Internet of Vehicles: Architectures, Applications and Challenges - Research Article

International Journal of Distributed Sensor Networks

International Journal of Distributed Sensor Networks
2019, Vol. 15(5)

DOI: 10.1177/1550147719847440
journals.sagepub.com/home/dsn
SAGE

Traffic congestion prediction based on GPS trajectory data

Shuming Sun[1,2], Juan Chen[1] and Jian Sun[3]

Abstract

Since speed sensors are not as widely used as GPS devices, the traffic congestion level is predicted based on processed GPS trajectory data in this article. Hidden Markov model is used to match GPS trajectory data to road network and the average speed of road sections can be estimated by adjacent GPS trajectory data. Four deep learning models including convolutional neural network, recurrent neural network, long short-term memory, and gated recurrent unit and three conventional machine learning models including autoregressive integrated moving average model, support vector regression, and ridge regression are used to perform congestion level prediction. According to the experimental results, deep learning models obtain higher accuracy in traffic congestion prediction compared with conventional machine learning models.

Keywords

GPS trajectory data, map matching, convolutional neural network, recurrent neural network, traffic congestion prediction

Date received: 7 September 2018; accepted: 4 April 2019

Handling Editor: Muhammed Ali Imran

Introduction

With the improvement of living standards, the vehicle possession per capita continues to grow. As a result, pressure on urban transportation system is increasing.[1] The transportation system is a complicated system which is composed of pedestrians, cars, and roads. The system is affected by many stable and unstable factors. The stable factors include the number of road lanes, road grades, and urban areas, and the unstable factors include road construction and traffic control. For cities with incomplete urban layouts, it is effective to alleviate traffic congestion by improving road infrastructure. In contrast, it is efficient to solve the problem by predicting traffic congestion precisely and carrying out proper deployment for cities with mature urban layouts.

Quality and quantity of data with traffic information will influence a lot in performing traffic congestion prediction. Traffic information collected by fix detection sensors is accurate. Therefore, most studies on traffic congestion prediction are based on these data. However, the quantity of fixed detection devices is not enough to cover most part of road networks because of their high cost. For those roads without fixed sensors, traffic congestion is hard to predict. Nevertheless, with the roaring growth of vehicles with GPS, it is cost effective to use GPS floating car devices to gather traffic information. GPS trajectory data are more comprehensive, in real time and in a large scale.[2]

[1]SHU-UTS SILC Business School, Shanghai University, Shanghai, China
[2]School of Computer Engineering and Science, Shanghai University, Shanghai, China
[3]Department of Traffic Engineering and Key Laboratory of Road and Traffic Engineering, Ministry of Education, Tongji University, Shanghai, China

Corresponding author:
Juan Chen, SHU-UTS SILC Business School, Shanghai University, Shanghai 201899, China.
Email: chenjuan82@shu.edu.cn

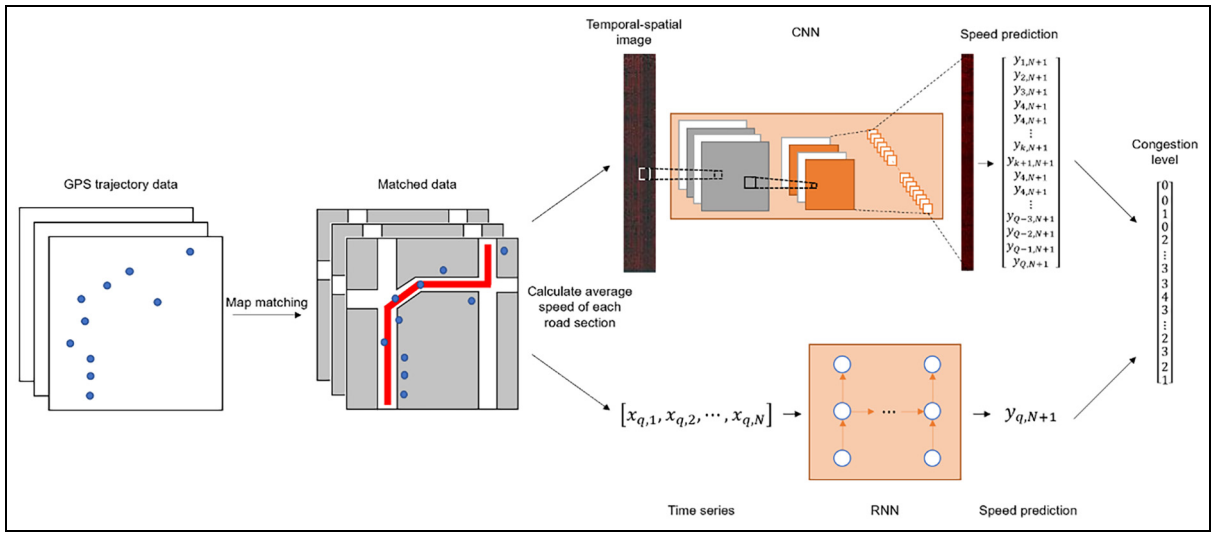

Figure 1. An illustration of the traffic congestion prediction based on GPS trajectory data.

Nonetheless, the data measured by the GPS equipment show deviation. An important step to process raw GPS trajectory data is to match them to the actual road network. Accurate prediction results of traffic congestion level can provide residents with reasonable decision support, save residents' time, and improve road utilization and traffic capacity. Consequently, the efficiency and security of the transportation network will be improved by predicting traffic congestion accurately.

In this article, a procedure shown in Figure 1 is proposed to perform traffic congestion level prediction based on GPS trajectory data. The main ideas of this article are as follows: (1) first, the GPS trajectory data are matched to roads, and experimental road networks are selected according to map matching results; (2) the average driving speed of road networks is calculated by the results of map matching step and then the calculation result is converted to time sequences and temporal–spatial image; (3) the convolutional neural network (CNN) and the recurrent neural network (RNN) series models are used to predict the average driving speed of the road networks; (4) finally, the traffic congestion levels are classified according to the traffic congestion level standard.

The article is organized as follows: section "Introduction" presents the introduction. Section "Literature review" presents a literature review, which summarizes the related map matching technology and traffic flow prediction research. Section "Map matching" discusses hidden Markov model (HMM)-based map matching algorithm and analyzes the matching results. Section "Traffic congestion prediction based on deep neural network" discusses four deep neural network models used in this article, namely, CNN, RNN, long short-term memory (LSTM), and gated recurrent unit (GRU). Section "Analysis of results" analyzes the experimental results. Section "Conclusion" summarizes the research and points out the future research direction.

Literature review

Map matching technology research

The map matching algorithms can be roughly divided into four categories: map matching algorithm based on geometric information, road topology information, probability information, and integrated information.

In terms of geometric information, a map matching model is proposed in Li and Huang[3] to measure the degree of geometric similarity between candidate routes and GPS trajectory. A genetic model is utilized which takes into consideration both path cost and shape similarity between the observed paths and GPS trajectories.[4] The A^* algorithm is used to calculate path cost, and the dynamic time wrapping technique is used to calculate shape similarity. Map matching models based on geometric information usually have a light model structure. Nevertheless, these kinds of models will generate many incorrect results when the roads in the area are dense.

In terms of road topology information, an algorithm based on the topological relationship of road traffic network on the existing map matching model was proposed.[5] The topological relationship of the road network was used in this model to determine the range of candidate road sections and performs map matching based on historical trajectory database. The disadvantage is that only the distance between the GPS points

and the candidate roads is considered, and the driving direction is not taken into account. Topological ordering and a local search tree were utilized in Rahmani and Koutsopoulos[6] to implement online map matching. However, map matching models based on topology information only take road connectivity into consideration. For GPS trajectories with many anomaly points, this kind of model will not perform well.

In terms of probability information, HMM that adopts direction, speed, and α-trimmed mean filters to reduce the scale of candidate sets was proposed by Mohamed et al.[7] Another HMM-based map matching model is proposed to use Viterbi algorithm to generate a partial route with a high confidence.[8] A cellular-based trajectory map matching SnapNet system that showed a difference with the actual road network was introduced by Mohamed et al.[9] Probability-based map matching models have a high accuracy. The models tend to generate complete matching trajectories because they take all the states into account, but they are trained by a large amount of data. When the amount of training data is little, the probability-based model will also mismatch many points.

In terms of integrated information, a method which fuses multiple kinds of traffic information (i.e. speed and direction) and takes the global effect into consideration was utilized by Hu et al.[10] In addition, a historical model was used to estimate the historical traffic speed and current surrounding traffic speed based on temporal and spatial analyses. The influential factor based on spatial distribution of the middle location between two continuous GPS coordinates on low-frequency GPS trajectories was considered by the spatial and temporal conditional random field (ST-CRF) map matching model.[11] Similar to most existing studies, both the consistency of driving direction and the temporal–spatial accessibility between two consecutive GPS coordinates are considered in the ST-CRF model. The model is basically an HMM map matching model based on Viterbi algorithms.[12] Map matching models considering integrated information are complex and time-consuming, so they are not online models.

In this article, an HMM-based map matching algorithm is adopted to match the GPS trajectory points to road. Considering the fact that the HMM-based map matching model is a global algorithm and its accuracy is higher than that of other local algorithms, HMM is used to implement map matching based on GPS trajectory data.

Traffic flow prediction research

The main idea of this article is to use the deep neural network to predict the average speed of selected road networks and then classify the traffic congestion level according to the traffic congestion level standard. Therefore, the traffic flow forecast will be reviewed below.

In the field of traffic flow prediction, research methods can be basically divided into two categories: parametric and nonparametric approaches.

Parametric approaches are based on empirical data and theoretical assumption to decide the model parameters, such as Kalman filter models and autoregressive integrated moving average (ARIMA)-based models, which are classical time series models.[13] A model with one parameter was proposed to predict traffic condition efficiently.[14] The autoregressive model was combined with other prediction methods to optimize prediction performance. The autoregressive model proposed in Davoodi et al.[15] is used for traffic flow prediction.

The advantage of parametric methods includes (1) low model complexity and (2) ease of implementation. However, the traffic network is complex and shows nonlinear and stochastic change, and the parametric approaches cannot handle the uncertainty well. The prediction performance of the parametric methods will be influenced by many external factors in the transportation system. Otherwise, parametric models are usually based on prior knowledge in specific field in practice.

For nonparametric approaches, large-scale training data are necessary to build the model structure. The common nonparametric models include support vector regression (SVR), *k*-nearest neighbor (KNN), artificial neural network (ANN), and the Bayesian network model.[16] A model taking singular point probabilities into account was proposed by Liu et al.[17] A prediction model which combined ANN with root mean square error (RMSE) was utilized by Qian et al.[18] The radial basis function (RBF) model taking the analysis of chaotic characteristics into account was implemented to predict traffic flow by Chen.[19] Compared with conventional nonparametric approaches, deep learning based parametric models can build higher dimensional space to match the real solution space.

RNNs and their variants can extract some sequence features in the input data. While the driving speed in road networks has temporal characteristics, RNN models such as RNN,[20] LSTM[21] and GRU (Cho et al., 2014)[22] can be considered to predict traffic flow. A deep bidirectional long short-term memory (DBL) model proposed in Wang et al.[23] is utilized to extract feature. A LSTM-based model which utilized attention mechanisms to perceive long-term intensive traffic flow and improves the long-term memory of LSTM was proposed by Yang et al.[24] Nonetheless, transportation system has both temporal and spatial characteristics, but the RNN series models focus only on the temporal features.

Table 1. Taxi GPS trajectory data field.

Field name	Data type	Description
VehicleID	Int32	Taxi number
Latitude	Double	Coordinate of taxi when sampling
Longitude	Double	
Att	Bool	Whether passenger is carried
Time	Date	Time stamp when sampling

Traffic flow data can be represented to a space-time matrix and can be converted to image, so the CNN model can be used to extract features from the image and learn temporal-spatial characteristics from traffic flow. The CNN model was used to predict traffic flow. The prediction results were converted to corresponding congestion level. Then the accuracy is calculated based on the congestion prediction.[25] The experimental results show that the CNN model has the highest accuracy when the mean square error (MSE) value is the lowest.

In the following, the taxi GPS data will be preprocessed by the HMM-based map matching algorithm. According to the map matching result, the experimental roads are selected and the average speed of the experimental road sections is calculated.

Map matching

Taxi GPS trajectory data preprocessing

The raw GPS trajectory data are the GPS trajectory data of taxis from the time period of 3–30 August 2014 in Chengdu, China. As shown in Table 1, the data include such fields as taxi number, time stamp, and the longitude and latitude of the vehicle at that moment. In addition, the original data also include information whether or not passenger is carried. The GPS trajectory data are disordered in the time dimension and some data are missing. In addition, for the convenience of experiment, only a part area of Chengdu and about 2000 taxis are selected to conduct the experiments, so raw data need to be preprocessed.

Data supplementation and sorting. The raw data are unordered on the timeline, so the GPS trajectory data for each vehicle per day need to be sorted by time stamp. In addition, the original data include the taxi GPS trajectory data of Chengdu from the time period of 3–30 August 2014. However, the data on the dates 7, 13, and 17 of August are missing. According to the analysis of traffic flow characteristics,[24] the missing 3-day data of 2000 taxis can be replaced by another 2000 taxis' GPS trajectory data on 14th, 20th, and 24th of August.

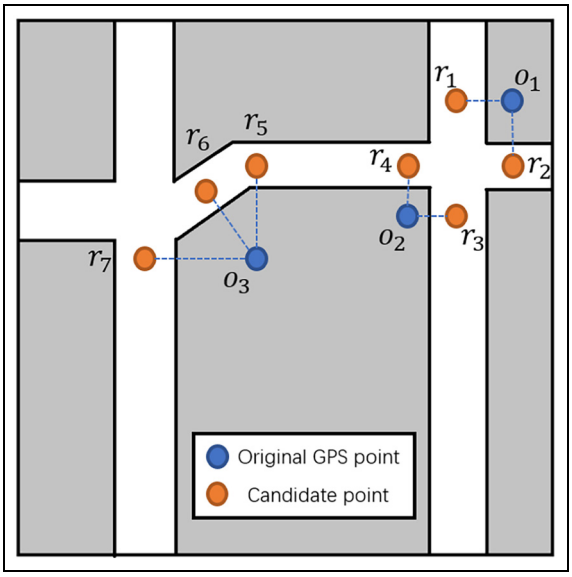

Figure 2. HMM-based map matching model.

Data filtering. The selected area is part of the urban area of Chengdu. The longitude ranges from 103.94 and 104.2, and the latitude ranges from 30.57 to 30.75. Therefore, some taxi GPS track points may be outside the map. In this article, the GPS track points outside the selected area are excluded, and the sequence with such kind of points is divided into many parts.

HMM-based map matching algorithm

The HMM is a probability model that can be used to process time series. The HMM describes a random sequence of randomly generated unobservable states from a hidden Markov chain. Then a corresponding observation based on each state is generated to form a process of observing random sequences. A sequence of random states generated by hidden Markov chains is called a sequence of states; each state corresponds to an observation and a sequence of random observations based on this is called an observation sequence.

As shown in Figure 2, the sequence (o_1, o_2, o_3) is the original taxi GPS trajectory data with deviation. Candidate points are selected in nearby roads by projecting the GPS points to roads. In the HMM-based map matching model, the confidence of each candidate is determined by a specific probability. As the number of GPS points increases, the trajectory will be more and more complete and the probability of matching candidates to correct road will increase. The sequence of track points with the highest confidence will be output as a matching track.

Figure 2 presents the illustration of the HMM-based map matching model in this article. Each GPS point (e.g. o_1, o_2, o_3 in Figure 2) is defined as an observation

Table 2. Map matching algorithm based on HMM.

HMM-based map matching
Input: taxi trajectory point sequence, $O = (o_1, o_2, \ldots, o_N)$
Output: optimal matching point sequence, *optimalpath* $= (r_1, r_2, \ldots, r_N)$
1. Use a list V to store information for each state, including joint probability, parent node number, and node coordinates corresponding to the parent node number
2. Use *optimalpath* to store the set of optimal matching points after backtracking
3. Use *candi* set to store the observation probability and the corresponding candidate matching point coordinates and obtain and initialize the *candi* set when i equals 0
4. For $i = 1$ to $N - 1$
5. Obtain a set of first candidate *lastcandi* matching points corresponding to o_{i-1} and calculate an observation probability
6. Obtain a set of second candidate *candi* matching points corresponding to o_i and calculate an observation probability
7. For each matching point r_j in *candi*
8. For each matching point r_k in *lastcandi*
9. Calculate the transition probability $P_t(r_k, r_j)$
10. Calculate $V[i][j][prob] = P_j(o_i/r_j) \max_{r_j \in lastcandi} \{P_t(r_k, r_j) \cdot V[i-1][k]["prob"]\}$
11. $r = \arg\max_{r_j \in candi} \{P_t(r_k, r_j) \cdot V[i-1][k]["prob"]\}$
12. $V[i][j]["prev"] = r$ // Save the point with the highest joint probability to list V
13. $optimalpath[N-1] = \arg\max_{1 \le l \le len(V[N-1])} \{V[N-1][l]["prob"]\}$
// Find the matching point with the highest joint probability and trace back from that point
14. For $i = N - 2$ to 1
15. $optimalpath[i] = V[i+1][optimalpath[i+1]]["prev"]$

HMM: hidden Markov model.

state in the HMM model. The corresponding candidate points are defined as hidden states. In Figure 2, the hidden states of o_1 include r_1 and r_2, the hidden states of o_2 include r_3 and r_4, and the hidden states of o_3 include r_5, r_6, and r_7. Each candidate point has an observation probability. This probability indicates how likely a collected GPS point is matched to the giving candidate point. The candidate points closer to the observed GPS point have larger observation probabilities.

However, it will cause a large deviation in practical applications by taking the observation probability into consideration solely. Such problems as winding, turning around, and unreachability during the time interval arise. Therefore, the reachability of adjacent candidate points should be considered. In the HMM, this can be represented by the transition probability of two adjacent candidate points. Then the hidden Markov chain is built.

In this article, $P_e(o_n / r_i)$ denotes the observation probability, where o represents an observed GPS point at time n, which is the observation state, and r_i represents a candidate point, which is the hidden state. $P_t(r', r_i)$ denotes the transition probability of the taxi traveling from the candidate point r'_j to r_i. Therefore, for a given taxi GPS trajectory sequence $S = (o_n | n = 1, \ldots, N)$, the optimal trajectory in the hidden Markov chain should have the biggest joint probability, which can be described as:

$$P_{n,r_i} = P_e(o_n/r_i) \max_{r'_j \epsilon K_{n-1}} \{P_t(r'_j, r_i) P_{n-1,r'_j}\} \quad (1)$$

For $n = 1$, there is $P_{1,r_i} = P_e(o_1/r_i)$, where K_{n-1} denotes all hidden states of the previous GPS points. Therefore, the whole optimal trajectory can be built by backtracking from the last optimal candidate. Then the entire optimal trajectory can be restored. The candidate in optimal trajectory can be represented as $r = \arg\max_{r' \epsilon K_n} \{P_{n,r_i}\}$.

Given an observed GPS trajectory point, the projection of this point in all road segments in the road network can be regarded as its candidate points. The GPS positioning error can be approximated as a Gaussian distribution with a mean of 0,[26] and the observation probability of r_i can be described as

$$P_e\left(\frac{o_n}{r_i}\right) = \frac{1}{\sqrt{2\pi}\sigma} e^{-0.5\left(\frac{\|o_n - m_{n,i}\|}{\sigma}\right)^2} \quad (2)$$

where $m_{n,i}$ denotes the matching point of the state o_n on the road segment r_i; $\|o_n - m_{n,i}\|$ denotes the large arc distance of o_n to $m_{n,i}$; and σ is the standard deviation of GPS measurement error. It is shown in equation (2) that the observation probability is only related to the current GPS point o_n. Meanwhile, the greater the distance between the GPS point and the candidate point is, the smaller the observation probability is.

Trajectory built by two adjacent candidate points may not be a normal driving trajectory. Taking the condition into consideration, the probability of driving from a candidate point to another candidate point is regarded as the transition probability. It is shown in

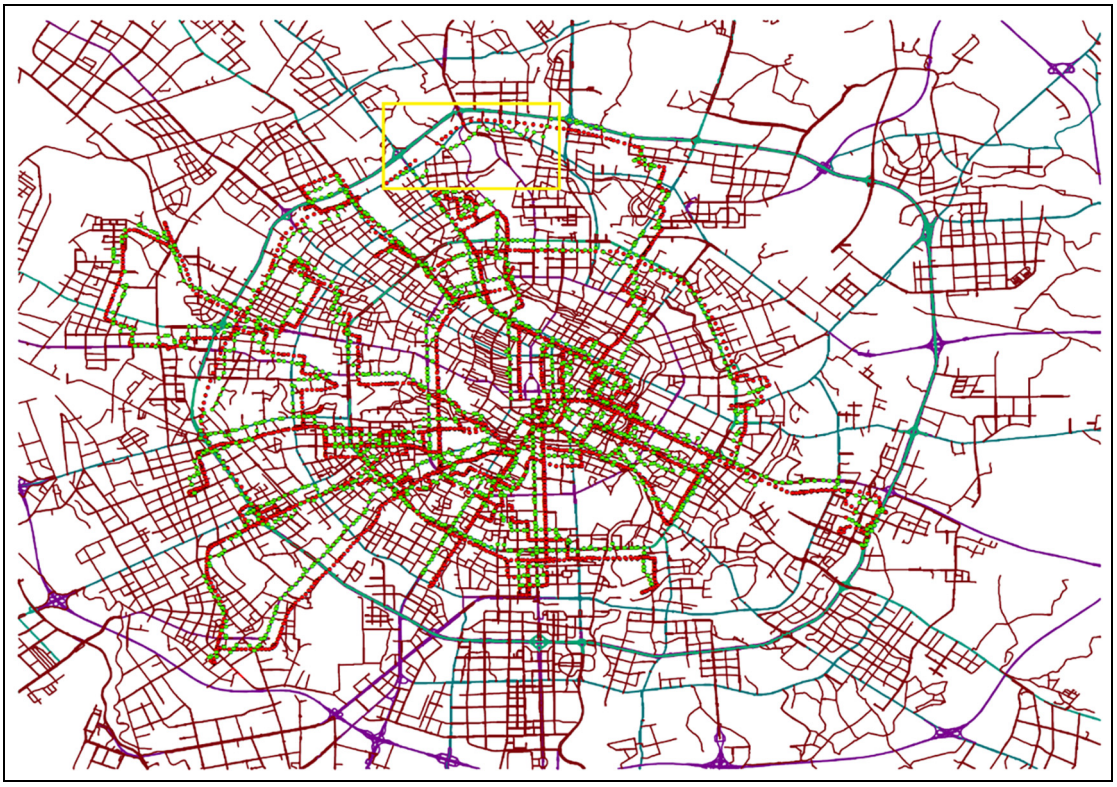

Figure 3. Map matching result with large sampling time interval points.

Newson and Krumm[27] that the correct travel path distance is often similar to the large arc distance of adjacent GPS points. Therefore, the transition probability can be described as

$$P_t(r, r') = \frac{1}{\beta} e^{-\frac{d_t}{\beta}} \tag{3}$$

where $d_t = |\|o_{n+1} - o_n\| - \|m_{n+1,r} - m_{n,r'}\||$ represents the large arc distance of adjacent GPS points and $\|m_{n+1,r} - m_{n,r'}\|$ denotes the trajectory distance of neighboring candidate points. The larger the β is, the smaller the similarity between the large arc distance of the GPS track point and the actual track distance is. β is an adjustable parameter.

The process of the hidden Markov map matching algorithm is shown in Table 2.

Map matching result analysis

The performance of the HMM-based map matching model is mainly influenced by two factors.

First, if the time interval in a GPS trajectory is long, GPS points are likely to be mismatched to other nearby roads. Figure 3 shows the map matching result. The time intervals in the trajectory in a yellow box in Figure 3 range from 10 to 319 s. Those points with a short sampling interval tend to match well, while some points with a long sampling interval tend to be mismatched to a wrong road. The green matched points in a yellow box in Figure 3 change from a main road to another road when the red true GPS points maintain its tendency.

Second, if there are many roads with high similarity to GPS trajectory, GPS points also tend to be wrongly matched to other routes. Figure 4 shows the map matching result of a trajectory and Figure 5 shows an enlarged view of the yellow box in Figure 4. In Figure 5, it can be estimated that the blue road should be the proper matching road, but red GPS points are mismatched to the roads which green points are in. One possible reason is that the wrongly matched roads' shapes are similar to the proper road and they are closer to red GPS points. This phenomenon happens frequently in wrongly matching results.

Road network selection

The complete road network in Chengdu is complicated. There are many roads with high similarity in some areas, which has a great influence on the performance of map matching. If GPS points are mismatched to some roads, the average speed calculated by map

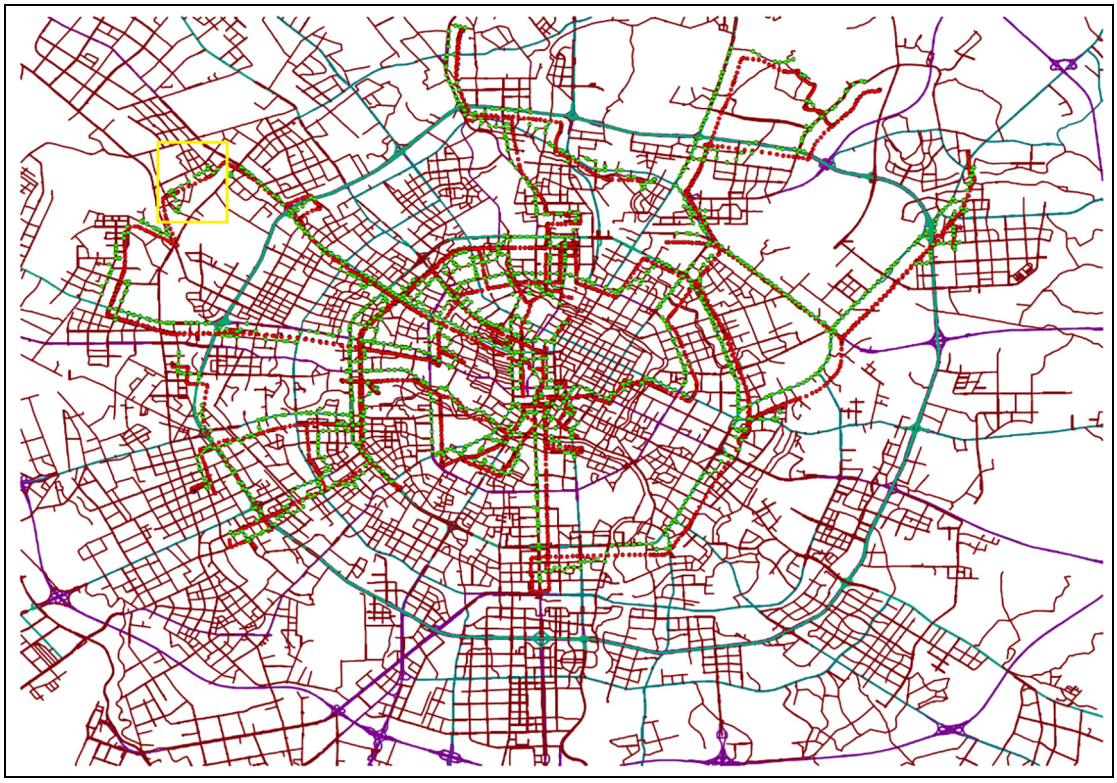

Figure 4. Map matching result with similar road in the road network.

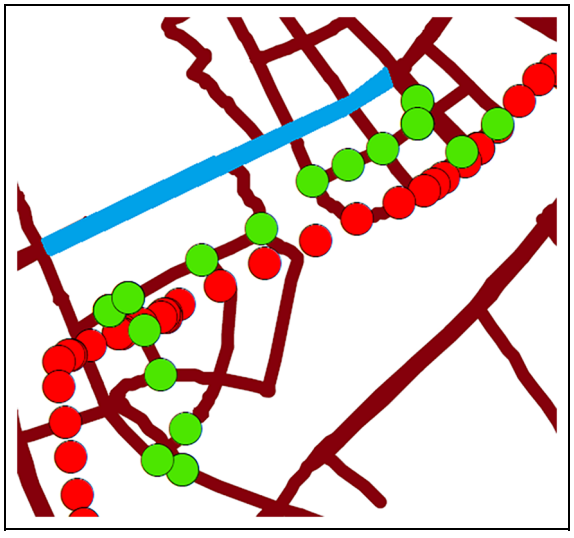

Figure 5. Enlarged view in map matching result with similar road in the road network.

matching results will be unreliable. Therefore, it is necessary to select road networks to avoid the problem mentioned above. The selected road networks should have fewer roads similar to nearby roads.

The road networks selected in this article are shown in Figure 6. The number of roads in Network 1 is 385 and that in Network 2 is 588. As shown in Figure 6, the red thick roads represent Network 1 and the green thick roads represent Network 2. There are fewer roads with similar degrees around the selected network and these two road networks are moderately dense.

Calculating the average speed of road section

Since it is necessary to predict the congestion level of selected road networks at a certain moment in the future, the average speed of selected road networks should be predicted first. Then, according to the prediction results, the congestion grade of selected road networks can be classified by the traffic congestion level classification standard of Chengdu. Therefore, the congestion of selected road networks at certain moment in the future can be predicted. This is actually a time series prediction problem. The input of the prediction model is the average driving speeds in the past. The output of the prediction model is the predicted traffic congestion levels in the future. For the purposes of predicting traffic congestion of selected road networks, the average driving speed of selected road networks is calculated by the following steps in this article. First, map matching is performed on a certain number of taxis' GPS trajectory data. The average driving speed of a road in selected road networks can be calculated using formula (4)

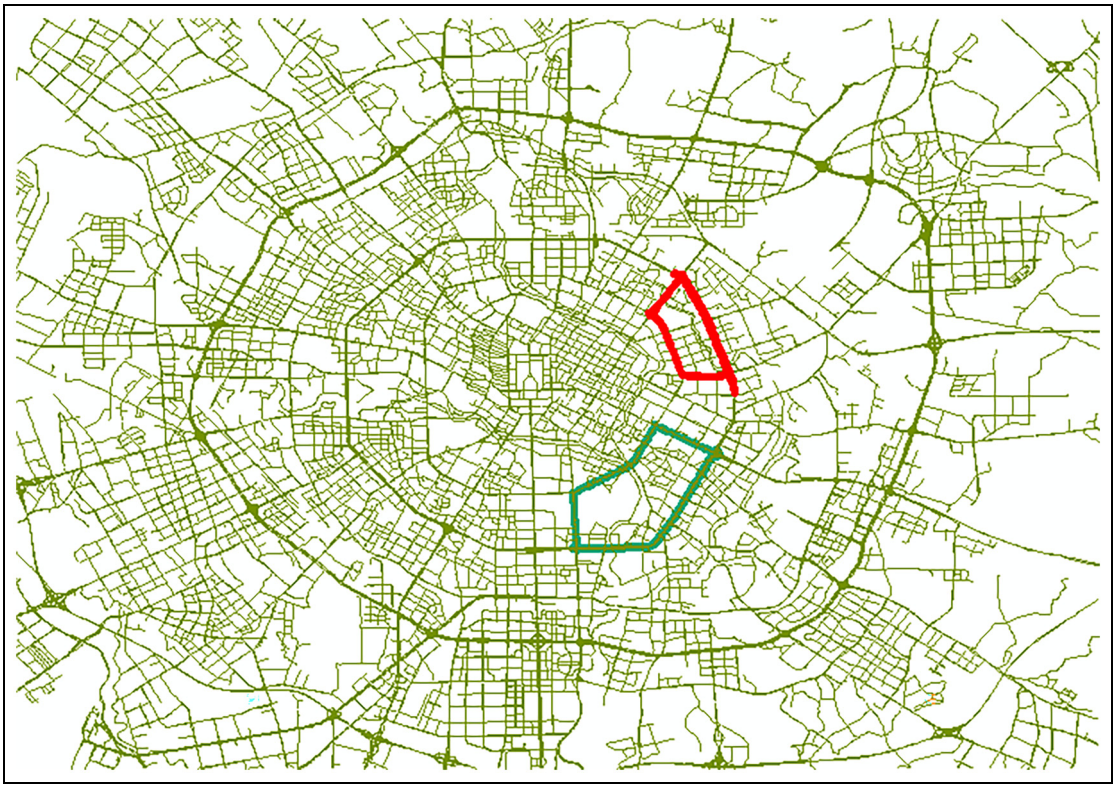

Figure 6. Road network selection.

$$x_{ab} = \frac{1}{M_r}\sum_{i=1}^{M_r}\left(\frac{1}{Tra_i - 1}\sum_{j=1}^{Tra_i-1}\frac{distance(o_{i,j}, o_{i,j+1})}{interval(o_{i,j}, o_{i,j+1})}\right) \quad (4)$$

where x_{ab} denotes the average driving speed in road a in time b, M_r denotes the number of trajectories passing road a, Tra_i denotes the number of GPS points in the part of the trajectory i passing road a, $o_{i,j}$ denotes the jth original GPS point in trajectory i passing road a, $distance(o_{i,j}, o_{i,j+1})$ represents the large arc distance of the matching points of $o_{i,j}$ and $o_{i,j+1}$, and $interval(o_{i,j}, o_{i,j+1})$ represents the sampling time interval between $o_{i,j}$ and $o_{i,j+1}$ The calculation results are used as the input to the prediction models.

In the next section, the CNN, RNN, LSTM, and GRU models will be introduced to perform the prediction of the average driving speed in selected road networks. Then the traffic congestion level classification will be carried out according to the traffic operation level classification standard.[28] Finally, the experimental results will be analyzed.

Traffic congestion prediction based on deep neural network

CNN model building

The deep learning model has been widely used in computer vision and image recognition, and has achieved remarkable results.[29] The traffic flow of the road networks can be transformed into a space–time image. In this image, temporal dimension features are related to a road's traffic information with the change of timeline. Spatial dimension features are related to the traffic flow information in a time stamp among all roads. The CNN can learn the temporal and spatial characteristics of the selected road network well. Therefore, the traffic congestion level of selected road networks at a certain moment in the future can be predicted by the CNN model.

The procedure to predict average driving speed in road networks is shown in Figure 7. The input of the CNN model is a space–time image. After extracting the temporal–spatial features of the input, prediction is made by a fully connected layer.

Figure 8 presents the form of the input image. In Figure 8, N is the length of input time window units used to perform prediction. Q is the number of roads in the selected road network. The time window unit is defined as the average driving speed in selected road networks within 5 min in this article. The input of the model can be described using formula (5)

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{Q1} & x_{Q2} & \cdots & x_{QN} \end{bmatrix} \quad (5)$$

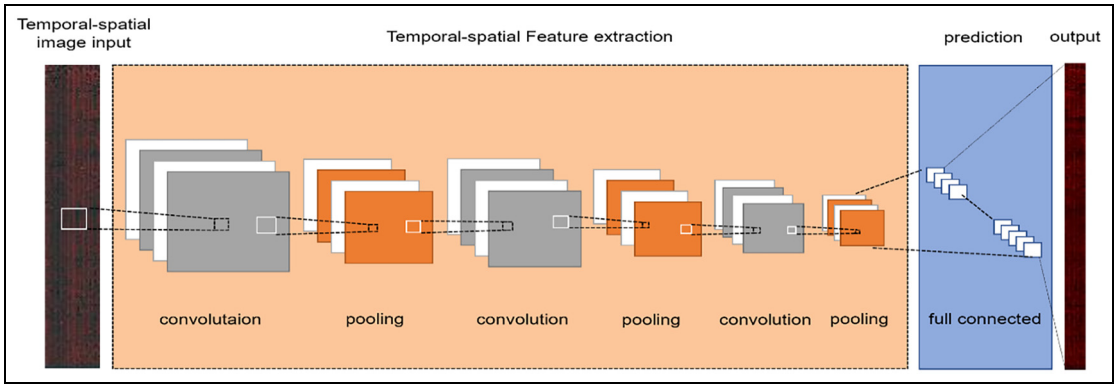

Figure 7. Illustration of the CNN structure in this article.

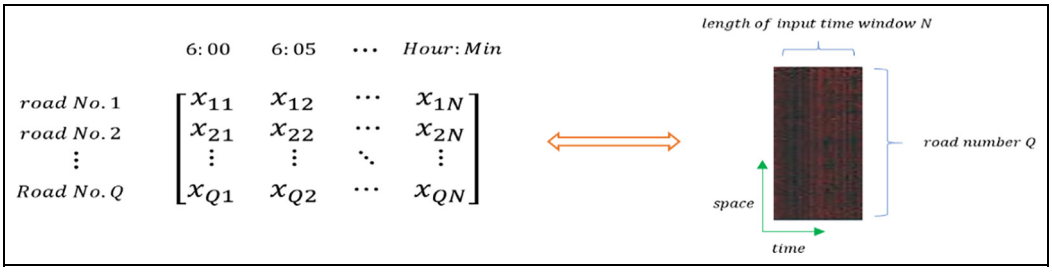

Figure 8. The input of the CNN model.

In this article, we select two road networks. For Network 1, the resolution of the input image is $385 \times N$. The resolution of the output image is $385 \times W$, where W denotes the output length of the time window unit. For Network 2, the resolution of the input image is $588 \times N$. The resolution of the output image is $588 \times W$.

The output of the convolution and pooling layer can be described as

$$o_d^j = pool\left(\sigma\left(W_d^j x_d^j + b_d^j\right)\right), \quad j \in [1, c_l] \tag{6}$$

where *pool* is a pooling function, $\sigma(x)$ is the activation function, and D is the depth of the CNN model. The input, output, and parameters of the dth layer are denoted as x_d^j, o_d^j and (W_d^j, b_d^j), respectively, where j represents the index of the convolution filter. c_d denotes the number of convolution filters in the dth layer.

The input of convolution and pooling layer of layer $d(d \neq 1, d \neq D)$ can be described as

$$o_d^j = pool\left(\sigma\left(\sum_{k=1}^{c_{d-1}} W_d^j x_d^j + b_d^j\right)\right), \quad j \in [1, c_d] \tag{7}$$

After extracting the features, the $flatten(x)$ function is used to connect the output traffic features to a dense vector. The dense vector contains deep features of the input. The dense vector can be described as

$$o_D^{flatten} = flatten\left(\left[o_D^1, o_D^2, \cdots, o_D^j\right]\right), \quad j = c_D \tag{8}$$

Finally, the vector is converted to the output of the model through a fully connected layer. Therefore, the output of the model can be described as

$$\begin{aligned} \hat{y} &= W_f o_D^{flatten} + b_f \\ &= W_f\left(flatten\left(pool\left(\sigma\left(\sum_{k=1}^{c_{d-1}} W_d^j x_d^j + b_d^j\right)\right)\right)\right) + b_f \end{aligned} \tag{9}$$

where W_f and b_f denote the parameters of the fully connected layer and $\hat{y}$ is the predicted value.

RNN model building

RNN model. RNNs can learn the sequence information and features in the data.[20] The traffic flow changes with time, so it has temporal characteristics. Therefore, it is reasonable to use a time series prediction model to predict traffic flow. The RNN and its two variants LSTM and GRU models are used to predict the average driving speed in selected road networks in this article.

The network structure of the RNN model used in this article is shown in Figure 9, where R denotes the hidden layer of the RNN model. In this article, the three-layer RNN model is adopted. In our experiments,

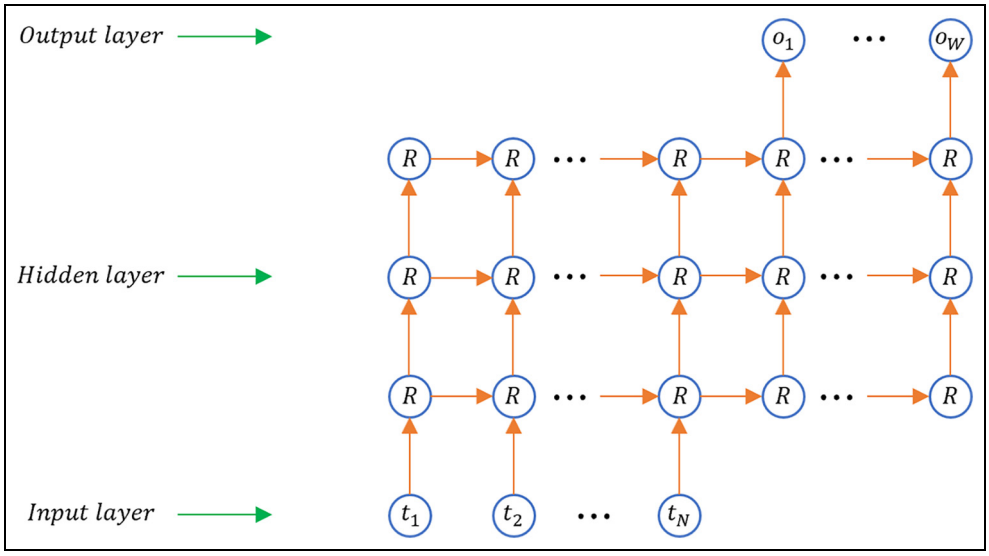

Figure 9. Illustration of the RNN model structure.

the RNN model we adopted includes one input layer, three hidden layers, and one output layer.

Figure 10 shows the time series input of the RNN model and its variants. The time series input in Figure 10 can be described as

$$u_i = \left[t_p^k, t_{p+1}^k, \cdots, t_{p+N-1}^k\right],\ p \in [1, M-W-N+1],\ k \in [1, Q] \tag{10}$$

where i is the sample index and x_t^k is the average driving speed of the kth roads at time t. The time window unit in the input is defined as the average driving speed in selected road networks within 5 min in this article.

The calculation in the hidden layer can be described as

$$s_t = f(Ws_{t-1} + Uu_i) \tag{11}$$

where f is a nonlinear activation function, such as tanh or ReLU, and s_{-1} is set to 0 in order to facilitate the calculation of s_0.

The calculation of the output layer can be described as

$$o_t = softmax(Vs_t) \tag{12}$$

LSTM model. The LSTM model is an improved model based on the RNN model.[21] The main improvement is that gate structure is added in the hidden layer.

Figure 11 presents the inner structure in a hidden layer. A hidden layer contains three gates, an input gate, a forgetting gate, and an output gate. The input of a hidden layer includes the output of the previous hidden layer, the recorded cellular information, and the input at the current time point. The input gate plays a role in selectively recording the input information into the cell state. The function of the forgetting gate is to forget some state information in the cell. The function of the output gate is to selectively output the result of the hidden layer.

The calculation in the hidden layer is described by formulae (13)–(18), where W is the weight matrix and b is the offset.

The calculation of the forgetting gate f_t can be described as

$$f_t = \sigma\left(W_f \cdot [o_{t-1}, x_t] + b_f\right) \tag{13}$$

where σ is a nonlinear activation function, o_{t-1} is the output of the hidden layer at time $t-1$, and x_t is the input at time t.

The input gate i_t can be described as

$$i_t = \sigma(W_i \cdot [o_{t-1}, x_t] + b_i) \tag{14}$$

The candidate cell information $\hat{C}_t$ at time t can be described as

$$\hat{C}_t = \tanh(W_c \cdot [o_{t-1}, x_t] + b_C) \tag{15}$$

where the tanh function is an activation function.

The cell information C_t at time t can be described as

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t \tag{16}$$

The output gate h_t can be described as

$$h_t = \sigma(W_o \cdot [o_{t-1}, x_t] + b_o) \tag{17}$$

The final output o_t can be described as

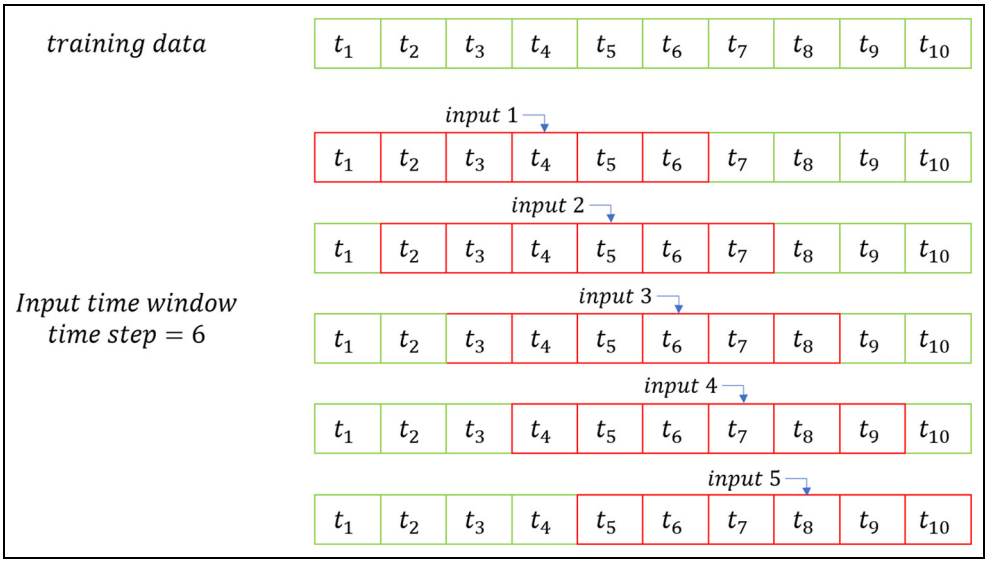

Figure 10. Time series input of the RNN model.

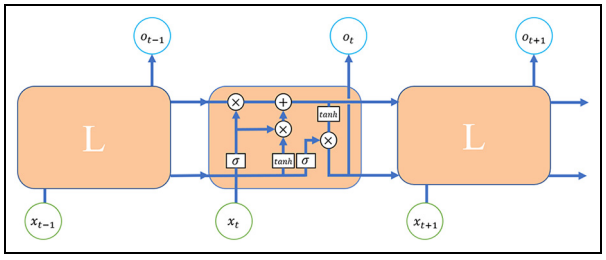

Figure 11. Illustration of the LSTM hidden unit structure.

$$o_t = h_t \cdot \tanh(C_t) \tag{18}$$

GRU model. The GRU model is a variant of the LSTM model (Cho et al., 2014). The GRU model combines the input gate and the forgetting gate into an update gate, which incorporates the cell state and the hidden state.

The inner structure in the GRU hidden unit is shown in Figure 12. A GRU hidden unit contains two gates: reset gate and update gate. The reset gate determines how much current input is combined with previous memory information. The update gate determines how much previous memory information is retained. When the reset gate is set to 1 and the update gate is set to 0, the GRU model is equivalent to the RNN model.

The update gate z_t can be described as

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{19}$$

where h_{t-1} is the hidden layer output at time $t-1$, x_t is the input at time t, and W_z is the weight matrix of the update gate.

The reset gate r_t can be described as

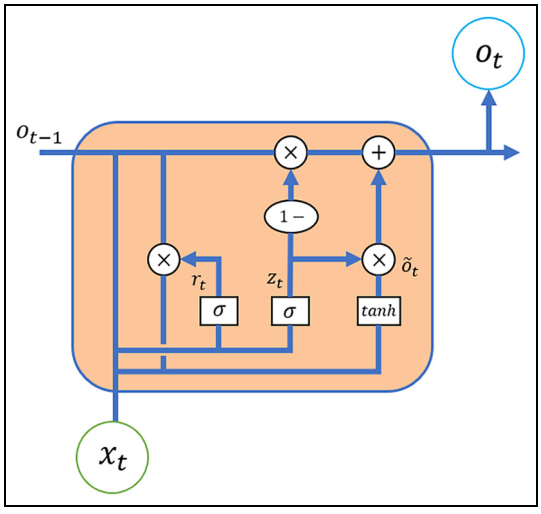

Figure 12. Illustration of the GRU hidden unit structure.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{20}$$

where W_r is the weight matrix of the reset gate.

The hidden layer output h_t can be described as

$$\tilde{h}_t = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t]) \tag{21}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{22}$$

where $\tilde{h}_t$ is the candidate cell information at time t.

Compared with the LSTM model, the GRU model has a simpler internal structure, so its training speed will be faster than that of LSTM when dealing with large training datasets.

The structures of the LSTM and GRU models applied in our article are similar to that of the RNN

Table 3. Experimental tasks.

Task 1: Predicting one unit using the last 6 time window units (30 min)
Task 2: Predicting one unit using the last 8 time window units (40 min)
Task 3: Predicting one unit using the last 12 time window units (60 min)
Task 4: Predicting one unit using the last 18 time window units (90 min)

Table 4. Classification of traffic congestion level of Chengdu in the main road.

Operation grade	Free	Basically free	Mild congestion	Moderate congestion	Heavy congestion
Main road	$\bar{V}>40$	$30<\bar{V}\leqslant 40$	$20<\bar{V}\leqslant 30$	$15<\bar{V}\leqslant 20$	$\bar{V}\leqslant 15$

Table 5. The performance of the ARIMA, SVR, and ridge regression models.

	Input units	6	8	12	18
MSE	ARIMA	190.04	199.38	234.69	240.33
	SVR	151.20	178.42	208.89	197.83
	Ridge	**116.87**	**132.02**	**154.64**	**143.07**
MAE	ARIMA	6.04	6.29	7.22	7.01
	SVR	4.74	5.51	6.48	6.29
	Ridge	**3.99**	**4.51**	**5.32**	**5.30**
RMSE	ARIMA	13.79	14.12	15.32	15.50
	SVR	12.30	13.36	14.45	14.07
	Ridge	**10.81**	**11.49**	**12.44**	**11.96**
Accuracy (%)	ARIMA	79.66	80.45	**80.02**	76.97
	SVR	83.21	**81.04**	77.91	**78.07**
	Ridge	**83.59**	80.69	76.31	75.92

ARIMA: autoregressive integrated moving average; SVR: support vector regression; MSE: mean square error; MAE: mean absolute error; RMSE: root mean square error.
The bold values show the best performance of different indexes in different experiment tasks.

model shown in Figure 9. Both the GRU and LSTM models have three hidden layers.

Time series of average driving speed is regarded as the input of the RNN, LSTM, and GRU models. The temporal characteristics of the input sequence can be learned by the RNN model and its variants. The spatial and temporal characteristics of the input can be learned by the CNN model.

The prediction results of three conventional machine learning models and four deep neural network models mentioned above are analyzed below.

Analysis of results

In this article, the GPS data of nearly 2000 taxis in 28 days are used as the dataset, where about 80% data (22 days) in the dataset are used as the training set and about 20% data (6 days) as the test set.

In terms of the experiments, four different lengths of input time window units are selected to predict the average driving speed in selected road networks. The description of experimental tasks is shown in Table 3. The time window unit in the input is set as the average driving speed in selected road networks within 5 min.

The raw GPS data are collected by taxis in Chengdu. The classification standard shown in Table 4 is made by Chengdu Transportation Department. Therefore, we adopt the standard in Table 4 to classify the road states.[28]

In this article, four deep learning models including the CNN, RNN, LSTM, and GRU models and three conventional machine learning models including ARIMA, SVR, and ridge regression models are used for prediction. Four performance indexes including MSE, mean absolute error (MAE), RMSE, and accuracy are used in this article. MSE, MAE, and RMSE are calculated based on the predicted speed and the calculated speed after map matching. Accuracy is calculated based on the congestion level of the predicted speed and the congestion level of the calculated speed after map matching.

Three conventional machine learning models are used to perform prediction in this article. Table 5 gives the performance of ARIMA, SVR, and ridge regression. As shown in Table 5, with the growth of the length of input time window units, the MSE, MAE, and RMSE of the three models are all increased. When the input length of units is set at 6, the three models

Table 6. Hyperparameters of the CNN, RNN, LSTM, and GRU models.

CNN	RNN	LSTM	GRU
Batch size = 128 Number of filters = 32 Kernel size = (3, 3)	Batch size = 128	Batch size = 128	Batch size = 128

CNN: convolutional neural network; RNN: recurrent neural network; LSTM: long short-term memory; GRU: gated recurrent unit.

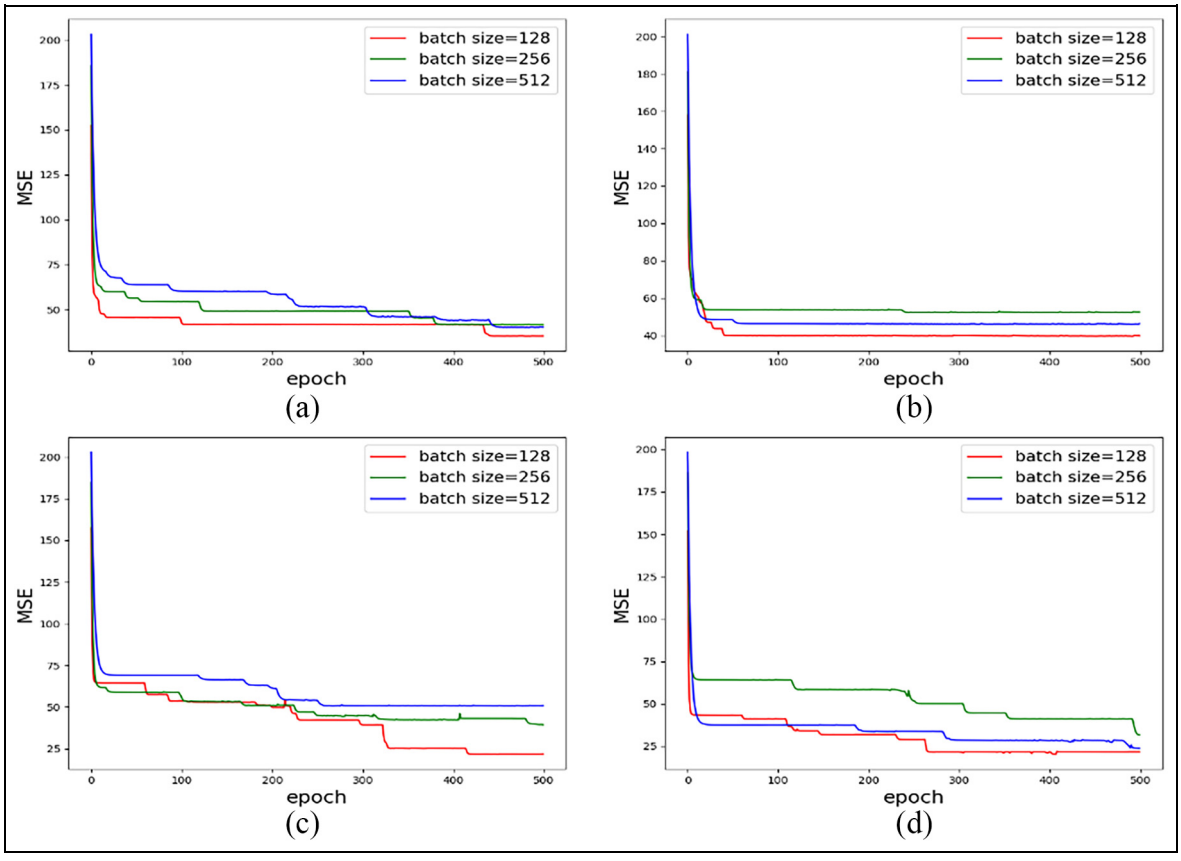

Figure 13. The loss curves of the RNN model: (a) input time window length = 6, (b) input time window length = 8, (c) input time window length = 12, and (d) input time window length = 18.

obtain the minimum MSE, MAE, and RMSE values compared with the input lengths of 8, 12, and 18. The phenomenon shows that conventional machine learning models perform better in short-term prediction than long-term prediction in our experiments. However, the MSE values of conventional models range from 116.87 to 240.33 and the accuracy ranges from 75.92% to 83.59%, which is not as good as the performance of deep learning models. One of the reasons may be that traffic flow contains both linear and nonlinear features, but conventional machine learning models cannot learn nonlinear features well from the input. In the short term, linear features influence the tendency of traffic flow more seriously than nonlinear features. In the long term, nonlinear features play a more important role. Consequently, conventional machine learning models perform not very well in our study.

In addition, four deep learning models are utilized to realize prediction. Table 6 gives the hyperparameters of the CNN, RNN, LSTM, and GRU models.

The hyperparameters shown in Table 6 are decided by the experimental results shown in Figures 13–17. Figure 13 shows the loss curves of the RNN model when the batch size is set at 128, 256, and 512, while the input length is set at 6, 8, 12, and 18, respectively. It can be seen from Figure 13(a) and (c) that the RNN model has the lowest MSE values when the batch size is 128 and it has the highest MSE values when the batch size

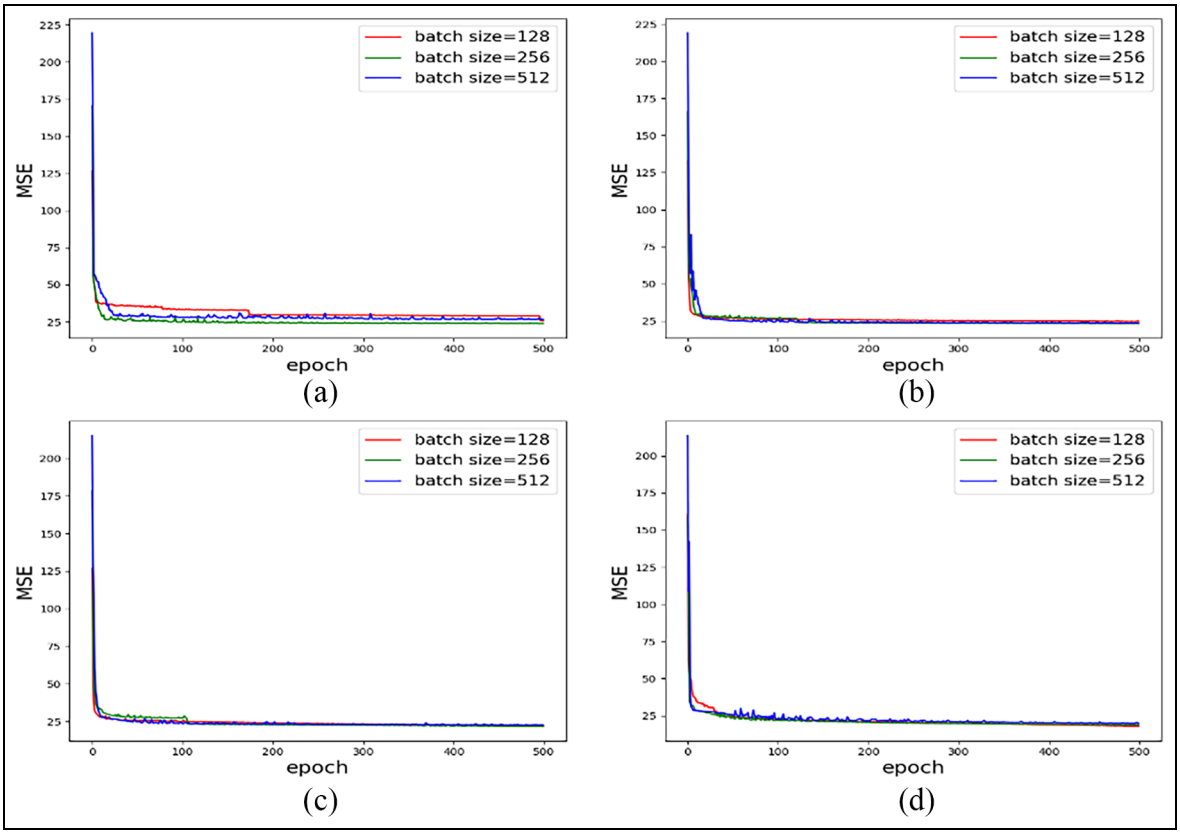

Figure 14. The loss curves of the CNN model (filter number = 32, kernel size = (3, 3)): (a) input time window length = 6, (b) input time window length = 8, (c) input time window length = 12, and (d) input time window length = 18.

is 512 and the training epoch is 500. It can be seen from Figure 13(b) and (d) that the RNN model is trained the best when the batch size is 128 and the RNN model is trained the worst when the batch size is 256. Therefore, the batch size of 128 is used for the RNN model and its variants in the remaining experiments.

Figure 14 presents the loss curves of the CNN model when the batch size is set at 128, 256, and 512, the filter number is 32, and the kernel size is 3×3, while the input length is set at 6, 8, 12, and 18. Figure 15 shows the loss curves of the CNN model when the batch size is set at 128, 256, and 512, the filter number is 32, and the kernel size is 5×5, while the input length is set at 6, 8, 12, and 18, respectively. As shown in Figure 14(a)–(d) and Figure 15(a) and (b), different batch sizes do not influence the tendency of loss curves much obviously. However, it can be found from Figure 15(d) that sometimes models do not converge when the batch size equals 512 and more epochs may be needed to realize the model training when the batch size equals 256. Therefore, the batch size of 128 is optimal for stable CNN training and the batch size of 128 is used in the remaining experiments.

Figure 16 shows the loss curves of the CNN model when the batch size is set at 128, the filter number is 32, 64, and 128, and the kernel size is 3×3, while the input length is set at 6, 8, 12, and 18, respectively. We can find from Figure 16(a)–(c) that the CNN model has the lowest training loss when the number of filters is 32. In Figure 16(d), the CNN model shows the best convergence performance when the number of filters is 128. Consequently, the filter number of 32 is adopted in the remaining experiments for considering stable training.

Figure 17 shows the loss curves of the CNN model when the batch size is 128, the filter number is 32, and the kernel size is set at 3×3, 5×5, and 7×7. It can be observed from Figure 17(c) and (d) that the CNN model has the lowest training loss and converges well when the kernel size is 3×3. In Figure 17(a) and (b), the size of the kernel does not influence the convergence of the CNN model obviously. Therefore, the kernel size of 3×3 is utilized in the remaining experiments.

It is shown that the CNN and RNN series models perform well in our study. The CNN model can learn the temporal–spatial features from the input. The RNN model and its variants can learn temporal features well from the short-term input and the long-term input. Table 7 gives the performance of the four models. It can be seen from Table 7 that the CNN model has the lowest MSE, MAE, and RMSE values than the RNN

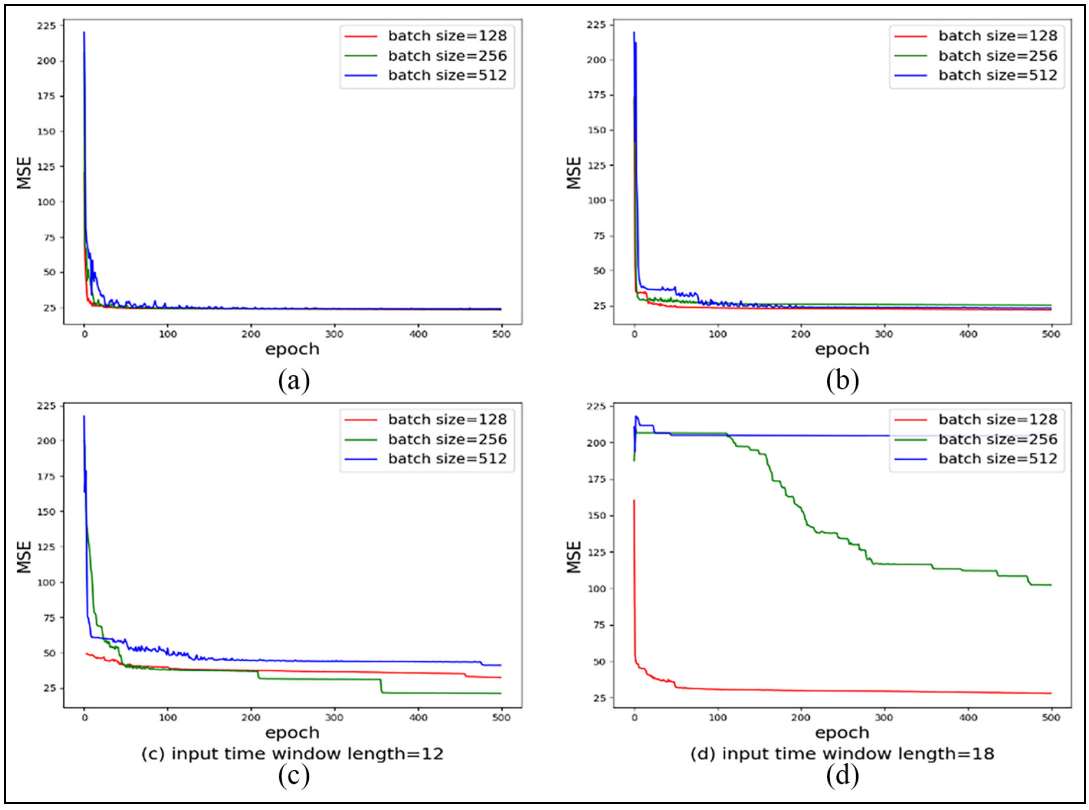

Figure 15. The loss curves of the CNN model (filter number = 32, kernel size = (5, 5)): (a) input time window length = 6, (b) input time window length = 8, (c) input time window length = 12, and (d) input time window length = 18.

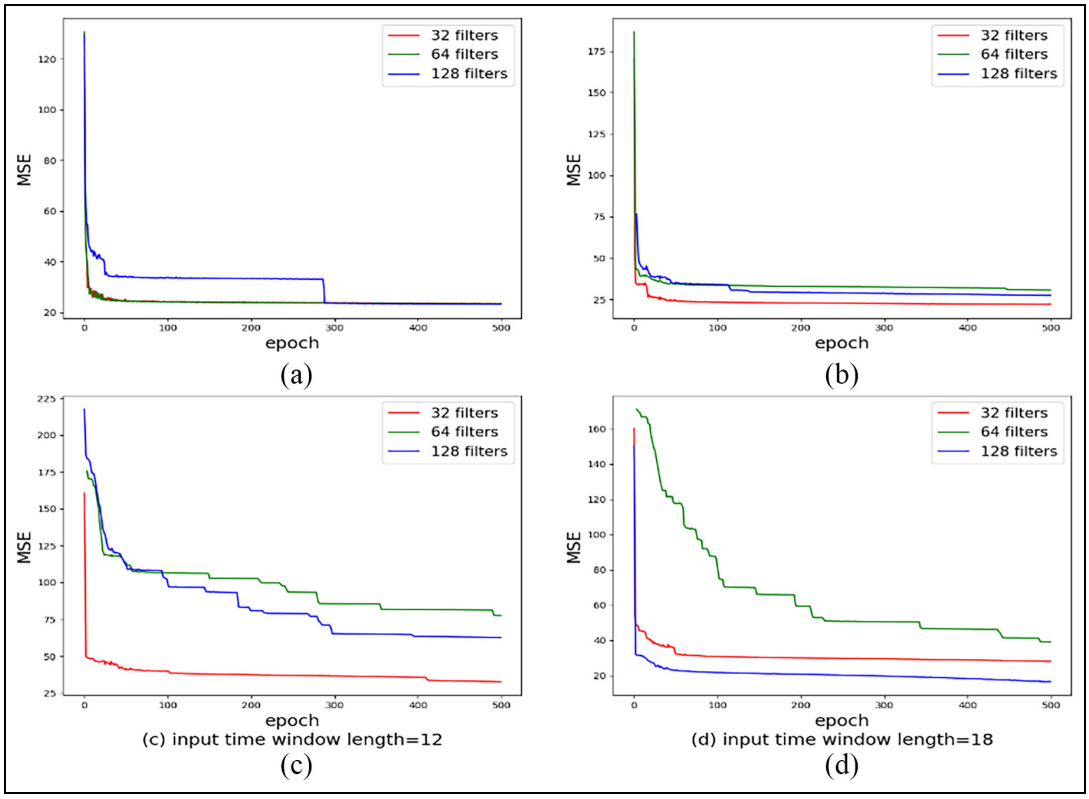

Figure 16. The loss curves of the CNN model (batch size = 128, kernel size = (3, 3)): (a) input time window length = 6, (b) input time window length = 8, (c) input time window length = 12, and (d) input time window length = 18.

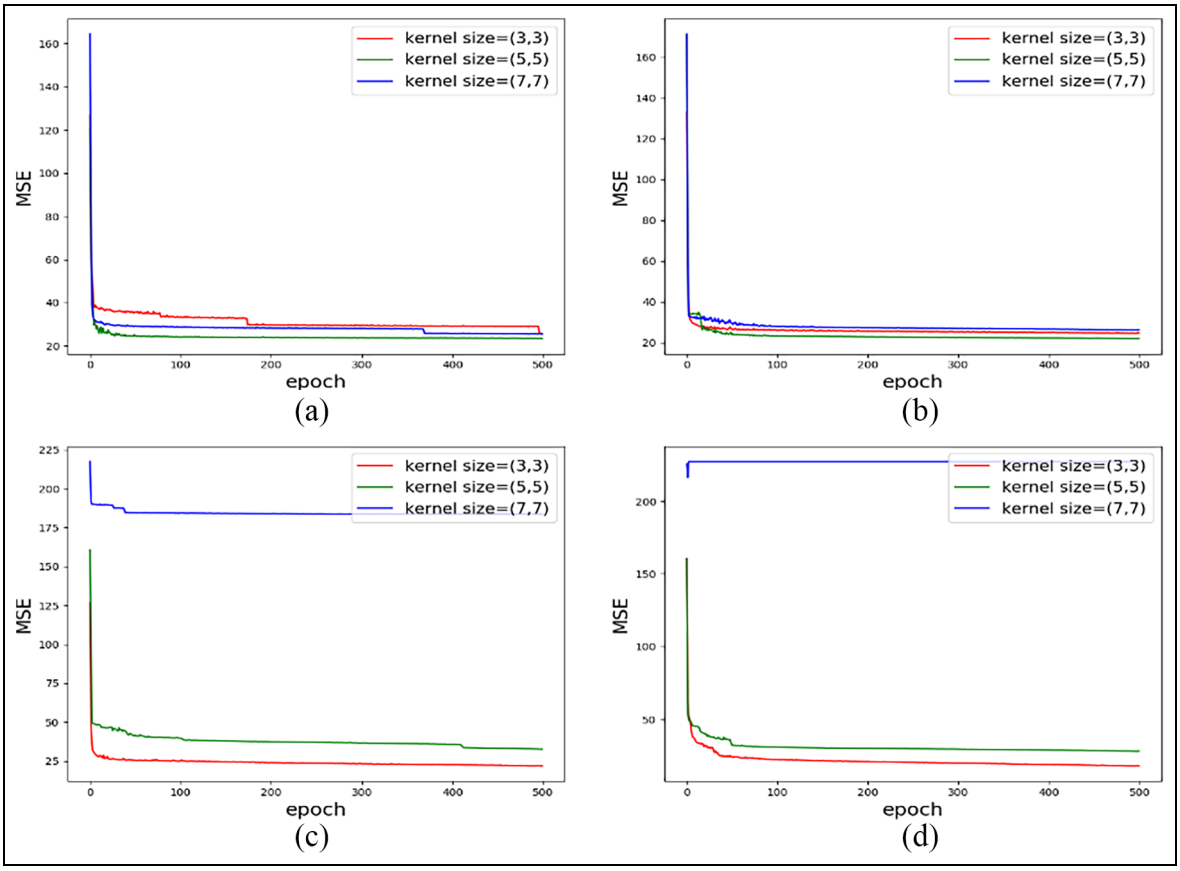

Figure 17. The loss curves of the CNN model (batch size = 128, filter number = 32): (a) input time window length = 6, (b) input time window length = 8, (c) input time window length = 12, and (d) input time window length = 18.

Table 7. The performance of the CNN, RNN, LSTM, and GRU models.

	Road	Road Network 1				Road Network 2			
	Units	6	8	12	18	6	8	12	18
MSE	CNN	**38.50**	**36.97**	**27.11**	**15.69**	**41.81**	**40.51**	30.71	**12.61**
	RNN	44.58	43.73	26.75	27.00	44.72	47.81	**20.39**	28.56
	LSTM	47.88	47.95	35.42	23.81	50.39	52.55	39.89	21.22
	GRU	48.39	44.77	37.21	30.49	45.52	43.78	39.23	35.08
MAE	CNN	**1.59**	**1.45**	**1.04**	**0.71**	1.65	1.60	1.19	**0.67**
	RNN	1.64	1.58	1.05	1.05	**1.60**	1.59	**0.92**	1.07
	LSTM	1.70	1.62	1.19	0.90	1.69	1.66	1.28	0.83
	GRU	1.73	1.60	1.24	1.09	1.61	**1.54**	1.27	1.17
RMSE	CNN	**6.20**	**6.08**	5.21	**3.96**	**6.47**	**6.36**	5.54	**3.55**
	RNN	6.68	6.61	**5.17**	5.20	6.69	6.91	**4.52**	5.34
	LSTM	6.92	6.92	5.95	4.88	7.10	7.25	6.32	4.61
	GRU	6.96	6.69	6.10	5.52	6.75	6.62	6.26	5.92
Accuracy (%)	CNN	90.55	93.20	94.99	**96.32**	90.38	91.66	94.10	96.23
	RNN	92.20	**93.34**	**95.59**	95.60	92.79	93.78	**95.78**	95.88
	LSTM	91.89	93.18	95.37	96.08	92.48	93.57	95.34	**96.65**
	GRU	**92.30**	92.96	95.02	95.40	**92.81**	**93.80**	95.38	95.39

CNN: convolutional neural network; RNN: recurrent neural network; LSTM: long short-term memory; GRU: gated recurrent unit; MSE: mean square error; MAE: mean absolute error; RMSE: root mean square error.
The bold values show the best performance of different indexes in different experiment tasks.

series models in most of the situations. It is proposed that the CNN model is able to learn temporal–spatial features from the space–time image. However, the accuracy of the CNN model is lower than that of the RNN series models in most of the situations. The reason may be that some prediction values (e.g. 19.5 km/h) have small gaps with real values (21 km/h), but they are classified into different congestion levels. In addition, with the increment of the length of the input units, the MSE value of the CNN model drops quickly. This shows that the CNN model tends to perform well in long-term traffic flow prediction. For the RNN model, the MSE, MAE, and RMSE values drop when the length of the input units is less than 12 and rises when the length of the input units is bigger than 12. In contrast, the MSE, MAE, and RMSE values of the LSTM and GRU models drop with the increment of the length of the input time window units. The minimum MSE, MAE, and RMSE values are achieved with an input length of 18. This indicates that the RNN model performs better in short-term prediction than long-term prediction. The GRU and LSTM models are good at predicting long-term time sequences.

To sum up, conventional machine learning models perform well in short-term linear prediction, while deep learning models show their advantages in our study. Moreover, the CNN model is able to learn the temporal–spatial features from traffic flow space–time images and its prediction ability is no less than that of the RNN series models in our study. In addition, the RNN series models can learn temporal features from the time series input and perform well in time sequence prediction. The RNN model predicts well in short-term traffic flow prediction and its performance drops when the length of the time series input becomes long. By comparison, the LSTM and GRU models are able to perform well in predicting long-term series.

Conclusion

Deep neural network is widely used to solve a variety of problems, including traffic flow prediction and traffic congestion prediction. However, the existing research on traffic flow prediction and traffic congestion prediction is mostly based on off-the-shelf traffic flow data gathered by fixed detection sensors. A large number of detection devices are required to collect such kind of data. Therefore, research in this field can only be applied to those roads where detection sensors are installed. As a result, the application range of the research is limited.

The experiments in this article are based on GPS trajectory data which can be easily obtained by floating cars. Our study can expand the application range of traffic flow prediction. The procedure of our methods includes the following steps. First, a map matching algorithm based on the HMM model is used in this article to preprocess taxi GPS trajectory data. After the map matching process, the average driving speed in selected road networks at each moment can be calculated. The calculation results are regarded as the input of the deep neural networks. The prediction models are used to predict the average driving speed of the road networks at a moment in the future. According to the traffic congestion level classification standard, the prediction results are classified into the corresponding congestion grade. In this article, four deep learning models and three conventional machine learning models are used for prediction. However, ARIMA, SVR, and ridge regression do not perform as good as deep learning models. In terms of deep neural networks, the RNN, LSTM, and GRU models show similar performance in these experiments. The GRU and LSTM models perform better in long-term prediction than the RNN model. In addition, the CNN model has the lowest MSE value compared with the RNN series models, which proves that it has the ability to learn the temporal and spatial characteristics in traffic flow space–time images.

There are still some shortcomings in this article and these will be considered as the research directions of our future study. First, the HMM-based map matching model can be replaced by some state-of-the-art map matching algorithms. Second, this article only uses the data from the time period of 3–30 August 2014. The GPS trajectory data of 2000 taxis in 28 days are still insufficient. It can be considered to use more GPS data. Finally, the network structure of the CNN model can be improved to enhance the performance.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The authors would like to thank the National Natural Science Foundation of China (Grant No. 61104166).

ORCID iD

Juan Chen https://orcid.org/0000-0003-0631-8370

References

1. Liu Z, Li J, Wang C, et al. Traffic congestion prediction model based on complex urban road network. *J Univ Electr Sci Tech* 2016; 45(1): 17–25.

2. Zhang YC, Zuo XQ, Zhang LT, et al. Traffic congestion detection based on GPS floating-car data. *Procedia Eng* 2011; 15: 5541–5546.
3. Li Q and Huang L. A map matching algorithm for GPS tracking data. *Acta Geodaet Cartograph Sinica* 2010; 39(2): 207–212.
4. Nikolić M and Jović J. Implementation of generic algorithm in map-matching model. *Exp Syst Appl* 2017; 72: 283–292.
5. Chen B, Wang M, Wang H, et al. Map matching algorithm based on network smashing relationship. *J Survey Map Sci Tech* 2006; 23(5): 331–334.
6. Rahmani M and Koutsopoulos HN. Path inference from sparse floating car data for urban networks. *Transp Res Part C* 2013; 30: 41–54.
7. Mohamed R, Aly H and Youssef M. Accurate and efficient map matching for challenging environments. In: *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*, Dallas, TX, 4–7 November 2014, pp.401–404. New York: ACM.
8. Jagadeesh GR and Srikanthan T. Online map-matching of noisy and sparse location data with hidden Markov and route choice models. *IEEE Trans Intel Transport Syst* 2017; 18(9): 2423–2434.
9. Mohamed R, Aly H and Youssef M. Accurate real-time map matching for challenging environments. *IEEE Trans Intel Transport Syst* 2017; 18(4): 847–857.
10. Hu G, Shao J, Liu F, et al. If-matching: towards accurate map-matching with information fusion. *IEEE Trans Knowledge Data Eng* 2017; 29(1): 114–127.
11. Liu X, Liu K, Li M, et al. A ST-CRF map-matching method for low-frequency floating car data. *IEEE Trans Intel Transport Syst* 2017; 18(5): 1241–1254.
12. Hsueh YL and Chen HC. Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions. *Informat Sci* 2017; 433–434: 55–69.
13. Jin S, Wang D, Xu C, et al. Short-term traffic safety forecasting using Gaussian mixture model and Kalman filter. *J Zhejiang Univ Sci A* 2013; 14(4): 231–243.
14. Kong QJ, Zhao Q, Wei C, et al. Efficient traffic state estimation for large-scale urban road networks. *IEEE Trans Intel Transport Syst* 2013; 14(1): 398–407.
15. Davoodi N, Soheili A, Hashemi SM, et al. A macro-model for traffic flow with consideration of driver's reaction time and distance. *Nonlinear Dyn* 2016; 83(3): 1621–1628.
16. Sun S, Zhang C and Yu G. A Bayesian network approach to traffic flow forecasting. *IEEE Trans Intel Transport Syst* 2006; 7(1): 124–132.
17. Liu B, Cheng J, Cai K, et al. Singular point probability improve LSTM network performance for long-term traffic flow prediction. In: Du D, Li L, Zhu E, et al. (eds) *National conference of theoretical computer science*. Singapore: Springer, 2017, pp.328–340.
18. Qian ZS, Li J, Li X, et al. Modeling heterogeneous traffic flow: a pragmatic approach. *Transport Res Part B* 2017; 99: 183–204.
19. Chen D. Research on traffic flow prediction in the big data environment based on the improved RBF neural network. *IEEE Trans Ind Informat* 2017; 13(4): 2000–2008.
20. Siegelmann HT and Sontag ED. Turing computability with neural nets. *Appl Mathemat Lett* 1991; 4(6): 77–80.
21. Hochreiter S and Schmidhuber J. Long short-term memory. *Neural Comput* 1997; 9(8): 1735–1780.
22. Cho K, Van Merriënboer B, Gulcehre C, et al. *Learning phrase representations using RNN encoder-decoder for statistical machine translation* (arXiv preprint arXiv:1406.1078), 2014.
23. Wang J, Hu F and Li L. Deep bi-directional long short-term memory model for short-term traffic flow prediction. In: Diu L, Xie S, Li Y, et al. (eds) *International conference on neural information processing*. Cham: Springer, 2017, pp.306–316.
24. Yang B, Sun S, Li J, et al. Traffic flow prediction using LSTM with feature enhancement. *Neurocomputing* 2018; 332: 320–327.
25. Ma X, Dai Z, He Z, et al. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* 2017; 17(4): 818.
26. Van Diggelen F. GNSSS accuracy: lies, damn lies, and statistics, 2007, https://www.gpsworld.com/gpsgnss-accuracy-lies-damn-lies-and-statistics-1134/
27. Newson P and Krumm J. Hidden Markov map matching through noise and sparseness. In: *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, Seattle, WA, 4–6 November 2009, pp.336–343. New York: ACM.
28. Zhang R. *The research for solving the traffic congestion of Chengdu City*. Chengdu, China: Southwest Jiaotong University, 2013.
29. Oquab M, Bottou L, Laptev I, et al. Learning and transferring mid-level image representations using convolutional neural networks. In: *Proceedings of the 2014 IEEE conference on computer vision and pattern recognition*, Columbus, OH, 23–28 June 2014, pp.1717–1724. New York: IEEE.