



Traffic Flow Prediction Model and Performance Analysis Based on Recurrent Neural Network

Haozheng Wu¹, Yu Tang¹, Rong Fei^{1(✉)}, Chong Wang²,
and Yufan Guo¹

¹ Xi'an University of Technology, Xi'an 710048, China
annyfei@xaut.edu.cn

² University of Electronic Science and Technology of China, Chengdu, China

Abstract. Traffic flow has the characteristics of randomness, nonlinearity and periodicity. Using deep learning as the method of traffic flow prediction can make full use of traffic flow characteristics for prediction. In this paper, a recurrent neural network-LSTM (Long Short-Term Memory) model and its variant GRU (Gated Recurrent Units) are used to solve the traffic flow prediction problem. According to the road traffic information collected by the automatic measurement station (LAM) of the Dutch traffic management department, the vehicle traffic flow data has been predicted every five minutes. It's verified that the GRU traffic flow prediction model performs well by comparing the prediction accuracy of LSTM and GRU.

Keywords: LSTM · GRU · Traffic flow

1 Introduction

Traffic flow refers to the vehicle flow formed by motor vehicles driving on the urban roads. It involves three parameters such as the number of vehicles currently driving on the road, speed of vehicle and road density [1]. The change laws of the can be studied to discover various characteristics of traffic flow, which are currently the first choice for many scholars to study traffic flow problems, and which are also an important basis for using intelligent transportation systems for traffic control and guidance [2].

At present, as there are few public datasets in China, and the traffic databases of most cities are still under construction. A large amount of high-quality data about traffic flow can't be obtained. Therefore, in this paper, the data used in the experiment comes from the Finnish Public Transport Authority, using the automatic measurement stations (LAM) to collect road traffic information. There are currently about 500 active LAM sites in Finland. Information is shared between the original form and the completed report.

This paper mainly analyzes and studies the LSTM and GRU models to effectively predict the short-term traffic flow, and compares their performance. First, the characteristics of the short-term traffic flow data in LAM points: influencing factors, processes and other characteristics are analyzed. In view of the characteristics such as

dependence, timing and nonlinearity of flow data, a short-term traffic flow prediction model based on recurrent neural network LSTM and GRU was designed, and experiments have verified that performance of GRU is better.

2 Related Theories

LSTM (Long short term memory) is a neural network improved the defects of recurrent neural network, which adds a “processor” to the RNN neurons to determine whether the current information is to be forgotten [3]. The structure of this processor is also named cell, thus making the RNN network to have a long-term memory function and solving the problem of handling long-distance dependence of RNN [9], which is proposed by Hochreiter and Schmidhuber (1997). There are four controllable gates (the forgotten gate, input gate, candidate gate, and output gate) in each memory cell of the hidden layer, and the sequential data are interacted and transmitted in a unique way between them. As shown in Fig. 1.

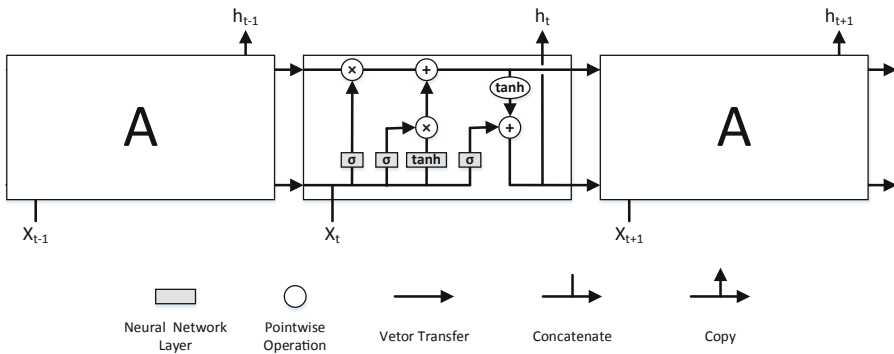


Fig. 1. The sketch of LSTM.

In the Fig. 1, vectors means to transfer, concatenate, and copy the data, as well as input and output. The circle represents the addition, multiplication and other operations of the corresponding vector, and the matrix A is the learned neural network layer.

The most important feature of LSTM is the existence of cell state. Combining the characteristics of the cycle, the cycle of the cell state is like a chain on which the information runs. There are only a few interactions, and the information can be continuously transferred. The controllable gate allows information to pass through selectively, which contains a sigmoid function and a vector multiplication operation. The Sigmoid function can receive the input value and then output a value between 0 and 1, which describes how much the current input value can pass through the gate. 0 means “not allow any quantity to pass” and 1 means “allow all quantities to pass” [4–6].

In the process of LSTM training, the first step of inputting information is to pass the forgotten gate, and then decide the information to forget from the cell state, that is, read h_{t-1} and x_t , and then output a number between [0, 1] to each cell state C_{t-1} , among

which, 1 means “completely reserved” and 0 means “completely abandoned”. The structure of the forgotten gate is shown in Fig. 2 below.

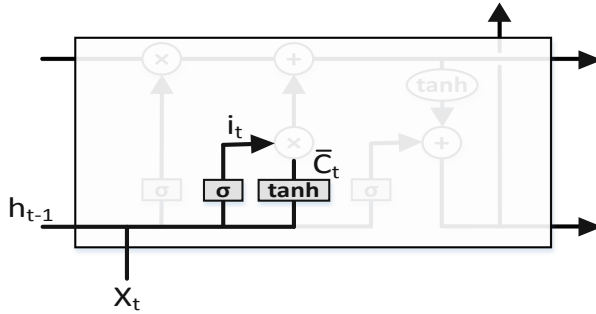


Fig. 2. The substructure of forgotten gate and function.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{1}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2}$$

where h_{t-1} indicates historical information, and x_t indicates new information currently flowing into the cell, and decides which historical information to forget.

The second Step of the training process is to prepare for updating the cell state, divided into two parts. The first part is the input gate where input data is saved into the cell unit. The second part is to create a new candidate vector and store the output value of the previous cell state, which can be compared to memorizing information of the previous cell state. Just like \tilde{C}_{t-1} will be added to the state, combining these two parts to create a new memory, which is the next step, using two pieces of information to update the cell state. The structure of input gate is shown in the Fig. 3.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

The third step is to update the old cell status, that is, update C_{t-1} to C_t . By multiplying C_{t-1} and f_t , the forgotten information is selected and forgotten. Adding $i_t * \tilde{C}_t$, this is the way to obtain C_t , and then update C_{t-1} .

The last Step is to pass the output gate. The information is filtered to determine the information needs to be output, by running a sigmoid function. Then, after processing the cell state through the tanh function, a value between $[-1, 1]$ can be obtained, which after being multiplied with the output of the sigmoid function can determine and output the information we need (Fig. 4).

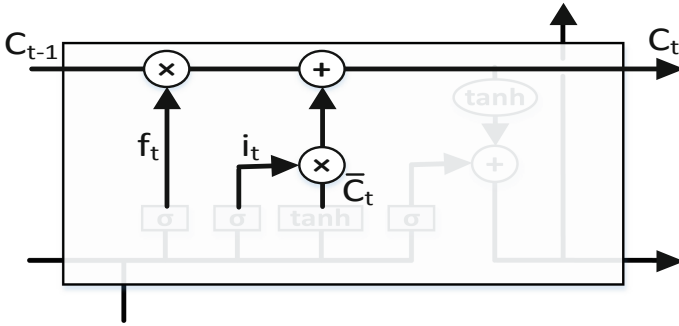


Fig. 3. Update old cell status and functions.

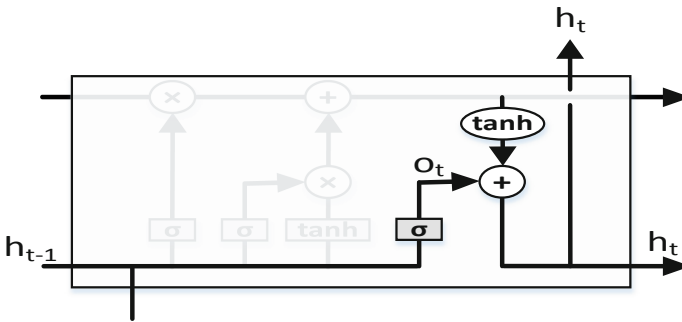


Fig. 4. The output process and function.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * \tanh(C_t) \tag{6}$$

A summary is as follow. The forget gate can determine the information to be abandoned in the memory cell, that is, how much information in the memory cell C_{t-1} at the previous moment can be transferred to the memory cell C_t at the current moment. The main function of the input gate is to control the input information to enter the memory cell C_t at the current moment selectively. The main function of the output gate is to determine how much information in the current memory cell C_t can enter the hidden layer state h_t . Use the forgotten gate and the input gate to update the memory cell, and use the tanh function to generate the candidate memory cell, that is, the information about to enter the output gate.

LSTM controls the current memory cell to forget, memory, update and output through the interaction of the four gates and also realize the continuous accumulation of historical information, that is, memory, through continuous cyclic update.

The training algorithm of LSTM is still BPTT, and the output value of each nerve is calculated forward, that is, the values of the five vectors f_t, i_t, C_t, O_t, S_t . The parameters that LSTM needs to learn and update are the weight matrix and the bias term of the four gates.

The state calculation formula of RNN is $S_t = f(W * S_{t-1} + U * x_t)$. According to the chain rule, the gradient will change to a multiplicative form, and the function value of sigmoid is less than 1, and the multiplicative form will lead to the situation that eventually approaches 0. In order to solve this problem, scientists used the accumulation form $S_t = \sum_{\tau=t}^t \nabla S_\tau$, whose derivative is also accumulation, so as to avoid the disappearance of the gradient. LSTM is used the accumulation form which can be seen from $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$.

The above is the basic principle of the LSTM model, and GRU is a variant of LSTM, proposed by Cho et al. [7, 8] in 2014. GRU only contains update and reset gates, which is z_t and r_t in the figure below. The role of the update gate is to determine how much information at the previous moment can enter the current cell state. The larger the value of the update gate is, the more information that enters the current cell state will be. The role of the reset gate is to select how much information to forgot at the previous moment. The larger the value of the reset gate is, the less information will be forgotten.

Through the Fig. 5, analogy with the LSTM model, the process and formula of its forward propagation can be obtained:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{7}$$

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{8}$$

$$\tilde{h}_t = \tanh(W_{\tilde{h}} [r_t * h_{t-1}, x_t]) \tag{9}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{10}$$

$$y_t = \sigma(W_o \cdot h_t) \tag{11}$$

z_t is the update gate, the logic gate when updating activation. r_t represents reset gate, and decides whether to abandon the previous activation h_t when candidate activation, \tilde{h}_t represents candidate activation to receive $[x_t, h_{t-1}]$, and h_t represents activation the hidden layer of GRU, to receive $[h_{t-1}, \tilde{h}_t]$.

The update gate mainly performs linear transformation on the information at the previous time and the current time, that is, multiplies the weight matrix by the right, and then enter the added data into the update gate, that is, multiplies the sigmoid function, and the value obtained is between $[0, 1]$.

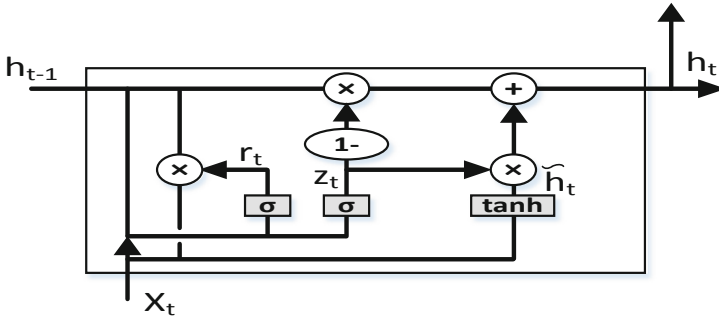


Fig. 5. The output process and function.

The main function of the reset door is to determine how much historical information cannot be passed on to the next moment. Similar to the data processing of the update gate, the information at the previous time and the current time is linearly transformed, that is, the weight matrix is respectively multiplied by the right, and then the added data is sent to the reset gate, which is multiplied by the sigmoid function and obtain the resulting value between [0, 1]. The difference lies in the value and use of these two weight matrix.

GRU no longer uses separate memory cells to store memory information, but directly uses hidden units to record historical states, using the reset gate to control the data volume of current information and memory information and generating new memory information to continue to forward. Because the output of the reset gate is between [0, 1], the amount of data that can continue to be transferred forward by using the reset gate to control the memory information. When the reset gate is 0, that means all the memory information is cleared, otherwise the reset gate is 1, it means all the memory information has passed.

The update gate is used to calculate the output of the hidden state at the current moment. The output information of the hidden state includes the hidden state information h_{t-1} at the previous moment and the hidden state output h_t at the current moment. The update gate is used to control the amount of data to pass on. The training method of GRU is similar to LSTM.

3 Model Design

3.1 Experimental Environment and Dataset

Operating System: Windows 10. Hardware platform: a universal PC. Development tools: Python3, Tensorflow, Keras.

Dataset: The one-month data from January 1, 2019 in Finland is used as the training set, and the data of the following week is used as the test set [9].

3.2 Experimental Environment and Dataset

In this paper, in order to compare the performance differences between the two models, the same construction method is used for both models, that is, they both involve a neural network including two hidden layers. Python is used for development, with Keras, a high-rise neural network API based on Tensorflow, Theano and CNTK backends. Using Keras framework, the speed we build models has been accelerated. This paper first generates a standardized object scaler from the processed training data and test data, using the object to standardize the dataset. Because the prediction task of sequential data needs to use historical data to predict the future data, the training data needs to be divided. A time-lagged variable is set up that lags = 12, the historical data of the previous one hour is used to predict the data of the next time node, and finally the training data set becomes the form to be (samples, lags).

The data set divided still has the characteristics of time sequence in order. Even though keras can choose to shuffle the data during training, the execution order is to sample the VAL data before shuffling, and the sampling process is still in order. Therefore, NP.random.shuffle is used to shuffle the data in advance to disrupt the order of the data.

Next is the construction of the model. Each hidden layer of the model contains 64 neurons. A dropout layer is added between the hidden layer and the output layer to reduce interdependence between nodes and realize the regularization of the neural network. Finally, the output layer contains 1 neural unit, that is, a number should be output.

There have been set 2 hidden layer LSTM2 and 2 hidden layer GRU networks.

LSTM and GRU are trained according to the normal RNN network, using the train_model () function to train ans RMSprop (lr = 0.001, rho = 0.9, epsilon = 1e-06) as the optimizer, with the batch_szie to be 256, epochs to be a total of 600, and lags to be 12 (scilicet time-lagged length is one hour).

Model regression prediction results are evaluated using several indicators as MAE, MSE, RMSE, MAPE, R2, explained_variance_score.

After the model training is completed, the processed test set is input into the model for prediction to obtain prediction accuracy, and the prediction result is shown in the Table 1. The results of evaluation indicators are as follows:

Table 1. Table captions should be placed above the tables.

	LSTM	GRU
MAPE	31.003164%	24.585686%
MAE	6.691722	6.703402
MSE	83.585697	85.958260
RMSE	9.142521	9.271368
R2	0.932984	0.931082
Explained_variance_score	0.933180	0.931228

In order to obtain the general rule, the parameters of the two models were adjusted, and the same dataset is used to train and test again. The process is the same as above. In this experiment, the optimizer was adjusted and adopted adam optimization (Lr = 0.001, beta_1 = 0.9, beta_2 = 0.999, epsilon = 1e-08) to train the model. The results of evaluation indicators are as follows (Table 2):

Table 2. The result of evaluation indicators.

	LSTM	GRU
MAPE	26.143534%	27.275156%
MAE	6.561796	6.570125
MSE	81.740679	81.411213
RMSE	9.041055	9.022816
R2	0.934463	0.934727
Explained_variance_score	0.934714	0.934815

At the same time, a three-layer LSTM and a GRU neural network model were also chosen to be built to compare the performance differences between the two models. Inputting the same dataset for training and testing, this experiment still used the RMSprop optimizer, compared with the results of the two-layer neural network:

The results of evaluation indicators are as follows (Table 3):

Table 3. The result of evaluation indicators.

	LSTM	GRU
MAPE	24.241568%	25.885307%
MAE	6.806903	7.819329
MSE	89.589734	125.560051
RMSE	9.465185	11.205358
R2	0.928170	0.899330
Explained_variance_score	0.928381	0.900794

Then change the optimizer used for training, the adma optimizer was used to train and test again. The following results were obtained:

The results of evaluation indicators are as follows (Table 4):

Table 4. The result of evaluation indicators.

	LSTM	GRU
MAPE	24.209921%	24.256364%
MAE	6.618004	6.649293
MSE	82.718285	86.228644
RMSE	9.094959	9.285938
R2	0.933679	0.930865
Explained_variance_score	0.933702	0.931027

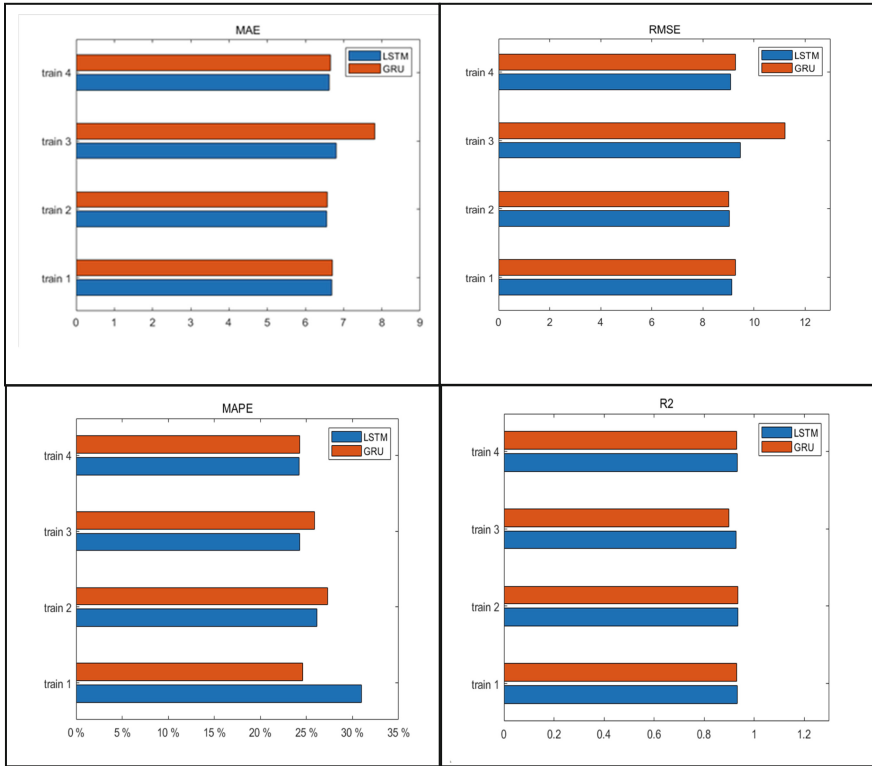


Fig. 6. The results of MAE, RMSE, MPE and R2.

4 Results Analysis

From Fig. 6, it can be seen from the above data that MAE, MSE and RMSE are the descriptions of the model prediction results and the error of the true value. It's known that LSTM and GRU perform poorly when using a three-layer neural network and using the RMSprop optimizer. Not only the error is relatively large, but the fitting effect is not good, and when the adma optimizer is used, as shown in training 4, the performance of the two models is almost the same, and the advantages and disadvantages cannot be distinguished. When using a two-layer neural network, the prediction error is significantly smaller than that of the three-layer neural network, and the fitting effect described by R2 is also much better than that of the three-layer. Therefore, two-layer neural network is taken as the analysis object to compare the two models. The MAPE chart should be focused to analyze, since the value of MAPE is a relative value, which is the average absolute percentage error. It not only take the error between the predicted value and the true value into consideration, but also considers the ratio of error to true value, used to evaluate the same set of data of different models. When using the RMSprop optimizer, the MAPE value of GRU is 5 percentage points smaller, indicating that GRU is more accurate than LSTM in the longer-term prediction is the

result. While using the Adam optimizer, the difference between LSTM and GRU is within one percentage point. The conclusion can be obtained that when predicting traffic flow data, the two-layer neural network performs better, and the performance of the GRU model is slightly better than LSTM.

5 Conclusion

This paper mainly analyzes and studies the LSTM and GRU models to effectively predict short-term traffic flow, and compares their performance. First, the research background of short-term traffic flow and the current research status based on this problem are introduced. Then, a systematic analysis of the problem is carried out. Taking traffic flow as the research object, the characteristics of traffic flow, influencing factors, and forecasting process are analyzed.

After analyzing and processing the data of the Dutch Transport Agency, the corresponding data sets were established. The mean method used to fill in missing data and the data normalized, the processed data is converted into the model's input matrix format for the training. In view of the characteristics such as dependence, timing and nonlinearity of flow data, a short-term traffic flow prediction model based on recurrent neural network LSTM and GRU is adopted, and the advantages and principles are understood and analyzed. Model established, trained and tested, from the result analysis GRU model possessed the better performance. In future research, the accuracy of the final prediction structure needs to be further improved.

References

1. Lv, Y., Duan, Y.: Traffic flow prediction with big data: a deep learning approach. *IEEE Trans. Intell. Transp. Syst.* **16**(2), 865–873 (2014)
2. Ahmed, M.S., Cook, A.R.: Analysis of Freeway Traffic Time Series Data by Using Box-Jenkins Techniques, vol. 722, pp. 1–9. Transportation Research Board, Washington, D.C. (1979)
3. Okutani, I., Stephanedes, Y.J.: Dynamic prediction of traffic volume through Kalman filtering theory. *Transp. Res. Part B* **18**(1), 1–11 (1984)
4. Castro-Neto, M., Jeong, Y.S., Jeong, M.K., et al.: Online-SVR for short-term traffic flow prediction under typical and a typical traffic conditions. *Expert Syst. Appl. Int. J.* **36**(3), 6164–6173 (2009)
5. Jeong, Y.S., Byon, Y.J., Castro-Neto, M.M., et al.: Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE Trans. Intell. Transp. Syst.* **14**(4), 1700–1707 (2013)
6. Smith, B.L., Demetsky, M.J.: Short-term traffic flow prediction: neural network approach. *Transp. Res. Rec.* (1453) (1994)
7. Cho, K.B., van Merriënboer, C., Gulcehre, F., Bougares, H., Schwenk, D., Bahdanau, Y., Bengio.: Learning phrase representations using RNN encoder-decoder for statistical machine translation, arXiv preprint [arXiv:1406.1078](https://arxiv.org/abs/1406.1078) (2014)
8. Kuremoto, T., Kimura, S., Kobayashi, K., et al.: Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neural Comput.* **137**, 47–56 (2014)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997)