A Deep Neural Network-based Nonlinear Dynamics Model for the Prediction of Lateral-longitudinal Vehicle Dynamics

Yongjun Pan (yongjun.pan@cqu.edu.cn)  
Chongqing University  
https://orcid.org/0000-0002-2521-5644

Xiaobo Nie  
Chongqing University

Wei Dai  
Chongqing University

Feng Xu  
Shanghai Turing Info Technology Co., Ltd.

Zhixiong Li  
Chongqing University

Research Article

Keywords: Lateral-longitudinal dynamics, Deep neural networks, Vehicle dynamics, Error functions, Multibody dynamics

DOI: https://doi.org/10.21203/rs.3.rs-629814/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
A deep neural network-based nonlinear dynamics model for the prediction of lateral-longitudinal vehicle dynamics

Xiaobo Nie · Yongjun Pan · Wei Dai · Feng Xu · Zhixiong Li

Received: date / Accepted: date

Abstract The vehicle multibody model can be used for accurate coupling dynamics, but it has higher computational complexity. Numerical stability during integration is also very challenging, especially in complicated driving situations. This issue can be substantially alleviated by using a data-driven nonlinear dynamics model owing to its computational speed and robust generalization. In this work, we propose a deep neural network (DNN)-based modeling approach for predicting lateral-longitudinal vehicle dynamics. Dynamic simulations of vehicle systems are performed based on a semirecursive multibody formulation for data acquisition. The data are then used for training and testing the DNN model. The DNN inputs are the torque applied on wheels and the initial vehicle speed that imitates a double lane change maneuver with acceleration and deceleration. The DNN outputs are the longitudinal driving distance, lateral driving distance, final longitudinal velocities, final lateral velocities, and yaw angle. The dynamic responses obtained from the DNN model are compared with the multibody results. Furthermore, the accuracy of the DNN model is investigated in terms of error functions. The DNN model is finally verified via the results of a commercial software package. The results show that the DNN vehicle dynamics model predicts accurate dynamic responses in real time. The DNN model can be used for real-time simulation and preview control in autonomous vehicles.

Keywords Lateral-longitudinal dynamics · Deep neural networks · Vehicle dynamics · Error functions · Multibody dynamics

Y. Pan (✉) · X. Nie · W. Dai
State Key Laboratory of Mechanical Transmission, College of Mechanical and Vehicle Engineering, Chongqing University, Chongqing, 400044, China
E-mail: yongjun.pan@cqu.edu.cn

F. Xu
Shanghai Turing Info Technology Co., Ltd., Shanghai, 200093, China

Z. Li
Yonsei Frontier Lab, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul 03722, Republic of Korea
1 Introduction

Neural networks can be understood as a set of algorithms, which are loosely modeled after the human brain recognizes patterns. They can interpret many kinds of data using a type of machine perception, clustering and labeling raw input. Neural networks are currently being applied to solve many complex engineering problems, and the demand is increasing with time. A wide range of applications starting from dynamics modeling to intelligent control is being handled by neural networks [4, 27, 26, 13].

Due to the advantages of neural networks, they have been widely used in lateral-longitudinal vehicle dynamics and control in recent years. Devineau et al. explored the capability of deep neural networks (DNNs) to capture key characteristics of vehicle dynamics and their ability to perform coupled lateral and longitudinal control of a vehicle [3]. Kumarawadu et al. presented a neural network adaptive model-based combined lateral and longitudinal vehicle control algorithm for highway applications. A nonlinear three-degree-of-freedom dynamic model of a passenger wagon was developed to simulate vehicle motion and for controller design [11]. Tork et al. performed simultaneous control of the lateral and longitudinal dynamics in the nonlinear system of an autonomous vehicle. The numerical results showed maximum tracking errors of 0.02 m/s for longitudinal velocity and 0.04 m for lateral path following [25]. Taghavifar et al. proposed a probabilistic estimation approach by hybridization of optimal robust control and damped least-square backpropagation neural networks. The approach can predict and control vehicle dynamics to improve vehicle handling and stability [23]. Ji et al. presented a novel lateral motion control method that consists of a robust steering controller and an adaptive neural network approximator. The simulation and experimental results suggest that the strategy can robustly track the reference path and maintain vehicle yaw stability at or near the physical limits of tire friction [8]. Melzi et al. developed a layered neural network approach that uses input dynamic parameters acquired from ordinary sensors to estimate the sideslip angle. The results show good consistency between the estimated and measured sideslip angles in different dynamic cases [14]. Kim et al. proposed a sideslip angle estimation scheme combining DNN and nonlinear Kalman filters. The algorithm was verified through both simulation and experimental results [9]. Acosta et al. combined a model predictive controller and feedforward neural networks to achieve autonomous drifting of a vehicle along a wide range of road radii [1].

The literature survey indicates that neural networks can be efficiently used in lateral-longitudinal vehicle dynamics and control. In most studies, lateral and longitudinal control problems have been analyzed in a decoupled way. However, there are strong couplings between the two dynamics [2, 15]. As a consequence, the simultaneous inclusion of lateral and longitudinal control becomes unavoidable for improving vehicle performance in a large operating range. However, the major difficulty of neural network-based methods is the number of driving situations required to build a representative training dataset. The direct estimation of vehicle states is sometimes unavailable because numerous road tests of high-performance vehicles are needed. Vehicle dynamics simulations based on commercial software packages are also time consuming since real-time simulations are still very challenging in low-cost hardware [19].
To partially address these issues, data-driven modeling approaches based on real-time data acquisition and deep neural networks provide an efficient solution. In this work, a deep neural network-based model for predicting lateral-longitudinal vehicle dynamics is presented. An efficient semirecursive multibody formulation, which performs faster than real-time simulations for data acquisition, is used to capture the key vehicle characteristics. The originality of this work lies in two contributions. The first contribution consists of the use of an efficient semirecursive multibody dynamics formulation to acquire many training and testing data. The vehicle multibody model can accurately consider and model the nonlinearities of complicated vehicle systems. The lateral-longitudinal coupled dynamics can therefore be acquired, which is much more accurate than the decoupled or simplified vehicle models. The second contribution consists of a DNN modeling approach for predicting lateral-longitudinal vehicle dynamics. Different initial velocities and applied torques over a large range are used to imitate all kinds of on-road driving situations. Furthermore, the DNN model is tested and verified by the testing set and the results obtained from a commercial software package.

The rest of the manuscript is organized as follows. In Section 2, an efficient semirecursive multibody formulation is introduced for data acquisition in real time. In Section 3, the DNN model for lateral-longitudinal dynamics of a vehicle is developed. The DNN training procedure is also provided. In Section 4, the effectiveness of the DNN model in different driving situations are investigated. The error functions are introduced to evaluate the model accuracy. In Section 5, the proposed DNN nonlinear dynamics model is verified in the framework of a commercial software package. Finally, in Section 6, we conclude our work.

2 Vehicle multibody formulation

In this work, we use an efficient semirecursive multibody method to establish a vehicle dynamics model. For open-loop multibody systems, the motion equations are formulated in terms of relative coordinates based on the Lagrange equations and the first velocity transformation, which can be described as [7, 21]:

\[ R_d^T M_T R_d \ddot{z} = R_d^T (Q - \dot{P}) \]  

where \( R_d \) denotes the first velocity transformation matrix, which can be used to express the Cartesian velocities and accelerations in terms of the relative velocities and accelerations. \( T \) denotes the path matrix representing the recursive system tree topology. \( M \) and \( Q \) denote the generalized mass matrix and external forces, respectively. \( \dot{P} \) denotes the velocity-dependent inertial forces. \( \dot{z} \) and \( \ddot{z} \) contain the relative velocities and accelerations, respectively.

For closed-loop multibody systems, a series of mechanical joints or slender rods need to be cut or removed to take advantage of the recursive system tree topology. The corresponding loop-closure constraint equations should be considered based on Eq. (1). By introducing the second velocity transformation, a subset of relative coordinates, which are called independent relative coordinates, can be abstracted to describe the relative coordinates. Therefore, the motion equations of the closed-loop multibody
system can be achieved in terms of independent relative coordinates. They can be expressed as [16]:

\[
R_z^T R_d^T \dot{M}_d^2 R_d R_z \ddot{z} = R_z^T R_d^T \left[ Q^2 - T^T \dot{M}_d \left( T R_d R_z \right) \right] dt \dot{z}
\]

(2)

where \( R_z \) denotes the second velocity transformation matrix, which can be used to express the relative velocities and accelerations in terms of independent relative velocities and accelerations. \( M_d^2 \) and \( Q^2 \) denote the accumulated generalized mass matrix and external forces, respectively. \( \dot{z} \) and \( \ddot{z} \) contain the independent relative velocities and accelerations, respectively.

This formulation is the double-step semirecursive multibody formulation, which was proposed by Javier Garcia de Jalon and his coworkers [20, 6]. Although the expression of Eq. (2) is more complicated than other multibody formulations, it leads to the smallest set of independent relative accelerations \( \ddot{z} \), which results in higher computational efficiency. Furthermore, the ordinary differential form of Eq. (2) enables the stable simulation of closed-loop multibody systems by using various numerical integrators. Low-order integration methods are very suitable for efficient simulations. However, it is important to note that the solution accuracy may be unable to satisfy the requirements as the numerical error accumulates over time. The Adams-Bashforth-Moulton and 4th-order Runge-Kutta methods are typical high-order integration methods that are often used in real-time simulations with longer simulation times [18, 17].

The training and testing data of lateral and longitudinal dynamics can be obtained in real time based on the dedicated multibody dynamics model. Thus, we collect historical data and build a vehicle lateral-longitudinal dynamics model by using a DNN. The capacity of the DNN modeling approach to capture the key vehicle characteristics is investigated based on the testing data and verified based on the results obtained from a commercial software package. The data-driven vehicle dynamics model is developed offline via DNN and used online for complex vehicle preview control. The computational burden of the data-driven vehicle model is much smaller than that of the vehicle multibody model. In addition, the data-driven model also improves the dynamics modeling robustness and vehicle preview control.

3 DNN model for lateral-longitudinal vehicle dynamics

3.1 DNN theory for vehicle dynamics modeling

Neural networks are computational technologies that resemble the human nervous system and involve the use of machine learning and efficient algorithms. A DNN is a kind of neural network with more than three structural layers but without feedback and no interconnection in each layer. It can learn and store several relationships between the input neurons and the output neurons and does not need to reveal the basic mathematical equations in advance. A general DNN is shown in Fig [22].

In the input layer, it is necessary to fix the number and structure of neurons. Input neurons should be highly reliant on the corresponding physical quantities and must be
very sensitive to the expected results. Therefore, the significant input parameters should be related to the output parameters. In the hidden layers, if a DNN with more hidden layers and hidden layers with more neurons are used, the network should have more accurate predictive performance. However, it may lead to low computational efficiency and apparent local convergence. Technically, there is no advisable method for resolving the specific network structure in each layer. Thus, it is critical to continuously test the influence of neural network prediction results with different numbers of neurons. The different network structures should be trained to select the best network. DNNs are widely used in many fields due to their advantages. They essentially realize a mapping function from the input layer to the output layer. Mathematical theory proved that DNNs can approach any nonlinear continuous function with high precision. Furthermore, the DNN model is robust and fault tolerant. If local or partial neurons of the DNN are destroyed, the overall training result will not be greatly affected.

The DNN training process is mainly composed of forward propagation of the input signal and backpropagation of errors. In forward propagation, the input matrix and the weight matrices and bias matrices are described as:

\[ Z_1 = (i_1, i_2, i_3, \ldots, i_m) \]  
\[ W_n = (w_{n1}, w_{n2}, w_{n3}, \ldots, w_{nm}) \]  
\[ B_n = (b_{n1}, b_{n2}, b_{n3}, \ldots, b_{nm}) \]

where \( m \) denotes the number of samples in the training set, \( n \) denotes the total number of hidden layers and input layer, \( Z_1 \) represents the input matrix of the model, \( W_n \) represents the weight matrix of the \( n \)-th layer, and \( B_n \) represents the bias matrix of the \( n \)-th layer. The forward propagation steps can be expressed as:

\[ Z_1 = A_1 \]  
\[ Z_{i+1} = W_i^T A_i + B_i, \quad i = 1 \ldots n \]  
\[ A_{i+1} = f^{i+1}(Z_{i+1}), \quad i = 1 \ldots n \]
where \( Z_i \) represents the input of the \( i \)-th layer, \( A_i \) represents the output of the \( i \)-th layer, and \( f \) denotes the \( i \)-th activation function. Note that \( A_{n+1} \) denotes the results of the DNN model. There are many types of activation functions that can help the DNN model fit the linear and nonlinear functions and to classify them.

It is important to select an appropriate loss function to measure the differences between the sample results and the DNN results before backpropagation. By means of backpropagation, the weight matrices and bias matrices can be updated to minimize the loss function. In general, the accuracy of the DNN model increases as the value of the loss function decreases. The mean square error (MSE) is the most widely used loss function for prediction. Note that when fitting the nonlinear function, it is easy to cause overfitting. To avoid this issue, regularization is normally used to enhance the generalization performance of the DNN model. The common generalization methods include L1 regularization, L2 regularization, and dropout regularization. In this work, L2 regularization is used to compute the loss function in terms of MSE to avoid overfitting. The MSE, L2 regularization and the loss function can be expressed as (13):

\[
MSE = \frac{1}{2m} \| Y - A_{n+1} \|^2 \\
L2 = \frac{\lambda \Sigma \| W_n \|^2}{2m} \\
e = MSE + L2 = \frac{1}{2m} \| Y - A_{n+1} \|^2 + \frac{\lambda \Sigma \| W_n \|^2}{2m}
\]

where \( e \) denotes the network’s cost, \( Y \) denotes the value of samples, and \( \lambda \) denotes the regularization coefficient.

Subsequently, the adaptive moment estimation optimization (Adam) algorithm is introduced in the backpropagation process [10]. It is one of the most efficient algorithms for accelerating convergence in deep learning and can be regarded as a combination of the adaptive gradient (AdaGrad) algorithm and root mean square prop (RMSProp) algorithm [5, 24]. The Adam algorithm is more appropriate for backpropagation than other algorithms because it has a higher computational efficiency in DNN training and does not require much memory. Furthermore, the Adam algorithm is good at handling sparse gradients and nonstationary targets. Hyperparameters of the Adam algorithm have intuitive explanations and require few adjustments. The key parameters of AdaGrad and RMSProp can be expressed as:

\[
v_{dW_n} = \beta_1 v_{dW_n} + (1 - \beta_1) dW_n \\
v_{dB_n} = \beta_1 v_{dB_n} + (1 - \beta_1) dB_n \\
s_{dW_n} = \beta_2 s_{dW_n} + (1 - \beta_2) dW_n^2 \\
s_{dB_n} = \beta_2 s_{dB_n} + (1 - \beta_2) dB_n^2
\]

where \( dW_n \) and \( dB_n \) denote the partial differential of the weight matrices and the bias matrices with respect to the loss function. \( v_{dW_n}, v_{dB_n}, s_{dW_n}, \) and \( s_{dB_n} \) represent the AdaGrad and RMSProp parameters in each training iteration for weight matrices and bias matrices, respectively. Note that the values of these parameters are 0 at the first
training iteration. The values of hyperparameters $\beta_1$ and $\beta_2$, in general, are 0.9 and 0.999.

Since the moving exponential weighted average results in relatively large errors to the initial value at the beginning of the iteration, the AdaGrad and RMSProp parameters obtained above need to be corrected as:

\[
v_{c_{\text{dwn}}} = \frac{v_{\text{dwn}}}{1 - \beta_1} \quad (16)
\]

\[
v_{c_{\text{dbn}}} = \frac{v_{\text{dbn}}}{1 - \beta_1} \quad (17)
\]

\[
s_{c_{\text{dwn}}} = \frac{s_{\text{dwn}}}{1 - \beta_2} \quad (18)
\]

\[
s_{c_{\text{dbn}}} = \frac{s_{\text{dbn}}}{1 - \beta_2} \quad (19)
\]

where these parameters are corrected in each training iteration. The weight matrices and bias matrices, in turn, are updated according to the Adam algorithm:

\[
W_n = W_n - \alpha \frac{v_{c_{\text{dwn}}}}{\sqrt{s_{c_{\text{dwn}}} + \epsilon}} \quad (20)
\]

\[
B_n = B_n - \alpha \frac{v_{c_{\text{dbn}}}}{\sqrt{s_{c_{\text{dbn}}} + \epsilon}} \quad (21)
\]

where $\epsilon$ denotes a smoothing parameter, which equals 10E-08 in this work. $\alpha$ represents the learning rate, which needs to be minutely adjusted in the deep learning process.

### 3.2 Lateral-longitudinal vehicle dynamics modeling

In this work, we use the semirecursive multibody formulation presented in the previous section to model the vehicle dynamics to obtain a lateral-longitudinal dynamics dataset. This vehicle system consists of McPherson suspensions in the front, multilink suspensions in the rear, and Pacejka tire models. The vehicle structure diagram is described in Fig. 2. Additional vehicle parameters are described in Table 1.

| Parameter                          | Value   | Parameter                          | Value   |
|-----------------------------------|---------|-----------------------------------|---------|
| Vehicle mass                      | 1155 kg | Tire rolling radius               | 0.4673 m|
| Distance from centroid to front axle | 0.7209 m| Centroid height                   | 0.5373 m|
| Distance from centroid to rear axle | 2.0791 m| Wheelbase                         | 2.8 m   |
| Stiffness of front absorber       | 40000 N/m| Damping of front absorber         | 1800 N/(m/s)|
| Stiffness of rear absorber        | 35000 N/m| Damping of rear absorber          | 1800 N/(m/s)|

In dynamics simulations, vehicles move with various initial longitudinal speeds and front-wheel driving torques to imitate different driving situations. The simulation
time is 5 s. The steering angle is defined to ensure the double lane change maneuver, as shown in Fig. 3. We randomly obtain several critical vehicle responses by introducing 500 different initial longitudinal speeds and front-wheel driving torques at a defined steering angle. Those responses, which include longitudinal driving distance, lateral driving distance, final longitudinal speed, final lateral speed, and vehicle yaw angle, represent the lateral-longitudinal dynamics of vehicles. Note that the range of the initial longitudinal speed is 15 m/s to 45 m/s, and the range of driving torque is -500 Nm to 500 Nm.

We randomly divide these 500 samples into a training set and testing set with 450 and 50 samples in the training set and testing set, respectively. The training set is used for training the DNN model, and the testing set is used to evaluate the trained neural network. Furthermore, the training data were processed to eliminate the inconvenience caused by the magnitude of values. It can avoid very small weight matrices that may cause stability problems during numerical integration. In this work, we choose Z-score
standardization for data processing, and the algorithm is mathematically expressed as:

\[ x^* = \frac{x - \mu}{\sigma} \]  

(22)

where \( x \) denotes the value of the samples, \( \mu \) denotes the mean of the samples, and \( \sigma \) denotes the standard deviation of the samples. \( x^* \) represents the standardization value. The mean value of the processed data is 0, and the standard deviation is 1. The calculation results of the outputs should be reversed standardized later for DNN training.

There are two approaches to updating the optimization algorithm parameters in deep learning. The first is batch gradient descent, which calculates all the dataset samples. Thus, the number of calculations is large, and the calculation speed is slow. The other approach is stochastic gradient descent, which calculates the loss function of each data point and finds the updated parameters. The calculation speed of this approach is faster, but the convergence performance is not very good. To overcome the drawbacks of these two approaches, minibatch gradient descent is used. This approach divides the data into several batches and updates the parameters according to the batches. Thus, the data in a batch jointly determines the direction of this gradient, and it will not vanish easily when falling and will reduce the randomness. In this work, the number of calculations is small; thus, the batch size is set to 32.

After data training, we finally built the DNN model for predicting lateral-longitudinal vehicle dynamics. The DNN modeling procedure is shown in Fig. 4. The inputs of the DNN are the initial longitudinal speed and driving torque of the vehicle, and the outputs are the longitudinal driving distance, lateral driving distance, final longitudinal speed, final lateral speed, and vehicle yaw angle. The DNN model has 4 hidden layers, and each hidden layer involves 28, 24, 20, and 15 neurons. The last activation function is linear, and other activation functions are rectified linear unit (ReLU) functions. The ReLU function is expressed as:

\[ f(x) = \max(0, x) \]  

(23)

The ReLU function can effectively avoid the gradual disappearance and explosion of the gradient. The concise calculation formula can reduce the calculation cost. These
merits ensure that ReLU is one of the most far-ranging activation functions in deep learning. After the neural network training is completed, the testing set needs to be used for DNN model verification. The MSE of the dataset also needs to be calculated. The trained DNN is qualified and can be used for the prediction if the MSE is very small. The procedure of DNN training for vehicle lateral-longitudinal dynamics modeling is described in Fig. 5.

![Fig. 5 The procedure of DNN training.](image)

### 4 DNN model results

To evaluate the accuracy of the trained DNN model, we compare the predicted results of the DNN model and the simulated results of the multibody model. Note that the vehicle initial longitudinal speeds of the DNN model and multibody model both range from 15 m/s to 45 m/s, and the front-wheel driving torque ranges from -500 Nm to 500 Nm. Fig. 6 shows the differences between the DNN model results and the multibody model results. The comparative responses are longitudinal driving distance, lateral driving distance, final longitudinal velocities, final lateral velocities, and yaw angle. Note that for better visualization, only part of the predicted results is included in the figures.

Fig. 7 to 11 show the absolute percentage error between the DNN model results and the multibody model results for these five vehicle responses, respectively.
According to the results comparison and absolute percentage error analysis, the following conclusions can be drawn.

Fig. 6 The comparative results.

initial speeds and driving torques in a wide range are used to imitate various driving situations.
Fig. 7 Box plot of absolute percentage error: final longitudinal distance.

Fig. 8 Box plot of percentage absolute error: final lateral distance.

Fig. 9 Box plot of absolute percentage error: final longitudinal velocities.
Fig. 10 Box plot of absolute percentage error: final lateral velocities.

Fig. 11 Box plot of absolute percentage error: yaw angle.

- The DNN prediction of longitudinal driving distance and final longitudinal velocities fit the multibody model results well. Less than a 5% absolute percentage error is observed for the DNN model.
- The lateral driving distance, final lateral velocity and yaw angle obtained from the DNN nonlinear dynamics model also fit the reference results (multibody model results) well.

The DNN model can be trained based on a large quantity of lateral and longitudinal data offline and used for the prediction of lateral-longitudinal dynamics later. Furthermore, because the DNN model completely omits the tedious process of numerical integration, it is especially suitable for the preview control of autonomous vehicles.
5 DNN model verification

5.1 Model accuracy and efficiency

In this section, we evaluate the accuracy of the proposed DNN model quantifiably by error functions. The computational efficiency of the DNN model is also investigated. The accuracy evaluation metrics comprise the maximum absolute error (ME), the mean absolute error (MAE), the mean absolute percentage error (MAPE), the root mean square error (RMSE) and \( R^2 \). The ME denotes the maximum absolute difference between the simulated results and the predicted results. The MAE denotes the average numerical value of the differences between the ground truth and predicted values. The MAPE denotes the homologous percentage differences. The RMSE denotes the root mean square of the differences between the simulated results and the predicted results. \( R^2 \) denotes the coefficient of determination, a statistical indicator used to reflect the regression model to illustrate the reliability of the dependent variable change. The expressions of these five error functions are written as:

\[
\text{ME} = \max_{1 \leq i \leq n} \{|y_i - \hat{y}_i|\} \tag{24}
\]

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{25}
\]

\[
\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{26}
\]

\[
\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{27}
\]

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y}_i)^2} \tag{28}
\]

where, \( y_i \) represents the \( i \)-th value of the simulated results, \( \hat{y}_i \) represents the \( i \)-th value of the predicted results, \( \bar{y}_i \) represents the \( i \)-th mean value of the predicted results, \( n \) represents the number of groups, and \( n \) equals 7701. These five evaluation metrics are used to quantify the accuracy of the DNN model. The smaller the first four metrics are, the more accurate the DNN model is. However, for \( R^2 \), if its value is closer to 1, the DNN model fits well.

The longitudinal driving distance, lateral driving distance, final longitudinal speed, final lateral speed and yaw angle predicted by the proposed DNN model are analyzed based on the evaluation metrics. The evaluation results are described in Table 2.
Table 2 The accuracy parameters of the DNN model (driving distance).

| Vehicle responses | ME (m/m/s&deg;) | MAE (m/m/s&deg;) | MAPE (%) | RMSE (m/m/s&deg;) | R²         |
|-------------------|------------------|-------------------|----------|-------------------|-----------|
| Longitudinal distance | 0.692661        | 0.062761          | 0.049755 | 0.079885          | 0.999997  |
| Lateral distance   | 0.134750         | 0.001898          | 2.006429 | 0.003909          | 0.999534  |
| Longitudinal speed | 0.346306         | 0.021783          | 0.068241 | 0.031725          | 0.999987  |
| Lateral speed      | 0.048930         | 0.000393          | 2.164173 | 0.001221          | 0.998392  |
| Yaw angle          | 0.001222         | 0.000016          | 2.531303 | 0.000034          | 0.998917  |

It is easy to observe in Table 2 that the ME, MAE, MAPE, and RMSE of the lateral-longitudinal responses are relatively small, and $R^2$ is very close to 1. This indicates that the DNN model has very high accuracy in predicting lateral-longitudinal dynamics. In addition, the longitudinal prediction is slightly more accurate than the lateral prediction.

To investigate the computational efficiency of the proposed DNN model, we compare the CPU time and MAPE of deep neural networks with different numbers of hidden layers. The results are described in Tables 3 and 4. It is important to note that the CPU time is the average time of ten random driving situations, and MAPE is measured based on 7,701 sets of data.

Table 3 Comparison of different DNN structures (CPU time).

| Hidden layers | Longitudinal distance | Lateral distance | Longitudinal speed | Lateral speed | Yaw angle |
|--------------|-----------------------|------------------|-------------------|--------------|----------|
| 4            | 1.0059                | 1.0025           | 1.0053            | 1.0092       | 0.9938   |
| 3            | 0.9981                | 1.0029           | 0.9960            | 0.9922       | 1.0043   |
| 2            | 1.0057                | 0.9964           | 1.0026            | 1.0000       | 1.0069   |

Table 4 Comparison of different DNN structures (MAPE).

| Hidden layers | Longitudinal distance | Lateral distance | Longitudinal speed | Lateral speed | Yaw angle |
|--------------|-----------------------|------------------|-------------------|--------------|----------|
| 4            | 0.0498                | 2.0034           | 0.0682            | 2.1642       | 2.5313   |
| 3            | 0.0530                | 7.1222           | 0.0731            | 4.9764       | 5.1395   |
| 2            | 0.0785                | 9.3461           | 0.0505            | 7.7543       | 6.8048   |

It is obvious in Tables 3 and 4 that the CPU time for lateral-longitudinal dynamics prediction is approximately 1 ms with 2, 3, and 4 hidden layers. The computational efficiency remains the same in these cases. However, in the case of 2 or 3 hidden layers, the MAPE is relatively large, and the predicted results are less accurate than the proposed DNN model.
5.2 Model verification

We use the commercial software package CarSim to further verify the effectiveness of the proposed DNN model. It is important to note that dynamic simulations with the same vehicle parameters are performed in the CarSim framework. Due to limited space, we only plot the comparative results of longitudinal distances, lateral distances and longitudinal velocities under the conditions of 300 Nm and -300 Nm driving torques. Figs. 12 to 14 depict the result comparison when the 300 Nm and -300 Nm driving torques are imposed respectively. The detailed view in each figure describes the maximum difference between the DNN result and the commercial software result.

Fig. 12 Comparison of longitudinal distances.

Fig. 13 Comparison of lateral distances.
As presented in Figs. 12 to 14, the predicted results via deep neural networks coincide with the results of the CarSim software package. The results also are consistent with our earlier multibody model results. The driving situation imitates that the vehicle accelerates and decelerates during a double lane change maneuver with a wide range of initial speeds. Finally, the effectiveness and robustness of the proposed DNN model for vehicle lateral-longitudinal dynamics are verified. It can be used for the preview control of autonomous vehicles.

6 Conclusion

In this work, a DNN-based approach for vehicle lateral-longitudinal dynamics was proposed. A semirecursive multibody dynamics formulation is used for vehicle modeling and data acquisition. An algorithm for obtaining lateral-longitudinal dynamics data and a DNN modeling procedure were developed. It is significant that massive parallel computations can be performed by using GPUs that are especially suitable for large volumes of data.

To verify the proposed DNN model, the testing data were randomly selected for accuracy evaluation. Error functions were introduced to quantifiably assess the model accuracy. The computational efficiency and solution accuracy were investigated for neural networks with three different structures. Finally, results from a commercial software package were obtained to further verify the effectiveness of the proposed DNN model. The results showed that the DNN model can predict the vehicle lateral-longitudinal dynamics accurately and efficiently. The DNN model used for modeling lateral, longitudinal and vertical coupling dynamics will be investigated in the future. Overall, a DNN-based approach was presented to predict the vehicle’s lateral-longitudinal dynamics in real time, which is suitable for the preview control of autonomous vehicles.
Conflict of Interest

The authors declare that they have no conflict of interest.

Funding

This study was funded by the National Natural Science Foundation of China (grant number 12072050).

Data Availability Statement

The data that support the findings of this study are not openly available right now, and are available from the corresponding author upon reasonable request.

References

1. Acosta, M., Kanarachos, S.: Teaching a vehicle to autonomously drift: A data-based approach using neural networks. Knowledge-Based Systems 153, 12–28 (2018)
2. Attia, R., Orjuela, R., Basset, M.: Combined longitudinal and lateral control for automated vehicle guidance. Vehicle System Dynamics 52(2), 261–279 (2014)
3. Devineau, G., Polack, P., Alchétè, F., Moutarde, F.: Coupled longitudinal and lateral control of a vehicle using deep learning. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 642–649. IEEE (2018)
4. Duan, K., Fong, S., Chen, C.P.: Multilayer neural networks-based control of underwater vehicles with uncertain dynamics and disturbances. Nonlinear Dynamics 100(4), 3555–3573 (2020)
5. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of machine learning research 12(7) (2011)
6. García de Jalón, J., Álvarez, E., de Ribera, F., Rodríguez, I., Funes, F.: A fast and simple semi-recursive formulation for multi-rigid-body systems. In: J. Ambrosio (ed.) Advances in Computational Multibody Systems, Computational Methods in Applied Sciences, vol. 2, Chapter 1, pp. 1–23. Springer Netherlands (2005)
7. Jerkovsky, W.: The structure of multibody dynamic equations. Journal of Guidance, Control and Dynamics 1(3), 173–182 (1978)
8. Ji, X., He, X., Lv, C., Liu, Y., Wu, J.: Adaptive-neural-network-based robust lateral motion control for autonomous vehicle at driving limits. Control Engineering Practice 76, 41–53 (2018)
9. Kim, D., Min, K., Kim, H., Huh, K.: Vehicle sideslip angle estimation using deep ensemble-based adaptive Kalman filter. Mechanical Systems and Signal Processing 144, 106,862 (2020)
10. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Kumarawadu, S., Lee, T.T.: Neuroadaptive combined lateral and longitudinal control of highway vehicles using RBF networks. IEEE Transactions on Intelligent Transportation Systems 7(4), 500–512 (2006)
12. Liu, Y., Yang, D., Zhang, C.: Relaxed conditions for convergence analysis of online back-propagation algorithm with 1.2 regularizer for Sigma-Pi-Sigma neural network. Neurocomputing 272, 163–169 (2018)
13. Luo, C., Lei, H., Li, J., Zhou, C.: A new adaptive neural control scheme for hypersonic vehicle with actuators multiple constraints. Nonlinear Dynamics 100, 3529–3533 (2020)
14. Melzi, S., Sabbioni, E.: On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results. Mechanical Systems and Signal Processing 25(6), 2005–2019 (2011)
15. Nguyen, A.T., Dinh, T.Q., Guerra, T.M., Pan, J.: Takagi-Sugeno fuzzy unknown input observers to estimate nonlinear dynamics of autonomous ground vehicles: Theory and real-time verification. IEEE/ASME Transactions on Mechatronics pp. 1–1 (2021)
16. Pan, Y., Callejo, A., Bueno, J.L., Wehage, R.A., García de Jalón, J.: Efficient and accurate modeling of rigid rods. Multibody System Dynamics 40(1), 23–42 (2017)
17. Pan, Y., Dai, W., Xiong, Y., Xiang, S., Mikkola, A.: Tree-topology-oriented modeling for the real-time simulation of sedan vehicle dynamics using independent coordinates and the rod-removal technique. Mechanism and Machine Theory 143, 103,626 (2020)
18. Pan, Y., He, Y., Mikkola, A.: Accurate real-time truck simulation via semirecursive formulation and Adams–Bashforth–Moulton algorithm. Acta Mechanica Sinica 35(3), 641–652 (2019)
19. Pan, Y., Nie, X., Li, Z., Gu, S.: Data-driven vehicle modeling of longitudinal dynamics based on a multibody model and deep neural networks. Measurement 180, 109,541 (2021)
20. Rodríguez, J.L., Jiménez, J.M., Funes, F.J., García de Jalón, J.: Recursive and residual algorithms for the efficient numerical integration of multi-body systems. Multibody System Dynamics 11(4), 295–320 (2004)
21. von Schwerin, R.: Multibody System Simulation, Numerical Methods, Algorithms and Software. Springer (1999)
22. Seo, J., Yun, B., Park, J., Park, J., Shin, M., Park, S.: Prediction of instantaneous real-world emissions from diesel light-duty vehicles based on an integrated artificial neural network and vehicle dynamics model. Science of The Total Environment 786, 147,359 (2021)
23. Taghavifar, H., Hu, C., Taghavifar, L., Qin, Y., Na, J., Wei, C.: Optimal robust control of vehicle lateral stability using damped least-square backpropagation training of neural networks. Neurocomputing 384, 256–267 (2020)
24. Tieleman, T., Hinton, G.: Lecture 6.5-rmsprop, coursera: Neural networks for machine learning. University of Toronto, Technical Report (2012)
25. Tork, N., Amirkhani, A., Shokouhi, S.B.: An adaptive modified neural lateral-longitudinal control system for path following of autonomous vehicles. Engineering Science and Technology, an International Journal 24(1), 126–137 (2021)
26. Yong, H., Seo, J., Kim, J., Choi, J.: State reconstruction in a nonlinear vehicle suspension system using deep neural networks. Nonlinear Dynamics pp. 1–17 (2021)
27. Zerari, N., Chemachema, M.: Event-triggered adaptive output-feedback neural-networks control for saturated strict-feedback nonlinear systems in the presence of external disturbance. Nonlinear Dynamics 104(2), 1343–1362 (2021)