SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2020, Vol. 63 212207:1–212207:21
https://doi.org/10.1007/s11432-019-2761-y

Motion trajectory prediction based on a CNN-LSTM sequential model

Guo XIE*, Anqi SHANGGUAN, Rong FEI, Wenjiang JI, Weigang MA & Xinhong HEI*

School of Automation and Information Engineering, School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048*, China*

Received 11 September 2019/Accepted 29 November 2019/Published online 15 October 2020

Abstract Accurate monitoring the surrounding environment is an important research direction in the field of unmanned systems such as bio-robotics, and has attracted much research attention in recent years. The trajectories of surrounding vehicles should be predicted accurately in space and time to realize active defense and running safety of an unmanned system. However, there is uncertainty and uncontrollability in the process of trajectory prediction of surrounding obstacles. In this study, we propose a trajectory prediction method based on a sequential model, that fuses two neural networks of a convolutional neural network (CNN) and a long short-term memory network (LSTM). First, a box plot is used to detect and eliminate abnormal values of vehicle trajectories, and valid trajectory data are obtained. Second, the trajectories of surrounding vehicles are predicted by merging the characteristics of CNN space expansion and LSTM time expansion; the hyper-parameters of the model are optimized according to a grid search algorithm, which satisfies the double-precision prediction requirement in space and time. Finally, data from next generation simulation (NGSIM) and Creteil roundabout in France are taken as test cases; the correctness and rationality of the method are verified by prediction error indicators. Experimental results demonstrate that the proposed CNN-LSTM method is more accurate and features a shorter time cost, which meets the prediction requirements and provides an effective method for the safe operation of unmanned systems.

Keywords bio-robots, unmanned system, outlier detection, hyper-parameters, trajectory prediction

Citation Xie G, Shangguan A Q, Fei R, et al. Motion trajectory prediction based on a CNN-LSTM sequential model. Sci China Inf Sci, 2020, 63(11): 212207, https://doi.org/10.1007/s11432-019-2761-y

1 Introduction

In the development of bio-robotic systems, the safety and reliability are important but challenging issues [1]. Traffic transportation, as one of the large application fields of bio-robotic systems, is facing increasing safety problem. It can be seen from historical traffic accidents that ensuring operation safety of traffic transportation is extremely important. Currently, there are many transportation safety performance evaluation methods, such as fault diagnosis [2–5], remaining useful life prediction [6, 7], and parameters identification [8].

As for the self-driving vehicles, they can only ensure their safety by judging whether the surrounding environment is safe during driving. Particularly, they can make corresponding safety decisions by predicting the future trajectories of moving objects around them. Therefore, it is of great significance to accurately and quickly obtain the future driving trajectories of the moving objects around the self-driving vehicle, which can reduce road traffic accidents and improve the safety level of self-driving vehicles.

* Corresponding author (email: guoxie@xaut.edu.cn, heixinhong@xaut.edu.cn)

The existing methods of trajectory prediction are mainly classified into two categories: model-based prediction and data-driven prediction. In the model-based prediction, most studies are based on the Kinematics model or a statistical model. Qiao et al. [9] used the Gaussian mixture model to describe the complex motion patterns of trajectories, calculate the probability distribution of different motion patterns, and predict the most likely trajectory of the moving object. Qiao et al. [10] proposed a trajectory prediction algorithm with self-adaptive parameter selection based on the hidden Markov model, that captures the parameters needed for real scenarios with dynamically changing objects. The results showed that it can improve the prediction efficiency but is extremely time-consuming. For different types of trajectories, Yang et al. [11] proposed an improved least squares algorithm to predict the target trajectory by polynomial fitting of different orders. Moreover, some scholars [12,13] used the constant yaw rate and acceleration motion model to predict trajectory, which achieved short-term trajectory prediction. Besides, different vehicle operating states and non-single model operation can be adapted in an interactive multi-model. Xie et al. [14] proposed a Bayesian-based online correction function that can be adapted to the transition probability of different model states and improve the model response speed and state estimation accuracy.

Although the model-based prediction methods have a great application in the trajectory prediction and are interpretable, they are only useful on the short term. Moreover, it is difficult to obtain precise operations of surrounding vehicles, so the model-based trajectory prediction methods are less effective in actual road scenarios. To overcome the shortcoming of the model-based prediction methods, numbers of data-driven prediction methods have gradually been proposed. Deep learning is considered to be a nonlinear function approximator in mapping nonlinear functions, which is superior in both prediction effect and performance. Various neural networks, such as multi-layer perception, recurrent neural network (RNN), long short-term memory (LSTM) network, and convolutional neural network (CNN) [15], have been applied to different tasks such as image processing, natural language processing, and time series analysis. Using historical data and learning ability of deep neural networks, self-driving vehicles can learn driving skills from human drivers [16]. Deo et al. [17] used an LSTM network to predict interactive perceptual motion trajectories of vehicles on highway. Park et al. [18] proposed an LSTM-based encoder to analyze the fundamental patterns in past trajectories and an LSTM-based decoder to generate future trajectory sequences. Altché et al. [19] applied an LSTM network to predict the vehicle trajectories from next generation simulation (NGSIM). Zhang et al. [20] proposed a two-channel fusion model combined with CNN and LSTM to extract model features. Kim et al. [21] predicted surrounding vehicle positions based on the deep dynamic neural network, by inputting the coordinates and speeds of the surrounding vehicles to an LSTM. Xie et al. [22–24] proposed an improved constrained sparse auto-encoder and correlation analysis method for diagnosing intermittent fault problems, and instrument fault problems are solved effectively with the advantages provided by deep learning. Zhang et al. [25] studied the adaptive neural network (NN) control method of a robot, in which the NN was used to model the uncertain dynamics of the robotic motion. He and Dong [26] used fuzzy NNs to develop a method for identifying uncertain plant models without obtaining prior knowledge of uncertainty and a sufficient number of observations. Xue et al. [27] presented an unmanned system-based precise exploitation and meticulous production solution for the Qarhan Salt Lake, which improved the rapidly exploring random tree algorithm.

The single moving model is the mainstream to predict trajectory in the above studies, lacking in timeliness and precision performance. In this paper, we propose a sequential model that combines the CNN with the LSTM to predict surrounding vehicle trajectories. The hyper-parameters of CNN-LSTM network are optimized by a grid search (GS) algorithm, which can improve the accuracy and speed of trajectory prediction. This enables the self-driving vehicle to make right decisions to avoid obstacles and drive in the complex urban roads safely and smoothly.

The main contributions of this paper are as follows.

(1) We use the box plot method to detect and eliminate the outliers from the original vehicle data, which can improve the trajectory prediction accuracy.

(2) By combining the feature extraction implemented by CNN with the time series prediction imple-

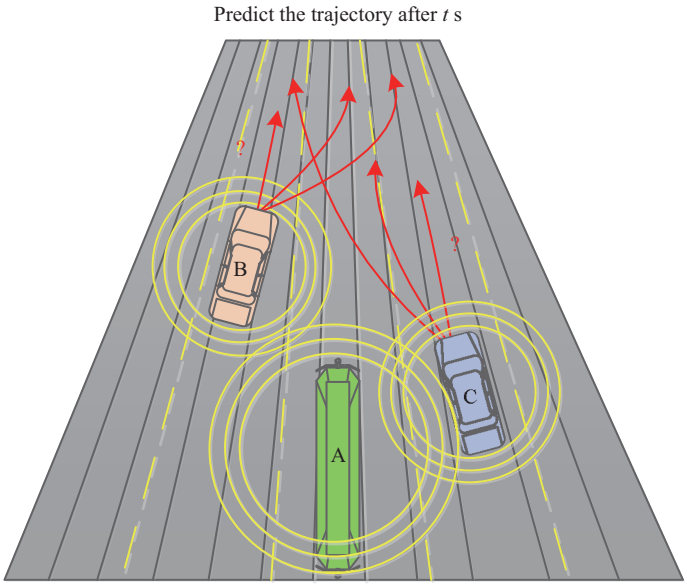

Figure 1 (Color online) A driving scenario of a self-driving vehicle.

mented by LSTM, we propose a vehicle trajectory prediction algorithm with high-precision in space and time and obtain optimal hyper-parameters of CNN-LSTM using a GS algorithm.

(3) We compare the performance of CNN-LSTM with other neural networks, e.g., CNN, LSTM, gated recurrent unit (GRU), in vehicle trajectory prediction. The root mean square error decreases by 9.1% on average, and the prediction efficiency increases by 74.7%.

The rest of the paper is organized as follows. In Section 2, the background of the driving environment of a self-driving vehicle and the overall process of trajectory prediction strategy are introduced. The preprocessing of vehicle trajectory data is presented in Section 3. The design idea and steps of establishing the sequential model of CNN and LSTM are introduced in Section 4. In Section 5, a GS algorithm is used to optimize the hyper-parameters of the CNN-LSTM network. The results of predicting trajectories of surrounding vehicles with CNN-LSTM and comparison with other neural networks are presented in Section 6. A conclusion is given in Section 7.

2 Multi-target trajectory prediction for complex traffic scenarios

2.1 Self-driving vehicle driving scenario

The prediction of the surrounding environment (e.g., pedestrians walking through or traffic light changing) and vehicle operations (e.g., going straight or changing lanes) should be accurate after obtaining the surrounding vehicles' information provided by vehicle-to-vehicle (V2V) technology, which can improve the safety of a self-driving vehicle in uncertain and dynamic environments.

The driving scenario of a self-driving vehicle is shown in Figure 1, which depicts three vehicles on the expressway. Vehicle A is a self-driving vehicle, and the driving information of vehicles B and C can be obtained through V2V technology when vehicle A is driving. Vehicles B and C travel on both sides of A at moment t, but their next-step operations are unknown. Hence, it is difficult for vehicle A to evaluate whether the future is safe. Therefore, it is necessary to predict the trajectories of vehicles B and C and then their driving intentions can be obtained. For example, by judging whether vehicle C drives straight or changes lanes, the obstacle avoidance system of vehicle A can be triggered to brake, decelerate, or maintain the operation in time, and therefore the driving safety of vehicle A can be improved.

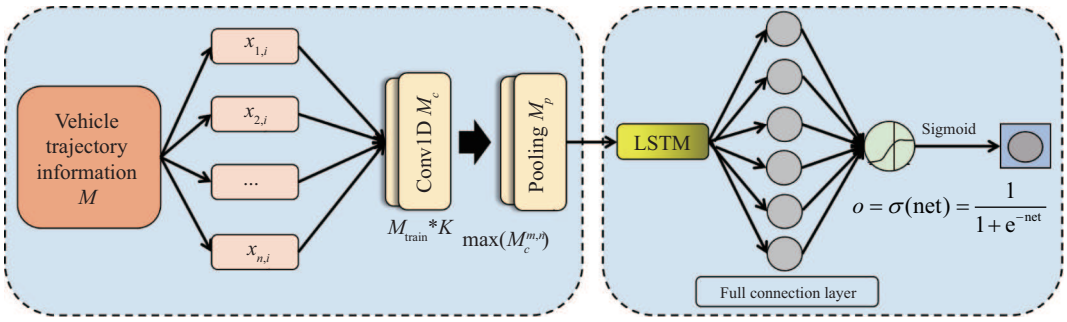

Figure 2 (Color online) The framework of vehicle trajectory prediction.

2.2 Multi-objective trajectory prediction strategy in traffic scenario

The characteristics of the surrounding vehicle trajectories are complex, which should be represented by a complex nonlinear model. Some scholars have used an LSTM network to predict driving vehicle trajectories on the expressway. An LSTM network is suitable for time-series data, but as it costs much time, it cannot meet the requirement for a prediction task. Therefore, a sequential model that combines the characteristics of CNN and LSTM is established, which improves the accuracy and rapidity of trajectory prediction. A sequential model here refers to network layers that are ordered and stacked linearly. The CNN network is used to extract features from input data and the LSTM network is used to predict time-series data.

NGSIM vehicle data are taken as the dataset for training and test in this paper, which are measured by the Federal Highway Administration of the United States [28]. There are many types of vehicle information in the dataset, and the data from NGSIM US-101 section are used in this paper. These data have been widely used in traffic flow applications and tasks, such as car-following model, lane-changing model, and trajectory prediction. The dataset is recorded by a high-definition camera installed on a road section, and the vehicle trajectory data are obtained by high-frequency video restoring. As for the field of data processing, Li et al. [29] proposed a method that combines grid search support vector machine power prediction and first-in-first-out robust smoothing, which smooths data effectively.

The prediction framework is shown in Figure 2, and the steps of the CNN-LSTM trajectory prediction are as follows.

Step 1. Select useful indexes from original data according to the analysis of trajectory prediction problem, eliminate abnormal values for each index, and obtain the valid data.

Step 2. Normalize and divide M into training set M_{train} and test set M_{test}.

Step 3. Feed M_{train} into the convolutional layer and the pooling layer of the CNN network, and then the features of the input data are extracted. The results of the convolutional layer and the pooling layer are as follows:

$$M_c = M_{\text{train}} * \boldsymbol{K}, \tag{1}$$

$$M_p = \max(M_c^{m,n}), \tag{2}$$

where M_c is the result of convolution, the symbol '$*$' represents convolution, and $\boldsymbol{K}$ is the convolutional kernel, which is the window size of convolution. M_p is the result of the pooling, and (m, n) is the pooling window size.

Step 4. Send M_p to the LSTM, and the process is expressed as

$$M_L = M_p \to \text{LSTM}(f_t, i_t, c_t, o_t), \tag{3}$$

where M_L is the result obtained by M_p passing through the forgetting gate, the input gate, the cell state and the output gate of the LSTM.

Step 5. The prediction model is obtained by training and learning, and the hyper-parameters in the model are optimized by a GS algorithm. The prediction result on M_{test} can then be obtained.

Table 1 Useful information in the NGSIM

Name	Description	Unit
x	Lateral offset	Feet
y	Longitudinal offset	Feet
v	Vehicle speed	Feet/s
l	Left lane distance	Feet
r	Right lane distance	Feet

3 Vehicle data preprocessing

3.1 Useful information selection

It is necessary to select useful indexes of vehicle information from the original data because the original data in NGSIM are too massive to feed into the deep learning framework.

The useful indexes of vehicle information in NGSIM are shown in Table 1. The relationship between feet and meters is that one foot equals 0.3048 m (1 foot = 0.3048 m), and the left and right lane distances can be calculated as

$$\begin{aligned} r &= x + k \times 3.6, \\ l &= x - k \times 3.6, \end{aligned} \tag{4}$$

where the lane width is 3.6 m and k is the lane number. The useful indexes of vehicle information can be represented as $M = \{(x, y, v, l, r)\}$.

3.2 Abnormal value detection and elimination

The process of video restoring cannot achieve an ideal result. By detecting and eliminating abnormal values, called the outliers, from the original data effectively, the accuracy of trajectory prediction can be improved. The classic outlier detection methods are the 3σ criterion and the box plot methods. In the 3σ criterion method, the test data must obey normal distribution, while in the box plot method they do not obey a certain distribution, making the method take the quartile and quartile spacing as the standard based on the actual data drawing. The definition of an outlier is usually expressed as

$$x_0 \in \begin{cases} x > Q_U + 1.5\text{IQR}, \\ x < Q_L - 1.5\text{IQR}, \end{cases} \tag{5}$$

where x_0 is an outlier, x is the value to be examined, Q_U is the upper quartile, Q_L is the lower quartile, and IQR represents the inter-quartile range, indicating the difference between Q_L and Q_U.

The box plot method mainly includes six elements, as shown in Figure 3. For a given index in M, the upper and lower limits, upper and lower quartiles, and the median values are calculated. Then every value is evaluated to determine whether it is an outlier.

Using the box plot method, the left distance, right distance, speed, and lateral offset of the vehicle trajectory data from NGSIM are examined and the outliers are detected and eliminated, as shown in Figure 4. The symbol '+' represents an outlier, and it is clear that both the left distance and the speed have outliers. The outliers are replaced by the mean of values on both sides of the outliers. The steps are described as follows.

Step 1. Input $M = \{(x, y, v, l, r)\}$.

Step 2. For each index in M, calculate the upper and lower quartiles Q_U, Q_L, and the upper and lower limits $[Q_U + 1.5\text{IQR}, Q_L - 1.5\text{IQR}]$.

Step 3. For each value of the index, examine the value and replace it if it is an outlier.

Step 4. Output valid data of M.

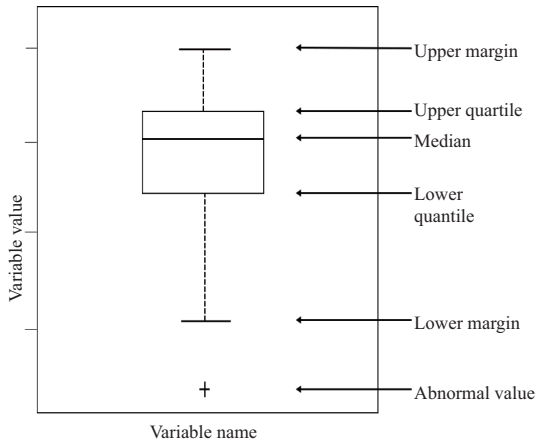

Figure 3 Basic elements of box diagram.

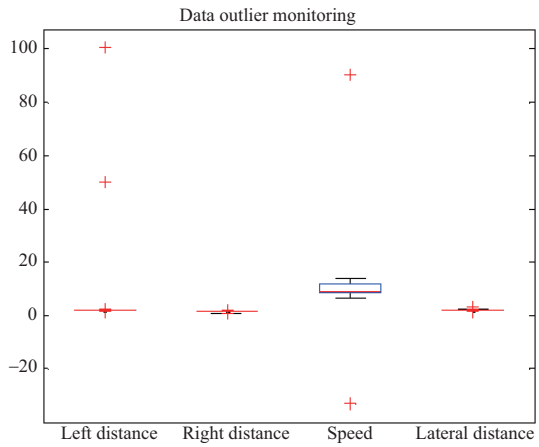

Figure 4 (Color online) Outlier detection results on NGSIM data.

4 Sequential trajectory prediction model

In the previous studies, the trajectory prediction method was carried out by establishing a vehicle motion model, which exhibits a large trajectory deviation in long-term prediction. The nonlinear function that approximates real data can be achieved by deep learning. The theory of deep learning was proposed by Hinton et al. [30] and has made great progress in the field of artificial intelligence. By integrating the characteristics of CNN and LSTM, a CNN-LSTM sequential model is established in this section.

CNN is an NN structure that proposed by Lecun et al. [31]. It consists of an input layer, an output layer, and several hidden layers. The hidden layers can be divided into a convolutional layer, a pooling layer, and a full connection layer. The convolutional layer and the pooling layer are the core, which can extract features, reduce the number of model parameters [32], and accelerate the training speed. The formula of the convolutional layer is as follows:

$$\begin{aligned} M_{c,t} &= f(M_{\text{train},t} * \boldsymbol{K} + b) \\ &= f\left(\sum_j M_{\text{train},j} * \boldsymbol{K}_{t-j} + b_j\right), \end{aligned} \tag{6}$$

where $M_{c,t}$ is the final convolution result, $M_{\text{train},t}$ is the data at moment t before inputting variable (x, y, v, l, r) into the training data, and $\boldsymbol{K}$ is the convolution kernel that does one-dimensional (1D) convolution on all the associated feature ranges. 1D convolution means that the convolution kernel is 1D, but not the input. j, b, and f are the current moment, the bias, and the activation function, respectively.

The specific performance is shown in Figure 5, where $x_1, x_2, x_3, \ldots, x_n$ are the input sequences, F_1, F_2, F_3 represent the convolution kernels, and w_1, w_2, w_3 are the weights generated by 1D convolution. Moreover, only a few inputs with typical characteristics are convoluted.

The pooling layer is also called the sampling layer. The results obtained by the convolutional layer are pooled, which can reduce the data size and improve the anti-jamming ability of the network. The calculation formula of the maximum pooling layer is as follows:

$$M_p = \max(M_{c,t}^{m,n}), \tag{7}$$

where $M_{c,t}$ is the output of the convolutional layer, and (m, n) represents the size of the pooling window; that is, the maximum value is sampled in the window.

The pooling layer is used to detect whether the feature is in the area covered by the window after convolution, which can preserve the relative position of the feature. The pooling window size is set in pooling layer, and its size is consistent with the step size because the pooling window cannot overlap in each pooling step.

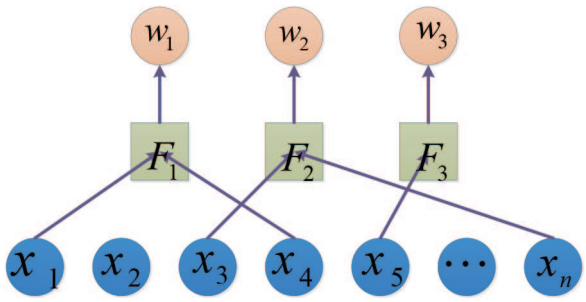

Figure 5 (Color online) 1D convolution for time series data.

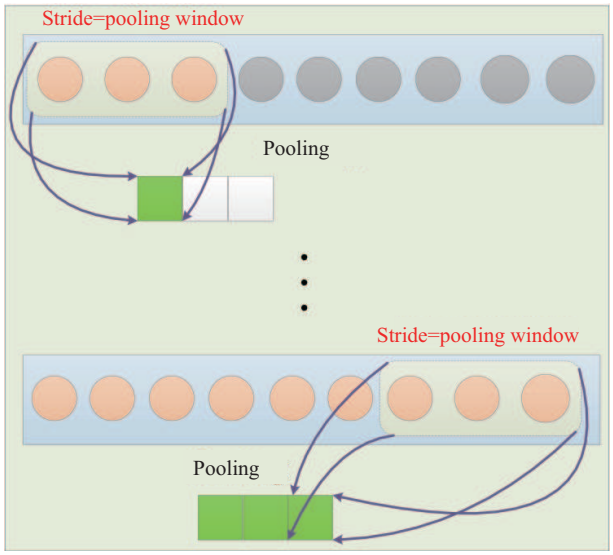

Figure 6 (Color online) Pooling for time series data.

The pooling performance is shown in Figure 6, where the pooling window size is equal to the move step size. The main features of the input are extracted by selecting the max value in the window, which can simplify the computational complexity of the network. Also, the pooling layer does not have parameters, which reduces the number of model parameters, thus improving the training efficiency.

LSTM is a variant of RNN [33], which was introduced in 1997 by Hochreiter and Schmidhuber [34]. Unlike RNN which tends to suffer from gradient disappearance or gradient explosion caused by passing previous units sequentially, LSTM adds a cell state in the RNN to remember the long-term state, and therefore it can determine which states should be left or forgotten, resulting in solving the gradient disappearance or gradient explosion of RNN. The advantage of LSTM in long-term memory makes it useful in long-term vehicle trajectory prediction.

Our proposed LSTM is composed of an input gate, the hidden unit state, a forgetting gate and an output gate. Its structure is shown in Figure 7, where x_t and h_t represent input unit and state output unit, respectively, and the intermediate cell memory unit c_t is controlled by three threshold units, i_t, o_t and f_t, which represent the input gate, the output gate, and the forgetting gate, respectively. The input gate is used to control the information input, the forget gate is used to control the retention of the historical state information of the cell, and the output gate is used to control the information output.

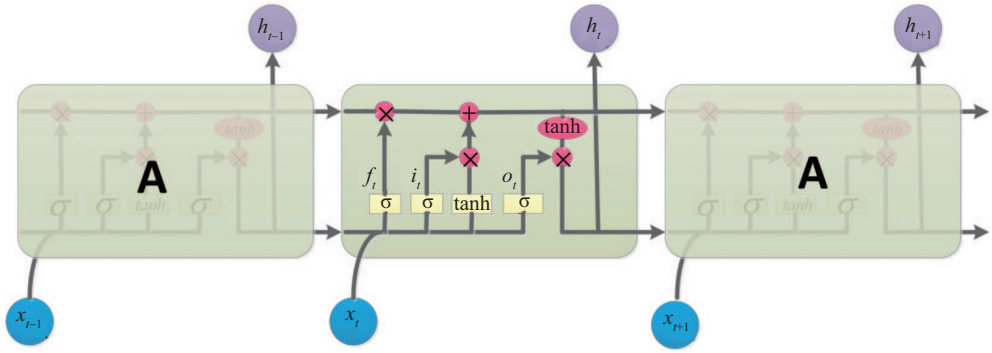

Figure 7 (Color online) LSTM structure.

The process can be expressed as follows:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i), \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f), \\ c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_{t-1} + b_o), \\ h_t &= o_t \tanh(c_t), \end{aligned} \tag{8}$$

where x_t denotes the current input sequence, which is also the pooling layer output M_p in this paper. t, h_t, i_t, and f_t represent the current time, the output, the output of input gate, and the output of forgetting gate, respectively. c_t is the cell unit at the current moment t, and o_t is the output of the output gate. W and b are the weight matrix and the offset vector, and σ represents the sigmoid activation function.

The main process of the trajectory prediction method is listed as follows.

Data preprocessing. The time series vehicle data after the abnormal value detection are represent as $M_N = \{(x, y, v, l, r)\}$, where N is the total length of the data. M_N are divided into training set M_{train} and test set M_{test}, where the length of training set is t, the length of the test set is $N - t$, (v, l, r) is the input of the CNN-LSTM network, and the offsets x, y are the output.

CNN-LSTM sequential model construction. The CNN-LSTM model including three layers: input layer, hidden layer, and output layer. Feed vehicle data (v, l, r) into the input layer, and the output layer outputs the offsets x, y. The hidden layer is composed by the convolutional layer and the pooling layer of the CNN, the LSTM network, and the fully connected (FC) layer.

Trajectory prediction model training. The weight parameter W and the deviation b in the network are initialized. The training set M_{train} includes the input (v, l, r) and the output x, y. Each sample in the dataset corresponds to the instantaneous information of a vehicle.

The model training process includes forward and back propagations. In the forward propagation, the relationship model between the input (v, l, r) and the offsets x and y is constructed. First, the training samples (v, l, r) are divided into several groups according to the the length of batch size, fed to the input layer, and transferred to the convolutional layer and the pooling layer according to (6) and (7). Second, the effective information M_p is extracted and then transferred to the LSTM according to (8), resulting in obtaining the output h_t. Finally, the predicted offsets x_p and y_p are obtained through the FC layer. The coefficients of the relationship model are the network's weight parameters.

In the backpropagation process, the relationship model is optimized by adjusting the weight parameters. The error between the true offsets x, y and the predicted offsets x_p, y_p is calculated by the loss function (11). The loss-value minimizing principle is used to reversely adjust the weight parameters, that can improve the prediction accuracy of the model.

Repeat the above propagations until all training samples have been trained.

Prediction on the test set. Each sample (v, l, r) in the M_{test} is put into the prediction model,

and the results of the offsets x, y are obtained in turn. The root mean square error (RMSE), the mean absolute error (MAE), and the deviation are calculated using (12)–(14).

5 Parameter optimization of the CNN-LSTM sequential model

Data normalization is the first step to eliminate the influence of data dimension in the network training procession. Every variable exhibits the same order of magnitude after normalization, which can improve the prediction accuracy. In this paper, maximum and minimum normalization is used to make the data's distribution have a mean value of 0 and a variance of 1. The formula is as follows:

$$x_{\text{norm}} = \frac{x_{\text{ori}} - x_{\text{ori}}^{\min}}{x_{\text{ori}}^{\max} - x_{\text{ori}}^{\min}}, \tag{9}$$

where $x_{\text{ori}}, x_{\text{ori}}^{\min}, x_{\text{ori}}^{\max}, x_{\text{norm}}$ represent the original data, the minimum of original data, and the maximum of original data, and the normalized data, respectively.

After data normalization, a sequential model is established. The input data is processed in the convolutional layer with 1D convolution, and then the calculation result is sent to the pooling layer. The maximum pooling method is used to find the maximum value in the area swept by the pooling window. The result is passed to the LSTM layer, then flattened into a column vector, and input to the FC layer. Finally, the output layer contains a neuron. The activation function of the output layer is a sigmoid function, and its expression is as follows:

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{10}$$

The loss function (loss), prediction bias (deviation(x)), root mean square error (RMSE), and the average absolute error (MAE) are used as evaluation indicators to verify the model prediction accuracy and its performance. These indicators are calculated as follows:

$$\text{loss} = \frac{\sum_{i=1}^{n} |x_i^o - x_i^p|}{n}, \tag{11}$$

$$\text{deviation}(x) = \frac{1}{n}\left(\frac{\sum_{i=1}^{n} |x_i^o - x_i^p|}{x_i}\right), \tag{12}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i^o - x_i^p)^2}, \tag{13}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i^o - x_i^p|, \tag{14}$$

where x_i^o and x_i^p represent truth data and predicted results, respectively. The predicted results tend to be true when the above indexes are close to zero, thus ensuring better prediction effect and model performance.

The above sequential model framework is shown in Figure 8.

The hyper-parameters have great influence on network performance, and the efficiency of trajectory prediction can be improved by a faster convergence rate. Therefore, a GS algorithm is used to optimize the hyper-parameters of the network. GS is used to exhaustively search the optimal value manually by comparing all results of a set of hyper-parameters. If a larger search range and a smaller step size are used, the GS will find the global optimal value with high probability. The most representative hyper-parameters mainly include the optimizer, the convolution filter size, the kernel size, the number of LSTM neurons, the number of LSTM layers, the number of neurons in the FC layer, and the batch size.

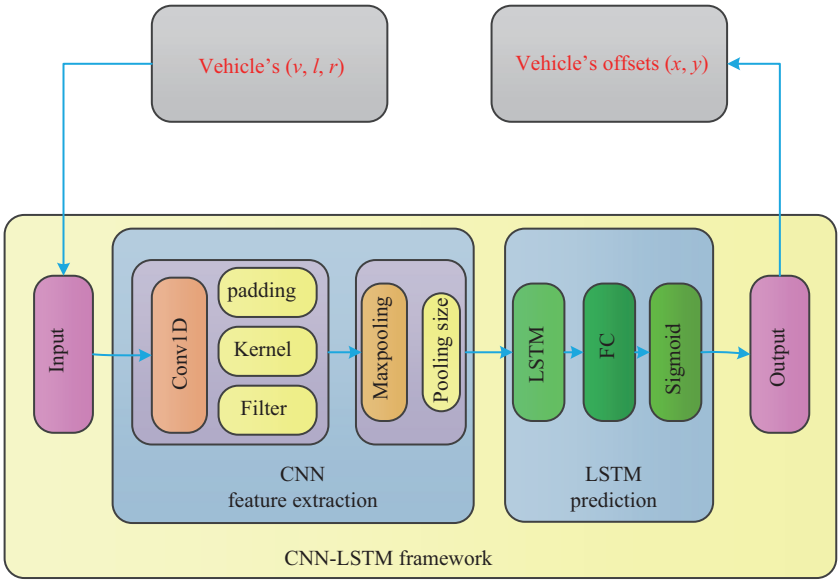

Figure 8 (Color online) CNN-LSTM sequential model framework.

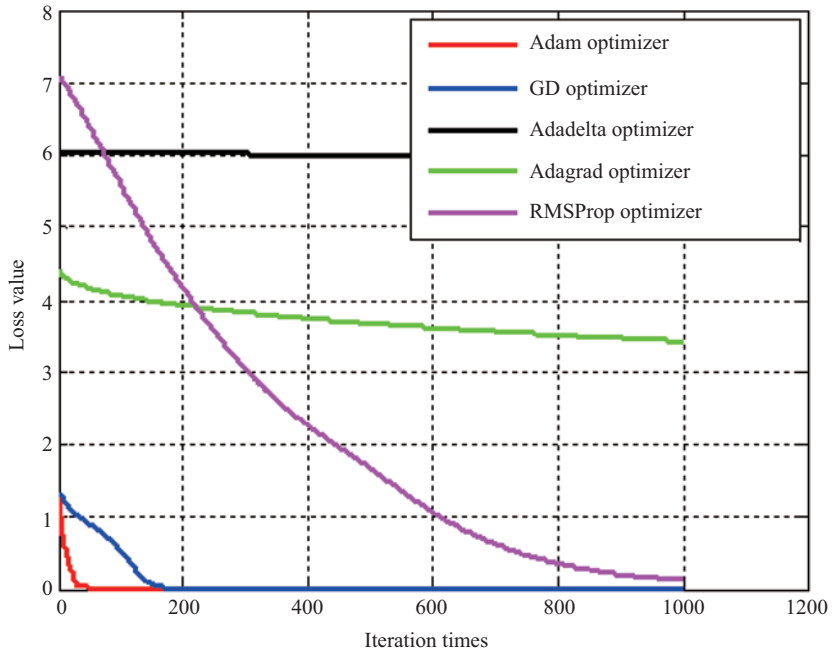

Figure 9 (Color online) Loss value for different optimizer.

5.1 Optimizer

The main optimizer used in the CNN-LSTM network is stochastic gradient descent (SGD) [35], adaptive moment estimation (Adam), adaptive delta (Adadelta), adaptive gradient (Adagrad), and root mean square prop (RMSProp). The optimal result is obtained by comparing the loss value of the loss function under the same number of iterations (Figure 9).

In Figure 9, the x-coordinate indicates the iteration times, and the y-coordinate indicates the test loss value. Under the same number of iterations, the convergence speed of the Adam optimizer is clearly better than that of the other optimizers, and its loss function has the lowest value and costs the least time. Therefore, Adam is selected as the optimizer.

Table 2 Prediction error of different filters

	KERNEL 5	KERNEL 10	KERNEL 15	KERNEL 20
RMSE	0.04731	0.02724	0.03417	0.03368
MAE	0.03504	0.01555	0.02079	0.02034
Deviation	0.00617	0.00268	0.00361	0.00353

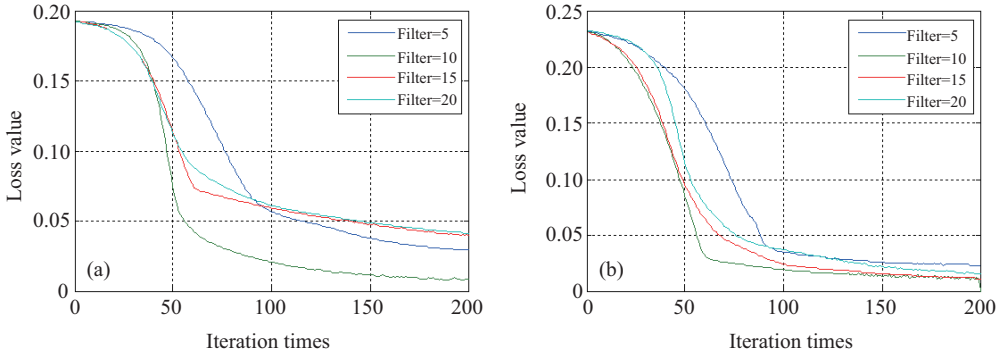

Figure 10 (Color online) Loss value for different convolution kernels. (a) Training dataset; (b) test dataset.

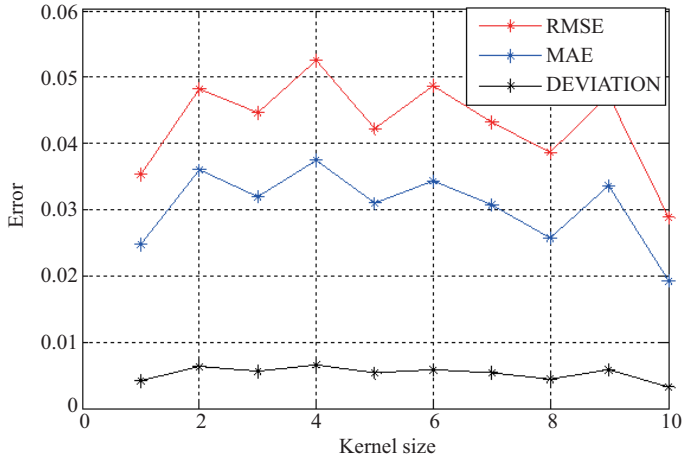

Figure 11 (Color online) Prediction error for different kernel size.

5.2 Convolution filter size

The convolution kernel is the core of the convolutional layer of the CNN, and the kernel can reduce the input data dimension and model parameters, which can increase the depth of the model such that the representation capability of the model is improved. The optimal kernel size is determined by considering the prediction error (shown in Table 2) and the loss function convergence (shown in Figure 10) of different convolution kernels. It can be seen from Table 2 that the RMSE, MAE, and the deviation are the smallest when the convolution kernel is 10. Moreover, the loss convergence of the kernel of 10 is shown to be the fastest. Therefore, the optimal kernel is chosen as 10.

5.3 Kernel size

The width of the convolution kernel is 1, and the kernel size is the length of the convolution kernel. The convolution kernel can tell us how many different windows there are, and the kernel size determines the window size. For example, when the convolution kernel is 100 and the kernel size is 4, the convolutional layer will create 100 different filters and the length of each filter is 4. Therefore, the optimal kernel size is determined by the prediction error (Figure 11) and the loss function convergence (Figure 12) of different kernel sizes.

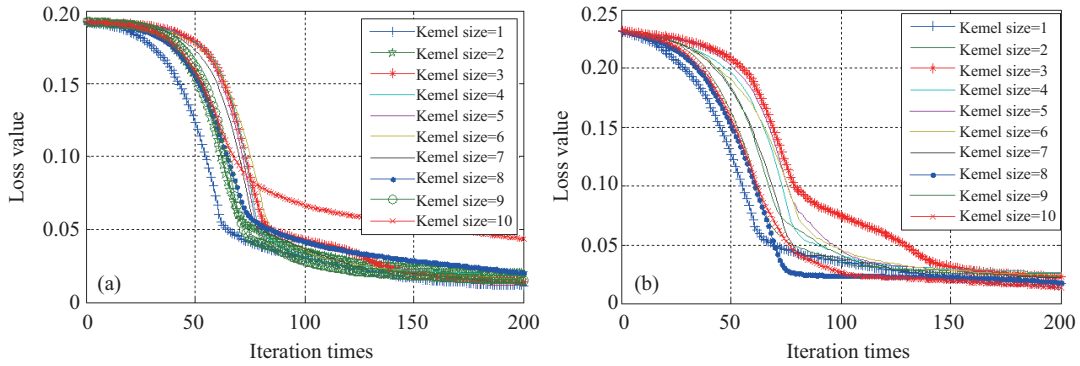

Figure 12 (Color online) Loss value for different kernel size. (a) Training dataset; (b) test dataset.

Table 3 Prediction error for different number of LSTM neurons

	3 neurons	6 neurons	9 neurons	12 neurons
RMSE	0.02923	0.02248	0.02656	0.04312
MAE	0.01746	0.01188	0.01844	0.03181
Deviation	0.00304	0.00205	0.00323	0.00556

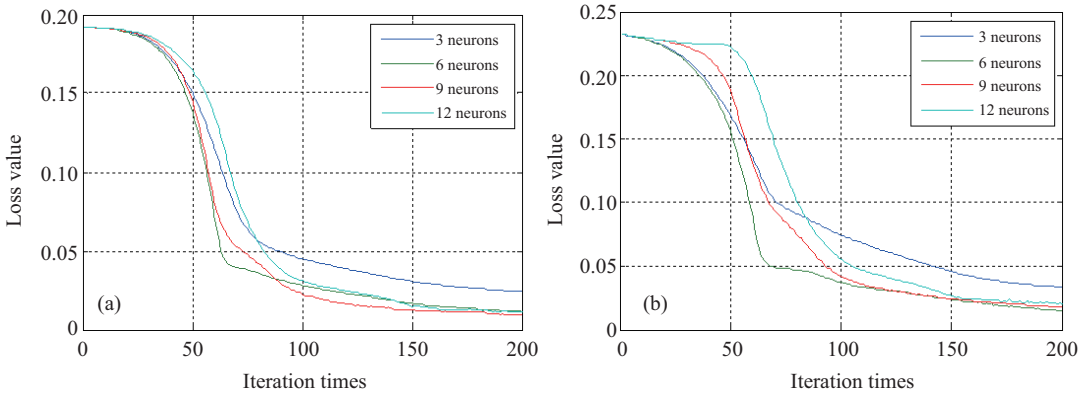

Figure 13 (Color online) Loss value for different number of LSTM neurons. (a) Training dataset; (b) test dataset.

In Figure 12, the loss function converges quickly when the kernel size of the training set and test set is 1, but the loss function of the test set converges most slowly when the kernel size is 8. In Figure 11, the RMSE, MAE, and the deviation are all smallest when the kernel size is 10 or 1, and the difference approaches zero. Moreover, the loss functions of the training set and test set exhibit the fastest convergence rate when the kernel size is 1. Therefore, the kernel size is chosen to be 1 here, which is more appropriate.

5.4 The number of LSTM neurons

The appropriate number of LSTM neurons (memory unit) has great influence on the network prediction accuracy. Therefore, the optimal unit is determined by changing the number of neurons. The range of the number of neurons are 3, 6, 9, and 12, and the optimal number of neurons is determined by evaluating the prediction error (shown in Table 3) and the loss value of different number of neurons (shown in Figure 13). In Figure 13, when the number of neurons in the LSTM is 6, the loss value is the lowest, and its prediction error is the lowest; hence the number 6 of neurons in the LSTM is suitable.

5.5 The number of LSTM layers

Considering that the increase of the depth of the NN can enhance the model representation, the number of LSTM layers is set to 2, and the number of neurons in the second layer is set to 3 or 6 for test. The

Table 4 Prediction error for different number of LSTM layers

	One LSTM layer	Two LSTM layers with 3 neurons	Two LSTM layers with 6 neurons
RMSE	0.03853	0.06607	0.35303
MAE	0.02473	0.04929	0.28778
Deviation	0.00427	0.00861	0.05022

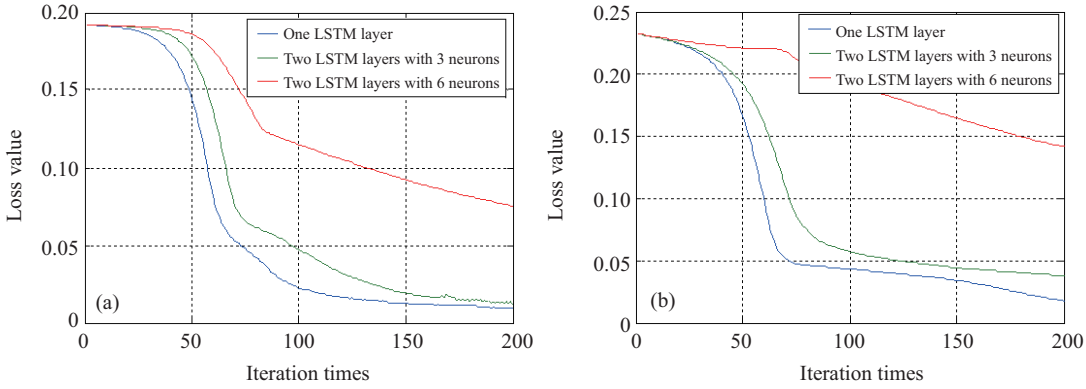

Figure 14 (Color online) Loss value for different number of LSTM layers. (a) Training dataset; (b) test dataset.

Table 5 Prediction error for different number of neurons in the FC layer

	3 neurons	6 neurons	9 neurons	12 neurons
RMSE	0.05429	0.03896	0.05138	0.04488
MAE	0.04028	0.02492	0.03751	0.03257
Deviation	0.00703	0.00431	0.00654	0.00568

optimal LSTM layer number is determined by evaluating the prediction error (shown in Table 4) and the loss value (shown in Figure 14) of different LSTM layers and the number of neurons. When the LSTM layer number is 2 and the number of neurons in the second layer is 3, the loss value increases, and the prediction accuracy decreases. If the number of neurons in the second layer is 6, the loss value is larger, and the prediction error is larger than other parameters. Therefore, the LSTM layer number is chosen as 1.

5.6 The number of neurons in the FC layer

Useful information can be retained by the FC layer. However, the parameters in the FC layer occupy about 80% of the overall network structure. Among them, the number of neurons in the FC layer has a great influence on the parameter learning. A suitable number of neurons can reduce the complexity and improve the learning capability of the model. The number of neurons should not be too large, or the network running time will be increased and the efficiency will be reduced. Therefore, the optimal number of FC layer neurons is determined by accessing the prediction error (shown in Table 5) and the loss value (shown in Figure 15) of different number of neurons. The neurons in the FC layer ranges among 3, 6, 9 and 12. It can be found that the loss value and prediction error are the optimized when the number of neurons in the FC layer is 6. Therefore, the number of neurons in the FC layer is set to 6.

5.7 Batch size

Batch size affects the optimization degree, the model convergence rate, and GPU memory consumption. The appropriate batch size can determine the best equilibrium between storage efficiency and capacity. The appropriate batch size is selected by considering the prediction error variation (shown in Figure 16), and the loss value (shown in Figure 17) under different batch sizes. Because batch size is strongly related to the optimization speed of the model, the network running time and CPU consumption time under different batch sizes are counted, and the result is shown in Figure 18. It can be seen from the above

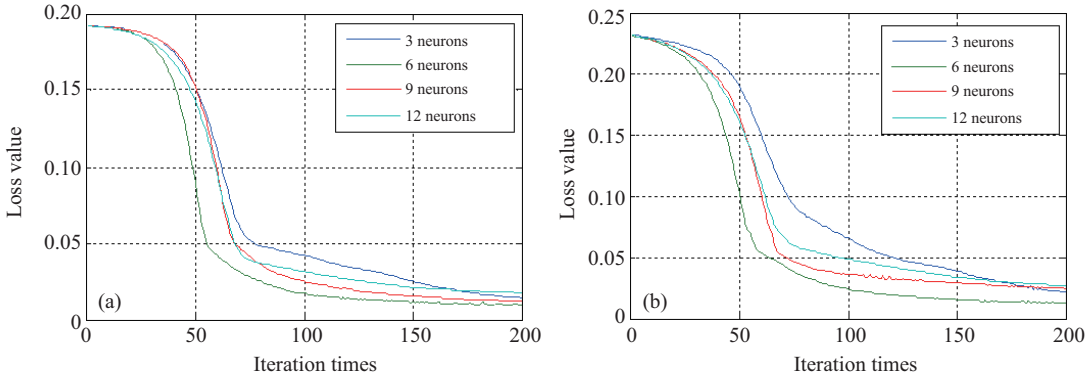

Figure 15 (Color online) Loss value for different number of neurons in the FC layer. (a) Training dataset; (b) test dataset.

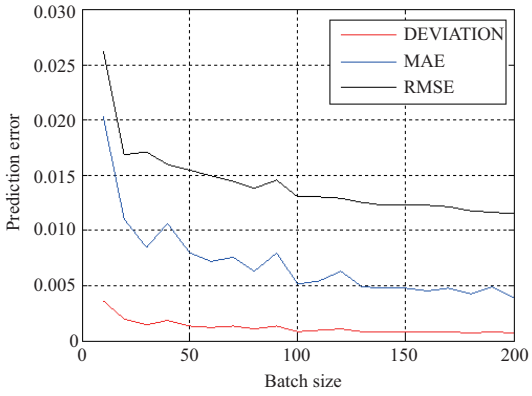

Figure 16 (Color online) Prediction error for different batch sizes.

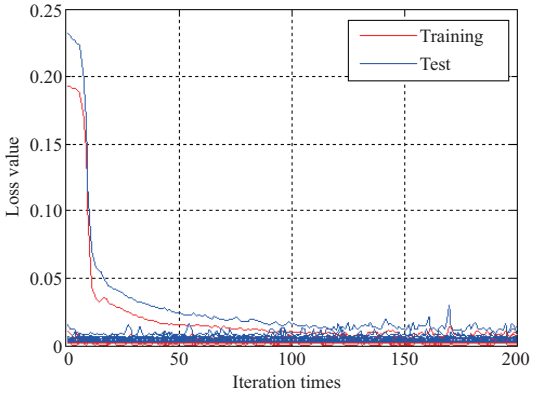

Figure 17 (Color online) Loss value for different batch sizes.

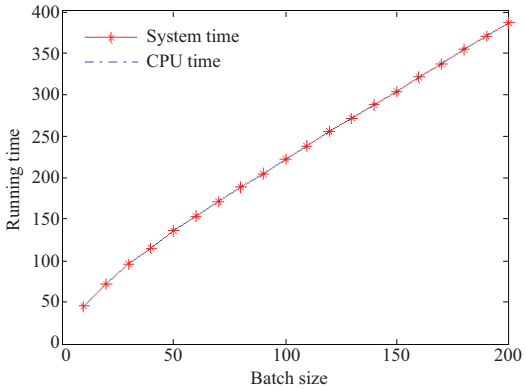

Figure 18 (Color online) Network consumption time for different batch sizes.

results that the larger the batch size is, the smaller the prediction error is. However, the consumption time is increased. The prediction error tends to be relatively flat and the consumption time is moderate when the batch size is greater than 100. Therefore, an appropriate batch size is 100.

Table 6 Vehicles in four lanes

	LANE 1	LANE 2	LANE 3	LANE 4
Lane-keeping	V1	V3	V5	V7
Lane-changing	V2	V4	V6	V8

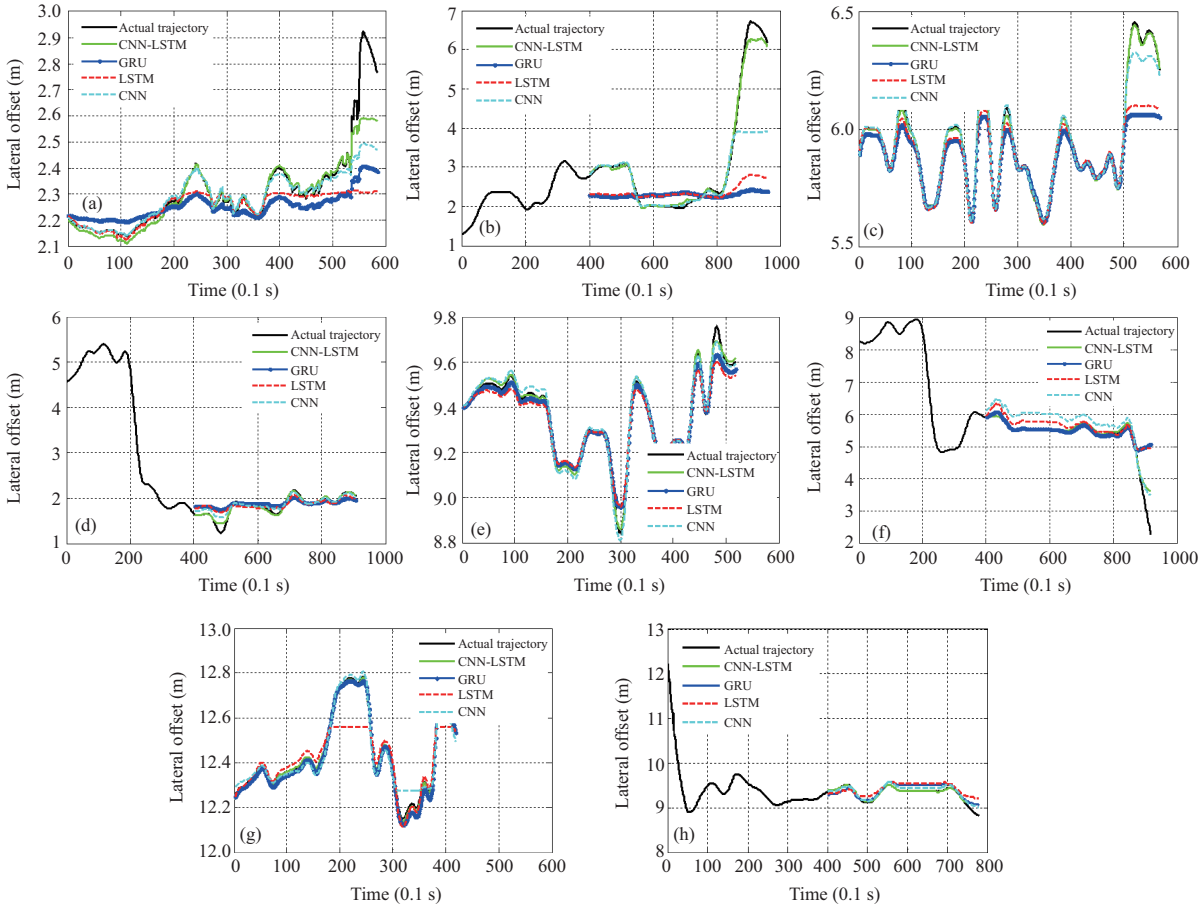

Figure 19 (Color online) Predicted lateral trajectories of surrounding vehicles. (a) V1; (b) V2; (c) V3; (d) V4; (e) V5; (f) V6; (g) V7; (h) V8.

6 Multi-objective location prediction and analysis of traffic scenario

6.1 Prediction results on the NGSIM

In this experiment, the trajectory data of eight vehicles are selected from the NGSIM. The vehicles performed lane-keeping and lane-changing modes in four lanes, as shown in Table 6.

Because the longitudinal offset is the vehicle's forward distance, the relationship between the lateral offset and other variables must be considered. So the relationship between x and the left lane spacing, the right lane spacing, and speed, $x = f(l, r, v)$, is established through CNN-LSTM. The GRU, LSTM, CNN-LSTM, and CNN models are used to predict the eight trajectories, and the relative error of each prediction method is obtained. The prediction results are shown in Figure 19, and the prediction error is shown in Figure 20.

In Figure 19, the prediction results of GRU, LSTM, CNN-LSTM, and CNN models have different deviations as compared with the real values, but the trajectories predicted by CNN-LSTM are closest to the real trajectories. The trajectories of GRU are inconsistent with those of LSTM, both having a large deviation from the real trajectory; that is to say, LSTM and GRU fail to correctly predict the true trajectory when the trajectory has a large offset, and the deviation with the real trajectory over

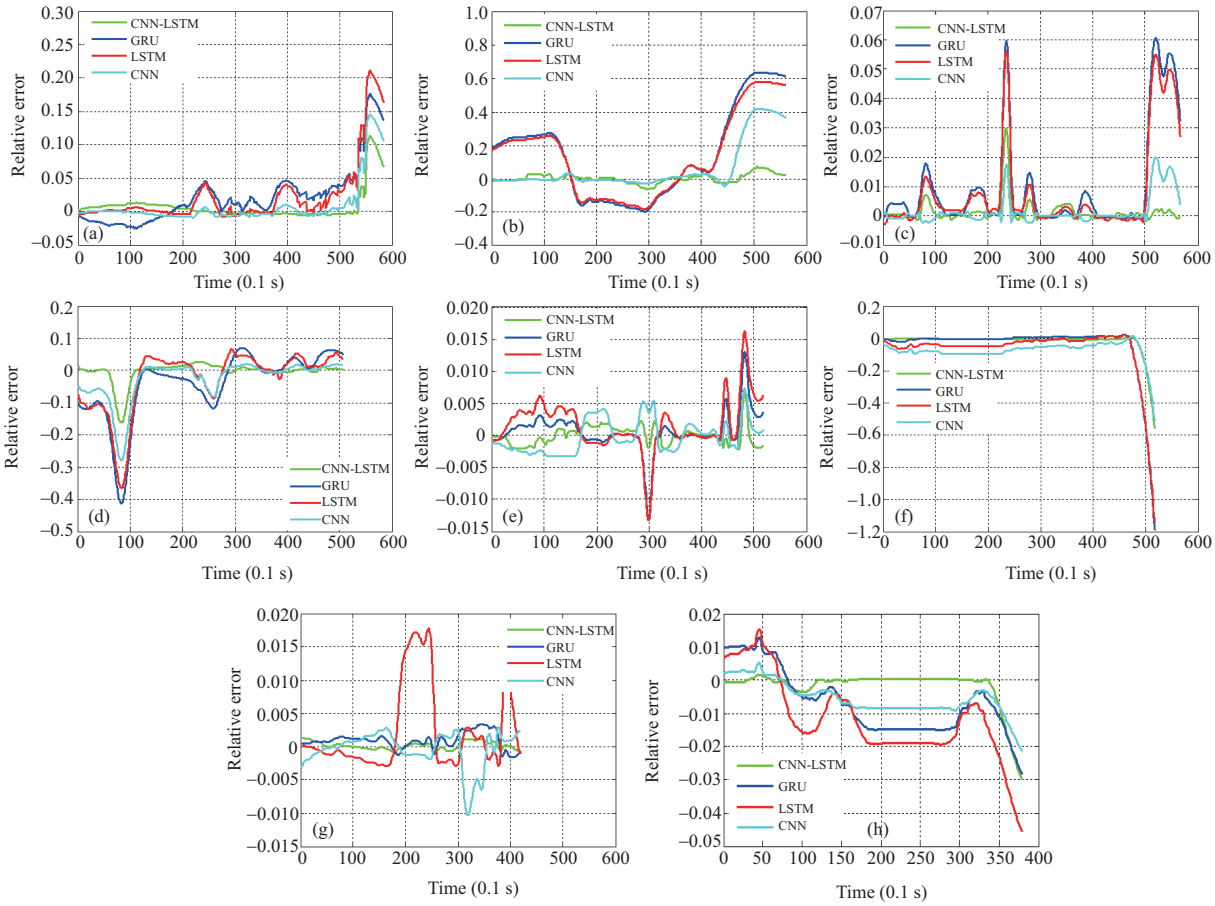

Figure 20 (Color online) Prediction relative error. (a) V1; (b) V2; (c) V3; (d) V4; (e) V5; (f) V6; (g) V7; (h) V8.

a long time is large. Therefore, the CNN-LSTM model is more robust, and the prediction trajectories over a long time are more accurate and stable. It can be seen from Figure 20 that the relative error of the CNN-LSTM is smaller, all fluctuations are almost close to zero, and the fluctuation range is smaller than that of GRU and LSTM. The change in the relative error between LSTM and GRU for V1 and V2 vehicles is consistent, and the relative error exceeds 20% after 50 s, whereas the CNN-LSTM relative error does not exceed 7%. For the V3 vehicle, the relative error of the CNN-LSTM, CNN, LSTM, and GRU models are less than 1%, less than 2%, more than 2%, and more than 4% after 50 s, respectively. The running times of GRU and LSTM are two times longer than that of the CNN-LSTM model, which greatly reduces the effectiveness of the trajectory prediction.

According to the various error indicators as shown in Table 7, the CNN-LSTM model performs better than single networks. In Table 7, the error indicators (RMSE, MAE, and deviation) of CNN-LSTM on the V1 trajectory data are reduced by 5.6%, 3.2%, and 1.2%, respectively, as compared with that of LSTM, GRU, and CNN. The prediction time of the CNN-LSTM is increased by 53.7%. The RMSE, MAE, and deviation on the V2 trajectory data are decreased by 12.6%, 7.2%, and 1.6%, respectively, while the prediction time of the CNN-LSTM is increased by 34.5%. In practice, the predicted trajectory offset ranges from 0.17 to 0.23 m based on the real trajectory when the deviation error decreases from 1.2% to 1.6%.

Although the prediction error and time cost of our CNN-LSTM model are reduced not so much than those of other single models, it is a meaningful improvement for real scenarios where 1-cm decrease of the distance between vehicles is enough to cause a car accident. Furthermore, the prediction time of CNN-LSTM model is less than 1 s when compared to other single models, which can make the vehicle better avoid surrounding obstacles or danger. For the safety performance of self-driving vehicles, the prediction results of CNN-LSTM model can be effectively applied to real scenarios.

Table 7 Various error indicators for trajectory prediction using different models

		CNN-LSTM	LSTM	GRU	CNN
V1	RMSE	0.06933	0.14911	0.13252	0.09680
	MAE	0.02830	0.06519	0.07954	0.03618
	Deviation	0.01095	0.02487	0.03183	0.01357
	Time	4.8037	10.3154	6.4992	8.5562
V2	RMSE	0.12953	1.52990	1.67531	0.99138
	MAE	0.07426	0.94095	1.02386	0.42756
	Deviation	0.01936	0.22333	0.24207	0.07478
	Time	8.7319	13.3231	9.0863	11.5380
V3	RMSE	0.02734	0.08685	0.12410	0.03316
	MAE	0.01086	0.04670	0.06239	0.01496
	Deviation	0.00178	0.00752	0.00995	0.00238
	Time	5.0013	9.1223	5.2013	6.9874
V4	RMSE	0.04316	0.14060	0.16167	0.09295
	MAE	0.02214	0.10137	0.11464	0.05610
	Deviation	0.01442	0.06222	0.07053	0.03613
	Time	5.5436	8.1660	8.4666	6.9444
V5	RMSE	0.01382	0.04065	0.02898	0.02418
	MAE	0.01043	0.02655	0.01620	0.02050
	Deviation	0.00111	0.00281	0.00172	0.00219
	Time	6.2157	7.8658	6.2764	6.7486
V6	RMSE	0.20344	0.52590	0.51133	0.40998
	MAE	0.06706	0.27635	0.17553	0.37104
	Deviation	0.02041	0.07026	0.05287	0.07420
	Time	5.3700	7.1364	7.3454	6.1584
V7	RMSE	0.00690	0.08620	0.01849	0.03323
	MAE	0.00546	0.05337	0.01541	0.02315
	Deviation	0.00041	0.00423	0.00122	0.00191
	Time	5.9349	9.0136	7.7046	7.2841
V8	RMSE	0.05230	0.15493	0.10782	0.06727
	MAE	0.01921	0.13760	0.09677	0.05844
	Deviation	0.00213	0.01485	0.01040	0.00630
	Time	6.2239	10.2061	9.0357	8.7624

6.2 Prediction results on Creteil roundabout data

To further verify the effectiveness of the CNN-LSTM model, another dataset is used to predict the vehicle's trajectory, which was recorded at Creteil roundabout in France from 5:00 p.m. to 7:00 p.m. on September 24, 2013. The useful indexes of vehicle information are selected and represented as $I = (x, y, w, v, p)$, where w, v, and p denote the vehicle's angle, speed, and position, respectively, and x and y are the vehicle's lateral and longitudinal offsets. The unit of offset is meter (m).

The outliers of Creteil roundabout data are processed by the box plot method, as shown in Figure 21. It can be seen that only the w index has outliers. The outliers are replaced by the mean of values on both sides of the outliers. The CNN-LSTM, CNN, LSTM and GRU models are used to predict the four vehicles' trajectories in the Creteil roundabout data. the prediction results of Creteil roundabout data are shown in Figure 22.

The trajectory prediction errors on Creteil roundabout data are calculated using (11)–(14), as shown in Figure 23 and Table 8. It can be seen that the prediction error of the CNN-LSTM model is 1%–3% less than that of other single models, and the prediction using CNN-LSTM model costs less time than using other models by a value varing from 1.5 to 3.5 s.

It can be seen from Figure 23 and Table 8 that the relative prediction error of the CNN-LSTM is less

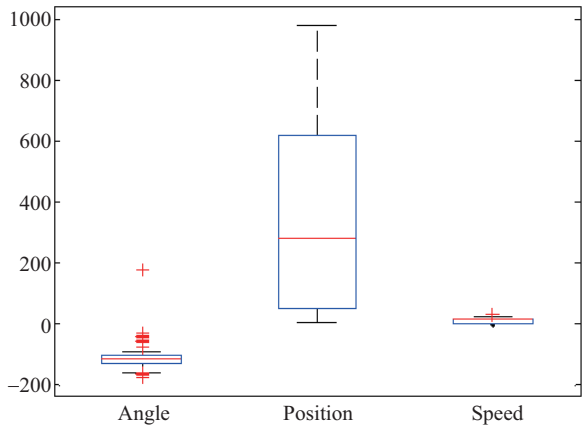

Figure 21 (Color online) Result of outlier value detection via input data at annular crossing.

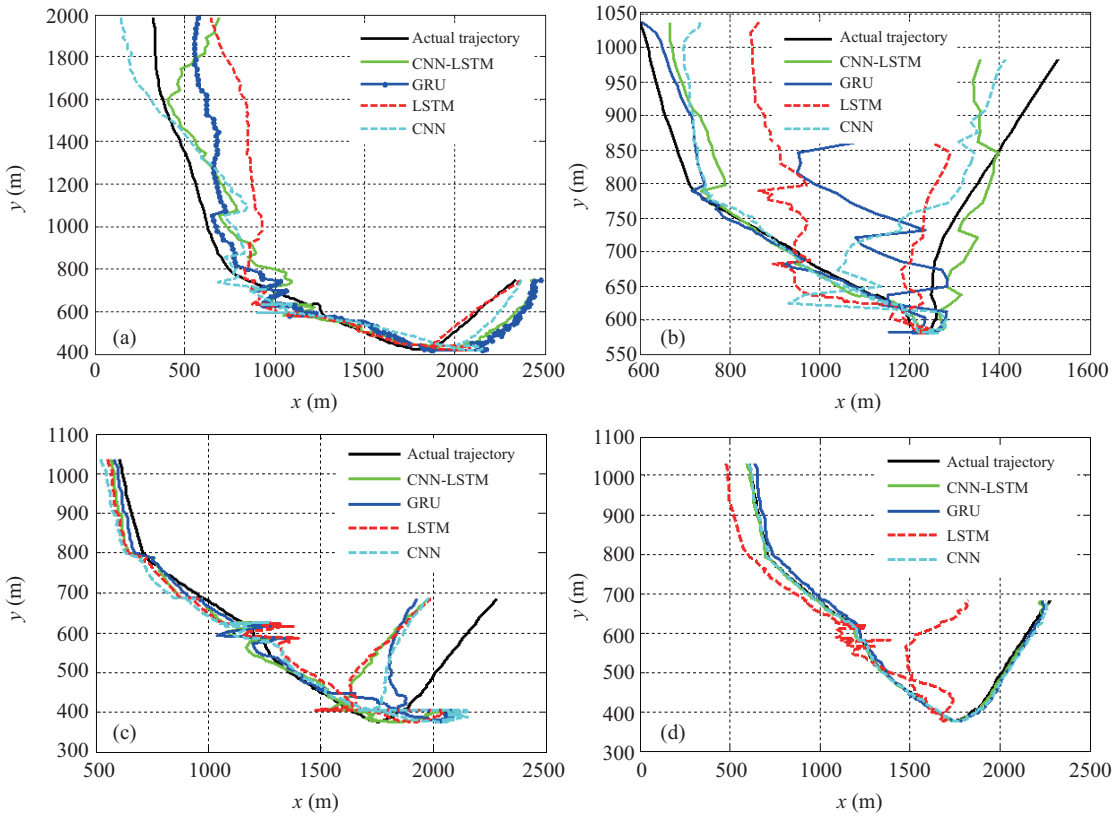

Figure 22 (Color online) Predicted trajectories of surrounding vehicles. (a) V1; (b) V2; (c) V3; (d) V4.

than 1%, which can predict the trajectory quickly and can provide an effective basis for decision-making evaluation of self-driving vehicles. The relative errors of GRU, CNN and LSTM are more than 2%, and up even 3% at the same time. Large errors can lead to incorrect decisions in assessing road safety. The prediction using CNN-LSTM model costs less time than using other models by a value varing from 1.5 to 3.5 s.

According to the above experimental results, the prediction metrics reflect the accuracy and rapidity of the CNN-LSTM model. Therefore, CNN-LSTM performs a better result and can build a solid foundation for the safe operation of a self-driving vehicle in complex traffic scenarios.

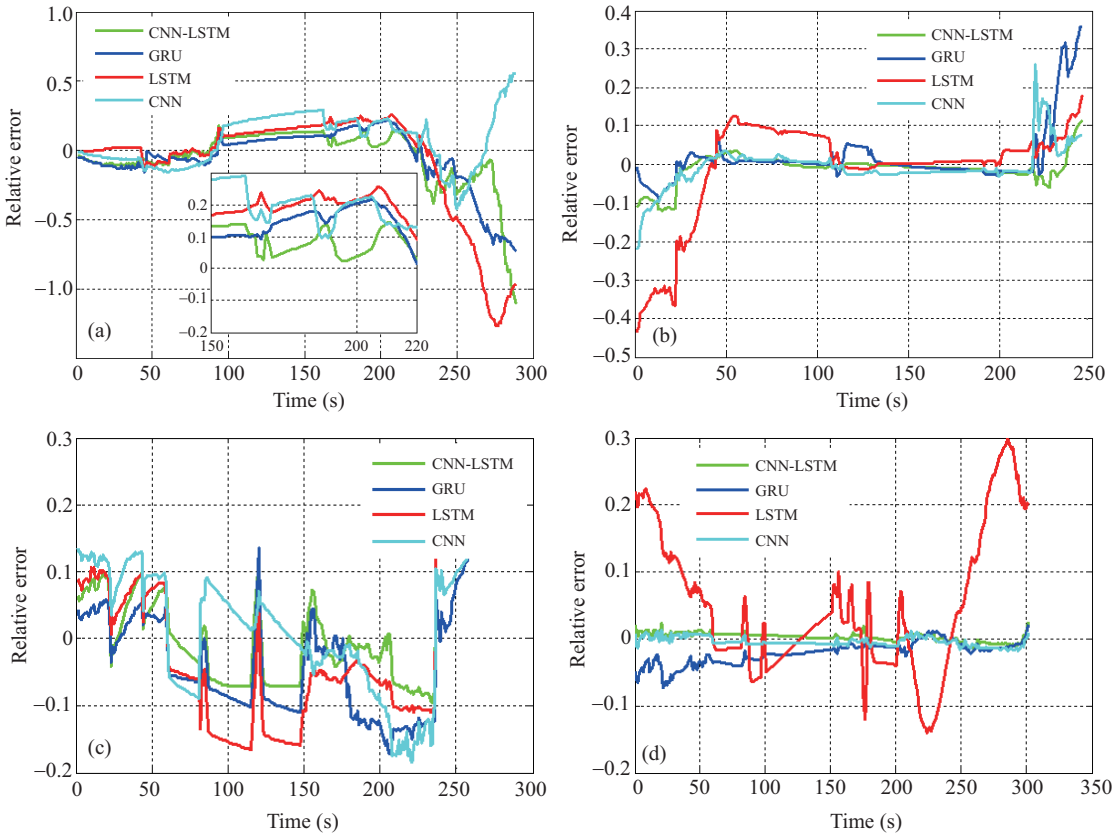

Figure 23 (Color online) Relative errors of predicted trajectories. (a) V1; (b) V2; (c) V3; (d) V4.

Table 8 Measured indicators of prediction results

		CNN-LSTM	LSTM	GRU	CNN
V1	RMSE	147.5097	212.7747	161.5150	210.6115
	MAE	131.4657	175.5494	145.9455	188.2853
	Deviation	0.1372	0.2318	0.1540	0.1723
	Time	6.6897	9.0634	8.7445	6.7868
V2	RMSE	36.8245	103.9171	108.3020	55.6265
	MAE	23.6988	74.6722	46.1783	35.7754
	Deviation	0.0236	0.0801	0.0377	0.0338
	Time	7.1114	10.4041	9.7067	9.8593
V3	RMSE	144.3097	172.3119	153.5478	148.0862
	MAE	101.5878	144.9382	119.0696	116.0209
	Deviation	0.0663	0.1004	0.0772	0.0800
	Time	9.7288	10.1961	9.6121	10.0884
V4	RMSE	10.2299	12.9799	27.7174	196.76836
	MAE	7.9365	9.9685	24.9205	124.1512
	Deviation	0.0060	0.0069	0.0218	0.08541
	Time	9.5135	10.8959	12.1631	11.8550

7 Conclusion

In this paper, a sequential model combining the characteristics of CNN and LSTM is proposed that can accurately predict the trajectories of surrounding vehicles. Based on the historical trajectory data, the trajectories within the next 30 s are predicted by the proposed method. The datasets of NGSIM and Creteil roundabout in Frence are used to evaluate the prediction performance of the CNN-LSTM

model by the prediction error, RMSE, MAE, and deviation. The prediction results are compared with CNN, LSTM, and GRU. The experimental results show that the RMSE, MAE, and the deviation of the proposed method are minimized, which indicates that the method is capable of accurately predicting the trajectories of surrounding vehicles. The proposed method obtains accurate evaluation results in real-time, thus providing an effective method for the safe operation and the correct decision-making of the unmanned systems such as a self-driving vehicle. How to deal with multi-vehicle information to obtain the possible distance between vehicles will be our next research topic.

Acknowledgements This work was supported by National Key R&D Program of China (Grant No. 2018YFB1201500), National Natural Science Foundation of China (Grant Nos. 61873201, 61773313, U1734210), Key Research and Development Program of Shaanxi Province (Grant No. 2018GY-139), Natural Science Foundation of Shaanxi Provincial Department of Education (Grant No. 19JS051), CERNET Innovation Project (Grant No. NGII20161201), and Scientific and Technological Planning Project of Beilin District of Xi'an (Grant No. GX1819).

References

1 He W, Zhang S. Control design for nonlinear flexible wings of a robotic aircraft. IEEE Trans Contr Syst Technol, 2017, 25: 351–357
2 Cao Y, Ma L, Zhang Y. Application of fuzzy predictive control technology in automatic train operation. Cluster Comput, 2019, 22: 14135–14144
3 Yu W K, Zhao C H. Online fault diagnosis for industrial processes with Bayesian network-based probabilistic ensemble learning strategy. IEEE Trans Automat Sci Eng, 2019, 16: 1922–1932
4 Chai Z, Zhao C H. A fine-grained adversarial network method for cross-domain industrial fault diagnosis. IEEE Trans Automat Sci Eng, 2020. doi: 10.1109/TASE.2019.2957232
5 Cao Y, Zhang Y Z, Wen T, et al. Research on dynamic nonlinear input prediction of fault diagnosis based on fractional differential operator equation in high-speed train control system. Chaos, 2019, 29: 013130
6 Xie G, Peng X, Li X, et al. Remaining useful life prediction of lithium-ion battery based on an improved particle filter algorithm. Can J Chem Eng, 2019, 41: 23675
7 Xie G, Li X, Peng X, et al. Estimating the probability density function of remaining useful life for Wiener degradation process with uncertain parameters. Int J Control Autom Syst, 2019, 17: 2734–2745
8 Xie G, Jin Y Z, Hei X H, et al. Adaptive identification of time-varying environmental parameters in train dynamics model. Acta Autom Sin, 2020. doi: 10.16383/j.aas.c190215
9 Qiao S J, Jin K, Han N, et al. Trajectory prediction algorithm based on gaussian mixture model. J Softw, 2015, 26: 1048–1063
10 Qiao S J, Shen D Y, Wang X T, et al. A self-adaptive parameter selection trajectory prediction approach via hidden Markov models. IEEE Trans Intell Transp Syst, 2015, 16: 284–296
11 Wu P J, Yang W T, Yu C, et al. Trajectory prediction method for high precision servo control system (in Chinese). Electric Mach Control, 2014, 18: 1–5
12 Houenou A, Bonnifait P, Cherfaoui V, et al. Vehicle trajectory prediction based on motion model and maneuver recognition. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, 2013. 4363–4369
13 Fei R, Li S S, Hei X H, et al. A motion simulation model for road network based crowdsourced map datum. J Intell Fuzzy Syst, 2020, 38: 391–407
14 Xie G, Sun L L, Wen T, et al. Adaptive transition probability matrix-based parallel IMM algorithm. IEEE Trans Syst Man Cybern Syst, 2019. doi: 10.1109/TSMC.2019.2922305
15 Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436
16 Li D Y, Liu M, Zhao F, et al. Challenges and countermeasures of interaction in autonomous vehicles. Sci China Inf Sci, 2019, 62: 050201
17 Deo N, Trivedi M M. Multi-modal trajectory prediction of surrounding vehicles with maneuver based LSTMs. In: Proceedings of IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018
18 Park S H, Kim B D, Kang C M, et al. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture. In: Proceedings of IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018
19 Altché F, Arnaud D L F. An LSTM network for highway trajectory prediction. In: Proceedings of IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017. 353–359
20 Zhang P, Yang T, Liu Y N, et al. QAR data feature extraction and prediction based on CNN-LSTM (in Chinese). Appl Res Comput, 2019, 36: 2958–2961
21 Kim B D, Kang C M, Lee S H, et al. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In: Proceedings of IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017. 399–404
22 Yang J, Xie G, Yang Y X, et al. Deep model integrated with data correlation analysis for multiple intermittent faults diagnosis. ISA Trans, 2019, 95: 306–319
23 Yang J, Xie G, Yang Y X, et al. An improved deep network for intelligent diagnosis of machinery faults with similar

features. IEEJ Trans Elec Electron Eng, 2019, 14: 1851–1864
24 Cao Y, Sun Y K, Xie G, et al. Fault diagnosis of train plug door based on a hybrid criterion for IMFs selection and fractional wavelet package energy entropy. IEEE Trans Veh Technol, 2019, 68: 7544–7551
25 Zhang S, Dong Y, Ouyang Y, et al. Adaptive neural control for robotic manipulators with output constraints and uncertainties. IEEE Trans Neural Netw Learn Syst, 2018, 29: 5554–5564
26 He W, Dong Y. Adaptive fuzzy neural network control for a constrained robot using impedance learning. IEEE Trans Neural Netw Learn Syst, 2018, 29: 1174–1186
27 Xue Z B, Liu J C, Wu Z X, et al. Development and path planning of a novel unmanned surface vehicle system and its application to exploitation of Qarhan Salt Lake. Sci China Inf Sci, 2019, 62: 084202
28 Thiemann C, Treiber M, Kesting A. Estimating acceleration and lane-changing dynamics based on NGSIM trajectory Data. Transport Res Record J Transport Res Board, 2008, 2088: 90–101
29 Li P, Dargaville R, Cao Y, et al. Storage aided system property enhancing and hybrid robust smoothing for large-scale PV systems. IEEE Trans Smart Grid, 2017, 8: 2871–2879
30 Hinton G E, Osindero S, Teh Y W. A fast learning algorithm for deep belief nets. Neural Computat, 2006, 18: 1527–1554
31 Lecun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition. Proc IEEE, 1998, 86: 2278–2324
32 Chan T A, Hermeking H, Lengauer C, et al. 14-3-3σ is required to prevent mitotic catastrophe after DNA damage. Nature, 1999, 401: 616–620
33 Gers F A, Schmidhuber J, Cummins F. Learning to forget: continual prediction with LSTM. Neural Computat, 2000, 12: 2451–2471
34 Hochreiter S, Schmidhuber J. Long short-term memory. Neural Computat, 1997, 9: 1735–1780
35 Wang Y X, Liu J Q, Misic J, et al. Assessing optimizer impact on DNN model sensitivity to adversarial examples. IEEE Access, 2019, 7: 152766–152776