A Multi-step Traffic Speed Forecasting Model Based on Graph Convolutional LSTM

Jinlong Guo
College of Control Science and Engineering
Zhejiang University
Hangzhou, China
guojinlong@zju.edu.cn

Chunyue Song*
College of Control Science and Engineering
Zhejiang University
Hangzhou, China
csong@zju.edu.cn

Hui Wang
College of Control Science and Engineering
Zhejiang University
Hangzhou, China
hwangzju@zju.edu.cn

***Abstract*—Traffic speed is an indicator reflecting the traffic state and it is important for traffic control and optimization. Traffic speed forecasting is a typical application of spatial-temporal forecasting problems. However, it is very difficult to accurately predict the traffic speed especially when we need a long-term forecasting result. And how to capture the spatial features is also a huge challenge. In this paper, a sequence to sequence (Seq2Seq) model based on Graph Convolutional LSTM (GC-LSTM) was proposed, which is an end-to-end deep learning model that uses the network topology and historical speed data. The Graph Convolutional Networks (GCN) is used to deal with spatial relationship features while LSTM for temporal relationship features. The Seq2Seq model is a structure that can predict the multi-step traffic speed. Then an experiment on a real-world dataset shows that our model can get a good result for the long-term traffic forecasting problem.**

Keywords—traffic forecasting, graph convolution, GC-LSTM, Seq2Seq

I. INTRODUCTION

In modern society, with the gradual acceleration of urbanization, the number of private cars has increased rapidly, which has caused some severe traffic problems. For example, more vehicles make traffic more congested and make more pollution. A large number of vehicles have also put forward higher requirements for the transportation industry. The intelligent transportation system (ITS) is considered to be an effective way to solve these problems. In the intelligent transportation system, various sensors collect traffic status data and upload it to the cloud. Then, the center can use these data to make appropriate decisions on different issues, which will be passed on to the terminal equipment. These devices execute corresponding instructions to control and optimize traffic conditions. Traffic speed prediction is the application which uses historical traffic speed data to predict future traffic speed. The predicted traffic conditions can provide route guidance suggestions for drivers and optimization basis for traffic policemen. Because of the expansion of the city scale, people travel longer and longer, so the long-term prediction of traffic speed is particularly important.

There are three types of data reflecting traffic state: speed, flow and travel time, and the forecasting methods of them are similar. The traffic state forecasting problem has been developed for decades and there are many models have been proposed. Some of them are traditional models such as the Kalman filter model [1] and the ARIMA model [2]. With the development of machine learning, some intelligent methods have been applied, among which support vector regression (SVR) [3] and neural networks (NN) [4] are two popular models. Zhou [5] proposed a traffic forecasting model based on the recurrent neural network (RNN), which is a deep learning model that can capture the correlation of traffic flow in the time dimension. Chang et al. [6] used KNN and non-parametric regression to predict multi-step traffic flow. However, the above methods are not accurate enough because they only consider temporal features.

In recent years, big data technology has developed vigorously. At the same time, traffic data is no longer limited to time series data. Spatial-temporal data, weather data, and social network data have attracted more and more attention from scholars. Among them, the use of spatial-temporal data has become a research hotspot. Convolutional neural network (CNN) [7] is a model that can analyze and predict the traffic situation of various urban areas from a macro perspective. However, it only regards the city map as an image, and the extraction of grid features can only represent the correlation between the upstream and downstream of the grid area or a single section, not the connectivity of the road network. Graph neural network is a hot field of deep learning and many convolution methods have been proposed [8], [9], [10]. Therefore, some graph-based forecasting models are also introduced in traffic forecasting. Cui et al. [11] defined a kind of traffic graph convolution with physical network topology. Wang et al. [12] used adjacency matrices rather than graph convolutions to represent the connectivity between road segments. However, these methods are still at the level of short-term prediction.

For long-term forecasting, Lv et al. [13] used the stacked autoencoder model (SAE) to capture spatiotemporal features, and it can give the multi-step forecasting results. Zhang et al. [14] considered the impact of upstream sections on downstream sections and used the gradient boosting regression tree (GBRT) to predict the traffic speed of the urban expressway. The forecasting duration can be up to 30 minutes with 5 minutes as a time step. Wu et al. [15] combined attention model, CNN, and RNN to increase the duration of prediction to 45 minutes, and analyzed the role of each network in detail. However, the model can only predict the traffic flow on one-way road segments. For complex road network structures, the model is not applicable. Liao et al. [16] used public social events data and online crowd queries data to increase the accuracy of the traffic forecasting using different models with the Seq2Seq structure. The model includes simple LSTM, DNN-LSTM and graph CNN-LSTM. The model reached a higher accuracy while the graph CNN is very complicated. Li et al. [17] proposed a diffusion convolutional recurrent neural network (DCRNN) to predict

This work was supported partially by the National Key Research and Development Program of China (No.2017YFA0700300), partially by the NSF under grant (61673342) of China, and partially by the Independent Project of State Key Laboratory of Industrial Control Technology (ITC1902).

traffic speed. DCRNN is a spatial-temporal model and the randomness of traffic dynamics can also be captured. But the computational cost of diffusion convolution is still very high.

In this paper, a traffic speed prediction model based on seq2seq is proposed. Its basic unit is graph convolution LSTM (GC-LSTM). The model can predict the multi-step speed data of the whole road network at the same time. In this model, the graph convolution network (GCN) can capture spatial features, and LSTM is used to capture temporal features. Seq2Seq is an encoder-decoder structure, which can simultaneously obtain multi-step prediction results. Besides, to improve the performance of the model, the attention mechanism is also applied.

The rest of this paper is arranged as follows. Section II gives the problem description, introduces the GC-LSTM model and the proposed Seq2Seq structure. Section III is an experiment to analyze the performance of the model and the comparison with other methods. Finally, Section IV summarizes the work of the paper.

II. Problem Description And Model

A. Multi-step Traffic Forecasting Problem

The traffic speed prediction problem is to predict the average speed of each section in the future given historical speed data and some other data, such as road network. A traffic road network can be represented by a directed graph $G = (V, E)$ with N nodes (vertices) $v_i \in V$ and edges $(v_i, v_j) \in E$. We define a section as a node, not an edge, because the value of the node is more predictable, and the edge represents the connection between two sections. Here we can use adjacency matrix $A \in \mathbb{R}^{N\times N}$ to represent the graph topology of the road network. Let A_{ij} be the value of A in row i and column j. If the vehicles can drive from road i to road j, the value of A_{ij} is set to be 1 and $A_{ij} = 0$ otherwise. Let $X_t \in \mathbb{R}^N$ be the value vector of the node on the graph and the value vector is the traffic speed of the road network in this paper. And $\tilde{X}_t \in \mathbb{R}^N$ is the predicted speed. The problem of traffic speed forecasting is to find a function $F(\cdot)$ to predict the future traffic speed of T steps by using historical traffic speed data of T' steps and the adjacency matrix A:

$$[\widetilde{X}_{t+1},...,\widetilde{X}_{t+T}] = F\left([X_{t-T'+1},...,X_t], A\right) \tag{1}$$

B. Graph Convolutional Network

Because spatial correlation has a great influence on traffic status, a graph-based model is needed. Here we use graph convolution network (GCN). There are two types of GCN, one based on spatial convolution [18] and another based on spectral convolution [10]. We use GCN based on spectral convolution because it is more general and needs less calculation.

Firstly, the degree matrix and Laplace matrix are introduced. The degree matrix of adjacency matrix A is a diagonal matrix $D = \text{diag}[D_{11}, D_{22}, \dots, D_{NN}]$ and D_{ii} is obtained by summing the elements of the i^{th} row of matrix A:

$$D_{ii} = \sum_j A_{ij} \tag{2}$$

There are three common Laplacian matrices and the symmetric normalized Laplacian matrix will be applied here:

$$L = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \tag{3}$$

The graph convolution is defined as the convolution of signal x with the filter $g_\theta = \text{diag}(\theta)$:

$$g_\theta * x = U g_\theta U^T x \tag{4}$$

where U is the matrix of eigenvectors of the Laplacian matrix in (3) and $U^T x$ is the graph Fourier transform of x [8], [19]. In the formula, $\theta \in \mathbb{R}^N$ is the parameter that should be learned.

As the matrix multiplication is computationally expensive, the graph convolution can be simplified using Chebyshev polynomials [9]. Reference [10] further simplified the graph convolution with the following expression:

$$g_\theta * x \approx \theta\left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right)x \tag{5}$$

where I_N is the unit matrix whose dimension is the same as A.

According to (5), GCN can be defined with the following layer-wise propagation rule:

$$H^{(l+1)} = \tanh\left(\widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{6}$$

$H^{(l)} \in \mathbb{R}^{N\times l}$ is the hidden state matrix of the l^{th} layer and $W^{(l)}$ is the parameter matrix which should be learned. Considering self-connections, $\tilde{A} = A + I_N$. $\widetilde{D}$ is the degree matrix of $\tilde{A}$. Here, $\tanh(\cdot)$ is the activation function which maps the input to [-1, 1].

Single-layer GCN means that only one step from the target node is considered. If we want to consider the K^{th}-order neighborhood, a K-layer GCN should be applied.

Since the adjacency matrix of a directed graph is an asymmetric matrix, GCN cannot be applied directly. Here, we split each node into two nodes: "in-node" and "out-node". Here we let $\hat{A} \in \mathbb{R}^{2N\times 2N}$ be the new adjacency matrix with self-connections, then for node v_i, $\hat{A}(i, i + N) = 1$ and $\hat{A}(i + N, i) = 1$, which means $v_{i,out}$ is connected with $v_{i,in}$, and the adjacency matrix is a symmetric matrix. For instance, if $\tilde{A}(1, 2) = 1$ means that node v_1 points to node v_2, then $\hat{A}(N + 1, 2) = 1$ and $\hat{A}(2, N + 1) = 1$. $\hat{A}$ can be expressed with the following expression:

$$\widehat{A} = \begin{bmatrix} 0 & \widetilde{A}^T \\ \widetilde{A} & 0 \end{bmatrix} \tag{7}$$

Since the dimension of the adjacency matrix has been extended to $2N\times 2N$, the dimension of X_t should also be extended with $X_t' = [X_t; X_t]$. Let $\widehat{D}$ be the degree matrix of $\hat{A}$ and $H^{(l)} \in \mathbb{R}^{2N\times l}$. The layer-wise propagation rule of GCN is:

$$H^{(l+1)} = \tanh\left(\widehat{D}^{-\frac{1}{2}}\widehat{A}\widehat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{8}$$

C. Graph Convolutional LSTM

Traffic speed data is a time series, so we use the long short-term memory network (LSTM) [20] to capture the temporal features. The propagation rule of LSTM is:

$$\begin{aligned} f_t &= \sigma\left(W_f \bullet [h_{t-1}, x_t] + b_f\right) \\ i_t &= \sigma\left(W_i \bullet [h_{t-1}, x_t] + b_i\right) \\ o_t &= \sigma\left(W_o \bullet [h_{t-1}, x_t] + b_o\right) \\ \widetilde{C}_t &= \tanh\left(W_C \bullet [h_{t-1}, x_t] + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{9}$$

where h_t is the output state vector and C_t is the hidden state vectors and x_t is the input vector of LSTM at time t. The input and output of LSTM are controlled by three gates. f_t is the forget gate which is responsible for finding how much of the historical state needs to be abandoned. i_t is the input gate that needs to find how much of the new state needs to be passed to the next step. o_t is the output gate that determines how the state is output. $\sigma(\cdot)$ is the sigmoid function that maps the values of gates to [0, 1]. The forget gate decides how much memory to forget and the input gate decides how much state to update. The output gate controls h_t according to C_t. In most cases, h_t is not the final output we need, but the input of the output network to obtain the final output. The output network can be a single-layer neural network.

To capture both temporal and spatial correlations, we proposed Graph Convolution LSTM (GC-LSTM) by replacing the multiplications in LSTM with graph convolution:

$$\begin{aligned} f_t &= \sigma\left(GCN_f\left([h_{t-1}, x_t]\right) + b_f\right) \\ i_t &= \sigma\left(GCN_i\left([h_{t-1}, x_t]\right) + b_i\right) \\ o_t &= \sigma\left(GCN_o\left([h_{t-1}, x_t]\right) + b_o\right) \\ \widetilde{C}_t &= \tanh\left(GCN_C\left([h_{t-1}, x_t]\right) + b_C\right) \\ C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned} \tag{10}$$

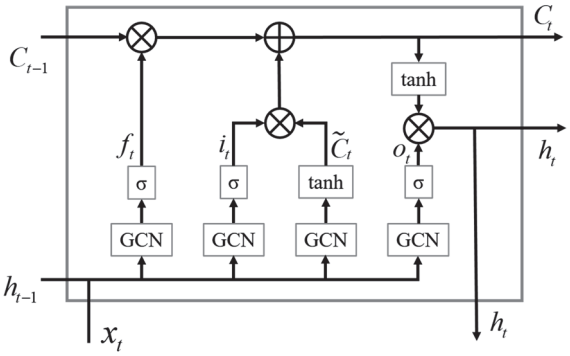

Fig. 1. Structure of GC-LSTM cell

GCN(·) in (10) is a multi-layer GCN model by stacking (8). The structure of GC-LSTM cell is shown in fig. 1.

D. Seq2Seq Model and Attention

For multi-step forecasting, a sequence to sequence (Seq2Seq) [21] structure is applied here.

Seq2Seq has an encoder and a decoder. The historical traffic speed data are the inputs of the encoder, and the outputs are the state vector h_t and C_t. Then we use a linear regression layer (LR) to transfer h_t to predicted speed $\tilde{X}_t$. For each step in the decoder, the input at step t are h_t, C_t and $\tilde{X}_t$, the output is $\tilde{X}_{t+1}$. Fig. 2 shows the Seq2Seq model. The GC-LSTM cell at every step is the same, which means there is only one GC-LSTM cell in Seq2Seq and the cell is reused at every step.

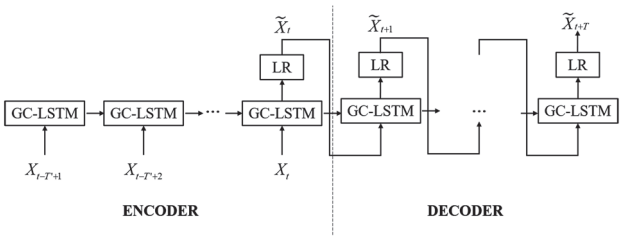

Fig. 2. Seq2Seq model with GC-LSTM cell

The output layer is a linear regression layer that maps hidden state to output speed:

$$\widetilde{X}_t = \tanh\left(W_h \bullet h_t + b_h\right) \tag{11}$$

Seq2Seq model can get a multi-step speed sequence. However, when the input sequence is too long, the Seq2Seq model may not capture very long-term features. Therefore, the attention mechanism [22] is used to enhance the ability of Seq2Seq to capture long-term features. The idea of attention mechanism is to design a model to pay more attention to the important state of the encoder. The model with the attention mechanism is shown in Figure 3.

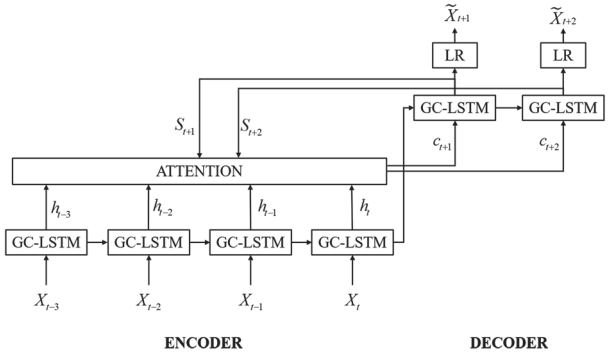

Fig. 3. Seq2Seq model with the attention mechanism. Here, as an example, let the encoder length equal 4 and the decoder length equal 2.

Here, let $m \in [t\text{-}T'\text{+}1, t]$ denotes the time step of the encoder, and $n \in [t\text{+}1, t\text{+}T]$ denotes that of the decoder. To avoid confusion, let S_n denotes the hidden state of the decoder at time n. As shown in fig. 3, the input at time step n of the decoder is calculated by the following expression:

$$c_n = \sum_{m=t-T'+1}^{t} \alpha_{nm} h_m \tag{12}$$

where α_{nm} is the weight that how much h_m contributes to c_n. α_{nm} can be calculated using a softmax function:

$$\alpha_{nm} = \frac{\exp(e_{nm})}{\sum_{k=t-T'+1}^{t} \exp(e_{nk})} \tag{13}$$

e_{nm} is the output of the function $G(S_{n-1}, h_m)$, which is always a linear regression layer or a shallow neural network. Here we use a linear regression layer:

$$e_{nm} = \sigma\left(W \bullet [S_{n-1}, h_m] + b\right) \tag{14}$$

III. Experiment

A. Dataset Description

The traffic speed data which we use are from [12]. The speed data were gathered from April 1, 2015 to April 30, 2015. The urban road network contains 156 road segments. The time interval is 10 minutes so the dataset is a 156×30×144 tensor. The network topology is represented by a file containing the connections between every two sections. We use the data of the first 22 days as the training set and the data from April 23 to April 25 as the validation set. The data of the last 5 days are used for the test.

Our research goal is to get the results of multi-step forecasting, so the input is a time series of 12 steps which are the speed data for 2 hours, and the output is a 6-step series for each road segment. So there are 126 time series every day. For example, we can use the data from 7:00 to 8:50 to predict the speed from 9:00 to 9:50. At the same time, we can also use the speed from 7:10 to 9:00 to predict that from 9:10 to 10:00.

B. Experiment Settings

1) Data Process

First, we need to process the data. Fortunately, our dataset has completed the processing of missing and outliers, so only normalization and dimensional transformations are required.

Since the traffic speed data in the training set may not contain the upper and lower limits of the actual speed, and the speed limit of each road is unknown, so the method of min-max normalization cannot be used. Here, we apply the z-score normalization, and the processed data conform to the standard Gaussian distribution:

$$X_i^{'} = \frac{X_i - \mu_{Xi}}{\sigma_{X_i}} \tag{15}$$

where μ_{Xi} is the speed average of the road i and σ_{Xi} is the variance.

Then the raw data needs to be converted into the input dimension acceptable to the model. The input is a 156×12 matrix and the output's shape is 156×6, and there are 126 pairs of data every day.

2) Baselines

The baseline models for comparison are as follows:

HA: Historical average forecast, using average values of past steps to predict the next step result.

Xgboost: A gradient boosting algorithm based on classification and regression tree (CART). Xgboost requires that loss functions should be derivable in the first and second order [23].

Seq2Seq + LSTM: We use a 3-layer Seq2Seq model based on LSTM.

GRNN: Graph recurrent neural network which uses GRU instead of LSTM and just multiplies the adjacency matrix with the hidden vectors as the representation of the graph [12].

3) GC-LSTM Model

In this paper, the Seq2Seq model based on GC-LSTM unit has 3 layers and the GCN model is 2 layers. For regression problems, MSELoss function is usually used as the loss function:

$$e = MSELoss\left(y_t^i, \hat{y}_t^i\right) = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(y_t^i - \hat{y}_t^i\right)^2 \tag{16}$$

where y_t^i denotes the real speed data of road i at time step t and $\hat{y}_t^i$ is the predicted speed. N and T are the number of road segments and time steps, respectively.

It is not easy to select suitable optimization functions for complex deep learning models. We tried many methods and then chose the Adam method [24], which is an adaptive learning method. Adam method has become the most popular method because of its high precision, fast and other advantages. Another advantage is that it does not need to adjust a lot of hyperparameters. In most cases, the default values in reference [24] can achieve good results.

4) Evaluation

Here three evaluation indicators are introduced to evaluate the model's performance: mean absolute percentage error (MAPE), mean absolute error (MAE) and root mean square error (RMSE). The definitions are:

$$MAPE = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left|\frac{y_t^i - \hat{y}_t^i}{y_t^i}\right| \times 100\% \tag{17}$$

$$MAE = \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left|y_t^i - \hat{y}_t^i\right| \tag{18}$$

$$RMSE = \sqrt{\frac{1}{NT}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(y_t^i - \hat{y}_t^i\right)^2} \tag{19}$$

C. Results and Analysis

We use Pytorch 1.0 to implement the proposed model, and the results are shown in Table. I.

From Table. I, the GC-LSTM model can achieve the highest accuracy of the above models, and the performance of the Seq2Seq structure is better than the others. HA, Xgboost and GRNN are single-step forecasting methods, so it is necessary to add the prediction results of time step t-1 to historical data when predicting the speed of time step t. This operation accumulates single-step errors, which can result in huge errors after multiple steps. For the Seq2Seq model, the historical information can be transmitted to the last step, and

the multi-step prediction model is optimized by reverse derivation to make it more suitable for the multi-step prediction problem.

TABLE I. PERFORMANCE OF DIFFERENT MODELS

Model	Evaluation Index		
	MAE	*RMSE*	*MAPE*
HA	3.56	4.95	17.53%
Xgboost	4.18	6.19	20.47%
GRNN	4.28	5.62	21.99%
Seq2Seq+LSTM	3.22	4.52	16.38%
Seq2Seq+GC-LSTM	**3.11**	**4.46**	**14.60%**

We show the MAE, RMSE and MAPE for each step of prediction in Fig. 4 to Fig. 6.

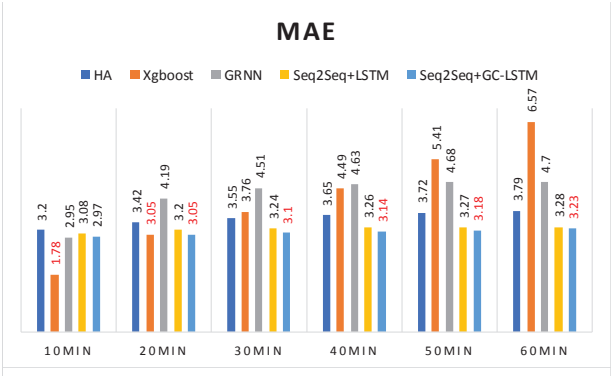

Fig. 4. MAE of the mentioned models at each step

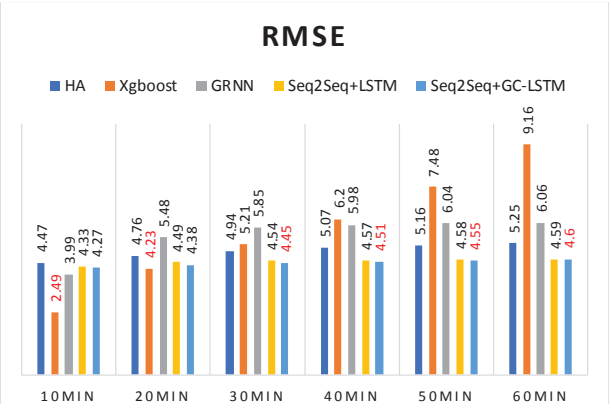

Fig. 5. RMSE of the mentioned models at each step

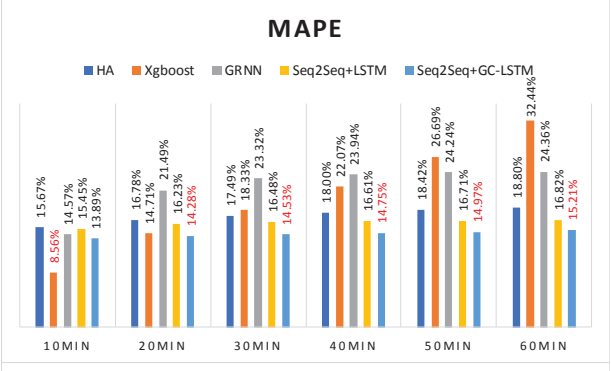

Fig. 6. MAPE of the mentioned models at each step

As shown in the figures, the proposed model in this paper performs the best among the model after the second step, and the error increases very slowly over time. These results prove that our model is highly accurate and stable under multi-step prediction conditions.

The accuracy of the first step of Xgboost is the highest, and that of the second step is similar to that of GC-LSTM. However, due to the cumulative error, its performance is not good after the second step. In contrast, the error of each step of the Seq2Seq model is relatively stable.

The GRNN model performed well in reference [12] but not in this paper, because the author used 144-step historical data, which equals one day, to predict only one step. So GRNN performed well in single-step forecasting. At the same time, GRNN training takes the longest time among the models. Compared with GC-LSTM, GRNN's ability to capture spatial features is poorer.

IV. CONCLUTION

This paper presents a multi-step traffic speed prediction model based on GC-LSTM. Firstly, GC-LSTM is introduced which is the combination of GCN and LSTM. The role of GCN is to capture spatial features and LSTM for temporal features. To predict the multi-step traffic speed, the Seq2Seq model is introduced, with GC-LSTM as the basic unit. The attention mechanism is also introduced here to achieve better accuracy for the long sequence. Then an experiment is introduced. The results show that GC-LSTM within the Seq2Seq structure can achieve better performance than HA, Xgboost, GRNN and normal Seq2Seq for multi-step prediction problems.

For future work, some other data may be introduced, such as weather data and some basic information of road sections. At the same time, if some statistical models of traffic engineering are combined with deep learning models, the prediction accuracy may be higher and the model interpretability will be better. For the model, the attention mechanism can replace the RNN model to realize parallel and accelerate computing. Besides, the combination of graph and attention mechanism in the spatial dimension is an exciting direction.

REFERENCES

[1] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through kalman filtering theory," Transportation Research Part B: Methodological, vol. 18, no. 1, pp. 1-11, February 1984.

[2] M. Van Der Voort, M. Dougherty and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," Transportation Research Part C: Emerging Technologies, vol. 4, no. 5, pp. 307-318, October 1996.

[3] C. H. Wu, J. M. Ho and D. T. Lee, "Travel-time prediction with support vector regression," in IEEE Transactions on Intelligent Transportation Systems, vol. 5, no. 4, pp. 276-281, December 2004.

[4] E. I. Vlahogianni, M. G. Karlaftis and J. C. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach," Transportation Research Part C: Emerging Technologies, vol. 13, no. 3, pp. 211-234, August 2005.

[5] C. Zhou, P. C. Nelson, "Predicting traffic congestion using recurrent neural networks," 9th World Congress on Intelligent Transport SystemsITS America, ITS Japan, ERTICO (Intelligent Transport Systems and Services-Europe), October 2002.

[6] H. Chang, Y. Lee, B. Yoon, S. Baek, "Dynamic near-term traffic flow prediction: system-oriented approach based on past experiences," IET Intelligent Transport Systems, vol. 6, no. 3, pp. 292-305, September 2012.

[7] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, "DNN-based prediction model for spatio-temporal data," Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 92:1-92:4, October 2016.

[8] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, "Spectral networks and locally connected networks on graphs," arXiv preprint arXiv: 1312.6203, December 2013.

[9] M. Defferrard, X. Bresson, P. Vandergheynst, "Convolutional nerral networks on graphs with fast localized spectral filtering," Advances in Neural Information Processing Systems, pp. 3844-3852, December 2016.

[10] T. N. Kipf, M. Welling, "Semi-supervised classification with graph convolutional networks," International Conference on Learning Representations (ICLR '17), April 2017.

[11] Z. Cui, K. Henrickson, R. Ke, Y. Wang, "Traffic graph convolutional recurrent neural network: a deep learning framework for network-scale traffic learning and forecasting," arXiv preprint arXiv:1802.07007v2, November 2018.

[12] X. Wang, C. Chen, Y. Min, J. He, B. Yang, Y. Zhang, "Efficient metropolitan traffic prediction based on graph recurrent neural network," arXiv preprint arXiv:1811.00740, November 2018.

[13] Y. Lv, Y. Duan, W. Kang, Z. Li, F. Y. Wang, "Traffic flow prediction with big data: a deep learning approach," IEEE Trans. Intelligent Transportation Systems, vol. 16, no. 2, pp. 865-873, September 2014.

[14] S. Zhang, Z. Zhan, X. Chen, "Gradient boosting regression tree for traffic flow prediction considering temporal and spatial correlations," 17th COTA International Conference of Transportation Professionals, pp. 2444-2454, July 2017.

[15] Y. Wu, H. Tan, L. Qin, B. Ran, Z. Jiang, "A hybrid deep learning based traffic flow prediction method and its understanding," Transportation Research Part C: Emerging Technologies, vol. 90, pp. 166-180, May 2018.

[16] B. Liao, J. Zhang, C. Wu, D. McIlwraith, T. Chen, S. Yang, Y. Guo, F. Wu, "Deep sequence learning with auxiliary information for traffic prediction," KDD '18 Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 537-546, August 2018.

[17] Y. Li, R. Yu, C. Shahabi, Y. Liu, "Diffusion convolutional recurrent neural network: data-driven traffic forecasting," International Conference on Learning Representations (ICLR '18), May 2018.

[18] M. Niepert, M. Ahmed, K. Kutzkov, "Learning convolutional neural networks for graphs," International Conference on Machine Learning, pp. 2014-2023, June 2016.

[19] D. Shuman, S. Narang, P. Frossard, A. Ortega, P. Vandergheynst, "The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains," IEEE Signal Processing Magazine, vol. 30, no. 3, pp. 83-98, April 2013.

[20] S. Hochreiter, J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, November 1997.

[21] I. Sutskever, O. Vinyals, Q. V. Le, "Sequence to sequence learning with neural networks," Advances in Neural Information Processing Systems, pp.3104-3112, December 2014.

[22] D. Bahdanau, K. Cho, Y. Bengio, "Neural machine translation by jointly learning to align and translate," International Conference on Learning Representations (ICLR '15), May 2015.

[23] T. Chen, C. Guestrin, "Xgboost: A scalable tree boosting system," KDD '16 Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 785-794, August 2016.

[24] D. P. Kingma, J Ba. "Adam: A method for stochastic optimization," International Conference on Learning Representations (ICLR '15), May 2015.