An Easy-portable Displacement-offset-based Trajectory Prediction Method for Autonomous Vehicles

Cheng Wei[1], Fei Hui[1,*], Xiangmo Zhao[1], Shanke Li[1], and Jie Wei[2]
[1]School of Information Engineering, Chang'an University, Xi'an, Shaanxi, China
[2]Shaanxi Heavy Duty Automobile Co.Ltd, Xi'an, Shaanxi, China
{chengwei, feihui, xmzhao, shankeli}@chd.edu.cn,weijie_jszx@sxqc.com
*corresponding author

Abstract—Trajectory prediction algorithm is an important component of autonomous driving system (ADS) or advanced driver assistance system (ADAS), which enable autonomous vehicles to evaluate critical tasks in advance thus reduce vehicle collisions and improve traffic safety. Most existing trajectory prediction methods suffer from difficulties in portability and application across coordinate systems. To address these difficulties, this study proposes an easy-portable, vehicle displacement-offset-based trajectory prediction model, which can be rapid deployed and reused in different highway road sections and does not require second training. Specifically, first a novel trajectory sampling method to homo-dimension the vehicle data of different lengths is proposed. Second, the traditional neural network used for sequence prediction is modified to develop an enhanced trajectory prediction model, and which is trained and tested using the input reduction method. Finally, the trained model is saved locally and embedded in a co-simulation environment composed of CARLA, SUMO and Keras, afterwards the proposed method is tested in real-time simulation and across coordinate systems. The experimental results show that the proposed method can be quickly ported and deployed for reuse without second training, and has a fairly high prediction accuracy and long prospective time.

Keywords-autonomous driving, trajectory prediction, model porting, simulation testing, neural network.

1. INTRODUCTION

1.1. Motivation

Autonomous driving has attracted tremendous attention worldwide due to its great benefits in improving transportation efficiency and safety [1]. An autonomous driving system (ADS) contains many algorithms, and the individual algorithms cooperate with each other to provide the different functions of the ADS. Currently, autonomous driving is in an important transformation stage from group testing to practical application, and virtual simulation testing has become one of the main methods for autonomous driving testing due to its advantages of low cost, repeatability, and rapid deployment [2,3]. Trajectory prediction, as one of the important technologies in autonomous driving, enables the driver or ADS in an autonomous vehicle to evaluate critical tasks in advance and adjust the vehicle motion state to avoid traffic congestion and collisions. Therefore, how to perform fast and efficient trajectory prediction algorithm development with high portability and test it has become a hot research topic, where Portability refers to the ability of a model to quickly transfer to another scenario for prediction. In addition, with simple traffic flow and less random disturbance in traffic environment, highways are one of the preferred road sections for autonomous driving deployment. In summary, the development, testing, and portability of trajectory prediction algorithms in highway scenarios have significant research value.

1.2. Related works

Currently, a number of research exist for trajectory prediction, which have modeled trajectory prediction in different scenarios based on different data characteristics. The focus of trajectory prediction is to analyze the relationship between the historical motion state and the future motion state of vehicles and model this relationship. According to different modeling methods, the currently existing trajectory prediction methods can be divided into mathematical inference models and data-driven models, and the following content will review the representative studies.

Mathematical inference models (MIMs): MIMs focus on analyzing the functional relationship between historical parameters and future trajectories, and describe the relationship as an explicit function. The most typical model based on mathematical inference is the use of polynomials for trajectory modeling. Back in 1989, Nelson [4] used polar coordinate polynomials and Cartesian polynomials to model the trajectory of a vehicle in a straight line to a circular arc, ensuring accuracy when tracking vehicle trajectories. Trieu Minh Vu et al. [5] proposed a method for generating vehicle trajectories based on multiple polynomials considering the restrictive relationship among different vehicle motion parameters, which can be applied to the route generation function of automatic parking. Calvin Kielas-Jensen et al. [6] presented a method for the generation of trajectories for ADS, which is based on Bernstein polynomial approximations to transcribe infinite dimensional optimization problems into nonlinear programming problems. Similarly, Wang et al. [7] predicted and generated lane change trajectories for autonomous vehicles on highways using seven polynomials, and the usability of their method was demonstrated by real vehicle experiments. Nunzio A. Letizia et al. [8] presented a novel recursive smooth trajectory (RST) generation algorithm for application in robotics based on polynomial, and the effectiveness of the proposed algorithm is demonstrated numerically via two illustrative scenarios. To find the optimal trajectory of an autonomous vehicle, Vu Trieu Minh and John Pumwa [9] modeled and analyzed the vehicle trajectory using symmetric polynomial, and experiments demonstrated that the proposed method can generate smooth trajectories for the control of autonomous vehicles. Ming et al. [10] proposed a quintic polynomial-based trajectory planning approach to generation comfort and safety trajectories. In addition to various polynomials, hidden Markov models (HMM) have also been applied to trajectory prediction. Ye et al. [11] proposed a new trajectory prediction model based on HMM, and their proposed method was experimentally shown to be quite accurate. Aiming at the problem that conventional models can not accurately describe the trajectory of vehicles in a network-constrained environment, Qiao et.al [12] proposed an HMM-based adaptive parameter selection trajectory prediction model for autonomous vehicle. Moreover, Gaussian related models are also widely used in trajectory prediction, in [13], a variational probability trajectory model based on Gaussian mixture and Bayesian is proposed, which can predict the trajectory of vehicles within 2s. Georges et al. [14] proposed a flexible non-parametric hybrid Bayesian model to represent the trajectory distribution and good results were achieved. There is no doubt that MIMs have made great achievements in trajectory prediction, but the disadvantage of MIMs is that their generalization ability and feature extraction ability are not strong, so the models cannot be ported quickly, therefore, data-driven models are introduced.

Data-driven models (DDMs): DDMs focus on extracting correlations between input and output, most of the data features are extracted in a black-box manner. DDMs represented by neural network-based models have been used extensively. Depending on the type of input data, DDMs can be broadly classified into models based on image and models based on time series data. Using images from the driver's perspective as input, Sun et al. [15] proposed a semantic segmentation model that can generate predicted trajectories directly on the current image. Similar to [15], Fang et al. [16] proposed a motion prediction framework with two phases for vehicle trajectory prediction using image data. In order to quantify the uncertainty of pedestrians, Probabilistic Population GAN (PCGAN) was proposed in [17], which can quantify pedestrian behavior under different traffic conditions to determine the impact of different behaviors on traffic, thereby adjusting the current traffic state. Different from the above research, Tang et al. [18] proposed a probabilistic-based framework for modelling future vehicle motion state using bird's-eye view images of a specific road section. As can be seen, the advantage of the image-based model is that the image data is easy acquired, sometimes only one camera is needed, and the collected data does not require much processing. However, when running the trajectory prediction algorithm, the image-based model requires powerful hardware support to meet the arithmetic requirements. Therefore, time series-based models are introduced. Time series data requires more sensors during acquisition as well as frame alignment and other operations after the acquisition is completed, but time series data-based models do not require strong hardware support after training and are less costly than image-based models in general. For example, Hui et al. [19] proposed a combinatorial neural network model to predict trajectories including highways, intersections, and roundabouts using time series data, and obtained good results. Based on the Encoder-Decoder structure, Wei et al. [20] proposed a fine-grained prediction model for highway scenario, which can predict the velocity and trajectory, in addition, they also propose a joint simulation environment for real-time simulation of the proposed prediction models in [21]. Similarly, a large number of studies [22-28] have used LSTM or combined variants of LSTM for trajectory prediction modeling. In summary, the extant studies may have the following shortcomings:

- Some of the studies focus on analyzing the relationship between historical and future trajectories and use coordinates in the absolute coordinate system as model inputs, which may result in the model not being usable across coordinate systems and maps.
- Some studies focus on the construction of trajectory prediction methods and theoretical performance analysis, without real-time simulation testing of the model, which may lead to the usability of the model in doubt.
- Some of the high-complexity models ignore the hardware and software costs in practical applications, resulting in low utility of the models.

Based on the above shortcomings, this study proposes a fast, efficient, and portable neural network model based on time series data for real-time simulation across coordinate systems to fill some research gaps.

1.3. Paper organization

The remainder of this study will be organized in the following way. Section 2 provides a detailed description of the dataset used and its processing method; Section 3 carries out the construction and description of the prediction model according to processed data characteristic. Section 4 tests the proposed model using a co-simulation approach. Finally, Section 5 concludes the paper and describes future research directions.

2. Data preprocess

It is known from previous studies that vehicle trajectories are not only related to historical trajectories, but also to historical vehicle motion states, so the data used needs to include both of these types of data. Based on the above requirements, HighD, a publicly available dataset from Germany [29], was selected as the data base for this study.

2.1. Data introduction

The data in the HighD dataset is derived from naturally driven vehicles on German highways [30], and all data is obtained from drones with fixed viewpoints above the highways, and the time series data has been extracted using a state-of-the-art computer vision approach, and with the error less than 10cm. Therefore, due to the characteristic of richness and accuracy of the data, this dataset is selected as the data basis for this study.

The HighD dataset contains data from several highway sections, and the length of its collected sections is about 400 meters. One of them is shown in Figure 1. According to the data requirements, the data under five identical road sections are selected as the basic data in this study. These data are stored under different files and their details are shown in Table 1. However, due to the large amount and similarity of the data, too much data overlay display will cover the details of the data, so the data in file No.25 will be used as the sample for subsequent data display in this study. In addition, the original data in the HighD dataset is high-dimensional and unequal in length, which cannot be directly applied to the neural network model in this study, therefore, this data needs to be processed in a secondary way, and the subsequent content will be carried out to introduce the data processing method.

2.2. Data preprocess

The HighD dataset is collected at a fixed viewpoint with equal frequency, so the faster vehicles have shorter data sequences and the slower vehicles have longer data sequences. However, the neural network model requires fixed-length inputs, and since the VKD and VTD of vehicles are correlated, this study uses the VTD as the basis for sampling the entire data.

Analyzing the data in the HighD dataset, it can be concluded that there are two types of vehicle trajectories on the highway, the first are lane keeping trajectories dominated by nearly straight lines and the second are lane changing trajectories dominated by curves, according to the above characteristics, this study proposes a data sampling dimensionality reduction method. Suppose the number of sampling points is denoted by $sampling\_times$, then it is expressed by:

$$sampling\_inter = \lfloor length / sampling\_times \rfloor, \quad (1)$$

where $length$ denotes the length of a vehicle trajectory, and the trajectory lengths of each vehicle are usually not equal and need to be acquired in real time during computation.

Figure 1. One highway section from the HighD (No. 25)

Table 1. Selected road sections

Selected file No.	4	7	11	25	31
Number of vehicles	1,163	855	1,776	2,850	2,254

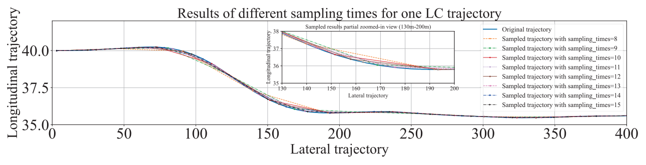
Further, assume the index of a coordinate on a trajectory is $trajectory\_index$, then, if $trajectory\_index$ and $sampling\_integer$ satisfy the following relationship:

$$trajectory\_index \% sampling\_inter = 0, \quad (2)$$

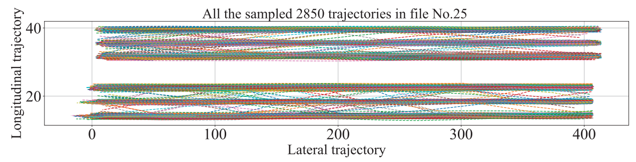
the trajectory coordinate point corresponding to $trajectory\_index$ will be selected as a sampling point, and the sampling operation will continue until the whole trajectory is facilitated. It should be noted that the "%" in (2) represents the modulo operation. Using the above method, the original data can be substantially downsized. However, in order to ensure that the sampled trajectories have the same composition of coordinate points, this study will process the sampled trajectories again, if the number of coordinates of the sampled trajectories is greater than the number of samples, the penultimate coordinate at the end of the trajectory will be removed.

In summary, it can be seen that the key parameter of the sampling algorithm is the $sampling\_times$, which represents the number of coordinate points of the trajectory after surrogate raising, in order to get the optimal $sampling\_times$, this study uses the method of multiple comparisons, and the results of the comparisons are shown in Figure 2(a), which shows that the sampled trajectory can be well fitted original trajectory when $sampling\_times = 14$, so the $sampling\_times$ of the trajectory in this study will be determined as 14.

After the sampling is completed, some irregular vehicle data (the 70th vehicle in file No. 25, the 28th and 29th vehicles in file No. 11) are again eliminated, and the basic data that can be used in this study can be obtained. Taking file 25 as the display sample, the trajectory after sampling using the above method is shown in Figure 2(b).



(a)



(b)

Figure 2. (a) Sampling results under different sampling times and (b) sampled trajectories in file No.25

However, considering that part of the sampled data is to be used as model inputs and some as model labels, it is also necessary to segment the sampled data. In addition, due to the characteristics of the above sampling method, the distance between two coordinate is not equal, which means that after segmenting the data with ratio 1:1, the data on both sides of the segmentation point also have different trajectory lengths, and this feature can solve the problem that the model has different sampling frequencies for sensors in practical applications. After segmenting the sampled trajectories in a 1:1 ratio, the segmentation points are visualized to obtain the results shown in Figure 3. However, it can be seen from Figure 3 that the data on both sides of the data segmentation point is not enough, when using such a data set in training the model, the model may converge to an intermediate state. In order to solve this problem, the model will gradually reduce the segmentation ratio during training to make the model generalize as much as possible. Once the data processing is complete, the subsequent content will carry out the construction of the prediction model.

3. Methodology

In this section, the types and formats of input and output data are described based on the sampled and segmented data, afterwards the trajectory prediction model is designed and trained based on the data characteristics.

3.1. Input and Output description

As mentioned before, one of the contributions of this study is to solve the problem that the model cannot be reused across coordinate systems to a certain extent. Therefore, different from existed studies, this study changes the idea of analyzing the relationship between historical trajectory and future trajectory, transposing the input and output of the model into the velocity sequence and displacement sequence of the vehicle, respectively.

Based on the processed data, it can be seen that the main input when the model starts training consists of seven velocity points and seven environmental data points. Based on the characteristics of data processing, this study selected seven distance headway point corresponding to the velocity sequence as auxiliary inputs, which are $[e_1, \dots, e_7]$ in (3). As the training advances, the actual coordinate number of the input will gradually reduce, but the input length will remain the same, so the vacant positions will be replaced by 0. In addition, the label data consists of six relative displacement values, and the length of the label data will become smaller as the model is trained, but in order to fit the network, the labels will be indented with the trailing data discarded to keep the data length constant. In summary, the input and output of first round can be expressed as:

$$input = [v_1, \dots, v_7, e_1, \dots, e_7], \tag{3}$$

$$label = [d_i - d_{i-1}] \quad (i = 6, 5, \dots, 2, 1). \tag{4}$$

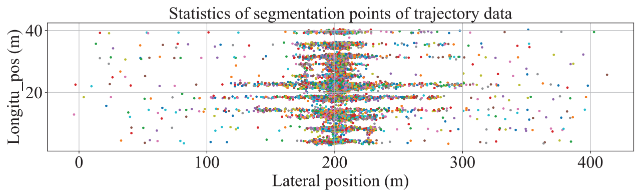


Figure 3. Statistics of segmentation points of trajectory data

where d_i denotes the coordinate of the vehicle. When the model starts the next round of training, the input and labels shrink, the label number remains the same, but the input will undergo data discard, which can be expressed as:

$$input = [v_1, \dots, v_6, 0, e_1, \dots, e_6], \tag{5}$$

After the completion of multiple rounds of training, the model in the optimal epoch will be saved locally in .h5 format for testing.

3.2. Trajectory prediction model design

According to the characteristics of the input and label described in the previous subsection, a neural network-based trajectory prediction model is designed, which has a vector of $[14\times1]$ as input and a vector of $[6\times1]$ as label, but the vehicle trajectory is divided into two directions: lateral and longitudinal, so the model should contain two parallel modules to predict the displacement in both directions simultaneously.

In conventional studies, as shown in Figure 4, the Encoder-Decoder structure with a long short term memory network (LSTM) as the computational core is often used for the prediction of time series, and its input and label are both a time series and mostly the same kind of data. However, such a structure may not be suitable for making predictions of different kinds of time series data, and cause a decrease in model accuracy. In addition, most of the studies can be classified into many-to-one (M2O) or many-to-many (M2M) problems depending on the relationship between the input and the output. In general, M2O is a classification problem, which requires fusion of the input data, and M2O is a regression problem, which requires sequence-to-sequence mapping. Since the input of this study contains both sequence data and some discrete environmental data, both the M2O problem and the M2M problem are included in this study, which requires the model to have both the ability of data fusion and data regression. In view of the excellent data fusion capability of the fully connected neural network and the sequence data mapping capability of the Encoder-Decoder structure, this study therefore modifies the Encoder-Decoder structure by adding a fully connected layer to its first layer to form a combined network module as the basis of the prediction model.

First, the conventional Encoder-Decoder is modified, and both Encoder and Decoder are changed to bi-directional Encoder and bi-directional Decoder with bi-directional propagation capability in order to improve its forward and reverse data mapping capability. In addition, due to the shortening characteristic of the input data in this study, then the first layer of the fully connected network is required to be able to perform adaptation for different input lengths, so some neurons in the first layer of the fully connected network are set as conditionally deactivated neurons to match different input lengths in this study. The trajectory prediction module designed according to the above description is shown in Figure 5.

It can be seen that this study uses a combined fully connected network and encoder-decoder structure, setting the input to this structure as a sequence $x = [x_1, x_2, x_3, \dots, x_n]$, when x crosses the first layer, the encoding work is performed to obtain the hidden-layer output at time t, which is:

$$h_t = f(x_t, h_{t-1}), \tag{6}$$

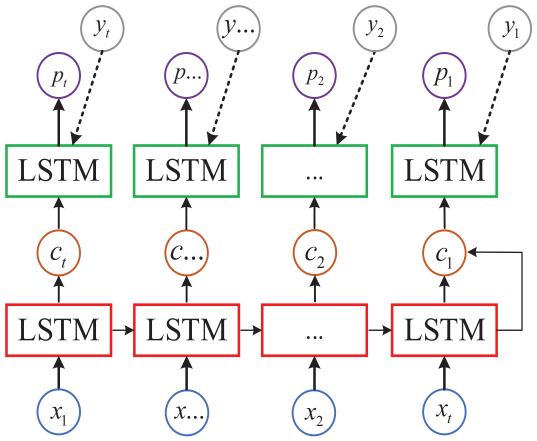


Figure 4. Conventional LSTM-based Encoder Decoder

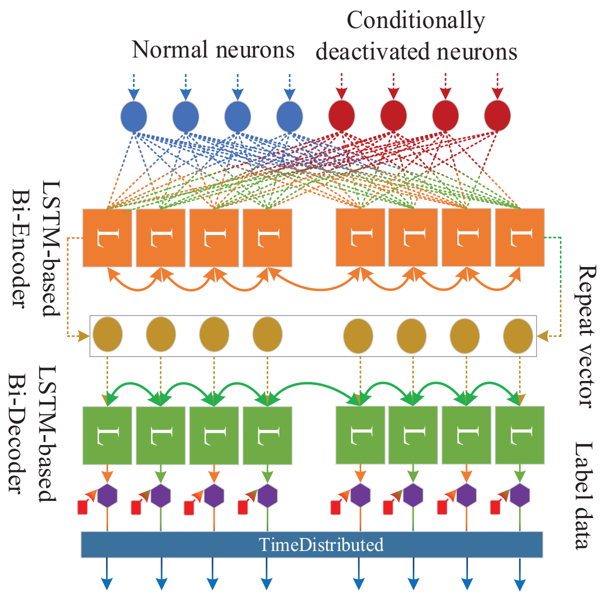


Figure 5. Enhanced trajectory prediction model

when the model starts its first forward computation, its internal weight parameters will be randomly generated by the deep learning library, and the initial hidden state h_1 is automatically calculated.

After the encoding work is completed, the final state of the encoder is the fixed-dimensional semantic vector c, which can be expressed as:

$$c = q(h_1, h_2, \dots, h_t), \tag{7}$$

subsequently, the decoder uses the semantic vector $C = \{C_1, C_2, \cdots C_n\}$ to decode and calculate the prediction result y_t at time t as follows:

$$p(y) = \Pi_{t=1}^{t} p(y_t \mid y_1, y_2, \dots, y_{t-1}, c). \tag{8}$$

Since the features of the encoder-decoder are propagated as a sequence, after considering the intermediate state of the decoder's hidden layer, the probability of a step y can be expressed as:

$$p(y_t \mid y_1, y_2, \dots y_{t-1}, c) = g(y_{t-1}, s_t, c), \tag{9}$$

where s_t denotes the state of the decoder's hidden layer at time t, and g is a nonlinear function. Therefore, the probability of obtaining the final output y is expressed as:

$$p(y) = \Pi_{t=1}^{t} g(y_{t=1}, s_t, c), \tag{10}$$

After obtaining the output of the encoder, the decoder starts its work with the task of modelling the conditional probability distribution of the predicted results. If no other information intervenes at this point, the decoder will output the conditional probability of y at moment t, which can be expressed as:

$$p(y_t \mid y_1, y_2, \dots y_{t-1}, x) = g(y_{t-1}, s_t, c_t), \tag{11}$$

where s_t denotes the output of the RNN's hidden layer at time t. Thus, it can be written that:

$$s_t = f(s_{t-1}, y_{t-1}, c_t). \tag{12}$$

After completing the forward propagation, the neural network uses a backward propagation algorithm to optimize the weights and thus obtain the optimal network.

The above explains how the model works and the prediction model, after completing the interpretation of the proposed trajectory prediction model, it can be used to construct a trajectory prediction model suitable for both lateral and longitudinal directions. The structure of this model is shown in Figure 6. It can be seen that the model is capable of performing parallel

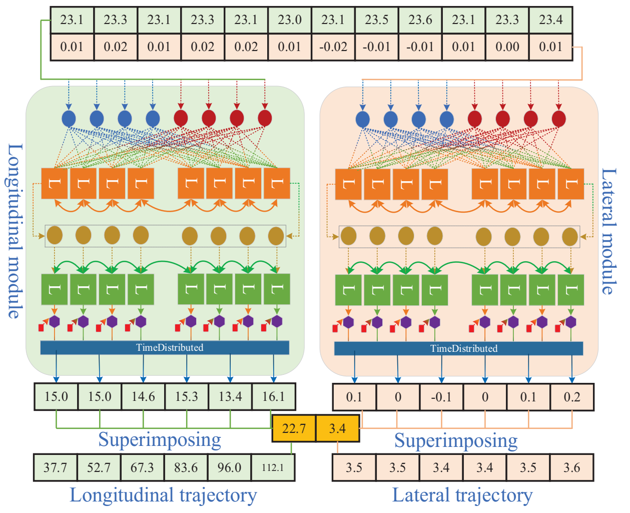


Figure 6. Trajectory prediction model

input of data in both directions, and after computation, it outputs two displacement vectors in both directions simultaneously. The predicted trajectory can be obtained by superimposing the displacements.

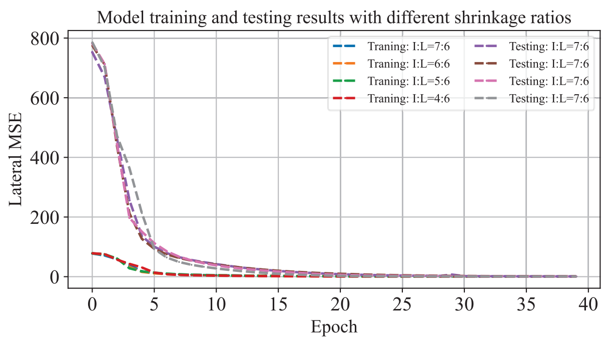
When the model predicts the displacement, the current coordinates are obtained, then the predicted trajectory is obtained by superimposing the predicted lateral and vertical displacements.

3.3 Model training and testing

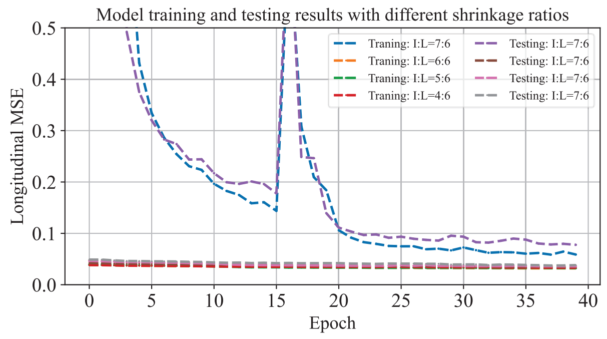
Once the model is designed, it is constructed and the trained using the data described in Section 2.

Since this study considers the challenges posed by real-time simulation tests on the prediction model, it is necessary to consider the time consumption of the model for one prediction task in addition to traditional studies comparing metrics such as root convergence mean square error (C-MSE) and epoch (CE). Using the data with different reduction ratios for training and testing, the MSE performance of the predicted model in both lateral and longitudinal directions are shown in Figure 7.

The MSE of the model during training and testing is shown in Figure 7. It can be seen that both the lateral and longitudinal MSEs converge in the region after a certain number of epochs, and the convergence MSE is relatively small. In addition, the time consumed by the model to execute a task is statistically analyzed, and the results shown in Figure 8 are obtained. It can be seen that the time consumption of one module in executing one task is under 0.04 microseconds, and even if the lateral and longitudinal modules work serially, the time consumption is under 0.1 microseconds, which can be said to complete the prediction task instantaneously. Therefore, at the level of time consumption, the model is able to meet the requirements of the simulation. The trained model is saved locally in .h5 format to enable real-time simulation, so in the next section, this study will test the trained model by real- time simulation. It should be noted that "I:L" in the Figure 7 refers to the ratio of effective input to output (IOR).



(a)



(b)

Figure 7. (a)Lateral and (b)longitudnial MSE in traning and testing process

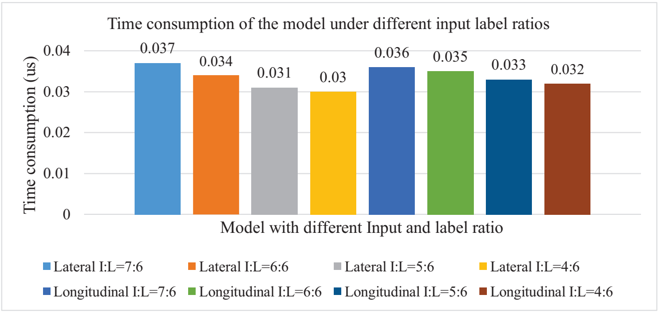


Figure 8. Time consumption of the model under different input label ratios

4. Real-time simulation

Real-time simulation of the proposed trajectory prediction model will be performed in this session. First, the simulation environment is constructed based on the model input data characteristics, and then the trained model is embedded into the simulation environment and acted on a certain vehicle to perform real-time trajectory prediction. Finally, the results of the simulation are analyzed. This study builds the model on a computer with a window 10 operating system and an i5-8500

CPU and 16 GB of RAM, the deep learning libraries used are TensorFlow 2.2.0 and Keras 2.3.1, the programming IDE used is Spyder 4.1.3.

4.1. Simulation environment construction

From the previous section, it is known that the model input of this study requires the vehicle velocity sequence and the distance headway sequence, and therefore, both of these data are required during the simulation. At present, the mainstream traffic simulation software includes CARLA, SUMO, CARSIM, etc., and each software has its own focused functions. In this study, the generation of highway traffic scenario and the collection of some traffic parameters are required for the simulation, and CARLA and SUMO can meet the above requirements, so CARLA and SUMO are used for the co-simulation in this study. The basic steps of CARLA-SUMO co-simulation can be found in the official document of CARLA [30], which achieves the synchronization of CARLA and SUMO, but other information need to design the independent communication function to transmission. Referring to our previous research [31], this study used UPD sockets for information transmission, and the same co-simulation architecture shown in Figure 9 is applied in this study.

Based on the co-simulation architecture shown in Figure 9 and the characteristics of the data required for this study, the functions of each software in this study as follows:

● CARLA: Perform the generation of highway scenario and vehicles, sensing of vehicle kinematic data, and visualization of the predicted trajectories for output.

● SUMO: Synchronize the traffic scenario generated by CARLA to obtain the traffic flow data that cannot be sensed by CARLA, i.e., the distance headway data sequence data in this study.

● Keras: Provide basic construction methods for trajectory prediction models, perform model training and testing, and localization, preservation of the completed training models.

● TensorFlow: Acts as a backend for Keras, providing support for the computation of vectors in the model, the output of predicted trajectories, and data regularization.

For the actual simulation, a UDP socket for sending and receiving information is also written to each process for the exchange of information between different modules, the flow of which is shown in Figure 9. After the simulation environment is constructed, the trained model is tested.

4.2. Simulation testing and results analyze

One of the contributions of this study is that no secondary training is required when porting the model to another scenario with a different coordinate system. Therefore, part road sections shown in Figure 10 of town04 in CARLA was selected as the test route for this study. In the simulation, the distance headway sequence is obtained from SUMO, the velocity sequence is obtained from CARLA, then the two data are combined and encoded into a byte stream sent to the trained model. The model calculates the predicted vehicle displacement, superimposes the displacement to get the predicted trajectory, and then sends it to CARLA for display.

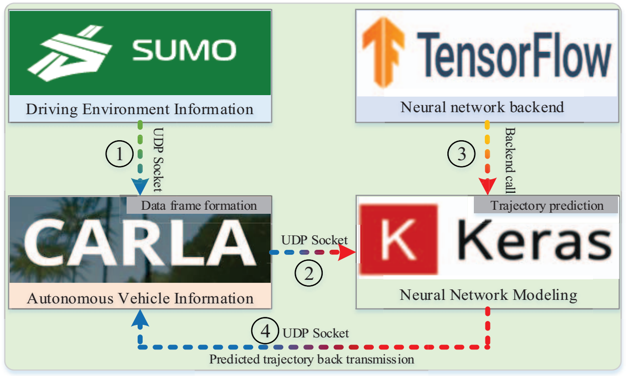


Figure 9. CARLA-SUMO-TensorFlow-Keras co-simulation environment architecture

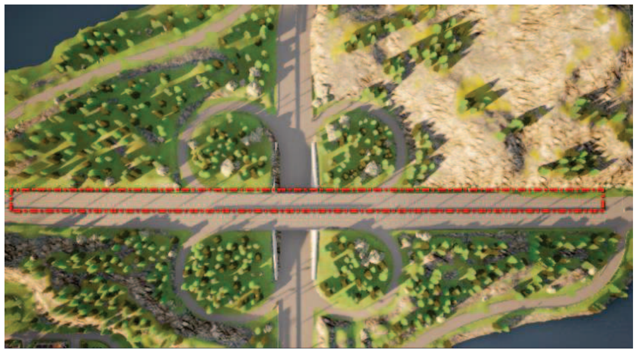


Figure 10. Selected test road section in CARLA

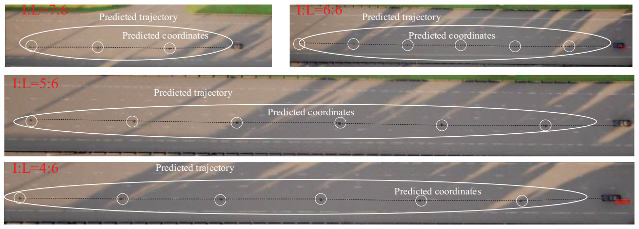


Figure 11. screenshots of predicted results

Based on the above-mentioned methods, different models were tested and part of the test screenshots are shown in Figure 11, the points in the figure are the predicted coordinates, the black line is the predicted trajectory formed by connecting the predicted coordinates. It can be seen that the predicted trajectory and the vehicle driving trajectory are compatible, the trajectory prediction model can operate normally. However, Figure 10 does not show the overall performance of the model, so the results of multiple experiments are statistically analyzed for numerical information in this study.

Based on the characteristics of the input and output data, it can be determined that the model has two key evaluation metrics, the first one is the average theoretical prospective

time (ATPT) for evaluating the prediction range and the second one is the average lateral/longitudinal prediction accuracy (ALPA) for evaluating the prediction accuracy.
In order to distinguish the difference between actual testing and theoretical performance, this study expands the evaluation metrics, where AAPTdT denotes the actual average prospective time during testing, and ALPA1 and ALPA2 denote the prediction accuracy in the lateral and longitudinal directions, respectively. The comparison results are shown in Table 2.

$$\begin{cases} E_i = \dfrac{1}{n}\sum_{i=1}^{n}\dfrac{abs(y_i - x_i)}{abs(y_i)}, \\ ALPA = (1-(\dfrac{1}{m}\sum_{j=1}^{m}E_j))*100\%. \end{cases} \tag{13}$$

In (13), y_i is the coordinates that the vehicle actually passes through, and in the sequence of actual coordinates, y_i and the predicted coordinate x_i have the closest straight-line distance. n is the number of predicted coordinates, and m is the number of predicted trajectories. It is also important to note that the values in Table 2 are derived from estimates and averages of multiple tests, and these values may change depending on the actual test conditions.

Table 2 Simulation testing results

I:L	7:6	6:6	5:6	4:6	Ave
ATPT (s)	~8.6	~7.8	~6.1	~4.3	~6.7
AAPTdT (s)	~7.5	~6.9	~5.8	~3.7	~5.9
ALPA1(%)	~91.8	~92.4	~93.1	~94.8	~93.0
ALPA2(%)	~92.3	~92.1	~94.7	~93.7	~93.2

According to the statistical results in Table 2, the model works properly with different IORs, with an average accuracy of 93.1% and an actual average look-ahead time of 5.9s.
Finally, in order to demonstrate the superiority of the proposed model in this study, a comparison of the mainstream sequence prediction methods is conducted, and the comparison results are shown in Table 3. Since the training data are the same, the ATPT of different prediction methods are the same, but their AAPTdT and ALPA have different results. In addition, in order to provide an intuitive comparison between different methods, the values in Table 3 were calculated by average under different IORs.

Table 3 Comparison of different mainstream sequence prediction methods

I:L	I	II	III	IV	Proposed
ATPT (s)	~6.7	~6.7	~6.7	~6.7	~6.7
AAPTdT (s)	~3.2	~3.4	~3.9	~4.1	~5.9
ALPA1(%)	~82.3	~85.6%	~87.5	~89.3	~93.0
ALPA2(%)	~83.7	~84.3%	~86.7	~89.9	~93.2

In Table 3, I represents the ordinary RNN structure, II and III represent the 2-layer GRU and 2-layer LSTM, respectively, and IV represents the Encoder Decoder structure based on LSTM. Compared to the other four structures, the model proposed in this study has loner actual prospective time and higher prediction accuracy.
In summary, the method proposed in this study achieves the expected results, and can be easily ported to highway scenarios with different coordinate systems, and has high prediction accuracy and prospective time, which can provide a certain research reference for autonomous driving trajectory prediction.

5. CONCLUSION

Trajectory prediction algorithms in autonomous vehicles plays a significant role in preventing vehicle collisions and improving traffic safety. However, some existing studies suffer from the problem of difficulty in portability and reuse. Based on the above motivation, we proposed a trajectory prediction model that is easily portable for collision avoidance to improve traffic safety. Specifically, this study first presented a novel data sampling algorithm that can reduce the dimensionality of high-dimensional data of different lengths to meet the model requirements. Second, a trajectory prediction neural network model based on vehicle displacement offset is developed according to the data characteristics, and the prediction model is trained by a stepwise reduction of the input data, which yields quite good theoretical training results. Finally, in order to test the actual performance of the model, the proposed model is embedded in a co-simulation environment with multiple software synchronization, and the experimental results show that the model can work properly without secondary training, and good prediction accuracy and prospective time was achieved. In summary, the model proposed in this study solves the problem that traditional models are difficult to be reused to a certain extent, which can provide some reference for autonomous driving research.
Although this study has made some progress, it still has the shortcomings of having a small amount of training data and no real vehicle testing. Therefore, in the future research, we improve the model by the following two aspects. The first is to increase the training data and test scenarios to increase the credibility, and the second is to use real-vehicle tests to improve the usability.

ACKNOWLEDGMENT

This work is supported by National Key R&D Program of China [Grant No. 2021YFB2501204], National Natural Science Foundation of China [Grant No. 52172380] and [Grant No. 52202489] as well as Shaanxi Key Industrial Innovation Chain (Cluster) - Industrial Field [Grant No. 2021ZDLGY04-06].

REFERENCES

[1] Yurtsever E, Lambert J, Carballo A, et al. A survey of

autonomous driving: Common practices and emerging technologies[J]. IEEE access, 2020, 8: 58443-58469.

[2] Wei C, Hui F, Khattak A J, et al. Controllable Probability-limited and Learning-based Human-like Vehicle Behavior and Trajectory Generation for Autonomous Driving Testing in Highway Scenario[J]. Expert Systems with Applications, 2023: 120336.

[3] Wei C, Hui F, Khattak A J, et al. Batch human-like trajectory generation for multi-motion-state NPC-vehicles in autonomous driving virtual simulation testing[J]. Physica A: Statistical Mechanics and its Applications, 2023, 616: 128628.

[4] Nelson W. Continuous-curvature paths for autonomous vehicles[C]//Proceedings, 1989 International Conference on Robotics and Automation. IEEE, 1989: 1260-1264.

[5] Vu T M, Moezzi R, Cyrus J, et al. Feasible Trajectories Generation for Autonomous Driving Vehicles[J]. Applied Sciences, 2021, 11(23): 11143.

[6] Kielas-Jensen C, Cichella V, Berry T, et al. Bernstein Polynomial-Based Method for Solving Optimal Trajectory Generation Problems[J]. Sensors, 2022, 22(5): 1869.

[7] Wang C, Zheng C Q. Lane change trajectory planning and simulation for intelligent vehicle[C]//Advanced materials research. Trans Tech Publications Ltd, 2013, 671: 2843-2846.

[8] Letizia N A, Salamat B, Tonello A M. A novel recursive smooth trajectory generation method for unmanned vehicles[J]. IEEE Transactions on Robotics, 2021, 37(5): 1792-1805.

[9] Minh V T, Pumwa J. Feasible path planning for autonomous vehicles[J]. Mathematical Problems in Engineering, 2014, 2014.

[10] Yue M, Hou X, Zhao X, et al. Robust tube-based model predictive control for lane change maneuver of tractor-trailer vehicles based on a polynomial trajectory[J]. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2018, 50(12): 5180-5188.

[11] Ye N, Zhang Y, Wang R, et al. Vehicle trajectory prediction based on Hidden Markov Model[J]. KSII Transactions on Internet and Information Systems. 2016,10(7):3150-3170.

[12] Qiao S, Shen D, Wang X, et al. A self-adaptive parameter selection trajectory prediction approach via hidden Markov models[J]. IEEE Transactions on Intelligent Transportation Systems, 2014, 16(1): 284-296.

[13] Shaojie Q, Kun J, Nan H, et al. Trajectory prediction algorithm based on gaussian mixture model[J]. Journal of software, 2015, 26(05): 1048-1063.

[14] Aoude G, Joseph J, Roy N, et al. Mobile agent trajectory prediction using Bayesian nonparametric reachability trees[M]//Infotech@ Aerospace 2011. 2011: 1512.

[15] Sun Y , Zuo W , Liu M . See the Future: A Semantic Segmentation Network Predicting Ego-Vehicle Trajectory With a Single Monocular Camera[J]. IEEE Robotics and Automation Letters, 2020, 5(2):3066-3073.

[16] Fang L, Jiang Q, Shi J, et al. TPNet: Trajectory Proposal Network for Motion Prediction[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020: 6797-6806.

[17] Eiffert S, Li K, Shan M, et al. Probabilistic crowd GAN: Multimodal pedestrian trajectory prediction using a graph vehicle-pedestrian attention network[J]. IEEE Robotics and Automation Letters, 2020, 5(4): 5026-5033.

[18] Tang C, Salakhutdinov R R. Multiple futures prediction[C]//Advances in Neural Information Processing Systems. 2019: 15424-15434.

[19] Hui F, Wei C, ShangGuan W, et al. Deep encoder–decoder-NN: A deep learning-based autonomous vehicle trajectory prediction and correction model[J]. Physica A: Statistical Mechanics and its Applications, 2022, 593: 126869.

[20] Wei C, Hui F, Yang Z, et al. Fine-grained highway autonomous vehicle lane-changing trajectory prediction based on a heuristic attention-aided encoder-decoder model[J]. Transportation Research Part C: Emerging Technologies, 2022, 140: 103706.

[21] Wei C, Hui F, Zhao X, et al. Real-time Simulation and Testing of a Neural Network-based Autonomous Vehicle Trajectory Prediction Model[C]//2022 18th International Conference on Mobility, Sensing and Networking (MSN). IEEE, 2022: 641-648.

[22] Kim B D, Kang C M, Kim J, et al. Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network[C]//2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2017: 399-404.

[23] Cui J, Zhou X, Zhu Y, et al. A road-aware neural network for multi-step vehicle trajectory prediction[C]//International Conference on Database Systems for Advanced Applications. Springer, Cham, 2018: 701-716.

[24] Xing Y, Lv C, Cao D. Personalized vehicle trajectory prediction based on joint time-series modeling for connected vehicles[J]. IEEE Transactions on Vehicular Technology, 2019, 69(2): 1341-1352.

[25] Park S H, Kim B D, Kang C M, et al. Sequence-to-sequence prediction of vehicle trajectory via LSTM encoder-decoder architecture[C]//2018 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2018: 1672-1678.

[26] Dai S, Li L, Li Z. Modeling vehicle interactions via modified LSTM models for trajectory prediction[J]. IEEE Access, 2019, 7: 38287-38296.

[27] Khakzar M, Rakotonirainy A, Bond A, et al. A dual learning model for vehicle trajectory prediction[J]. IEEE Access, 2020, 8: 21897-21908.

[28] Xiao H, Wang C, Li Z, et al. UB-LSTM: A Trajectory Prediction Method Combined with Vehicle Behavior Recognition[J]. Journal of Advanced Transportation, 2020, 2020.

[29] Krajewski R, Bock J, Kloeker L, et al. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems[C]//2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018: 2118-2125.

[30] Dosovitskiy A, Ros G, Codevilla F, et al. CARLA: An open urban driving simulator[C]//Conference on robot learning. PMLR, 2017: 1-16.

[31] C. Wei, F. Hui, X. Zhao and S. Fang, Real-time Simulation and Testing of a Neural Network-based Autonomous Vehicle Trajectory Prediction Model, 2022 18th International Conference on Mobility, Sensing and Networking (MSN), Guangzhou, China, 2022, pp. 641-648