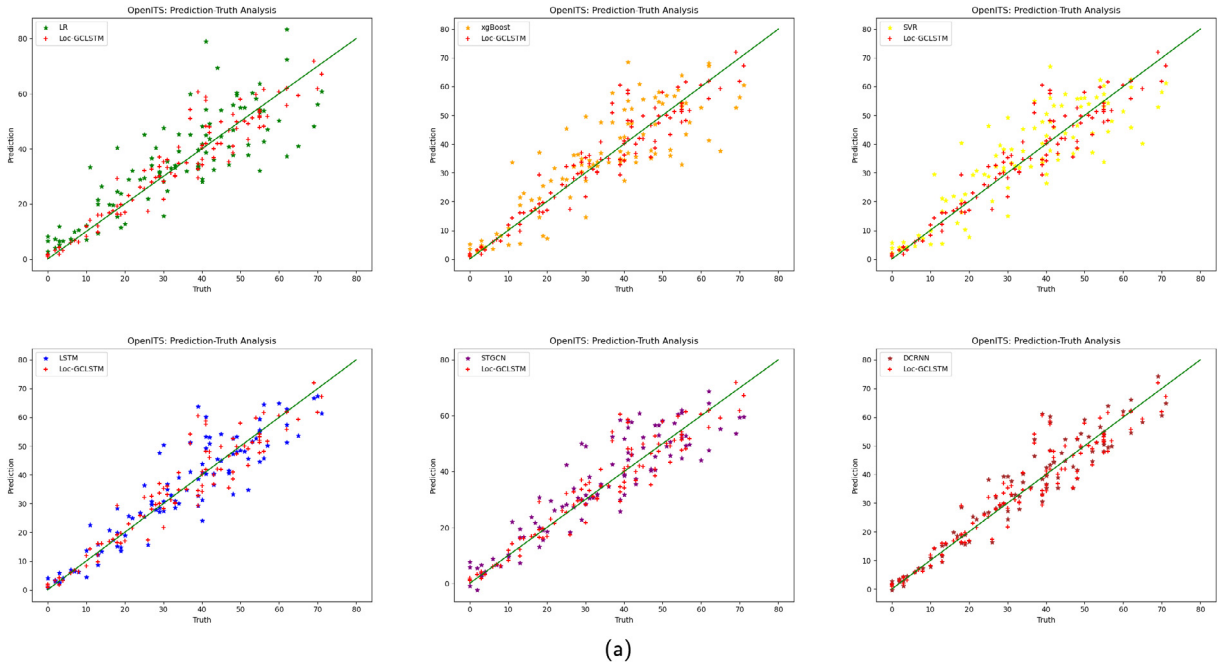

Fig. 12. Random-sampled comparison between the prediction results and the ground truth on (a) OpenITS dataset and (b) METR-LA dataset.

ing, Methodology, Software. **Hongliang Zhong:** Data curation, Investigation, Methodology, Writing - original draft, Software. **Yishi Zhang:** Writing - review & editing, Conceptualization, Formal analysis, Methodology, Supervision, Funding acquisition. **Jie Xue:** Investigation, Resources, Software, Visualization. **Chaozhong Wu:** Conceptualization, Formal analysis, Validation, Funding acquisition.

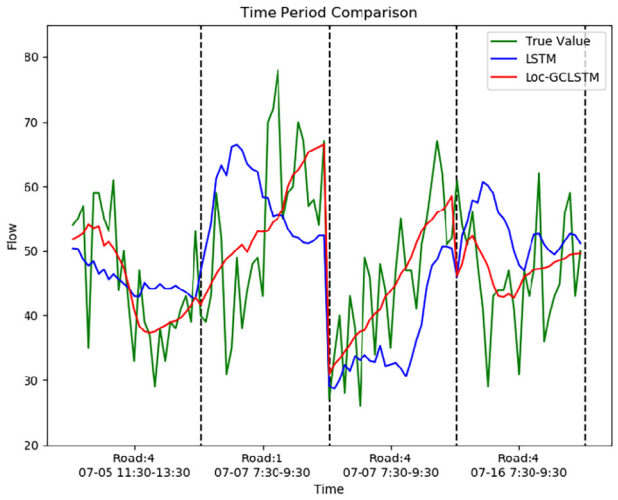

Fig. 13. Comparison between predicted results and ground truth on the OpenITS dataset.

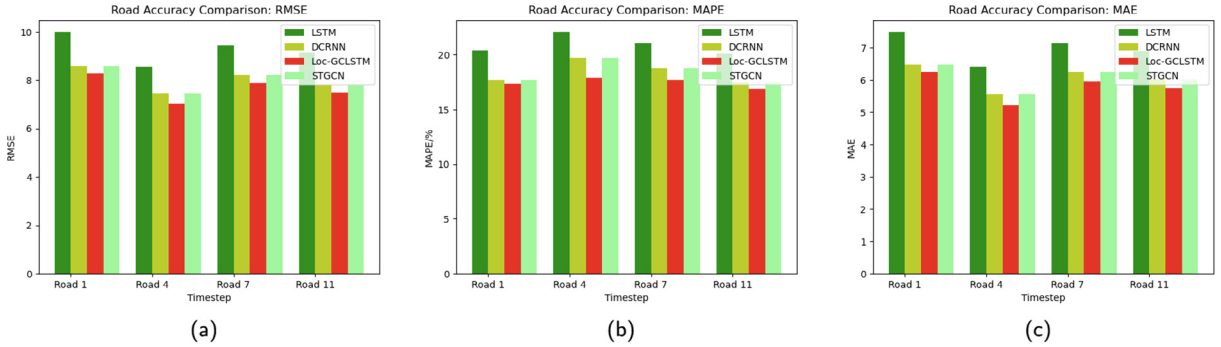

Fig. 14. Comparison of prediction results on each road among Loc-GCLSTM, DCRNN, LSTM, and STGCN. (a) Comparison of RMSE; (b) Comparison of MAPE; (c) Comparison of MAE.

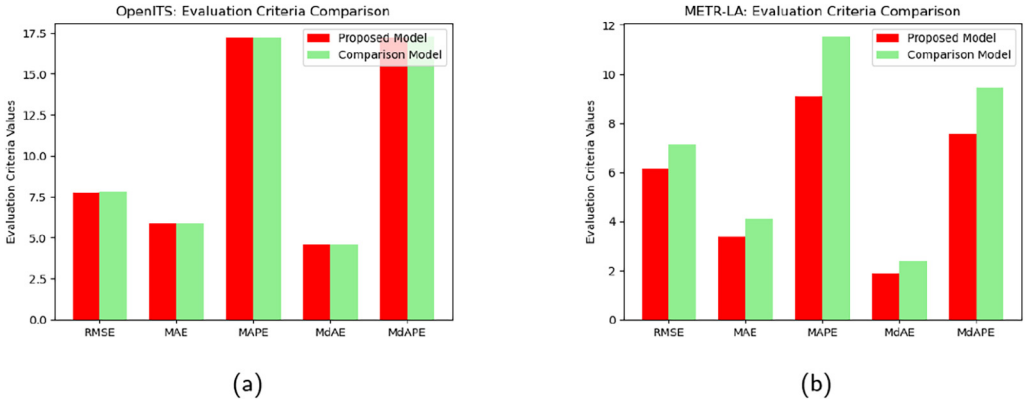

Fig. 15. Effect of the location module on (a) OpenITS dataset and (b) METR-LA dataset.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors wish to thank the anonymous reviewers and editors of the journal for their constructive comments, which improved the quality of this paper.

Appendix A

The appendix describes further analysis of the grid settings for each parameter and its effects on the prediction performance of the proposed model. To optimize the model for more accurate traffic flow prediction results, the parameters of the model must be selected, which is often known as parameter tuning. Several approaches exist for tuning. For our study, we chose the method of grid search, which is an extensive search of a subset of the hyperparameter space of the algorithm [30], i.e., each possible combination of parameters was attempted, and finally the optimal combination of parameters was selected by evaluating the performance o=f each combination according to the evaluation criteria mentioned in the previous section.

To obtain the optimal parameter combination of the proposed model, we conducted experiments on the same dataset as [19], that is, the METR-LA dataset, and searched left and right based on its grid settings when performing the grid search. The grid search hyperparameters considered in this study is listed in Table 4. The specific grid for each model parameter combination and performance evaluation are listed in Table 5. Fig. 16 shows the MAE values of each parameter setting when the number of layers was set to four in the proposed model. Batch size refers to the number of samples traversed before calculating the loss function. We observed that a smaller batch size enables the model to have higher prediction accuracy at the expense of increasing computational cost when units were set to 32, 64, and 128. A similar effect was observed when the number of units was increased. However, when units were set to 256, with the decrease in batch size, the errors on the validation dataset first rapidly decreased and then slightly increased. Considering the sensitivity and margin effects of the aforementioned two parameters on the prediction accuracy and the impact on the computational cost, we set the batch size and units to 64 and 256 in the proposed model, respectively.

Table 4
Grid search hyperparameters settings.

Number	Hyperparameters	Values
1	batch size	16,32,64,128
2	units	32,64,128,256
3	layers	3,4

Table 5
Grid search parameter combination settings and performance evaluation scores when layers = 4.

Layers = 4	Metrics	units = 32	units = 64	units = 128	units = 256
batch size = 16	MSE	39.261	38.501	38.137	39.124
	RMSE	6.260	6.200	6.171	6.251
	MAE	3.460	3.405	3.356	3.371
	MAPE	9.293	9.180	9.082	9.092
	MdAE	1.986	1.938	1.886	1.878
	MdAPE	7.732	7.603	7.526	7.553
batch size = 32	RMSE	39.661	38.607	38.326	38.122
	MSE	6.293	6.209	6.186	6.170
	MAE	3.490	3.422	3.398	3.368
	MAPE	9.358	9.217	9.153	9.047
	MdAE	2.022	1.956	1.934	1.883
	MdAPE	7.769	7.640	7.589	7.480
batch size = 64	MSE	40.684	39.447	38.671	38.015
	RMSE	6.372	6.275	6.231	6.161
	MAE	3.566	3.487	3.426	3.365
	MAPE	9.570	9.382	9.214	9.104
	MdAE	2.105	2.016	1.958	1.898
	MdAPE	7.954	7.776	7.641	7.548
batch size = 128	MSE	41.191	40.216	39.344	38.549
	RMSE	6.412	6.335	6.268	6.204
	MAE	3.606	3.538	3.486	3.419
	MAPE	9.721	9.495	9.355	9.249
	MdAE	2.152	2.068	2.022	1.955
	MdAPE	8.093	7.904	7.791	7.671

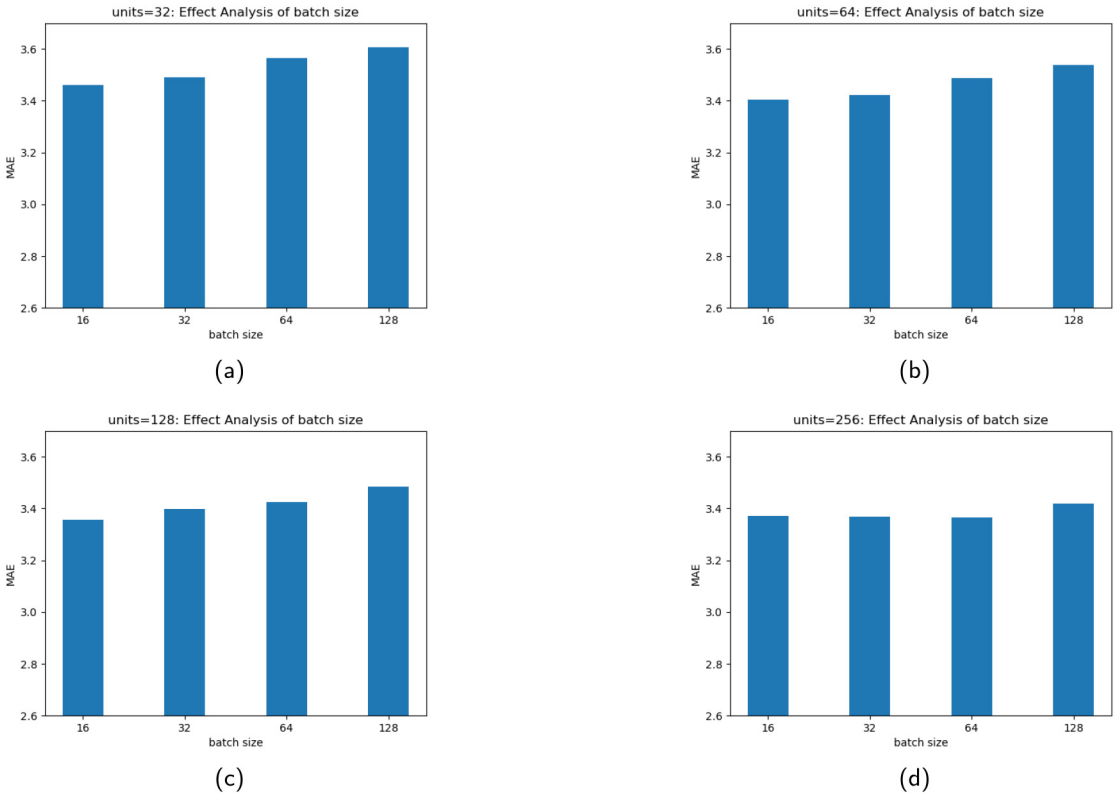

Fig. 16. Effects of different units and batch size setting (a) MAE value of different batch size when units = 32; (b) MAE value of different batch size when units = 64; (c) MAE value of different batch size when units = 128; (d) MAE value of different batch size when units = 256.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Zheng, X., 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467.

[2] A. Ali, Y. Zhu, M. Zakarya, Exploiting dynamic spatio-temporal correlations for citywide traffic flow prediction using attention based neural networks, Information Sciences 577 (2021) 852–870.

[3] B. Chen, D. Sun, J. Zhou, W. Wong, Z. Ding, A future intelligent traffic system with mixed autonomous vehicles and human-driven vehicles, Information Sciences 529 (2020) 59–72.

[4] Chen, Z., Chen, Q., Zhang, J., Zhang, Y., Yang, S., Dong, Y., Chen, C., 2020b. Traffic Flow Prediction Based on Cooperative Vehicle Infrastructure for Cloud Control Platform. Technical Report. SAE Technical Paper.

[5] Z. Cui, R. Ke, Z. Pu, X. Ma, Y. Wang, Learning traffic as a graph: A gated graph wavelet recurrent neural network for network–scale traffic prediction, Transportation Research Part C: Emerging Technologies 115 (2020) 102620.

[6] Cui, Z., Ke, R., Pu, Z., Wang, Y., 2018. Deep bidirectional and unidirectional LSTM recurrent neural network for network–wide traffic speed prediction. arXiv preprint arXiv:1801.02143.

[7] L. Dai, W. Qin, H. Xu, T. Chen, C. Qian, Urban traffic flow prediction: A MapReduce based parallel multivariate linear regression approach, in: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), IEEE, 2014, pp. 2823–2827.

[8] De, S., Mukherjee, A., Ullah, E., 2018. Convergence guarantees for RMSProp and ADAM in non–convex optimization and an empirical comparison to Nesterov acceleration. arXiv preprint arXiv:1807.06766.

[9] X. Dong, T. Lei, S. Jin, Z. Hou, Short–term traffic flow prediction based on XGBoost, in: 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), IEEE, 2018, pp. 854–859.

[10] X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, Y. Liu, Spatiotemporal multi–graph convolution network for ride–hailing demand forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 3656–3663.

[11] Y. Gu, W. Lu, L. Qin, M. Li, Z. Shao, Short–term prediction of lane–level traffic speeds: A fusion deep learning model, Transportation Research Part C: Emerging Technologies 106 (2019) 1–16.

[12] Guo, J., Song, C., Wang, H., 2019. A multi-step traffic speed forecasting model based on graph convolutional LSTM, in: 2019 Chinese Automation Congress (CAC), IEEE. pp. 2466–2471.

[13] K. He, Y. Huang, X. Chen, Z. Zhou, S. Yu, Graph attention spatial–temporal network for deep learning based mobile traffic prediction, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[14] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[15] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J.E., Weinberger, K.Q., 2017. Snapshot ensembles: Train 1, get m for free. arXiv preprint arXiv:1704.00109.

[16] Innamaa, S., 2000. Short–term prediction of traffic situation using MLP–neural networks, in: Proceedings of the 7th World Congress on Intelligent Transport Systems, Turin, Italy, pp. 6–9.

[17] J. Jiang, Q. Chen, J. Xue, H. Wang, Z. Chen, A novel method about the representation and discrimination of traffic state, Sensors 20 (2020) 5039.

[18] Kurbiel, T., Khaleghian, S., 2017. Training of deep neural networks based on distance measures using RMSProp. arXiv preprint arXiv:1708.01911.

[19] Li, Y., Yu, R., Shahabi, C., Liu, Y., 2017. Diffusion convolutional recurrent neural network: Data–driven traffic forecasting. arXiv preprint arXiv:1707.01926.
[20] B. Liao, S. Tang, S. Yang, W. Zhu, F. Wu, Multi–modal sequence to sequence learning with content attention for hotspot traffic speed prediction, Pacific Rim Conference on Multimedia, Springer. (2018) 212–222.
[21] G. Lin, A. Lin, D. Gu, Using support vector regression and k-nearest neighbors for short-term traffic flow prediction based on maximal information coefficient, Information Sciences 608 (2022) 517–531.
[22] S. Lu, Z. Jin, Improved Stochastic gradient descent algorithm for SVM, International Journal of Recent Engineering Science (IJRES) 4 (2017) 28–31.
[23] T. Ma, C. Antoniou, T. Toledo, Hybrid machine learning algorithm and statistical time series model for network–wide traffic forecast, Transportation Research Part C: Emerging Technologies 111 (2020) 352–372.
[24] X. Ma, H. Zhong, Y. Li, J. Ma, Z. Cui, Y. Wang, Forecasting transportation network speed using deep capsule networks with nested LSTM models, IEEE Transactions on Intelligent Transportation Systems (2020).
[25] W. Min, L. Wynter, Real–time road traffic prediction with spatio–temporal correlations, Transportation Research Part C: Emerging Technologies 19 (2011) 606–616.
[26] T.G. Molnár, D. Upadhyay, M. Hopka, M. Van Nieuwstadt, G. Orosz, Delayed lagrangian continuum models for on–board traffic prediction, Transportation Research Part C: Emerging Technologies 123 (2021) 102991.
[27] H. Peng, B. Du, M. Liu, M. Liu, S. Ji, S. Wang, X. Zhang, L. He, Dynamic graph convolutional network for long-term traffic flow prediction with reinforcement learning, Information Sciences 578 (2021) 401–416.
[28] H. Peng, H. Wang, B. Du, M.Z.A. Bhuiyan, H. Ma, J. Liu, L. Wang, Z. Yang, L. Du, S. Wang, Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting, Information Sciences 521 (2020) 277–290.
[29] P. Peng, D.W. Xu, H. Gao, Q. Xuan, Y. Liu, H.F. Guo, et al, Short–term traffic flow prediction using attention–based long short–term memory network, in: 2019 IEEE Fourth International Conference on Data Science in Cyberspace (DSC), IEEE, 2019, pp. 403–409.
[30] I. Priyadarshini, C. Cotton, A novel lstm–cnn–grid search-based deep neural network for sentiment analysis, The Journal of Supercomputing 77 (2021) 13911–13932.
[31] Y. Rajabzadeh, A.H. Rezaie, H. Amindavar, Short–term traffic flow prediction using time–varying Vasicek model, Transportation Research Part C: Emerging Technologies 74 (2017) 168–181.
[32] Y. Song, J. Lu, RNN–based traffic flow prediction for dynamic reversible lane control decision. Data Sci, Knowl. Eng. Sens. Decis. Support 1 (2018) 323–330.
[33] J. Wang, Q. Chen, H. Gong, STMAG: A spatial-temporal mixed attention graph-based convolution model for multi-data flow safety prediction, Information Sciences 525 (2020) 16–36.
[34] Y. Wang, C. Jing, S. Xu, T. Guo, Attention based spatiotemporal graph attention networks for traffic flow forecasting, Information Sciences 607 (2022) 869–883.
[35] B.M. Williams, L.A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results, Journal of Transportation Engineering 129 (2003) 664–672.
[36] T. Wu, F. Chen, Y. Wan, Graph attention LSTM network: A new model for traffic flow forecasting, in: 2018 5th International Conference on Information Science and Control Engineering (ICISCE), IEEE, 2018, pp. 241–245.
[37] T. Wu, K. Xie, G. Song, C. Hu, A multiple SVR approach with time lags for traffic flow prediction, in: 2008 11th International IEEE Conference on Intelligent Transportation Systems, IEEE, 2008, pp. 228–233.
[38] Z.B. Xian, L. Qiang, ARMA–based traffic prediction and overload detection of network, Journal of Computer Research and Development 12 (2002).
[39] G. Xiao, R. Wang, C. Zhang, A. Ni, Demand prediction for a public bike sharing program based on spatio–temporal graph convolutional networks, Multimedia Tools and Applications (2020) 1–19.
[40] Z. Xu, R. Zhu, Q. Yang, L. Wang, R. Wang, T. Li, Short–term bus passenger flow forecast based on the multi–feature gradient boosting decision tree, in: The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, Springer, 2019, pp. 660–673.
[41] Yu, B., Yin, H., Zhu, Z., 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. arXiv preprint arXiv:1709.04875.
[42] F. Yu, D. Wei, S. Zhang, Y. Shao, 3D CNN–based accurate prediction for large–scale traffic flow, in: 2019 4th International Conference on Intelligent Transportation Engineering (ICITE), IEEE, 2019, pp. 99–103.
[43] D. Zhang, M.R. Kabuka, Combining weather condition data to predict traffic flow: A GRU–based deep learning approach, IET Intelligent Transport Systems 12 (2018) 578–585.
[44] J. Zhang, D. Mucs, U. Norinder, F. Svensson, LightGBM: An effective and scalable algorithm for prediction of chemical toxicity–application to the Tox21 and mutagenicity data sets, Journal of Chemical Information and Modeling 59 (2019) 4150–4158.
[45] M. Zhang, X. Fei, Z. Liu, Short–term traffic flow prediction based on combination model of XGBoost–LightGBM, in: 2018 International Conference on Sensor Networks and Signal Processing (SNSP), IEEE, 2018, pp. 322–327.
[46] Z. Zhang, M. Li, X. Lin, Y. Wang, F. He, Multistep speed prediction on traffic networks: A deep learning approach considering spatio–temporal dependencies, Transportation Research Part C: Emerging Technologies 105 (2019) 297–322.
[47] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, H. Li, T–gcn: A temporal graph convolutional network for traffic prediction, IEEE Transactions on Intelligent Transportation Systems 21 (2019) 3848–3858.
[48] Z. Zheng, D. Su, Short–term traffic volume forecasting: A k–nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm, Transportation Research Part C: Emerging Technologies 43 (2014) 143–157.
[49] Y. Zhou, J. Li, H. Chen, Y. Wu, J. Wu, L. Chen, A spatiotemporal hierarchical attention mechanism-based model for multi-step station-level crowd flow prediction, Information Sciences 544 (2021) 308–324.
[50] Z. Zou, P. Gao, C. Yao, City–level traffic flow prediction via LSTM networks, in: Proceedings of the 2nd International Conference on Advances in Image Processing, 2018, pp. 149–153.