Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4726763, 18 pages
https://doi.org/10.1155/2020/4726763

Research Article

The Driver Time Memory Car-Following Model Simulating in Apollo Platform with GRU and Real Road Traffic Data

Rong Fei,[1] **Shasha Li**,[1] **Xinhong Hei**,[1] **Qingzheng Xu**,[2] **Fang Liu**,[3] **and Bo Hu**[4]

[1]*School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China*
[2]*College of Information and Communication, National University of Defense Technology, Xi'an 710106, China*
[3]*School of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China*
[4]*Beijing Huadian Youkong Technology Co., Ltd., Beijing 100193, China*

Correspondence should be addressed to Rong Fei; annyfei@xaut.edu.cn and Xinhong Hei; heixinhong@xaut.edu.cn

Received 15 January 2020; Accepted 11 February 2020; Published 17 March 2020

Guest Editor: Chi-Hua Chen

Car following is the most common phenomenon in single-lane traffic. The accuracy of acceleration prediction can be effectively improved by the driver's memory in car-following behaviour. In addition, the Apollo autonomous driving platform launched by Baidu Inc. provides fast test vehicle following vehicle models. Therefore, this paper proposes a car-following model (CFDT) with driver time memory based on real-world traffic data. The CFDT model is firstly constructed by embedded gantry control unit storage capacity (GRU assisted) network. Secondly, the NGSIM dataset will be used to obtain the tracking data of small vehicles with similar driving behaviours from the common real road vehicle driving tracks for data preprocessing according to the response time of drivers. Then, the model is calibrated to obtain the driver's driving memory and the optimal parameters of the model and structure. Finally, the Apollo simulation platform with high-speed automatic driving technology is used for 3D visualization interface verification. Comparative experiments on vehicle tracking characteristics show that the CFDT model is effective and robust, which improves the simulation accuracy. Meanwhile, the model is tested and validated using the Apollo simulation platform to ensure accuracy and utility of the model.

1. Introduction

Car-following (CF) behaviour is the most basic micro driving behaviour, referring to the interaction between two adjacent vehicles in a vehicle fleet driving on a single-lane road that does not allow passing [1]. The concept of CF originated in the early 1950s. Over the past six decades, CF models have been extensively and systematically studied, and fruitful achievements have been made [2]. Since the 1990s, research in related fields has gradually emerged in China. Researchers from various fields have attempted to interpret the observed microscopic phenomena from different perspectives [3].

There are currently many types of CF models that can be divided into two categories based on their origins: theory-driven and data-driven CF models [4].

In the development of the theory-driven models, the stimulus-response models are the most classic CF models, of which the General Motors (GM) model [5] is the most important. The GM model has been gradually developed and used since the late 1950s; it is the basis of many of the subsequent stimulus-response models. The GM model clearly reflects the characteristics of CF behaviour; it has a simple form and a clear physical meaning based on its originality. However, this model is prone to change with changes in traffic operational conditions and hence lacks universality.

With the increasing popularity of artificial intelligence, data-driven models have gradually become a focus of research of CF models. In 1988, Rumelhart proposed the back-propagation neural network (BPNN) [6], which is a multilayer feedforward neural network (FNN) that uses the error back-propagation algorithm to adjust weights. It is the most widely used NN model. Chen et al. proposed a deep learning method for learning potentially complex and irregular

probability distributions, which can accurately estimate the values of CDF and PDF [7].

With the wide application of NNs in the field of traffic simulation, in 1998, Kehtarnavaz [8] applied the BPNN model to CF behaviour modelling for the first time, using the speed of the following car and the distance between the two cars as the model inputs and the relative speed of the two cars as the model output, thus verifying the validity of the model. Zhang et al. [9] established a closed-loop driving following model based on BP neural network and verified the adaptability of the model to different driving groups through experiments.

Support vector regression (SVR) [10] is a machine learning algorithm that converts the original problem into a convex quadratic programming problem and solves it using the optimality theory to obtain the global optimal solution. Wei and Liu [11] proposed a vehicle following model based on support vector regression and studied the asymmetric characteristics of vehicle following behaviour and its influence on the evolution of traffic flow. Parham et al. [12] propose car-following modelling using an efficient support vector regression method and prove that it has appropriate validity after inputting the driver's instantaneous reaction time. However, studies based on such models are in their infancy.

CF behaviour is a continuous behaviour, so a driver can make a corresponding decision based on the memory of the previous time period [13–15]. However, a large number of existing models do not fully consider the driver's memory effect and only consider the instantaneous interaction between the following and leading cars. To process the CF time series data to use their historical information, the model must have a memory capability; this is lacking in both the BPNN- and SVR-based models.

Recurrent neural network (RNN) [16] is a class of neural network with memory capability. Yang [17] proposed a car-following model based on recurrent neural network (RNN) to effectively describe the state changes of vehicles while driving and road traffic congestion.

When an input sequence is long, the gradient explosion and vanishing problems, also known as the long-term dependency problem, will occur. To solve this problem, various modifications have been made on RNNs; the most effective method is to introduce various gating mechanisms such as the long short-term memory (LSTM) [18, 19] and the gated recurrent unit (GRU) [20] networks.

Based on the previously described studies, Wang et al. [21] proposed the use of GRU to model CF behaviour and embed the driver's memory effect in the model, which used the speed of the following car, the relative speed of the two cars, and the distance between the two cars observed in the last several time intervals as inputs and the estimated speed of the following car at the next time point as the output. The test results showed that the proposed model has higher simulation accuracy than the existing CF models and provides a new concept for the study of traffic flow theory and simulation. However, the driver's decision-making and reflection time are not considered in the judgment process.

However, influenced by multiple sources of information, a driver's decision-making and judgment process exhibits a complex nonlinear modality during driving, and the driver's psychological decision cannot be described with a simple mathematical expression. Fuzzy theories and artificial neural networks show certain operational advantages in handling complex nonlinear issues and also exhibit a good learning capacity under big data samples. Therefore, the fuzzy theory and artificial neural network are often used for simulating driving behaviours under different environments. However, the current schemes utilizing fuzzy theories and artificial neural networks only focus on the velocities and accelerations of the leading car and the following car, as well as the spacing there between, without considering driving environments [22]. In addition, how to obtain real-time traffic information (such as average speed, travel time, traffic flow, and traffic conditions) is also an important issue for unmanned driving. Many scholars have also done a lot of research on real-time traffic information. Chen proposed a cell probe (CP)-based method to analyse cellular network signals with an estimated accuracy of 97.63%, which is easier to obtain than traditional methods [23].

In April 2017, Baidu released its open platform Apollo for autonomous driving; after iterations of multiple versions, the platform has been enabled for localization, sensing, decision, and simulation. Apollo may help its partners in the automotive and autonomous driving industries to quickly develop a set of their own autonomous driving systems in consideration of vehicles and hardware systems. In the Apollo simulation environment, environment information including traffic signs, index lines, and the relationships with surrounding vehicles may be inputted into Dreamview via corresponding interfaces to thereby construct a driving environment. Besides, the Apollo platform is further enabled for validating the car-following model and optimizing the relevant algorithm through a 3D visual interface.

In this study, based on the previously described studies and combined with actual road conditions, we designed a CFDT model based on the data-driven model in combination with the improved RNN. In our model, the speed of the leading vehicle in the previous time interval, the speed of the following vehicle, and the distance between the two vehicles are used as inputs to predict the acceleration of the following vehicle at the next time point. Furthermore, the established model was calibrated using the CF data to determine the optimal parameters and optimal structure of the model, which were then verified through simulation. Finally, the proposed model was compared with the BPNN- and SVR-based models. It was confirmed that, compared with the traditional CF models, the RNN network-based CF model has high robustness and improved simulation accuracy, providing a methodological basis for studying the car-following behaviour.

The remainder of the paper is divided into four sections. Section 2 introduces conventional car-following models and the neural network based car-following model. Section 3 models CF behaviour mainly using the RNN network. Section 4 processes the data and briefly analyses the driver's response time. Section 5 trains the proposed model to obtain the optimal parameters, compares it with the other two

existing models, and verifies that the proposed model can achieve better simulation results. Section 6 conducts an empirical study of the three types of CF models based on the data and describes in detail the models' verification experiments. Section 7 uses the Apollo simulation platform to verify the model and ensure the accuracy and practicability of the model. Section 8 introduces the summary and prospects.

2. Background

2.1. Traditional Car-Following Models. The stimulus-response framework is the most traditional modelling idea of the car-following behaviour, which embodies many essential characteristics of the car-following behaviour, while the GM model is the most important stimulus-response type of the traditional vehicle following model. The GM model assumes that a vehicle does not show passing or lane-changing behaviour when following. The driving dynamics theory is used to derive the basic equation:

$$a_{n+1}(t+T) = \lambda v_{n+1}^{m}(t+T)\frac{\Delta v(t)}{[\Delta x(t)]^{l}}. \tag{1}$$

In equation (1), $a_{n+1}(t+T)$ is the instantaneous acceleration of a following car at time $(t+T)$; $v_{n+1}(t+T)$ is the instantaneous speed of a following car at time $(t+T)$; $\Delta v(t)$ is the relative speed of the two cars at time t; $\Delta x(t)$ is the distance between the two cars at time t; T is the response time; λ is the sensitivity parameter to be calibrated; and m and l are additional parameters to be calibrated. Numerous studies have been focused on the parameter calibration and extension of the GM model.

The GM model clearly reflects the characteristics of CF behaviour; it has a simple form and a clear physical meaning based on its originality. However, this model is prone to change with changes in traffic operational conditions and hence lacks universality.

2.2. Basics of NN. NN is a highly nonlinear model with a neuron as its basic unit. When a neuron receives a set of input signals, it generates an output signal. A typical structure is shown in Figure 1.

where $x = [x_1\ x_2\ \cdots\ x_d]^T \in \mathbb{R}^d$ is the input, $w = [w_1\ w_2\ \cdots\ w_d]^T \in \mathbb{R}^d$ is the weight, and b is a bias unit. The sum of the weighted inputs is described using the net input z; then, we have equation (2), as follows:

$$z = \mathbf{w}^T\mathbf{x} + b, \tag{2}$$

where $f(x)$ is the activation function. Then, the output can be expressed as follows in equation (3):

$$a = f(z). \tag{3}$$

The commonly used activation functions, which are nonlinear, are as follows:

Sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{4}$$

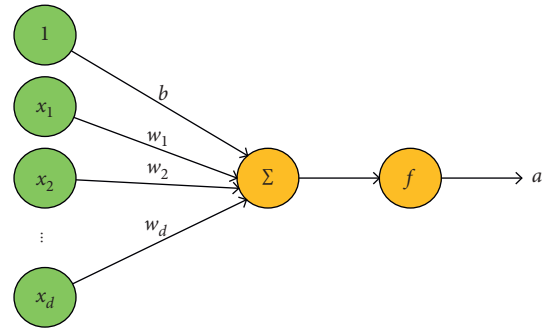


Figure 1: Structure of a neuron.

Tanh function:

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}. \tag{5}$$

ReLU function:

$$\mathrm{ReLU}(x) = \max(0,\ x). \tag{6}$$

Leaky ReLU function:

$$\mathrm{LeakyReLU}(x) = \max(\alpha x,\ x). \tag{7}$$

In practical application, the appropriate activation function can be selected according to the actual situation.

Historically, various NN structures have been proposed; those most commonly used include the feedforward NN, the feedback NN, and the graph network. In this study, the feedback NN was adopted. The neurons in a feedback NN can receive not only the signals of other neurons but also their own. Compared with those in a feedforward NN, the neurons in a feedback NN have a memory function and have different states at different times. The basic structure of a feedback NN is shown in Figure 2.

In a feedback NN, signals can propagate in one or both directions. This type of network includes RNN and the Boltzmann machine.

2.3. Gated RNN. To solve the long-term dependence problem of RNNs in the long sequence of training, a gating mechanism is introduced to selectively add new information while selectively forgetting the retained information. Such networks are collectively referred to as gated RNNs; the most popular include the LSTM and GRU networks.

2.3.1. LSTM Network. An LSTM network adds the new internal states $\mathbf{c}_t$ and introduces three "gates"—the forgetting gate ($\mathbf{f}_t$), input gate ($\mathbf{i}_t$), and output gate ($\mathbf{o}_t$). The value of a "gate" is within $(0,\ 1)$; it is used to control the amount of information passed.

Specifically, the forgetting gate $\mathbf{f}_t$ controls the amount of information to be forgotten by the internal state of the last time point ($\mathbf{c}_{t-1}$); the input gate $\mathbf{i}_t$ controls the amount of information to be retained by the candidate state $\tilde{c}_t$ of the current time point; and the output gate $\mathbf{o}_t$ controls the

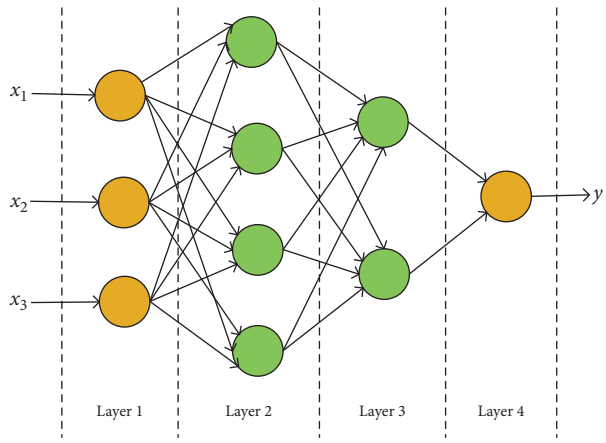


Figure 2: Feedback NN.

amount of information to be output to the external state $\mathbf{h}_t$ by the internal state $\mathbf{c}_t$ of the current time point.

Figure 3 shows the internal structure of the LSTM unit, where $\times$ represents the multiplication of vector elements and $+$ represents the addition of vector elements; $\sigma(x)$ is the sigmoid activation function. Thus, the three gates of $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$ can be calculated with equations (8)–(10), respectively. $W_*(* = f, i, o)$ and $U_*(* = f, i, o)$ stand for the weight matrix from the cell to gate, and $b_*(* = f, i, o)$ denotes the vector of each gate.

$$f_t = \sigma\left(U_f h_{t-1} + W_f x_t + b_f\right), \tag{8}$$

$$i_t = \sigma\left(U_i h_{t-1} + W_i x_t + b_i\right), \tag{9}$$

$$o_t = \sigma\left(U_o h_{t-1} + W_o x_t + b_o\right). \tag{10}$$

The methods for status updating are described in equations (11)–(13) for $\mathbf{f}_t$, $\mathbf{i}_t$, and $\mathbf{o}_t$, respectively:

$$\tilde{c}_t = \tanh\left(U_c h_{t-1} + W_c x_t + b_c\right), \tag{11}$$

$$c_t = f_t * c_{t-1} + i_t * \tilde{c}_t, \tag{12}$$

$$h_t = o_t * \tanh\left(c_t\right). \tag{13}$$

In equations (12) and (13), *represents the multiplication of vector elements.

In an LSTM network, the internal state of the unit $\mathbf{c}_t$ can retain certain key information for a significant amount of time.

2.3.2. GRU Network. The internal concept of a GRU network is similar to that of an LSTM network, and a GRU network can achieve a comparable effect. However, a GRU network has fewer parameters, lower training difficulty, and higher practicality.

Because the input and forgetting gates in the LSTM unit are complementary, in a GRU unit, they are combined into one gate, i.e., the update gate $\mathbf{z}_t$, while the output gate of the LSTM unit is deleted and a reset gate $\mathbf{r}_t$ is added without introducing a new internal state. Its structure is shown in Figure 4.

Where the update gate controls the amount of information to be retained by the state of the current time point $(\mathbf{h}_t)$ from the state of the last time point $(\mathbf{h}_{t-1})$, as well as the amount of new information received from the candidate state $(\tilde{h}_t)$, the reset gate controls the amount of information to be retained from the state of the last time point $(\mathbf{h}_{t-1})$ by the candidate state of the current time point $(\tilde{h}_t)$. The methods for the calculation of the two gates are shown in equations (14) and (15). $W_*(* = z, r, h)$ and $U_*(* = z, r, h)$ stand for the weight matrix from the cell to gate, and $b_*(* = z, r, h)$ denotes the vector of each gate.

$$z_t = \sigma\left(U_z h_{t-1} + W_z x_t + b_z\right), \tag{14}$$

$$r_t = \sigma\left(U_r h_{t-1} + W_r x_t + b_r\right). \tag{15}$$

The methods for updating the states are shown in equations (16) and (17):

$$\tilde{h}_t = \tanh\left(U_h\left(r_t * h_{t-1}\right) + W_h x_t + b_h\right), \tag{16}$$

$$h_t = z_t * h_{t-1} + \left(1 - z_t\right) * \tilde{h}_t. \tag{17}$$

In the case where $\mathbf{z}_t = 0$ and $\mathbf{r}_t = 1$, the GRU network degenerates into a simple RNN.

3. Approach to the Proposed Model

Herein, we adopted the RNN network to model CF behaviour. The proper choice of model inputs and outputs can improve the simulation accuracy of the model. Based on the GM model, we use the speed of the leading car at time t $(v_n(t))$, the speed of the following car at time t $(v_{n+1}(t))$, and the distance between the two cars $(\Delta x(t))$ as inputs and the acceleration of the following car at time $t + T$ $(a_{n+1}(t + T))$ as the output. Then, we have equation (18):

$$a_{n+1}(t+T) = f\begin{pmatrix} v_n(t), v_{n+1}(t), \Delta x(t), v_n(t-T), v_{n+1}(t-T), \Delta x(t-T), \dots \\ v_n(t-(N-1)T),\ v_{n+1}(t-(N-1)T), \Delta x(t-(N-1)T) \end{pmatrix}, \tag{18}$$

where N is the length of the time interval of "memory." Figure 5 is the specific structure diagram of the model proposed in this paper.

To eliminate the influence of dimension on the simulation accuracy and convergence rate of the model, the paper normalizes the vehicle following data (leading vehicle speed, following vehicle speed, and following vehicle acceleration). In different traffic environments, the behaviour of following a car is easily affected by the propagation of the slight disturbance of the speed of the leading car, the habit of acceleration and deceleration, the driver's cognition of the environment, and the driver's cognition of driving

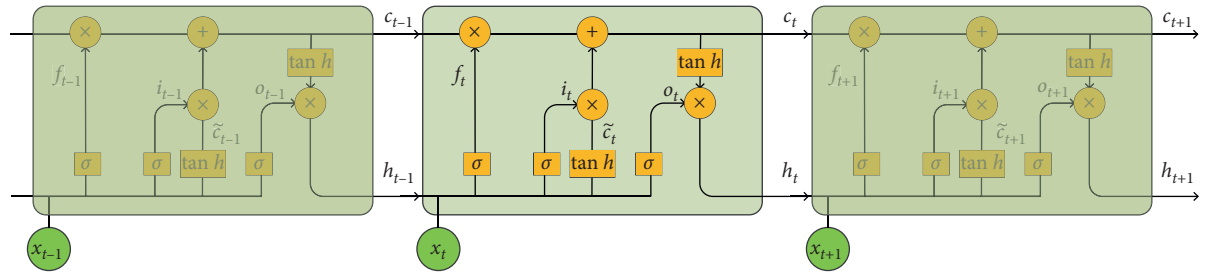


Figure 3: LSTM unit.

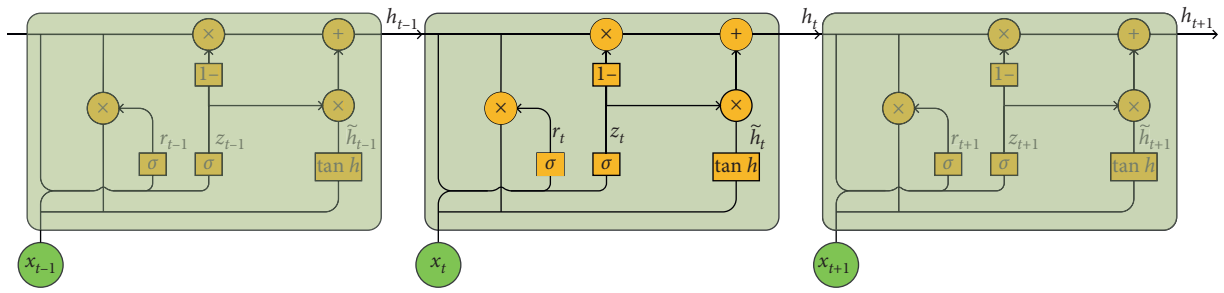


Figure 4: GRU unit.

behaviour. The distribution of the car-following data cannot judge whether it is close to a Gaussian distribution. Therefore, the MinMaxScaler of the simplest method to eliminate the influence of dimensionality and data value range is selected in this paper to preserve the relationship existing in the original data. The formula for MinMaxScaler is as follows:

$$X_{\text{scaled}} = \frac{X - X.\min}{X.\max - X.\min} \cdot (\max - \min) + \min, \tag{19}$$

where max and min are the maximum and minimum values of a given zoom range, respectively; i.e., the original data are scaled to the range of [max, min]. Specifically, the original data are scaled to the range of [0, 1]. Then, we have the following:

$$X_{\text{scaled}} = \frac{X - X.\min}{X.\max - X.\min}. \tag{20}$$

The CF data are constructed to the appropriate input and output shapes to comply with the input and output structures of the GRU model. Because the length of the "memory" interval (N) is closely related to the constructed data, it also directly affects the prediction accuracy of the model and so it was tested in detail in this study. Then, the dataset was randomly divided into training and test sets. Next, ReLU was selected as the activation function of the output layer to establish the GRU model. Because the number of hidden layers and the number of neurons in each hidden layer can have numerous combinations, it is necessary to separately test each of the models with different structures to obtain the optimal structure of the model.

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2. \tag{21}$$

The mean square error (MSE) (21) was used to construct the loss function and the Adam optimizer [24–26], due to its excellent performance in most cases, was adopted.

In the input layer, the time step is set to N, and each step contains three input variables $(v_n, v_{n+1}, \Delta x)$. Since it is necessary to predict the acceleration of the following vehicle with a continuous time of m, the input value is constructed into a matrix $X \in \mathbb{R}^{m \times N \times 3}$ through the normalization of formula (19). Finally, the predicted acceleration of d for a continuous period of time of the following car was constructed into output $y \in \mathbb{R}^{m \times 1}$ through the GRU gate unit of multiple hidden layers. The pseudocode for the construction of the RNN-based CF model is shown in Algorithm 1.

4. Data Preparation

4.1. Processing of the Following Vehicle Data. The Next Generation Simulation (NGSIM) program [27] was initiated by the U.S. Federal Highway Administration (FHWA). Through the established synchronous digital camera network, detailed vehicle trajectory data were acquired at a time interval of 0.1 seconds from the US-101 Freeway and the southbound direction of Lankershim Boulevard in Los Angeles, California; Interstate I-80 in Emeryville, California; and the eastbound direction of Peachtree Street in Atlanta, Georgia.

In this study, the detailed trajectory data of the eastbound vehicles on Interstate I-80 in Emeryville, California, were used. The data were acquired at 10 frames per second by seven cameras mounted on the 30-story Pacific Park Plaza Building located at Christie Avenue. The study road section is 503 m long and has six lanes. Lane 1 is a high occupancy vehicle (HOV) lane and Lane 6 is a collector-distributor lane, as shown in Figure 6.

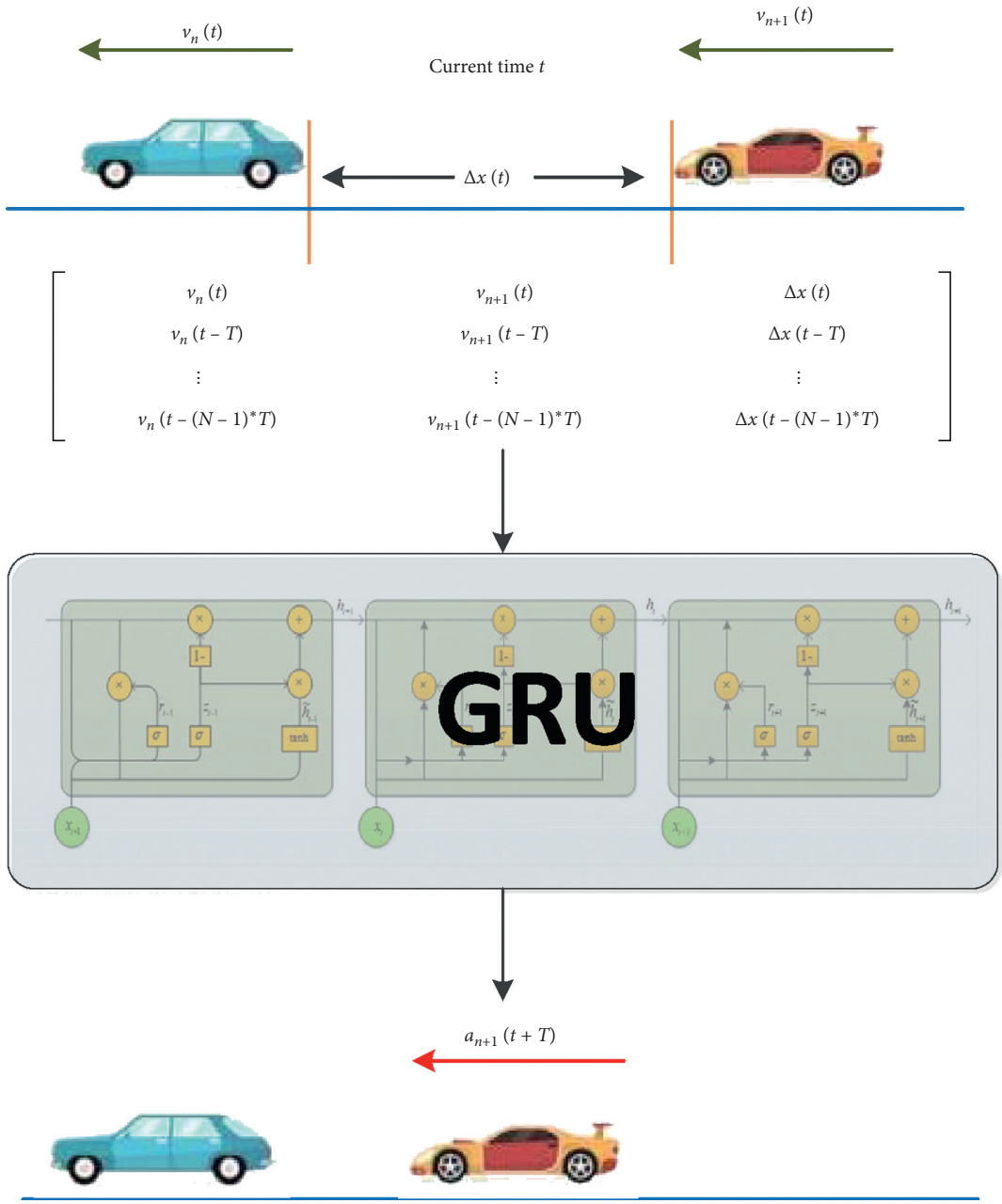


Figure 5: Schematic structure of the GRU-based CF model.

The I-80 dataset includes vehicle trajectory data for three time periods. The road conditions are shown in Table 1.

In this study, we mainly analysed the CF behaviour of cars and their microscopic characteristics. The CF behaviour is closely related to the road conditions. To ensure the universality of the CF behaviour, we examined the vehicle trajectory data between 4:00 pm and 4:15 pm, which contain 1,028,575 trajectory records. Each record contains 18 fields, as fully described in Table 2.

4.1.1. Data Preprocessing. First, data of the following vehicles were found and filtered according to the following rules:

(1) To avoid the potential difference in CF behaviour between different types of vehicles, we only focused on the CF behaviour of small cars (i.e., v_Class = 2)

(2) Cars from the HOV lane (i.e., Lane_ID = 1) and the collector-distributor lane (i.e., Lane_ID = 6) were excluded to ensure that the vehicles under study are associated with similar driving behaviour, i.e., to ensure the consistency of driving behaviour

(3) The single-lane data of driving vehicles were adopted to avoid the influence of lane-changing behaviour on CF behaviour

(4) Only the data for cars with a following time greater than 45 s (i.e., 450 records) were retained to ensure

GRU-based car-following model
Input: $v_n(t), v_{n+1}(t), \Delta x(t), v_n(t-T), v_{n+1}(t-T), \Delta x(t-T), \ldots,$
$v_n(t-(N-1)T),\ v_{n+1}(t-(N-1)T), \Delta x(t-(N-1)T)$
Output: $a_{n+1}(t+T)$

(1) Data normalization using MinMaxScaler.
(2) The input is constructed as $X \in \mathbb{R}^{m\times N\times 3}$, The output is constructed as $y \in \mathbb{R}^{m\times 1}$
(3) The constructed data is divided into training set and testing set.
(4) Modelling Sequential(), add GRU Layer and Dense Layer
(5) Compiler model, loss function = "MSE," optimizer = "Adam"
(6) **while** model convergence **do**
(7) Training model
(8) **end while**
(9) Test model

Algorithm 1: Pseudocode for the construction of the GRU-based CF model.

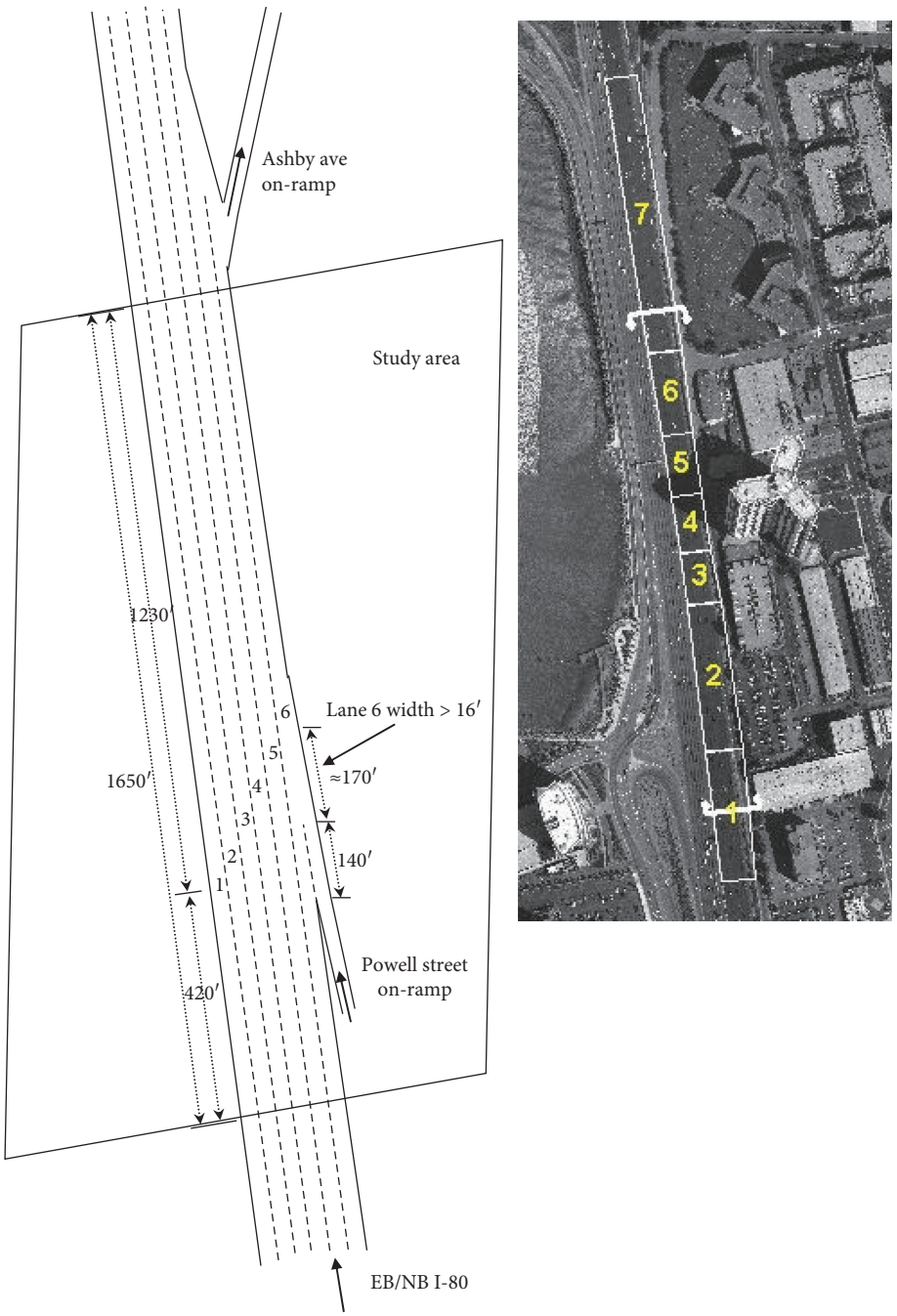


Figure 6: Outline of the study area.

Table 1: Road conditions.

Acquisition period	Road condition
4:00 p.m.~4:15 p.m.	Transition from noncongestion to congestion
5:00 p.m.~5:15 p.m.	Congestion
5:15 p.m.~5:30 p.m.	Congestion

Table 2: Field descriptions.

Field	Description
Vehicle_ID	Vehicle identification number (increase by entry time)
Frame_ID	Frame identification number (increase by start time)
Total_Frames	The total number of frames in the dataset of vehicles
Global_Time	Time elapsed since 1970.1.1 (unit: ms)
Local_X	X-value in study regional coordinate system
Local_Y	Y-value in study regional coordinate system
Global_X	X-value in standard geographic coordinate system
Global_Y	Y-value in standard geographic coordinate system
v_length	Vehicle length (unit: feet)
v_Width	Vehicle width (unit: feet)
v_Class	Vehicle type (1: motorcycle, 2: compact vehicle, 3: large vehicle)
v_Vel	Instantaneous vehicle speed (unit: feet/s)
v_Acc	Instantaneous vehicle acceleration (unit: feet/s^2)
Lane_ID	Number of the current lane of the vehicle
Preceding	Vehicle identification number of leading vehicle
Following	Vehicle identification number of following vehicle
Space_Headway	Head spacing (unit: feet)
Time_Headway	Headway time distance (unit: s)

The acquisition interval is 0.1 s; 1 ft = 0.3048 m.

the integrity of the CF process and to ensure an adequate number of samples for model training

The pseudocode for the data processing is shown in Algorithm 2.

Table 3 is a part of vehicle tracking data obtained through data preprocessing, whose lead vehicle ID is 66 and the following vehicle ID is 74.

4.1.2. Driver Reaction Time. The driver reaction time refers to the time between the driver perceiving a change in the surrounding environment and responding [28, 29]; it is also an important parameter in the CF model.

According to the restrictions of the CF, as the driving state of the leading car changes, the following car changes accordingly. However, changes in the driving states of the two are asynchronous. This is because the driver of the following car must have a reaction process to respond to a change by the leading vehicle. This reaction process includes four parts: perception, judgment, reaction initiation, and reaction execution; the required time is referred to as the reaction time. Assuming that the reaction time is T when the leading car makes a change at time t, the following car can only make the corresponding change at time $(t + T)$.

The driver reaction time has been extensively investigated by many researchers. Kim et al. [30] analysed the braking reaction time of young and old drivers. Jin [31] studied the driver reaction time using least squares analysis via SPSS 13.0 and generated a reaction time distribution map (Figure 7). The calculation showed that the weighted average of driver reaction time is 1.077 s.

Lu et al. [32] obtained statistical information on driver reaction time through 63 samples, as shown in Table 4.

Based on the above discussion, we set the reaction time to 1 s in this study. Because the acquisition time interval of the adjacent two records of the CF data is 0.1 s, we excerpted one record for every 1 s (i.e., 10 records); the new dataset was saved for later use.

4.2. Evaluation Index. In this paper, MSE is used as the evaluation index of CFDT model. MSE is a risk function, related to the expected value of the squared error loss or quadratic loss. MSE is arguably the most important criterion used to evaluate the performance of a predictor or an estimator. It measures how close a fitted line is to data points. For every x data point, take the distance vertically from the point to the corresponding y value on the curve fit (the error), and square the value. The lower estimation value of MSE represents the lower error [33, 34]. The specific calculation formula of MSE has been shown in equation (21).

5. Training and Test

In this section, we used Keras, a Python-based deep learning library, to construct and train the CFDT model, and used TensorFlow as a back-end tool. The hardware environment of our experiment is as follows: processor Intel Xeon 2.10 GHz E5-2683 v4, memory 64 GB 2400 MHz, operating system Windows Server 2012 R2 standard, IDE: PyCharm.

According to this model training process, the optimal length of a "memory" time interval and the optimal structure

```
Processing of vehicle trajectory data.
Input: Vehicle Trajectory Data, the number of trajectory records n
Output: Car-following Data
(1)  for pos_Lv ← 0 to n
(2)    if Following[pos_Lv] ≠ ∅ then
(3)      time ← 0
(4)      for pos_Fv ← 0 to n
(5)        if Vehicle_ID [pos_Fv] == Following[pos_Lv] and Fram_ID[pos_Lv] == Fram_ID[pos_Fv] then
(6)          time ← time + 1
(7)          Data ← Data + [v_Vel[pos_Lv], v_Vel[pos_Fv], v_Acc[pos_Fv], Space_Headway − v_length[pos_Lv]]
(8)        end if
(9)      end for
(10)     if time > 450 then
(11)         The Data that holds the data is stored in CSV format
(12)     end if
(13)   end for
```

Algorithm 2: The pseudocode for the CF data processing.

Table 3: A group of following team part of the data.

Lead vehicle speed	Following vehicle speed	Following vehicle acceleration	Space
22.07	18.5	0	27.72
22.07	18.5	0	28.07
22.07	18.5	0	28.41
22.22	18.5	0	28.77
22.26	18.5	0	29.17
21.94	18.5	0	29.57
21.17	18.57	1.07	29.93
20.05	18.71	1.84	30.14
19.01	18.78	0.39	30.19
18.33	18.7	−2.08	30.12
18.01	18.46	−3.41	30.03
17.96	18.19	−2.24	29.99
17.99	18.04	−0.59	30
18	18	−0.11	30
18	17.99	0.05	30
18	18	0.16	30
18	18.02	0.26	30
18	18	−0.48	29.99

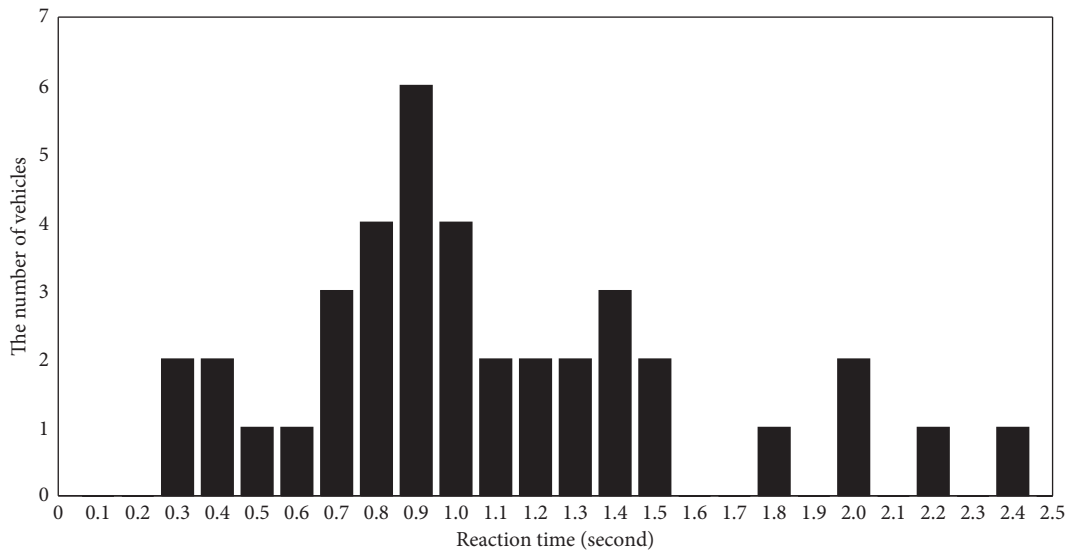


Figure 7: Reaction time distribution map.

Table 4: Reaction time statistics.

	Response time $\tau(s)$
Mean value	0.7016
Standard error	0.05236
Median	0.50
Standard deviation	0.41562
Variance	0.173
Range of change	1.70
Minimum	0.30
Maximum	2.00

of the model must be determined so the model has its best performance. The results of previous studies have shown that the optimal length of a "memory" time interval is not related to the structure of the model, and the NGSIM dataset does not need more than three hidden layers for model performance [21]. On this basis, we performed the following experiment.

First, we fixed the structure of the model. To reduce the training time, a simple structure with a single hidden layer and 20 neurons was adopted to separately perform the experiment with different N values. The results are shown in Figure 7.

Figure 8 shows that when the length of the "memory" time interval was 10, i.e., when the driver of the following car only considers the historical information within the time period of the last 10T, the model had the best performance.

On this basis, assuming $N = 10$, we separately trained and tested the models with the nine structures listed in Table 5.

The results showed that, for models with one hidden layer (Structures 1 through 3), the performance value of the model with Structure 3 is the minimum; for models with two hidden layers (Structures 4 through 6), the performance value of the model with Structure 5 is the minimum; and for models with three hidden layers (Structures 7 through 9), the performance value of the model with Structure 8 is the minimum. In terms of model performance, the performance values of the models are ranked in an ascending order: the model with Structure 3< the model with Structure 5< the model with Structure 8.

Next, the CF data, in which the vehicle ID of the leading car is 66 and that of the following car is 74, were trained using the model with Structure 3, the model with Structure 5, and the model with Structure 8. The simulation results were visualized and are shown in Figures 9–11.

In each of the figures, the first subplot shows the actual data, the second subplot shows the simulation data cluster belonging to 100 different training models, and the third subplot shows the simulation data cluster with error bars, i.e., mean ± std.

In summary, for the RNN-based CF model, the model with a "memory" time interval length of $N = 10$ and three hidden layers that contain 30, 10, and 10 neurons had the highest prediction accuracy and generated satisfactory simulation results for a road section that had continuous acceleration and deceleration behaviours.

6. Comparison with Other CF Models

To test the simulation accuracy of the RNN-based CF model, two other models, i.e., BPNN and SVR, were selected from the

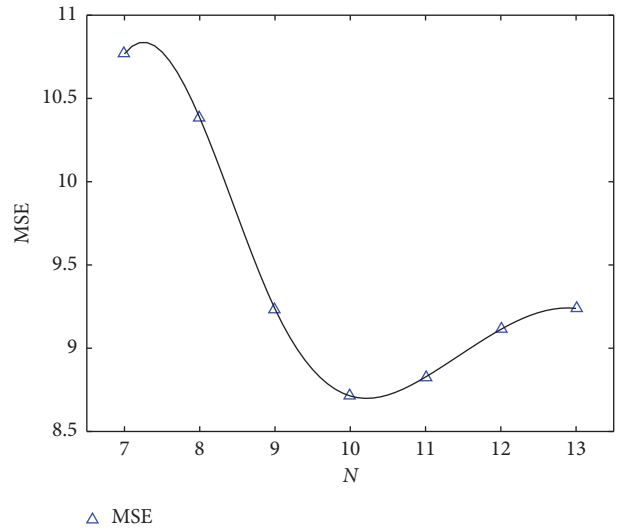


Figure 8: Influence of the length of the "memory" time interval.

Table 5: Structures of the model.

Structure number	Hidden layer		
	1	2	3
1	10	0	0
2	30	0	0
3	**50**	**0**	**0**
4	10	5	0
5	**30**	**10**	**0**
6	50	20	0
7	10	5	5
8	**30**	**10**	**10**
9	50	20	20

data-driven models to conduct a comparative experiment. Similarly, to ensure the fairness of the comparison, the speed of the leading car at time t $(v_n(t))$, the speed of the following car at time t $(v_{n+1}(t))$, and the distance between the two cars at time $(\Delta x(t))$ were used as the inputs to the model, and the acceleration of the following car at time $(a_{n+1}(t+T))$ was used as the output of the model. MSE continued to be adopted as the criterion for model evaluation.

6.1. BPNN-Based CF Model. First, a model as shown in Figure 12 was constructed based on the BPNN. The model can have various structures in terms of the number of hidden layers and the number of neurons in each layer. According to Kolmogorov's theorem [35], a back-propagation neural network with three layers is sufficient to complete any mapping from n dimensions to m dimensions. Therefore, we choose the BPNN network with two hidden layers as the structure of the BPNN-based CF model. After multiple tests, the optimal structure, which included two hidden layers containing 20 and 10 neurons, was selected. The Tan h function was used as the activation function of the neurons.

The model was constructed and trained using Keras. Methods such as the holdout method were used to randomly create the training and test sets.

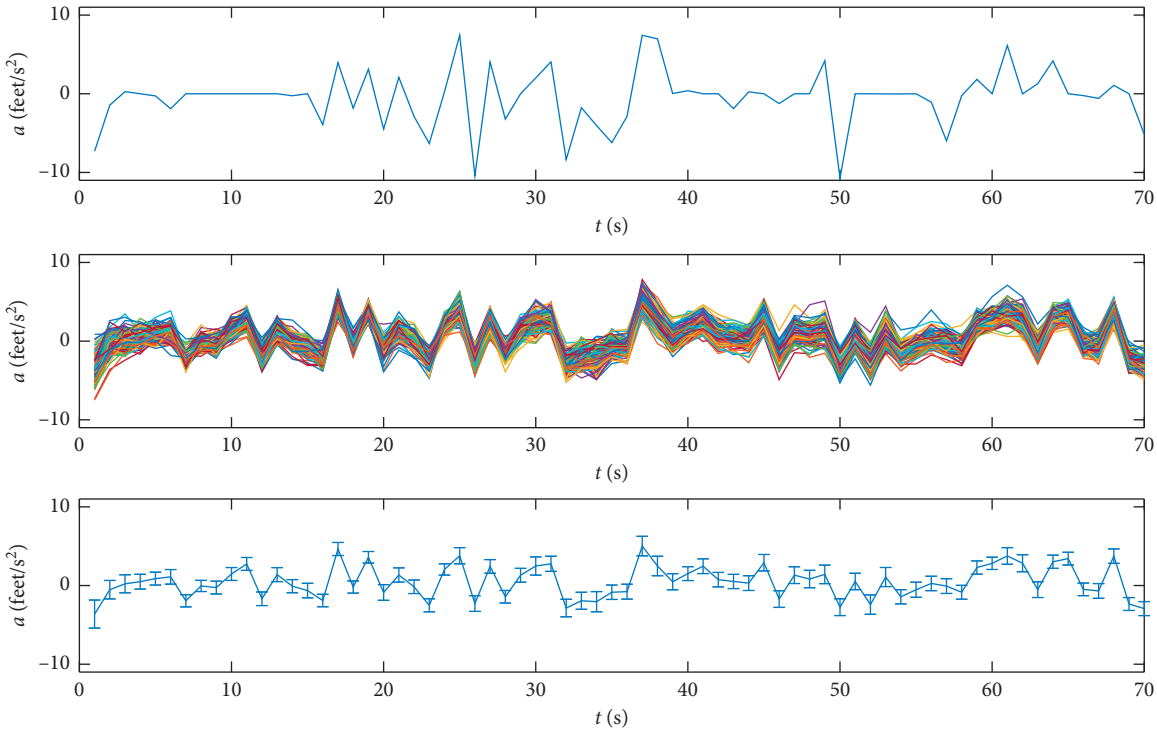


Figure 9: The model with Structure 3.

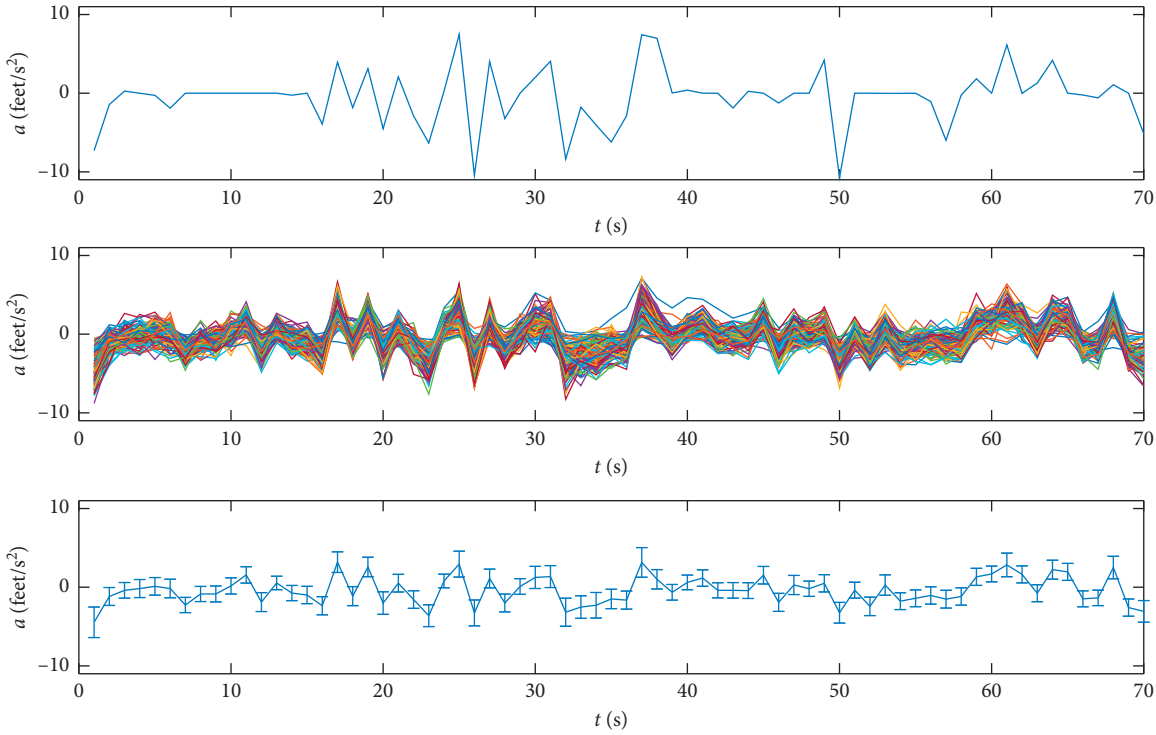


Figure 10: The model with Structure 5.

The CF data with the leading car ID of 2771 and the following car ID of 2812 were randomly selected; their simulation results are shown in Figure 13.

Figure 12 shows that because this dataset had a significant amount of "noise" (i.e., the following car was constantly in an accelerating or decelerating state), the resulting simulation results of the model have a low accuracy (i.e., there is a certain error when compared to the real data). Nevertheless, the results reflect the variation trend in the acceleration of the following car. Moreover, the model has a

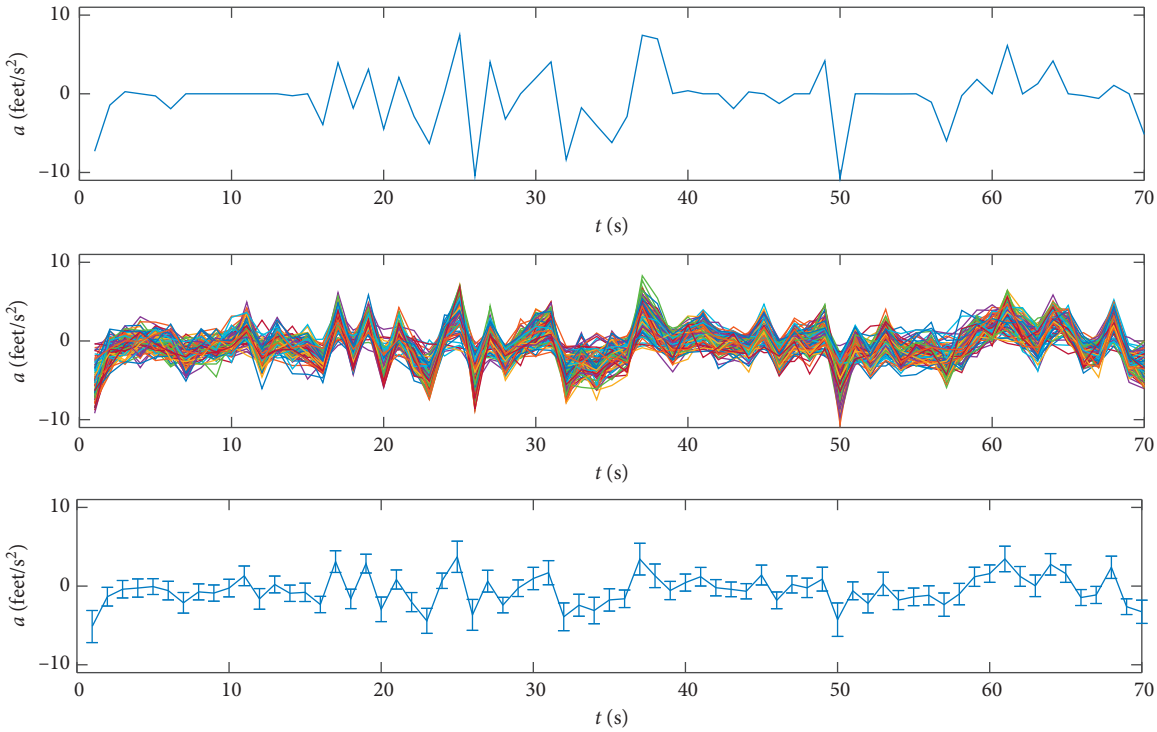


Figure 11: The model with Structure 8.

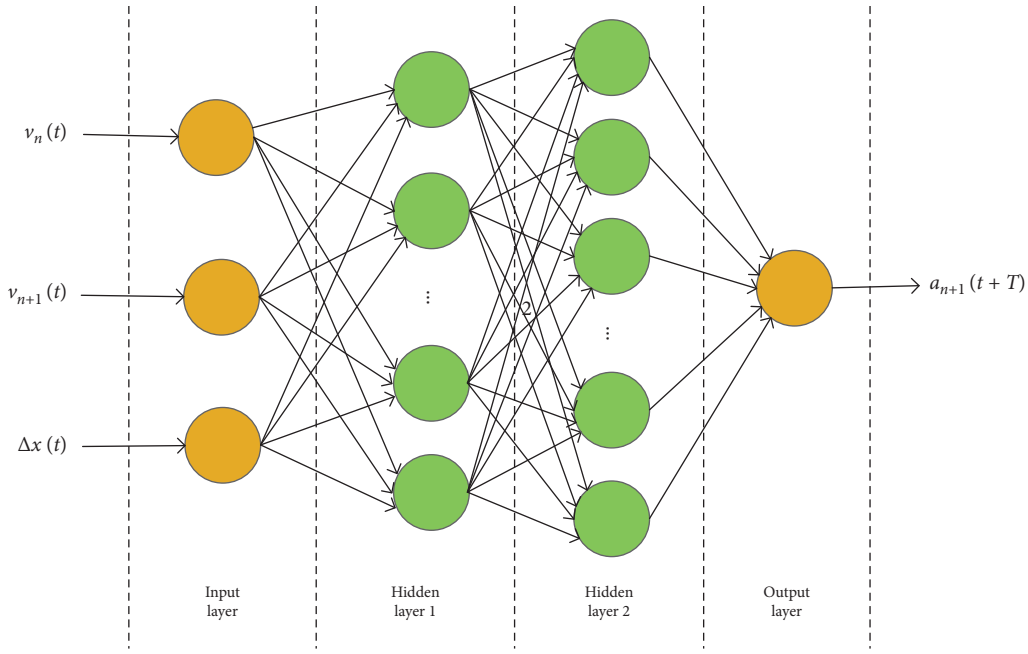


Figure 12: The BPNN-based CF model.

relatively simple structure and fast convergence rate. The above experiment demonstrates the validity of the BPNN-based CF model.

6.2. SVR-Based CF Model. we constructed an SVR-based CF model, as shown in Figure 14.

where the Gaussian kernel function was selected as the kernel function for the model.

$$K_{\text{Gaussian}}(x, z) = \exp\left(-\frac{\|x - z^2\|}{2\sigma^2}\right). \tag{22}$$

To simplify the experimental process, we used the svm. SVR in the existing Scikit-learn framework for training and

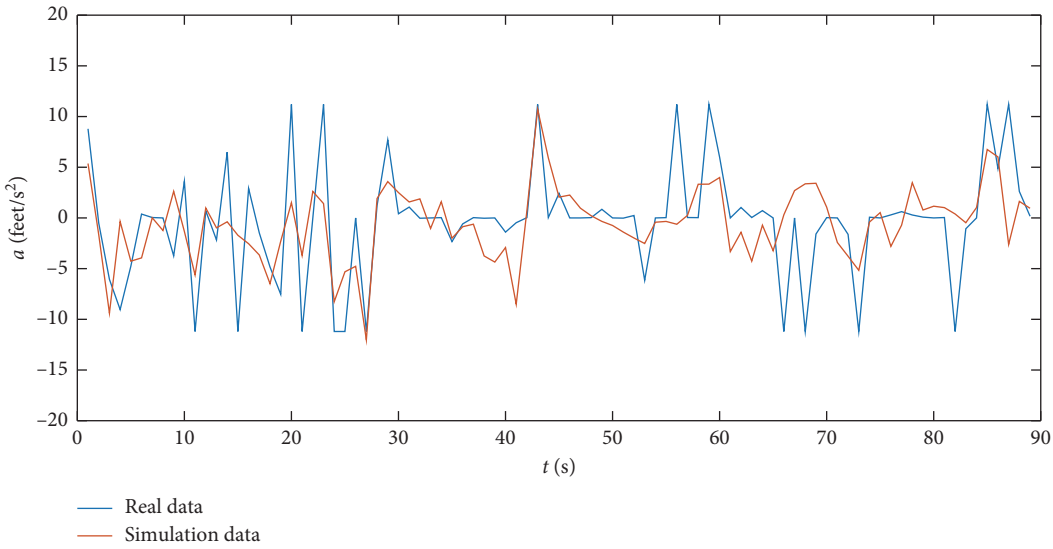


Figure 13: Simulation results of the BPNN-based CF model.

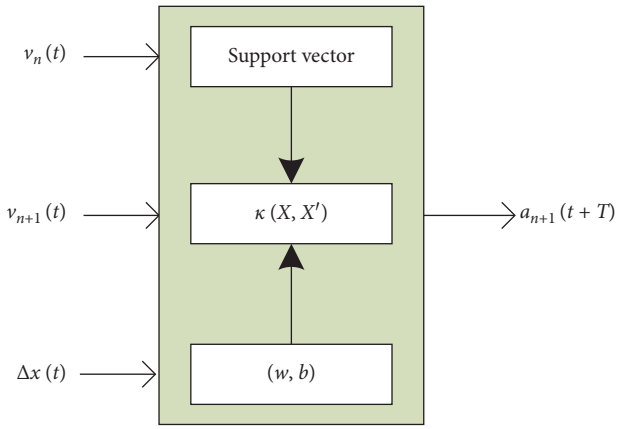


Figure 14: The SVR-based CF model.

testing and assumed the parameter kernel of svm. SVR was "rbf." As in the case of the BPNN-based model, the dataset was randomly split.

We again randomly selected a set of CF data that was associated with leading car 79 and following car 87. In addition, the following car exhibited frequent acceleration and deceleration behaviour. As shown in Figure 15, the simulation results indicate that the prediction error of the SVR-based model is small.

6.3. Comparison between Different CF Models. To compare the above three CF models, we tested each of the models on the same CF dataset (i.e., leading car 1503 and following car 1507). The simulation results are shown in Figure 16.

In this CF dataset, the acceleration and deceleration behaviour of the following car was infrequent, and the following car was driving at a constant speed nearly 40% of the time. As reflected in Figure 16, the polyline of the real data is relatively smooth. Therefore, all three models performed well in the simulation. However, in Figure 16, the GRU predicted acceleration is closer to the real acceleration than the BPNN model and the SVR model when the real acceleration of 0 to 5 seconds and 50 to 60 seconds has a significant continuous jump. Therefore, the GRU model proposed in this paper is superior to the SVR model and the BPNN model.

Figure 17 shows the MSE evaluation values of the above three models. The lower estimation value of MSE represents the lower error. Through the comparison of MSE evaluation index values in the above three model simulation experiments, it is found that the MSE value of BPNN model is the highest, while the model proposed in this paper is the lowest, which is only half of the MSE value of BPNN model. Therefore, the comparison of the MSE evaluation index values in the above three model simulation experiments shows that the proposed model is superior to the SVR model and the BPNN model.

To conclude, the above three models can not only reflect the variation trend in the acceleration of the following car but also accurately predict the values. However, for vehicles driving at variable speeds over a long period of time, the proposed model, which includes a memory unit, had better simulation results but, correspondingly, a slower convergence rate than the other two models.

7. Test Verification on Apollo Simulation Platform

The proposed method for predicting a car-following behaviour is combined with the Apollo platform as follows:

Differentiating scene information in an autonomous driving process of a vehicle into static information and dynamic information, and importing the static information and the dynamic information into Dreamview of the Apollo platform to construct a road scene, specifically including obtaining three-dimensional information of a traffic scene

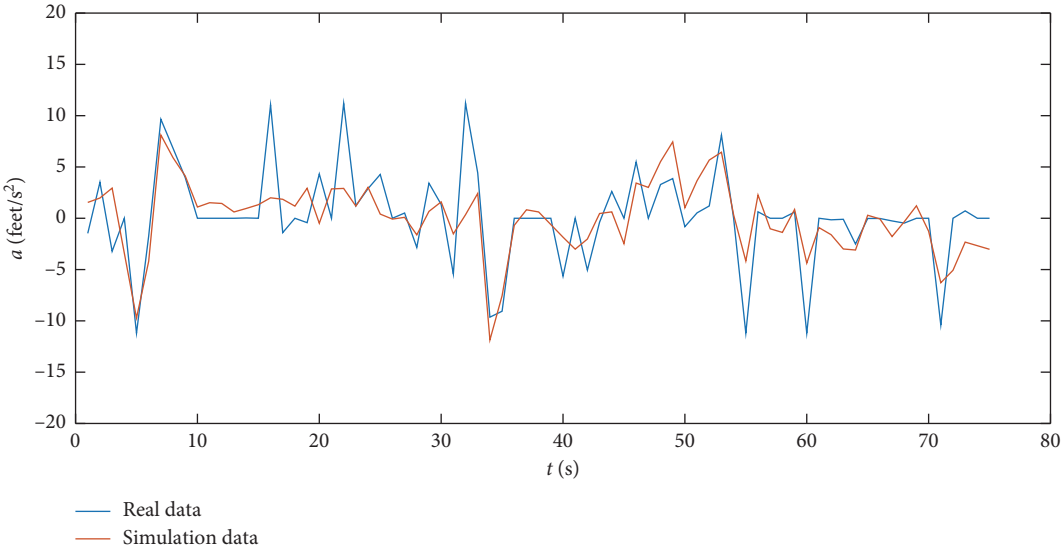


Figure 15: Simulation results of the SVR-based CF model.

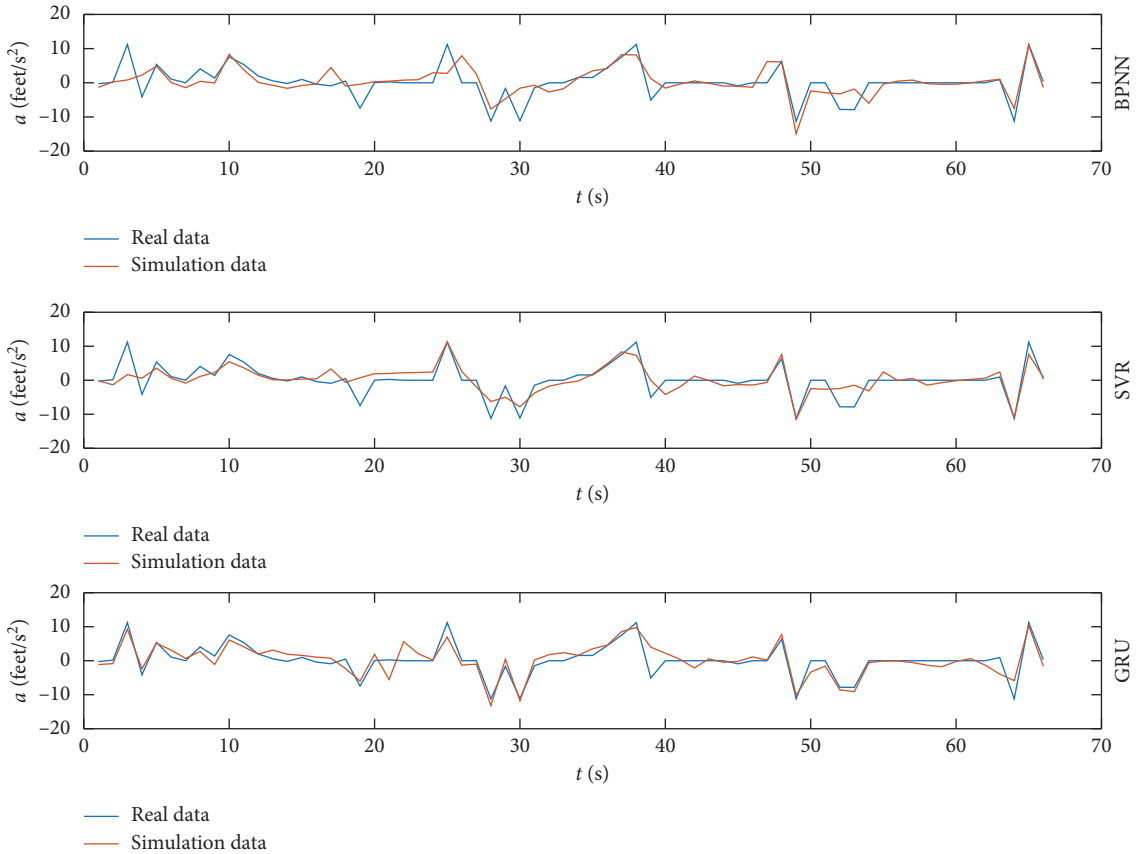


Figure 16: Comparison of the simulation results of the three CF models.

and motion information, wherein the three-dimensional information of the traffic scene refers to static information in the corresponding scene information and the motion information of the traffic scene refers to dynamic information in the scene information; preliminarily constructing a topological structure of the scene, wherein the topological information of the scene includes information such as the number of surrounding vehicles, the lanes occupied by surrounding vehicles, and the distance from a road edge; inputting such information into Dreamview via a corresponding interface of Apollo; configuring paths to specific modules based on the Table of Module Output Interface

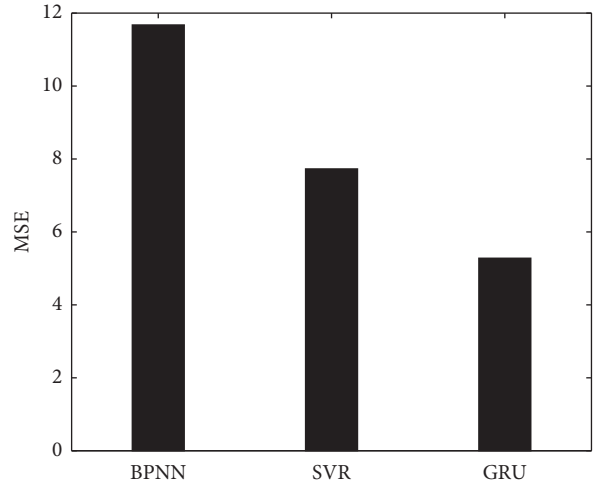


Figure 17: Comparison of the MSE evaluation values of the three CF models.

Table 6: Table of module output interface standards provided by simulation.

Module	Topic	Description	Field
Localization	/apollo/localization/pose	Output data such as the position and orientation of the following car	Position, orientation, linear_velocity, linear_acceleration, angular_velocity in pose
Perception	/apollo/perception/obstacles	Output data such as positions, orientations, velocities, shapes, etc. of respective obstacles	The simulation will provide id, position, theta, velocity, length, width, height, type, polygon_point in PerceptionObstacle
	/apollo/perception/traffic_light	Output traffic light signals	The simulation will provide color, id and tracking_time in Traffic Light,
CAN bus	/apollo/canbus/chassis	Output data such as the velocity and drive mode of the following car	The simulation will provide speed_mps
Router	/apollo/routing_response	Output the navigation result	The simulation will provide routing response, including the planned navigation route from the current position to the destination

Standards (Table 6) provided by the simulation environment, and performing, by respective modules in the Standard, environment construction with reference to the traffic flow and simulated environment information resulting from understanding of the scene, as shown in Table 6.

During the construction process, the dynamic information and the static information during vehicle driving are obtained through understanding of the scene; the desired distance and reaction time are obtained by capturing the driver's behaviour features, and the process of the car-following model is improved using RNN, computing the safety-and-comfort-based optimal solution of the following-car acceleration range. Meanwhile, the model is tested and validated using the Apollo simulation platform to ensure accuracy and utility of the model.

The method for predicting a car-following behaviour under the Apollo platform as disclosed is tested using the Apollo simulation platform. After the Apollo software environment is configured, the output interface of the Apollo platform is docked with the method. After successfully predicting information such as the following-car acceleration using the proposed method, the method is docked with the decision planning module Planning of Apollo; finally, the Apollo software implements testing and validation of the method; during multiple times of simulation process, the parameters are constantly adjusted, and the algorithm is optimized over the Apollo visual platform, specifically:

Deploying the environment (e.g., Docker environment), and pulling the container mirror of Apollo; entering the Apollo container, and compiling the simulation environment (e.g., Dreamview simulation environment); running the simulation environment after successful compilation; testing and validating the efficacy of the model using the corresponding simulation environment; the testing and validating interface refers to the simulation environment interface, wherein the interface is shown in Figure 18.

Docking the traffic flow and environment information outputted by Apollo with the input of the model; then, converting the predicted acceleration value obtained using the model into the Planning input simulation platform, wherein the specific docking path is shown in Table 7.

Docking the traffic flow and environment information outputted by Apollo with the input of the model; then, converting the predicted acceleration value obtained through the model into the Planning input simulation platform, wherein the specific docking path is shown in Figure 19.

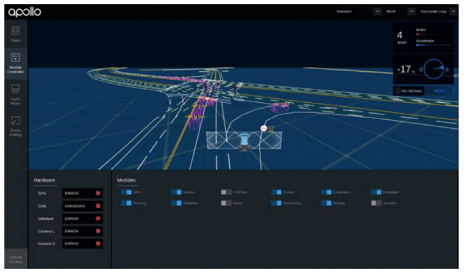


Figure 18: A simulation Apollo platform.

Table 7: Table of decision planning module interface standards.

Module	Topic	Description	Fields
Planning	/apollo/ planning	Output the planned trajectory of the following car in a future period of time	The developer must provide (1) timestamp_sec in header (2) v, a, relative_time in trajectoryPoint (3) x, y, z, theta, kappa in pathPoint
Prediction	/apollo/ prediction	Output respective obstacles and their predicted trajectories	The developer optionally outputs trajectories in predictionObstacle, which may be used for displaying predicted trajectories of respective obstacles
Decision	/apollo/ decision	Output decisions with respect to various obstacles and the main decisions	The developer optionally outputs mainDecision and objectDecisions, which may be used for displaying the main decisions and the decisions with respect to respective obstacles

Testing and validating the model in the Apollo simulation environment.

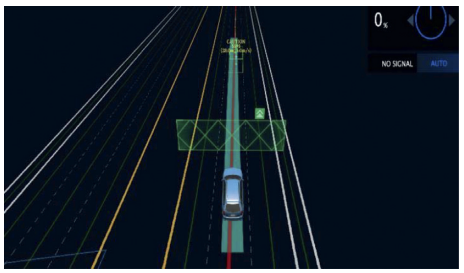
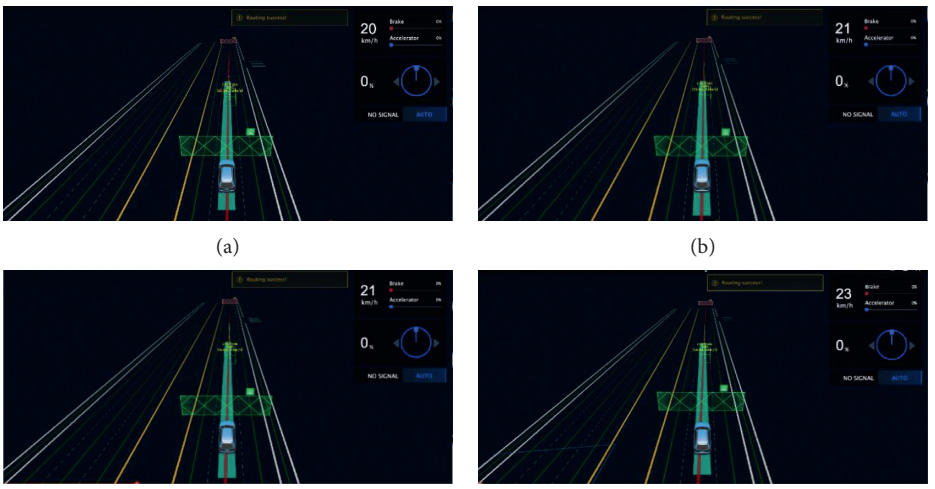


Figure 19: Simulation environment.



(a) (b) (c) (d)

Figure 20: The following process under Apollo platform.

Figure 20 is part of the following process diagram of the vehicle following behaviour visualization on the Apollo platform using the model proposed in this paper. In the figure, the cube with green border is the leading car and its current speed is shown directly above it. The blue car is the following car and its current speed is shown at the top right of the figure. In Figure 20(a), the speed of the leading car is 5 m/s (equivalent to 18 km/s), and the speed of the following car is 20 km/s. In Figure 20(b), the speed of the leading car is 5 m/s (equivalent to 18 km/s), and the speed of the following car is 21 km/s. In Figure 20(c), the speed of the leading car is 5 m/s (equivalent to 18 km/s), and the speed of the following car is 21 km/s. In Figure 20(d), the speed of the leading car is 5 m/s (equivalent to 18 km/s), and the speed of the following car is 23 km/s. In the comparison of Figures 20(a)–20(d), it is found that the speed of the following car has been fluctuating from the speed of the leading car so that the speed of the following car is not longer than that of the leading car. Moreover, a relatively long distance is maintained between the following car and the leading car, which provides sufficient braking distance for the following car to ensure the driving safety of the following car. Through the visual simulation of the model proposed in this paper on the Apollo simulation platform, it was found that there would be no collision between the following car and the guiding car during the long time of following. Therefore, by using Apollo simulation platform to test the model visually, it is found that the model proposed in this paper is valid and practical.

8. Conclusions

In this study, we used high-precision vehicle trajectory data from Interstate I-80 in the NGSIM dataset to obtain the CF data through data preprocessing. Based on the characteristics of the CF behaviour, we verified the correctness of the data through experiments and chose the reaction time of $T = 1$ s to further filter the CF data. We modelled the CF behaviour based on the RNN network, using the speed of the leading car, the speed of the following car, and the distance between the two cars, all at time t, as inputs and the acceleration of the following car at time $(t + T)$ as output. Finally, we performed in-depth examination and experiments on the length of the "memory" time interval and the network structure of the constructed model and obtained the parameters that enabled the model to have its best performance. Further, we compared the simulation results of the constructed model with those of the BPNN- and SVR-based models and demonstrated that the constructed model with a memory unit added had higher simulation accuracy. Both the BPNN- and SVR-based CF models were unstable; i.e., when the following vehicle had frequent acceleration and deceleration behaviour, their simulation results were poor. In comparison, the RNN-based CF model was able to make more accurate predictions because it considered the relevant information of the last several time intervals. At the same time, Apollo simulation platform was used to test and verify the model, ensuring the accuracy and practicability of the model.

The RNN-based CF model established in this study was only used to study the CF behaviour between small cars. In reality, the CF behaviours between different types of vehicles are different. In a follow-up study, we will construct models that consider various types of vehicle so that the existing model can be further improved according to the actual conditions such as asymmetric driving behaviour and multiple leading vehicles.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge the support from CERNET Innovation Project under Grant NGII20161201, Scientific and Technological Planning Project of Beilin District of Xi'an under Grant GX1819 supported by National Key Research and Development Program of China (2018YFB1201500), Natural Science Foundation of China (no. 61773313), and Teaching Research Foundation of Xi'an University of Technology (XJY1866).

References

[1] B. G. Cao, "A car-following dynamic model with headway memory and evolution trend," *Physica A: Statistical Mechanics and Its Applications*, vol. 539, Article ID 122903, 2020.

[2] G. H. Peng, S. H. Yang, D. X. Xia, and X. Q. Li, "Delayed-feedback control in a car-following model with the combination of V2V communication," *Physica A: Statistical Mechanics and Its Applications*, vol. 526, Article ID 120912, 2019.

[3] L. Zheng, C. Zhu, Z. He, T. He, and S. Liu, "Empirical validation of vehicle type-dependent car-following heterogeneity from micro- and macro-viewpoints," *Transportmetrica B: Transport Dynamics*, vol. 7, no. 1, pp. 765–787, 2018.

[4] L. Bin, *Modeling and Analysis of Car-Following Behavior Using Data-Driven Methods*, Southwest Jiaotong University, Chengdu, China, 2017.

[5] L. Li, D. Zhang, Z.-G. Xu, P. Wang, and G.-P. Wang, "The roles of car following and lane changing drivers' anticipations during vehicle inserting process: a structural equation model approach," *Journal of Advanced Transportation*, vol. 2018, Article ID 6372861, 19 pages, 2018.

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 696–699, 1988.

[7] C. H. Chen, F. Y. Song, F. J. Hwang, and L. Wu, "A probability density function generator based on neural networks," *Physica A: Statistical Mechanics and Its Applications*, vol. 541, Article ID 123344, 2020.

[8] N. Kehtarnavaz, N. Groswold, K. Miller, and P. Lascoe, "A transportable neural-network approach to autonomous vehicle following," *IEEE Transactions on Vehicular Technology*, vol. 47, no. 2, pp. 694–702, 1998.

[9] J. H. Zhang, Q. Li, and D. P. Chen, "Safety assistance algorithm for longitudinal collision avoidance based on BP neural network," *Journal of Xi'an Jiaotong University*, vol. 51, no. 7, pp. 140–147, 2017.

[10] B. Lu, N. Shaoquan, and S. S. Washburn, "A support vector regression approach for investigating multianticipative driving behavior," *Mathematical Problems in Engineering*, vol. 2015, Article ID 701926, 10 pages, 2015.

[11] D. Wei and H. Liu, "Analysis of asymmetric driving behavior using a self-learning approach," *Transportation Research Part B: Methodological*, vol. 47, pp. 1–14, 2013.

[12] P. Parham, M. P. A. Moghadam, and B. Bigdeli, "Car following prediction based on support vector regression and multi-adaptive regression spline by considering instantaneous reaction time," *Iranian Journal of Science and Technology, Transactions of Civil Engineering*, vol. 43, no. S1, pp. 67–79, 2019.

[13] X. L. Huang, J. Sun, and J. Sun, "A car-following model considering asymmetric driving behavior based on long short-term memory neural networks," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 346–362, 2018.

[14] X. Zhang, J. Sun, X. Qi, and J. Sun, "Simultaneous modeling of car-following and lane-changing behaviors using deep learning," *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 287–304, 2019.

[15] Y. K. Wu, H. C. Tan, X. X. Chen, and B. Ran, "Memory, attention and prediction: a deep learning architecture for car-following," *Transportmetrica B: Transport Dynamics*, vol. 7, no. 1, pp. 1553–1571, 2019.

[16] C. Hua, "A new car-following model considering recurrent neural network," *International Journal of Modern Physics B*, vol. 33, no. 26, 2019.

[17] D. Yang, D. Wu, Y. Liu, and B. Ran, "A novel car-following control model combining machine learning and kinematics models for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 6, pp. 1991–2000, 2019.

[18] Y. Li, X. Lu, C. Ren, and H. Zhao, "Fusion modeling method of car-following characteristics," *IEEE Access*, vol. 7, pp. 162778–162785, 2019.

[19] D.-F. Xie, Z.-Z. Fang, B. Jia, and Z. He, "A data-driven lane-changing model based on deep learning," *Transportation Research Part C: Emerging Technologies*, vol. 106, pp. 41–60, 2019.

[20] B. Xiao, Y. G. Liu, and B. Xiao, "Accurate state-of-charge estimation approach for lithium-ion batteries by gated recurrent unit with ensemble optimizer," *IEEE Access*, vol. 7, pp. 54192–54202, 2019.

[21] X. Wang, R. Jiang, L. Li, Y. Lin, X. Zheng, and F.-Y. Wang, "Capturing car-following behaviors by deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 910–920, 2018.

[22] J. Wang, F. Sun, and R. Cheng, "An extended car-following model considering the self-stabilizing driving behavior of headway," *Physica A Statistical Mechanics & Its Applications*, vol. 507, pp. 347–357, 2018.

[23] C.-H. Chen, "A cell probe-based method for vehicle speed estimation," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E103.A, no. 1, pp. 265–267, 2020.

[24] C. J. Ou, S. M. Bedawi, A. B. Koesdwiady, and F. Karray, "Predicting steering actions for self-driving cars through deep learning," in *Proceedings of the 2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, IL, USA, August 2018.

[25] J. Webber, N. Suga, A. Y. Mehbodni, K. Yano, and T. Kumagai, "Study on fading prediction for automated guided vehicle using probabilistic neural network," in *Proceedings of the 2018 Asia-Pacific Microwave Conference (APMC)*, Kyoto, Japan, November 2018.

[26] A. Chernikova, A. Oprea, C. Nita-Rotaru, and B. G. Kim, "Are self-driving cars secure? Evasion attacks against deep neural networks for steering angle prediction," in *Proceedings of the 2019 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, May 2019.

[27] FHWA, *Next Generation Simulation (NGSIM)*, FHWA, Washington, DC, USA, 2017.

[28] N. Arbabzadeh, M. Jafari, M. Jalayer, S. Jiang, and M. Kharbeche, "A hybrid approach for identifying factors affecting driver reaction time using naturalistic driving data," *Transportation Research Part C: Emerging Technologies*, vol. 100, pp. 107–124, 2019.

[29] Y. Wang, J. Zhang, and G. Lu, "Influence of driving behaviors on the stability in car following," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 3, pp. 1081–1098, 2019.

[30] C. K. Kim, B. Yacoubi, and E. A. Christou, "Visual load and variability of Muscle activation: effects on reactive driving of older adults," *Human Movement Science*, vol. 63, pp. 172–181, 2019.

[31] Q. Jin, *Reasearch on Parameters Calibration and Verification of Car-Following Models*, Shanghai Jiao Tong University, Shanghai, China, 2008.

[32] G. Lu, B. Cheng, Y. Wang, and Q. Lin, "A car-following model based on quantified homeostatic risk perception," *Mathematical Problems in Engineering*, vol. 2013, Article ID 408756, 13 pages, 2013.

[33] A. Kumar, A. Prakash, S. Sharma, and K. Jyoti, "Vehicle authentication and message hiding protocol for vehicle to vehicle communication," in *Proceedings of the 2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, IEEE, Dehradun, India, September 2015.

[34] N. F. Mustamin, "Relative distance measurement between moving vehicles for manless driving," in *Proceedings of the 2017 International Seminar on Application for Technology of Information and Communication*, Semarang, Indonesia, October 2017.

[35] V. Kurková, "Kolmogorov's theorem and multilayer neural networks," *Neural Networks*, vol. 5, no. 3, pp. 501–506, 1992.