



Article

An Improved Deep Neural Network Model of Intelligent Vehicle Dynamics via Linear Decreasing Weight Particle Swarm and Invasive Weed Optimization Algorithms

Xiaobo Nie ¹, Chuan Min ¹, Yongjun Pan ^{1,2,*} , Zhixiong Li ³ and Grzegorz Królczyk ³ 

¹ College of Mechanical and Vehicle Engineering, Chongqing University, Chongqing 400044, China; xiaobo.nie@cqu.edu.cn (X.N.); chuan.min@cqu.edu.cn (C.M.)

² State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian 116024, China

³ Faculty of Mechanical Engineering, Opole University of Technology, 45758 Opole, Poland; z.li@po.edu.pl (Z.L.); g.krolczyk@po.opole.pl (G.K.)

* Correspondence: yongjun.pan@cqu.edu.cn

Abstract: We propose an improved DNN modeling method based on two optimization algorithms, namely the linear decreasing weight particle swarm optimization (LDWPSO) algorithm and invasive weed optimization (IWO) algorithm, for predicting vehicle's longitudinal-lateral responses. The proposed improved method can restrain the solutions of weight matrices and bias matrices from falling into a local optimum while training the DNN model. First, dynamic simulations for a vehicle are performed based on an efficient semirecursive multibody model for real-time data acquisition. Next, the vehicle data are processed and used to train and test the improved DNN model. The vehicle responses, which are obtained from the LDWPSO-DNN and IWO-DNN models, are compared with the DNN and multibody results. The comparative results show that the LDWPSO-DNN and IWO-DNN models predict accurate longitudinal-lateral responses in real-time without falling into a local optimum. The improved DNN model based on optimization algorithms can be employed for real-time simulation and preview control in intelligent vehicles.

Keywords: longitudinal-lateral dynamics; vehicle multibody model; deep neural networks; particle swarm optimization; invasive weed optimization



Citation: Nie, X.; Min, C.; Pan, Y.; Li, Z.; Królczyk, G. An Improved Deep Neural Network Model of Intelligent Vehicle Dynamics via Linear Decreasing Weight Particle Swarm and Invasive Weed Optimization Algorithms. *Sensors* **2022**, *22*, 4676. <https://doi.org/10.3390/s22134676>

Academic Editor: Felipe Jiménez

Received: 23 May 2022

Accepted: 17 June 2022

Published: 21 June 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The modeling of complex multibody system for vehicle dynamics and control is always a challenging task. First, the dynamic tire forces and vehicle-tire-road interactions are difficult to calculate, partially due to external disturbances and parameter perturbation [1–3]. Second, the multibody-based formulations lead to full-vehicle models with increased computational burden, which always makes the real-time simulation and control unavailable [4,5]. Furthermore, the numerical stability is also a challenging issue to consider for cases where an off-road vehicle passes through abnormal road surfaces [6–8]. Correspondingly, to overcome these issues, deep neural networks (DNNs) based on historical data are widely used for lateral dynamics and longitudinal-lateral dynamics modeling [9–11]. The DNN-based vehicle models are able to deal with complexity through automation. They are used to simulate various scenarios, collect a large amount of data, verify it in an automated simulation environment, and then test it on real vehicles. In this way, intelligent vehicles are constantly learning to deal with complex and realistic scenarios [12].

Theoretically, neural networks can approximate nonlinear continuous functions under the reasonable structure and appropriate weights. Hence, they are suitable for solving various inherent complex problems and can adapt to complex nonlinear mapping. Deep neural

networks, on the other hand, can be regarded as a more complex form of neural networks. They possess many unmatched features compared to traditional statistical methods and algorithms, e.g., self-adaptation, self-organization, self-learning, high non-linearity, and good fault tolerance [13]. In one work, Rutherford et al. investigated the nonlinear modeling ability of neural networks to identify and control the dynamic systems [14]. In another related work, Kim et al. developed a scheme for sideslip angle estimation, which combines DNNs and Kalman filters for accurate prediction [15]. Similarly, Devineau et al. explored the capabilities of DNNs to perform lateral and longitudinal modeling of a vehicle [16]. Melzi et al. developed a layered neural-network method using dynamic parameters obtained from vehicle sensors to accurately estimate the sideslip angle [17]. Ji et al. developed a novel lateral motion control approach consisting of a robust steering controller and an adaptive neural network approximator [18]. Progressively, Acosta et al. used feed-forward neural networks and a model predictive controller for autonomous vehicle drifting along a large range of road radii [19]. Lio et al., in one recent work, investigated neural networks of different architectures for modeling longitudinal dynamics of a medium-scale vehicle [20].

Conclusively, the literature survey indicates that DNN-based approaches can be effectively used for vehicle dynamics modeling. The main difficulty for DNN methods, however, lies in the required number of dynamics maneuvers to build a representative dataset. Additionally, the direct estimation of vehicle characteristics is sometimes unavailable because numerous vehicle tests are needed for it. Vehicle dynamics maneuvers based on the commercial software, such as ADAMS and RecurDyn, are also time consuming [21–23]. To address this issue, data-driven modeling methods based on efficient data acquisition and DNNs provide an effective solution. Accordingly, in this study, a DNN-based dynamics model for vehicle's longitudinal-lateral dynamics is presented. Furthermore, an efficient semirecursive multibody formulation, which performs real-time simulations for data acquisition, is used to capture the vehicle key characteristics.

Activation functions are used to determine nonlinear properties in DNNs. Generally, they are nonlinear functions, such as the tanh and sigmoid functions. The use of rectified linear function (ReLU) has increased recently. Moreover, to calculate and update the weight matrices of DNNs, a backpropagation approach is widely used as a de-facto standard algorithm for improved recognition performance while training DNNs [24]. However, the backpropagation approach frequently requires a longer time to converge. Besides, the solutions of weight matrices and bias matrices tend to fall into the local optimum. To address this issue, two optimization methods, called as linear decreasing weight particle swarm optimization (LDWPSO) and invasive weed optimization (IWO), are introduced in this study to calculate the weight and bias matrices. The introduced optimization algorithms can improve the training results of the DNNs with faster convergence.

The highlights of this work lie in three phases. First, an efficient semirecursive multibody dynamics formulation is implemented for a vehicle system to acquire training data. The semirecursive vehicle dynamics model can accurately take into account the nonlinearities of a vehicle system. Therefore, the vehicle's longitudinal-lateral dynamics can be obtained. The data in this study are more accurate compared to the results obtained from decoupled or simplified models. Second, a DNN modeling method for vehicle's longitudinal-lateral dynamics is developed based on the training data. Most importantly, LDWPSO and IWO algorithms are introduced to improve the robustness and accuracy of the DNN model [25–27]. Various applied torques and initial velocities, spanning over a large range, are used to imitate diverse driving situations (accelerating and decelerating). Lastly, the obtained numerical results are validated in details.

The rest of the work is organized as follows. In Section 2, we present an efficient semirecursive multibody formulation for real-time data acquisition. Furthermore, a vehicle dynamics modeling method via DNNs and obtained data is developed. In Section 3, we introduce LDWPSO and IWO algorithms relying on DNN vehicle model, to improve the training results. In Section 4, we analyze the effectiveness and accuracy of the proposed DNN model for different driving situations. Finally, in Section 5, we conclude our work.

2. DNN Modeling for Vehicle's Longitudinal-Lateral Dynamics

2.1. Vehicle Dynamics Data Acquisition

In this study, we used an accurate and efficient semirecursive multibody formulation to model the vehicle dynamics and acquire the longitudinal-lateral characteristics. This semirecursive formulation was first proposed by García de Jalón et al. [28–30]. Based on this approach, the equations of motion for a vehicle multibody system can be concisely expressed as:

$$\mathbf{R}_z^T \mathbf{R}_d^T \mathbf{M}^\Sigma \mathbf{R}_d \mathbf{R}_z \ddot{\mathbf{z}}^i = \mathbf{R}_z^T \mathbf{R}_d^T \left[\mathbf{Q}^\Sigma - \mathbf{T}^T \bar{\mathbf{M}} \frac{d(\mathbf{T} \mathbf{R}_d \mathbf{R}_z)}{dt} \dot{\mathbf{z}}^i \right] \quad (1)$$

where, \mathbf{R}_d represents the first velocity transformation matrix that can be used to express the Cartesian velocities and accelerations via relative velocities and accelerations. \mathbf{R}_z represents the second velocity transformation matrix, which can be utilized to describe relative velocities and accelerations via independent relative velocities and accelerations. \mathbf{T} describes the path matrix of the system, and can express the recursive system tree-topology. $\bar{\mathbf{M}}$ corresponds to generalized mass matrix of the system. \mathbf{M}^Σ and \mathbf{Q}^Σ refer to the accumulated generalized mass matrix and external forces, respectively. Lastly, $\dot{\mathbf{z}}^i$ and $\ddot{\mathbf{z}}^i$ denote the independent relative velocities and accelerations, respectively. See [30] for more details.

We can see that the motion expression (Equation (1)) is more complicated than other multibody formulations reported in [31]. However, this expression yields a small set of independent relative accelerations $\ddot{\mathbf{z}}^i$, which in turn results in a higher computational efficiency. Therefore, the equations of motion (Equation (1)) can be used for real-time simulation in low-cost hardware [32–34]. On the other hand, to mitigate the issues related to unavailability of direct estimation of vehicle characteristics and high computational costs linked to commercial software based vehicle dynamics simulations, this efficient multibody formulation is introduced for vehicle simulation and data acquisition.

The vehicle system investigated in this study consists of Pacejka tire models, five-link suspensions in the rear, and McPherson suspensions in the front. Likewise, a schematic diagram of the vehicle multibody system is illustrated in Figure 1. Additional information related to the vehicle system is listed in Table 1. The vehicle multibody model has 34 relative (joint) coordinates in total, where 14 of them are independent of each other and can be utilized to describe the kinematics and dynamics of the full-vehicle model.

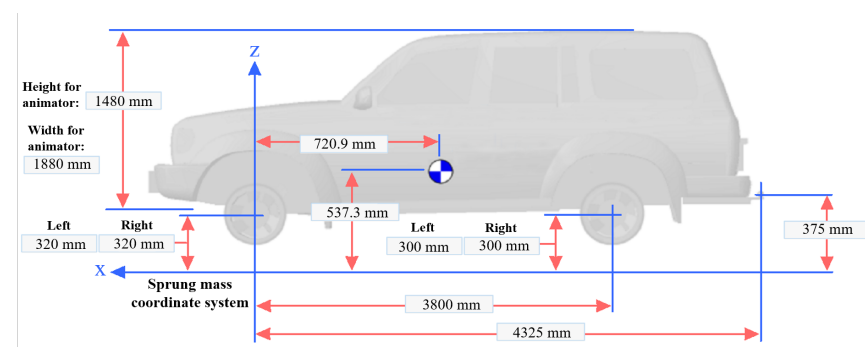
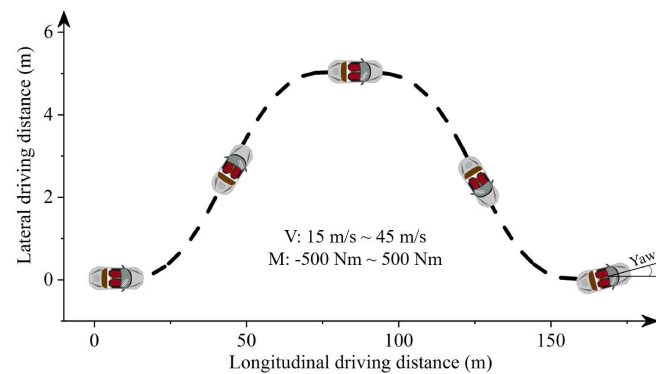


Figure 1. Vehicle system structure.

Table 1. Critical parameters of the vehicle model.

Parameter	Value
Vehicle mass	1155 kg
Wheelbase	2.8 m
Centroid height	0.5373 m
Tire rolling radius	0.4673 kg
Stiffness of front absorber	40,000 N/m
Stiffness of rear absorber	35,000 N/m
Damping of front absorber	1800 N/(m/s)
Damping of rear absorber	1800 N/(m/s)
Distance from centroid to front axle	0.7209 m
Distance from centroid to rear axle	2.0791 m

In the process of dynamic simulation and data acquisition, the vehicle is driven by various torques applied on front wheels and initial speeds. The driving torques range from -500 Nm to 500 Nm, to imitate the deceleration and acceleration conditions comprehensively. The initial speeds of the vehicle are set in a range spanning between 15 m/s and 45 m/s. The vehicle moves in a double lane change maneuver, as described in Figure 2. The vehicle simulation lasts for 5 s with time-step of 1 ms. Subsequently, 500 longitudinal-lateral dynamics datasets are collected. Each dataset includes vehicle initial speed, driving torque, longitudinal and lateral driving distances, final longitudinal and lateral velocities, and vehicle yaw angle.

**Figure 2.** The driving track of the vehicle.

The collected 500 datasets are randomly divided into the training and testing sets, containing 450 and 50 datasets, respectively. The training set is used to develop the DNN model, while the testing set is employed to evaluate the effectiveness of DNN model. In addition, all sample data is standardized during the data processing. Accordingly, the original data is transformed into non-dimensional index evaluation values via standardization. It prevents higher values from weakening the effects of lower values, which is essential to balance the contribution of each feature. Furthermore, standardization can also speed up the gradient descent to find optimal solutions. In this study, Z-score standardization was used for the data processing, which can be mathematically expressed as [35]:

$$x^* = \frac{x - \mu}{\sigma} \quad (2)$$

where, x represents raw data of samples, μ represents the mean value of raw data, and σ represents the standard deviation of raw data. The term x^* represents the processed data, which constitutes the input values for DNN model. After standardization, the standard deviation of processed data is 1 with an average value of 0 . Additionally, reversed standardization is performed on the output values of DNN model.

2.2. DNN Model of the Vehicle

The principle of DNNs is briefly demonstrated in Figure 3. In DNN process, the relationship between training data is mapped to the output layer. By continuously adjusting the values of weight matrix and bias matrix, errors between the output results and expected values are reduced and controlled. As shown in Figure 3, the general DNNs mainly include an input layer, several hidden layers, and an output layer. Each neuron in one layer has a direct connection with a certain neuron in the next layer. The units of these networks use different activation functions for propagation in different applications, and there are no cycles or loops in the neural networks. In addition, the number of layers and the number of neurons in each layer are not limited. They require repeated adjustments to ensure that the accuracy requirements for predicted results are satisfied. An increase in number of layers and neurons corresponds to lower training efficiency of the neural networks. Besides, the elements of input layer neurons must be highly correlated with the elements predicted by an output layer. Moreover, they should be sensitive to changes in the predicted elements [36,37].

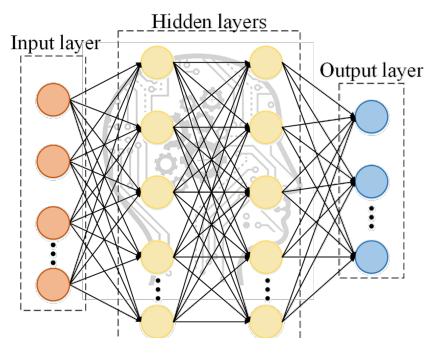


Figure 3. Deep neural network structure.

The learning process of DNNs involves forward propagation of the signal from an input layer and the backward propagation of error from the output layer. The input matrix, weight matrices, and bias matrices need to be defined in forward propagation. Accordingly, these matrices can be written as:

$$\mathbf{Z}_1 = (\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3, \dots, \mathbf{i}_m) \quad (3)$$

$$\mathbf{W}_n = (\mathbf{w}_{n_1}, \mathbf{w}_{n_2}, \mathbf{w}_{n_3}, \dots, \mathbf{w}_{n_m}) \quad (4)$$

$$\mathbf{B}_n = (\mathbf{b}_{n_1}, \mathbf{b}_{n_2}, \mathbf{b}_{n_3}, \dots, \mathbf{b}_{n_m}) \quad (5)$$

where, \mathbf{Z}_1 represents the input matrix of DNNs, \mathbf{W}_n represents n -th layer's weight matrix, \mathbf{B}_n represents n -th layer's bias matrix, m represents the number of samples in a training set, and n represents the total number of hidden and input layers. Progressively, the procedure for forward propagation can be expressed as:

$$\mathbf{Z}_1 = \mathbf{A}_1 \quad (6)$$

$$\mathbf{Z}_{i+1} = \mathbf{W}_i^T \mathbf{A}_i + \mathbf{B}_i, \quad i = 1 \dots n \quad (7)$$

$$\mathbf{A}_{i+1} = \mathbf{f}^{i+1}(\mathbf{Z}_{i+1}), \quad i = 1 \dots n \quad (8)$$

where, \mathbf{Z}_i contains i -th layer's input, \mathbf{A}_i contains i -th layer's output, and \mathbf{f}^i represents the i -th activation function. Note that \mathbf{A}_{n+1} contains the output of the DNN model. In this study, neural networks consist of four layers, among which there are two hidden layers. The two hidden layers involve 28 and 15 neurons, respectively. The first two activation functions of DNNs are ReLU functions, while the last activation function is a linear function. The ReLU function is mathematically expressed as:

$$f(x) = \max(0, x) \quad (9)$$

For backward propagation, we must choose a suitable loss function to judge the errors between the results of DNN model and the results of the multibody model. By continuously adjusting the values of the weight matrices and bias matrices, value of loss function keeps decreasing during the process of backward propagation. Generally, with the decrease in value of loss function, the accuracy of the DNN model increases. In this study, the mean square error (MSE) was opted as a loss function, which can be expressed as:

$$L = \frac{\|\mathbf{Y} - \mathbf{A}_{n+1}\|_2^2}{2m} \quad (10)$$

where, L denotes the loss of DNN model, \mathbf{Y} contains the value of the samples, \mathbf{A}_{n+1} contains the results of DNN model, and m denotes the number of samples.

Traditionally, gradient descent is widely used for backward propagation in neural network training. Nevertheless, in recent years, the adaptive moment estimation (Adam) optimization algorithm has gained more attention in deep learning [38]. Likewise, Adam algorithm offers independent adaptive learning rates for different parameters by computing the first and second moment estimations of gradients [39]. It can be implemented easily and directly, thereby improving the computational efficiency of DNNs. The hyperparameters of Adam algorithm with intuitive interpretation usually do not require much tuning. Similarly, the Adam algorithm used in this study can be mathematically expressed as:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) dm_t \quad (11)$$

$$n_t = \beta_2 n_{t-1} + (1 - \beta_2) dn_t \quad (12)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (13)$$

$$\hat{n}_t = \frac{n_t}{1 - \beta_2^t} \quad (14)$$

$$\Delta\theta_t = -\frac{\hat{m}_t}{\sqrt{\hat{n}_t + \varepsilon}} \alpha \quad (15)$$

where, m_t and n_t represent the first and second moments of weight and bias matrices for t -th training iteration, respectively. The term \hat{m}_t and \hat{n}_t denote the corrected value of first and second moments for t -th training iteration, respectively. $\Delta\theta_t$ represents the corrected value of weight and bias matrices for t -th training iteration. β_1 and β_2 , respectively, denote the attenuation coefficients of first and second moments, whose values generally are 0.9 and 0.999, respectively. Moreover, α can be regarded as a learning rate for gradient descent, and ε is a smoothing parameter with a value of 10×10^{-8} , in this study.

Additionally, the minibatch gradient descent is used to replace the stochastic gradient descent in backward propagation. In contrast to the stochastic gradient descent method, minibatch method randomly divides the entire dataset into a number of smaller batches, and performs Adam optimization according to the training results of each batch. Although its computational efficiency is relatively low, its training convergence is better than the stochastic gradient descent, in this way randomness of Adam optimization is reduced. Following the development of forward and backward propagation, a DNN model to predict the vehicle's longitudinal-lateral dynamics is estimated. The DNN modeling procedure is described in Figure 4. However, in the process of updating the DNN model, the solutions of weight and bias matrices tend to fall into a local optimum [40]. Hence, to solve this issue, we introduce LDWPSO and IWO algorithms to improve the training results of the neural networks in the next section.

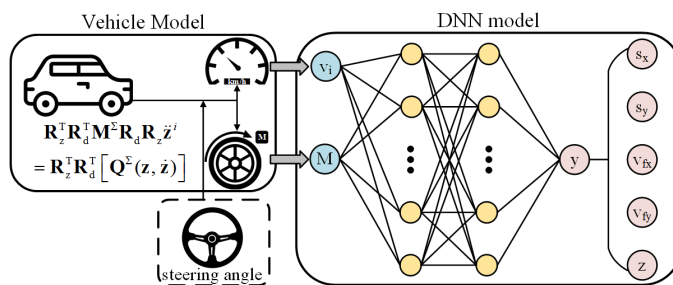


Figure 4. DNN model for vehicle’s longitudinal-lateral dynamics.

3. Optimization Algorithms

3.1. Linear Decreasing Weight Particle Swarm Optimization

The key advantages linked to particle swarm optimization (PSO) are its simple structure and rapid convergence. It has been widely used for applications in neural network training, kinetic modeling, multimodal function optimization, and control system [41,42]. In such an algorithm, a particle represents an individual and corresponds to a set of solutions, noting that particles have no mass and have only two attributes, namely position and speed of each particle. In each iteration, every particle of PSO explores for an optimal solution individually in a search space. Particles share information among each other to find the best individual value, which can be regarded as a current global optimal solution. Next, each particle corrects its position and speed according to the current global optimal solution. Thus, the PSO can effectively improve the performance of DNNs, and avoid its shortcomings of easily falling into local optimal values and network instability.

The updating equations of PSO can be described as follows. Note that, in this study, the objective function of PSO algorithm was used as a loss function of the neural networks.

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 r_1 (pbest_i^{(t)} - x_i^{(t)}) + c_2 r_2 (gbest^{(t)} - x_i^{(t)}) \tag{16}$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \tag{17}$$

where, $x_i^{(t)}$ and $v_i^{(t)}$, respectively, denote the position and speed of i -th particle in t -th iteration. In this study, the dimensions of x_i and v_i are equal to the number of elements in W_n and B_n , respectively. $pbest_i^{(t)}$ and $gbest^{(t)}$, respectively, denote the best position of i -th particle and the particle swarm in t -th iteration. c_1 and c_2 denote the corresponding acceleration factors, with values between 0 and 4. r_1 and r_2 denote the two random coefficients distributed from 0 to 1. Lastly, ω refers to the inertia factor and its value is non-negative. The particle position is updated as shown in Figure 5.

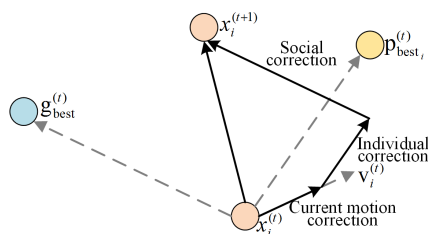


Figure 5. The procedure of particle position updating in PSO.

Noticeably, the value of inertia factor affects the optimization ability of PSO algorithm. Accordingly, a large value of inertia factor refers to a stronger global optimization ability, but weakens the local optimization ability simultaneously. In this study, the linear decreasing weight (LDW) method is utilized, i.e., the inertia factor decreases linearly with iterations. The LDW method is mathematically expressed as:

$$\omega^{(t)} = (\omega_{ini} - \omega_{end}) \frac{(T - t)}{T} + \omega_{end} \tag{18}$$

where, ω_{ini} and ω_{end} denote the initial inertia factor and the end inertia weight, respectively. T represents the maximum number of iterations. The LDW equation ensures that PSO algorithm has better global search ability at the beginning of an iteration, and has more local search ability near the end of iteration.

The algorithm framework of the LDWPSO-DNN model is illustrated in Figure 6, while the LDWPSO-DNN modeling procedure is shown in Figure 7. Note that the input of LDWPSO is the acquired vehicle states, as mentioned above.

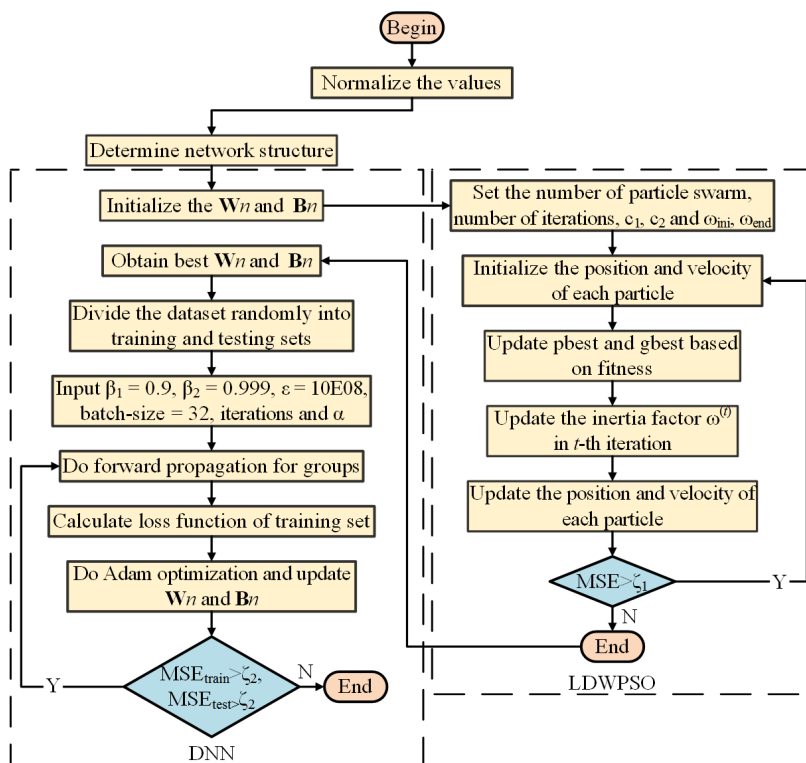


Figure 6. The framework of LDWPSO-DNN training.

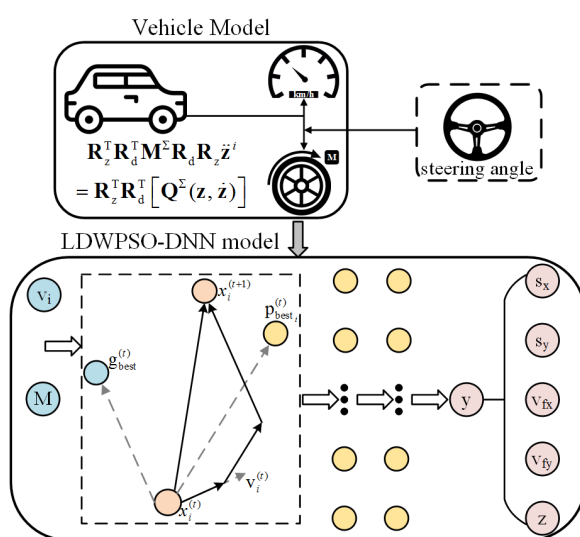


Figure 7. LDWPSO-DNN model for vehicle’s longitudinal-lateral dynamics.

3.2. Invasive Weed Optimization

Invasive weed optimization (IWO) is a population-based numerical optimization algorithm. It corresponds to a meta-heuristic algorithm designed by simulating the colonial

behavior of invasive weeds [43]. Contrary to other evolutionary algorithms, every weed reproduces an offspring during the process of evolution in IWO. Likewise, individuals with higher fitness produce more new individuals. Therefore, this algorithm strengthens the local search around superior individuals, while considering the diversity of the population. Owing to its strong robustness and adaptability, the IWO algorithm is widely used to solve practical engineering problems.

The IWO algorithm works in the following four phases. The first phase is population initialization, where a set of initial solutions is randomly generated in the D -dimensional search space. The number of initial solutions equals the initial population number. Note that, in this study, size of D is equal to the total number of elements in \mathbf{W}_n and \mathbf{B}_n . Accordingly, the i -th initial solution can be written as:

$$\mathbf{x}_i = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{iD}) \quad (19)$$

Subsequently, the second phase refers to reproduction. In this phase, we calculate the fitness of each individual according to the defined objective function. Note that, in this study, objective function is the reciprocal of a loss function. The number of seeds that each individual can produce varies from the minimum to maximum based on the fitness value. The number of seeds reproduced by each weed can be expressed as follows:

$$s_i = F_{floor} \left[s_{min} + \frac{f_i - f_{min}}{f_{max} - f_{min}} (s_{max} - s_{min}) \right] \quad (20)$$

where, s_i represents the number of seeds reproduced by i -th weed, F_{floor} denotes the round down function, f_i denotes the fitness of i -th weed, s_{max} and s_{min} represent the maximum and minimum number of seeds that can be produced, respectively. f_{max} and f_{min} denote the maximum and minimum fitness values in the evolution of current generation, respectively.

The third phase of IWO algorithm is spatial dispersal. During this phase, the reproduced seeds are randomly distributed in D -dimensional search space, near the parent weeds, with a normal distribution. The mean value of the normal distribution is 0, and the standard deviation is δ_t . The position of s -th seed produced by i -th weed is given as follows:

$$x_{i,s} = x_i + N(0, \delta_t), \quad s_{min} \leq s \leq s_{max} \quad (21)$$

where, $x_{i,s}$ denotes the position of s -th seed produced by the i -th weed. δ_t denotes the standard deviation of t -th iteration. The term δ_t can be calculated as follows:

$$\delta_t = (\delta_{ini} - \delta_{end}) \left(\frac{T-t}{T} \right)^n + \delta_{end} \quad (22)$$

where, δ_{ini} and δ_{end} denote the initial and end standard deviation, respectively. T represents the maximum number of iterations, and n is the non-linear adjustment factor, which equals 3 in this study.

The fourth and last phase of algorithm is competitive exclusion. When a population size reaches its maximum, we sort all individuals according to the fitness value, exclude individuals with poor fitness, and keep the rest, which continue to evolve. The relevant framework and modeling procedure for IWO-DNN algorithm are elaborated in Figures 8 and 9, respectively. Similar to the LDWPSO algorithm, input of IWO is the obtained vehicle states.

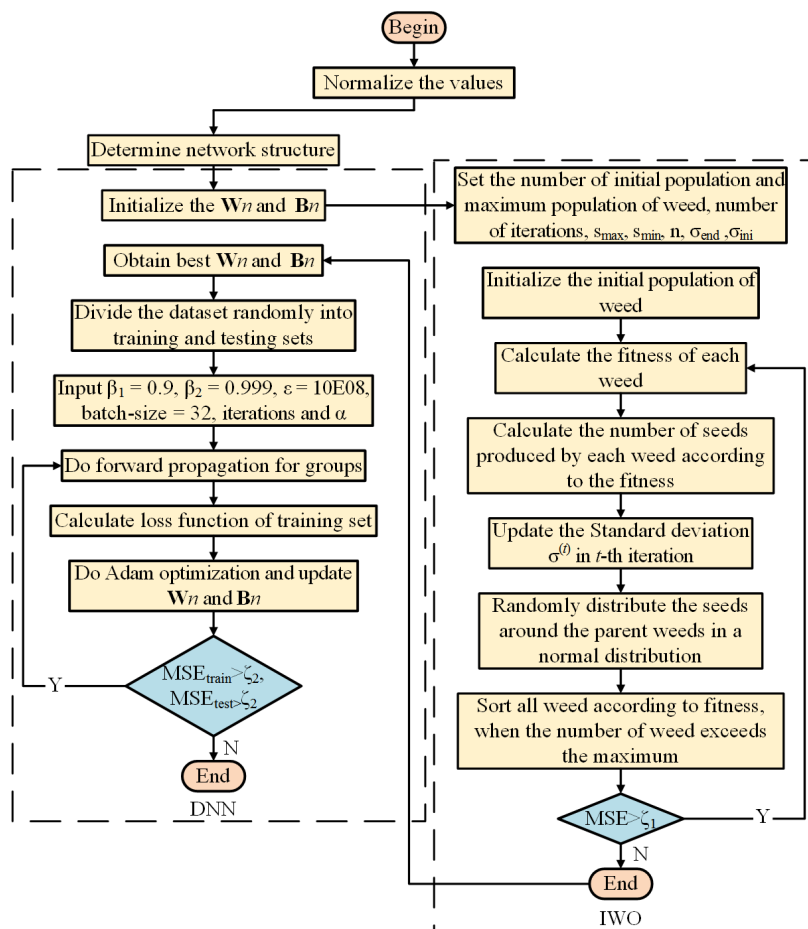


Figure 8. The framework of IWO-DNN training.

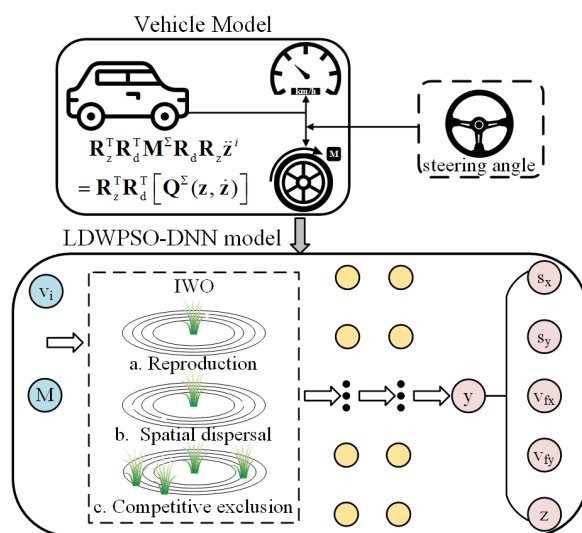


Figure 9. IWO-DNN model for vehicle’s longitudinal-lateral dynamics.

4. Numerical Results and Errors

4.1. Numerical Results

To investigate the improvements, obtained through LDWPSO and IWO, in the training results of DNNs, the DNN outputs including longitudinal and lateral driving distances, final longitudinal and lateral velocities, and vehicle yaw angle are calculated and compared (given in Figure 10). Each figure involves multibody model results, DNN predicted results,

and improved DNN predicted results. The driving situations, i.e., initial longitudinal speed varying from 15 m/s to 45 m/s and the driving torque ranging from -500 Nm to 500 Nm, were used to imitate the accelerating and decelerating.

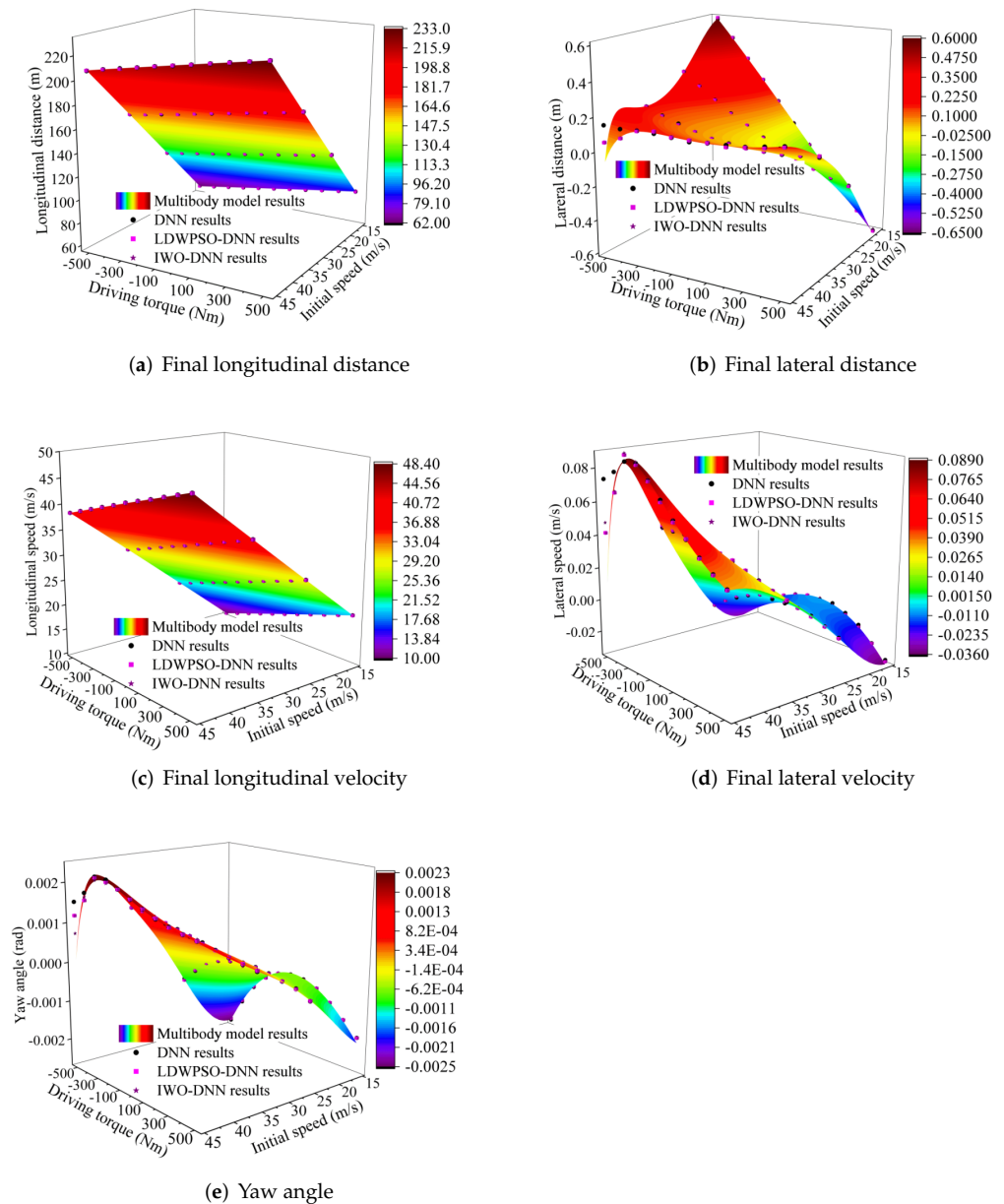


Figure 10. The comparison of DNN results.

Additionally, Figures 11–15 depict the absolute percentage errors for the above reported results in terms of five vehicle responses. Note that different initial longitudinal speeds are considered in these figures.

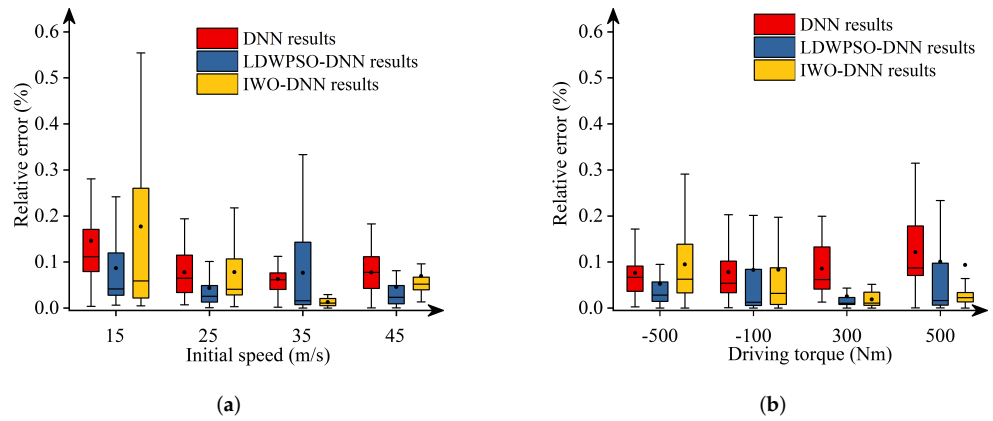


Figure 11. Absolute percentage error: final longitudinal distance.

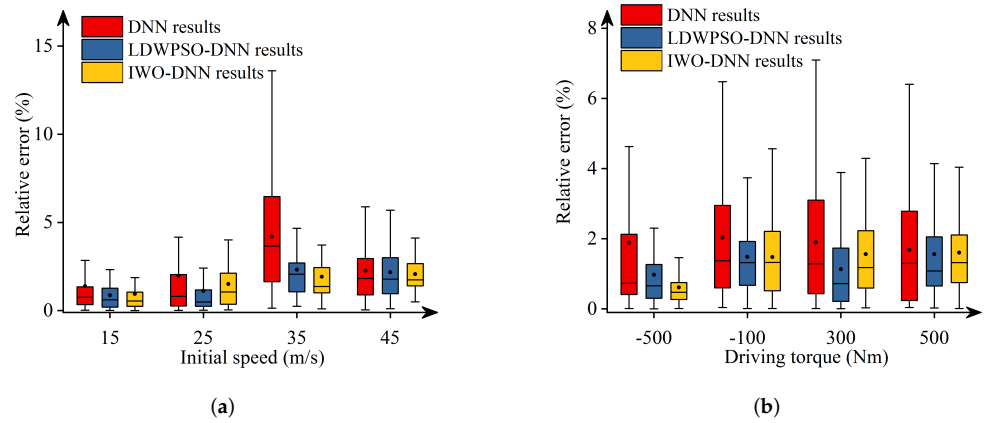


Figure 12. Absolute percentage error: final lateral distance.

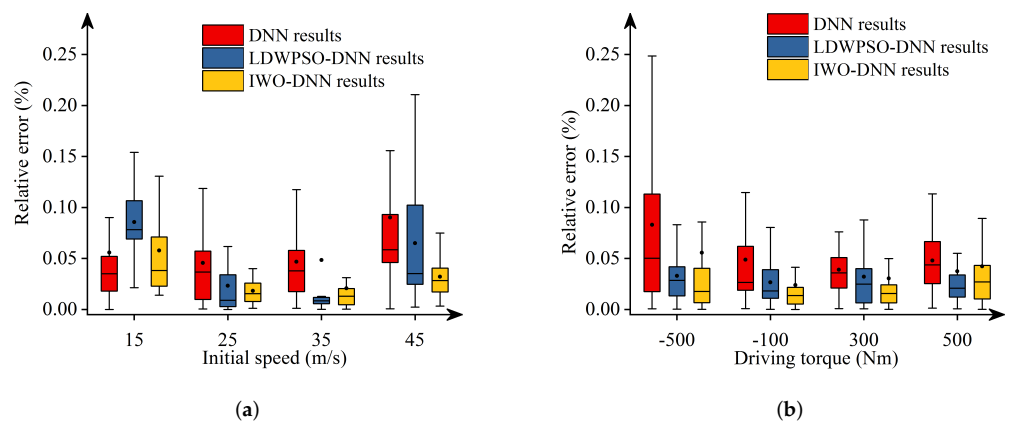


Figure 13. Absolute percentage error: final longitudinal velocities.

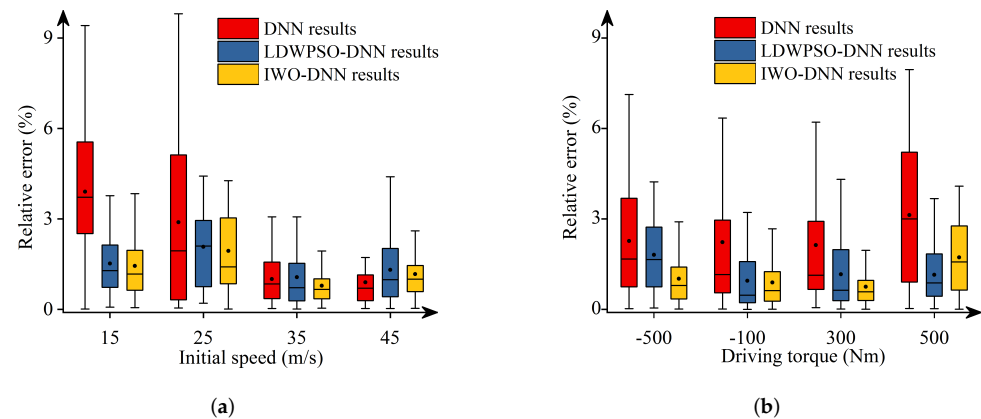


Figure 14. Absolute percentage error: final lateral velocities.

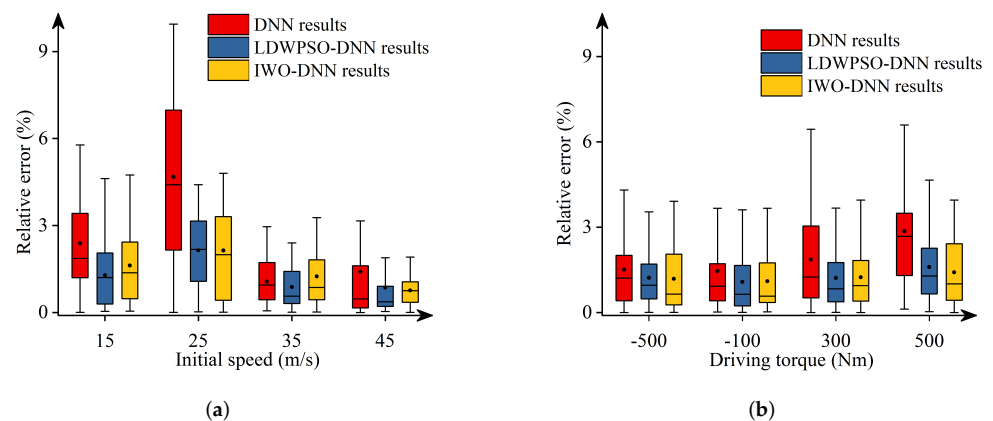


Figure 15. Absolute percentage error: yaw angle.

By analyzing the above numerical results, we can draw conclusions as follows:

- The predicted results of DNNs, LDWPSO-DNNs, and IWO-DNNs for longitudinal responses can fit the results of the vehicle multibody model well. The absolute percentage errors of less than 1% are observed.
- The absolute percentage errors of LDWPSO-DNNs and IWO-DNNs for lateral and longitudinal-lateral responses are smaller than that of DNNs. The results verify that LDWPSO and IWO algorithms improve the prediction accuracy of the DNN model.

4.2. Error Analysis

To quantify the improvements of LDWPSO and IWO algorithms to the accuracy of DNN results, we introduce four error functions, and compare the pros and cons for the predicted results. These four error functions are mean absolute error (MAE), mean absolute percentage error (MAPE), root mean square error (RMSE), and R^2 . The function MAE represents the average of absolute errors, and directly reflects the actual error of predicted results. The second one, MAPE is one of the widely used metrics for evaluating predictive performance, and can be calculated based on MAE easily. RMSE corresponds to arithmetic square root of MSE, which is more intuitive. For these three error functions, the smaller the value is, the better the predicted results are. Lastly, R^2 denotes the coefficient of determination, with a range varying from 0 to 1. The model fits well if the value of R^2 is closer to 1. Correspondingly, these four error functions are mathematically expressed as:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (23)$$

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right| \quad (24)$$

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (25)$$

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y}_i)^2} \quad (26)$$

where, y_i represents the i -th value of reference 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, and n represents the number of predicted results, which is 7701 for this study. Tables 2–6, respectively, depict evaluation results of final longitudinal and lateral distances, final longitudinal and lateral velocities, and yaw angle.

Table 2. The accuracy analysis of final longitudinal distance.

Final Longitudinal Distance	MAE (m)	MAPE (%)	RMSE (m)	R ²
DNN	0.107476	0.078521	0.135354	0.999990
LDWPSO-DNN	0.071606	0.049622	0.138596	0.999990
IWO-DNN	0.062153	0.048728	0.135021	0.999990

Table 3. The accuracy analysis of final lateral distance.

Final Lateral Distance	MAE (m)	MAPE (%)	RMSE (m)	R ²
DNN	0.002647	9.346083	0.006920	0.998541
LDWPSO-DNN	0.002158	4.741057	0.003724	0.999577
IWO-DNN	0.001912	3.201640	0.003467	0.999634

Table 4. The accuracy analysis of final longitudinal velocity.

Final Longitudinal Velocity	MAE (m/s)	MAPE (%)	RMSE (m/s)	R ²
DNN	0.013769	0.050521	0.018431	0.999996
LDWPSO-DNN	0.009383	0.032383	0.015797	0.999997
IWO-DNN	0.007688	0.025955	0.015518	0.999997

Table 5. The accuracy analysis of final lateral velocity.

Final Lateral Velocity	MAE (m/s)	MAPE (%)	RMSE (m/s)	R ²
DNN	0.000524	7.754304	0.001778	0.996591
LDWPSO-DNN	0.000341	4.861110	0.000791	0.999325
IWO-DNN	0.000290	3.255596	0.000848	0.999224

Table 6. The accuracy analysis of the yaw angle.

Yaw Angle	MAE (rad)	MAPE (%)	RMSE (rad)	R ²
DNN	0.000014	6.804836	0.000040	0.998480
LDWPSO-DNN	0.000010	3.879418	0.000026	0.999348
IWO-DNN	0.000008	3.176424	0.000016	0.999734

It is convenient to observe from Tables 2–6 that MAE, MAPE, and RMSE for LDWPSO-DNN and IWO-DNN models are smaller than the DNN model. The R² values for LDWPSO-DNN and IWO-DNN models are closer to 1. Furthermore, for the final lateral and longitudinal velocities and yaw angle, the MAPE of LDWPSO-DNN and IWO-DNN models drops sharply. While analyzing the error functions of LDWPSO-DNN and IWO-DNN models, we can observe that most of the indicators for IWO-DNN model are better than the LDWPSO-DNN model.

5. Conclusions

In this study, we proposed a DNN-based method to model the longitudinal-lateral dynamics of a vehicle. Two additional optimization algorithms, namely LDWPSO and IWO, were used to enhance the performance of DNNs. To verify the effectiveness of the LDWPSO-DNN and IWO-DNN models, a vehicle system was modeled and the predicted vehicle states of three DNNs were compared with the reference results in terms of error functions. The comparative results reveal that the prediction accuracy of IWO-DNN model is higher than that of the other two DNN models. Overall, an improved DNN model using LDWPSO and IWO algorithms was introduced to describe the vehicle's longitudinal-lateral dynamics in real time. The proposed method can be used effectively for the real-time simulation and control of intelligent vehicles in complex driving conditions to enhance the vehicle safety. The scenario-based testing of autonomous vehicles by using sensor networks will be performed to validate the proposed method in the future.

Author Contributions: Conceptualization, Y.P.; methodology, X.N. and C.M.; software, X.N. and C.M.; validation, X.N. and C.M.; formal analysis, C.M.; investigation, X.N.; resources, Y.P.; data curation, X.N.; writing—original draft preparation, Y.P. and Z.L.; writing—review and editing, Z.L. and G.K.; visualization, Z.L.; supervision, Y.P. and G.K.; project administration, Y.P.; funding acquisition, Y.P. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the National Natural Science Foundation of China (Project No.12072050 and No.12211530029), the Research Project of State Key Laboratory of Mechanical System and Vibration (Project No.MSV202216), and the Fundamental Research Funds for the Central Universities (Project No.2021CDJQY-032).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Guo, H.; Yin, Z.; Cao, D.; Chen, H.; Lv, C. A Review of Estimation for Vehicle Tire-Road Interactions Toward Automated Driving. *IEEE Trans. Syst. Man, Cybern. Syst.* **2019**, *49*, 14–30. [[CrossRef](#)]
- Yang, P.; Zang, M.; Zeng, H.; Guo, X. The interactions between an off-road tire and granular terrain: GPU-based DEM-FEM simulation and experimental validation. *Int. J. Mech. Sci.* **2020**, *179*, 105634. [[CrossRef](#)]
- Pang, H.; Yao, R.; Wang, P.; Xu, Z. Adaptive backstepping robust tracking control for stabilizing lateral dynamics of electric vehicles with uncertain parameters and external disturbances. *Control. Eng. Pract.* **2021**, *110*, 104781. [[CrossRef](#)]
- Ye, Y.; Huang, P.; Sun, Y.; Shi, D. MBSNet: A deep learning model for multibody dynamics simulation and its application to a vehicle-track system. *Mech. Syst. Signal Process.* **2021**, *157*, 107716. [[CrossRef](#)]
- Rodríguez, A.J.; Sanjurjo, E.; Pastorino, R.; Ángel Naya, M. State, parameter and input observers based on multibody models and Kalman filters for vehicle dynamics. *Mech. Syst. Signal Process.* **2021**, *155*, 107544. [[CrossRef](#)]

6. Sattar, S.; Li, S.; Chapman, M. Road surface monitoring using smartphone sensors: A review. *Sensors* **2018**, *18*, 3845. [[CrossRef](#)]
7. Gat, G.; Franco, Y.; Shmulevich, I. Fast dynamic modeling for off-road track vehicles. *J. Terramech.* **2020**, *92*, 1–12. [[CrossRef](#)]
8. Yamashita, H.; Chen, G.; Ruan, Y.; Jayakumar, P.; Sugiyama, H. Parallelized Multiscale Off-Road Vehicle Mobility Simulation Algorithm and Full-Scale Vehicle Validation. *J. Comput. Nonlinear Dyn.* **2020**, *15*, 091007. [[CrossRef](#)]
9. Ji, X.; He, X.; Lv, C.; Liu, Y.; Wu, J. A vehicle stability control strategy with adaptive neural network sliding mode theory based on system uncertainty approximation. *Veh. Syst. Dyn.* **2018**, *56*, 923–946. [[CrossRef](#)]
10. Xing, Y.; Lv, C. Dynamic state estimation for the advanced brake system of electric vehicles by using deep recurrent neural networks. *IEEE Trans. Ind. Electron.* **2019**, *67*, 9536–9547. [[CrossRef](#)]
11. Nguyen, T.; Nguyen-Phuoc, D.Q.; Wong, Y.D. Developing artificial neural networks to estimate real-time onboard bus ride comfort. *Neural Comput. Appl.* **2021**, *33*, 5287–5299. [[CrossRef](#)]
12. Tuncali, C.E.; Fainekos, G.; Prokhorov, D.; Ito, H.; Kapinski, J. Requirements-Driven Test Generation for Autonomous Vehicles With Machine Learning Components. *IEEE Trans. Intell. Veh.* **2020**, *5*, 265–280. [[CrossRef](#)]
13. Singh, P.; Chaudhury, S.; Panigrahi, B.K. Hybrid MPSO-CNN: Multi-level Particle Swarm optimized hyperparameters of Convolutional Neural Network. *Swarm Evol. Comput.* **2021**, *63*, 100863. [[CrossRef](#)]
14. Rutherford, S.J.; Cole, D.J. Modelling nonlinear vehicle dynamics with neural networks. *Int. J. Veh. Des.* **2010**, *53*, 260–287. [[CrossRef](#)]
15. Kim, D.; Min, K.; Kim, H.; Huh, K. Vehicle sideslip angle estimation using deep ensemble-based adaptive Kalman filter. *Mech. Syst. Signal Process.* **2020**, *144*, 106862. [[CrossRef](#)]
16. Devineau, G.; Polack, P.; Altché, F.; Moutarde, F. Coupled longitudinal and lateral control of a vehicle using deep learning. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 642–649.
17. Melzi, S.; Sabbioni, E. On the vehicle sideslip angle estimation through neural networks: Numerical and experimental results. *Mech. Syst. Signal Process.* **2011**, *25*, 2005–2019. [[CrossRef](#)]
18. 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. Eng. Pract.* **2018**, *76*, 41–53. [[CrossRef](#)]
19. Acosta, M.; Kanarachos, S. Teaching a vehicle to autonomously drift: A data-based approach using neural networks. *Knowl.-Based Syst.* **2018**, *153*, 12–28. [[CrossRef](#)]
20. Da Lio, M.; Bortoluzzi, D.; Rosati Papini, G.P. Modelling longitudinal vehicle dynamics with neural networks. *Veh. Syst. Dyn.* **2020**, *58*, 1675–1693. [[CrossRef](#)]
21. El-Nagar, A.M.; El-Bardini, M. Hardware-in-the-loop simulation of interval type-2 fuzzy PD controller for uncertain nonlinear system using low cost microcontroller. *Appl. Math. Model.* **2016**, *40*, 2346–2355. [[CrossRef](#)]
22. Dai, X.; Ke, C.; Quan, Q.; Cai, K.Y. RFLySim: Automatic test platform for UAV autopilot systems with FPGA-based hardware-in-the-loop simulations. *Aerosp. Sci. Technol.* **2021**, *114*, 106727. [[CrossRef](#)]
23. 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* **2021**, *180*, 109541. [[CrossRef](#)]
24. Arai, R.; Imakura, A.; Sakurai, T. An improvement of the nonlinear semi-NMF based method by considering bias vectors and regularization for deep neural networks. *Int. J. Mach. Learn. Comput.* **2018**, *8*, 191–197. [[CrossRef](#)]
25. Jatoth, C.; Gangadharan, G.; Fiore, U. Optimal fitness aware cloud service composition using modified invasive weed optimization. *Swarm Evol. Comput.* **2019**, *44*, 1073–1091. [[CrossRef](#)]
26. Xin, J.; Li, S.; Sheng, J.; Zhang, Y.; Cui, Y. Application of Improved Particle Swarm Optimization for Navigation of Unmanned Surface Vehicles. *Sensors* **2019**, *19*, 3096. [[CrossRef](#)]
27. Nie, X.; Min, C.; Pan, Y.; Li, K.; Li, Z. Deep-neural-network-based modelling of longitudinal-lateral dynamics to predict the vehicle states for autonomous driving. *Sensors* **2022**, *22*, 2013. [[CrossRef](#)] [[PubMed](#)]
28. Rodríguez, J.I.; Jiménez, J.M.; Funes, F.J.; de Jalón, J.G. Recursive and residual algorithms for the efficient numerical integration of multi-body systems. *Multibody Syst. Dyn.* **2004**, *11*, 295–320. [[CrossRef](#)]
29. de García Jalón, J.; Álvarez, E.; De Ribera, F.; Rodríguez, I.; Funes, F. A fast and simple semi-recursive formulation for multi-rigid-body systems. In *Advances in Computational Multibody Systems*; Springer: Berlin, Germany, 2005; pp. 1–23.
30. Pan, Y.; Callejo, A.; Bueno, J.L.; Wehage, R.A.; de Jalón, J.G. Efficient and accurate modeling of rigid rods. *Multibody Syst. Dyn.* **2017**, *40*, 23–42. [[CrossRef](#)]
31. Laulusa, A.; Bauchau, O.A. Review of Classical Approaches for Constraint Enforcement in Multibody Systems. *J. Comput. Nonlinear Dyn.* **2007**, *3*, 011004. [[CrossRef](#)]
32. Pan, Y.; Dai, W.; Huang, L.; Li, Z.; Mikkola, A. Iterative refinement algorithm for efficient velocities and accelerations solutions in closed-loop multibody dynamics. *Mech. Syst. Signal Process.* **2021**, *152*, 107463. [[CrossRef](#)]
33. He, L.; Pan, Y.; He, Y.; Li, Z.; Królczyk, G.; Du, H. Control strategy for vibration suppression of a vehicle multibody system on a bumpy road. *Mech. Mach. Theory* **2022**, *174*, 104891. [[CrossRef](#)]
34. Pan, Y.; Xiang, S.; He, Y.; Zhao, J.; Mikkola, A. The validation of a semi-recursive vehicle dynamics model for a real-time simulation. *Mech. Mach. Theory* **2020**, *151*, 103907. [[CrossRef](#)]
35. Elen, A.; Avuçlu, E. Standardized Variable Distances: A distance-based machine learning method. *Appl. Soft Comput.* **2021**, *98*, 106855. [[CrossRef](#)]

36. Kim, T.Y.; Cho, S.B. Optimizing CNN-LSTM neural networks with PSO for anomalous query access control. *Neurocomputing* **2021**, *456*, 666–677. [[CrossRef](#)]
37. Parcham, E.; Ilbeygi, M.; Amini, M. CBCapsNet: A novel writer-independent offline signature verification model using a CNN-based architecture and Capsule Neural Networks. *Expert Syst. Appl.* **2021**, *185*, 115649. [[CrossRef](#)]
38. Chang, Z.; Zhang, Y.; Chen, W. Electricity price prediction based on hybrid model of adam optimized LSTM neural network and wavelet transform. *Energy* **2019**, *187*, 115804. [[CrossRef](#)]
39. Khan, A.H.; Cao, X.; Li, S.; Katsikis, V.N.; Liao, L. BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 461–471. [[CrossRef](#)]
40. Zhang, P.; Li, H.; Ha, Q.; Yin, Z.Y.; Chen, R.P. Reinforcement learning based optimizer for improvement of predicting tunneling-induced ground responses. *Adv. Eng. Inform.* **2020**, *45*, 101097. [[CrossRef](#)]
41. Amer, N.H.; Hudha, K.; Zamzuri, H.; Aparow, V.R.; Kadir, Z.A.; Abidin, A.F.Z. Hardware-in-the-loop simulation of trajectory following control for a light armoured vehicle optimised with particle swarm optimisation. *Int. J. Heavy Veh. Syst.* **2019**, *26*, 663–691. [[CrossRef](#)]
42. Guo, X.; Ji, M.; Zhao, Z.; Wen, D.; Zhang, W. Global path planning and multi-objective path control for unmanned surface vehicle based on modified particle swarm optimization (PSO) algorithm. *Ocean. Eng.* **2020**, *216*, 107693. [[CrossRef](#)]
43. Le, T.L.; Lin, C.M.; Huynh, T.T. Self-evolving type-2 fuzzy brain emotional learning control design for chaotic systems using PSO. *Appl. Soft Comput.* **2018**, *73*, 418–433. [[CrossRef](#)]