World Electric Vehicle Journal

MDPI

Article

Deep Reinforcement Learning Lane-Changing Decision Algorithm for Intelligent Vehicles Combining LSTM Trajectory Prediction

Zhengcai Yang [1,2,*], Zhengjun Wu [1,2], Yilin Wang [3] and Haoran Wu [1,2]

[1] Vehicle Engineering Department, Hubei University of Automotive Technology, Shiyan 442002, China; wuzhengjun3770@163.com (Z.W.); wuhaoran1228@163.com (H.W.)
[2] Hubei Key Laboratory of Automotive Power Train and Electronic Control, Hubei University of Automotive Technology, Shiyan 442002, China
[3] College of Electrical and New Energy, China Three Gorges University, Yichang 443000, China; 18186782353@163.com
* Correspondence: yang516516@163.com

Abstract: Intelligent decisions for autonomous lane-changing in vehicles have consistently been a focal point of research in the industry. Traditional lane-changing algorithms, which rely on predefined rules, are ill-suited for the complexities and variabilities of real-world road conditions. In this study, we propose an algorithm that leverages the deep deterministic policy gradient (DDPG) reinforcement learning, integrated with a long short-term memory (LSTM) trajectory prediction model, termed as LSTM-DDPG. In the proposed LSTM-DDPG model, the LSTM state module transforms the observed values from the observation module into a state representation, which then serves as a direct input to the DDPG actor network. Meanwhile, the LSTM prediction module translates the historical trajectory coordinates of nearby vehicles into a word-embedding vector via a fully connected layer, thus providing predicted trajectory information for surrounding vehicles. This integrated LSTM approach considers the potential influence of nearby vehicles on the lane-changing decisions of the subject vehicle. Furthermore, our study emphasizes the safety, efficiency, and comfort of the lane-changing process. Accordingly, we designed a reward and penalty function for the LSTM-DDPG algorithm and determined the optimal network structure parameters. The algorithm was then tested on a simulation platform built with MATLAB/Simulink. Our findings indicate that the LSTM-DDPG model offers a more realistic representation of traffic scenarios involving vehicle interactions. When compared to the traditional DDPG algorithm, the LSTM-DDPG achieved a 7.4% increase in average single-step rewards after normalization, underscoring its superior performance in enhancing lane-changing safety and efficiency. This research provides new ideas for advanced lane-changing decisions in autonomous vehicles.

Keywords: LSTM; DDPG; reinforcement learning; lane-changing decision

Citation: Yang, Z.; Wu, Z.; Wang, Y.; Wu, H. Deep Reinforcement Learning Lane-Changing Decision Algorithm for Intelligent Vehicles Combining LSTM Trajectory Prediction. *World Electr. Veh. J.* **2024**, *15*, 173. https://doi.org/10.3390/wevj15040173

Academic Editor: Guoxing Bai

Received: 21 March 2024
Revised: 16 April 2024
Accepted: 19 April 2024
Published: 21 April 2024

1. Introduction

Among the many functions of intelligent driving assistance systems, autonomous lane changing is important [1]. Through intelligent lane-changing decisions, traffic accidents caused by driver errors can be significantly reduced, and the efficiency of lane changing can be improved. Consequently, systems related to autonomous vehicle lane-changing [2,3] have emerged as a focal point of research for various vehicle enterprises and the robotics field [4].

Currently, lane-changing decision models include two common methods: a rule-based decision algorithm and a reinforcement learning decision algorithm. The rule-based decision algorithm defines the behavior mode of the vehicle in different scenarios and uses characteristic variables as the basis for judgment when the driving condition switches.

The primary representative algorithms are the finite-state machine (FSM) and fuzzy logic algorithms. An FSM is a mathematical model of a discrete input and output system composed of a finite number of states. Within this system, the current state interprets events and triggers corresponding actions, leading to state transitions. For instance, under the mixed framework of an FSM, a reference [5] determines the results of different submodules according to their priorities, and determines the final decision results through state estimation and goal determination. Fuzzy logic, on the other hand, is instrumental for systems with indeterminate or expansive models. By emulating the human brain's approach to uncertainty and reasoning, fuzzy logic addresses challenges posed by conventional methods in deciphering regular information. A study highlighted in reference [6] introduced an automatic overtaking method leveraging fuzzy logic, enabling overtaking on two-way roads. Although rule-based decision-making methods are straightforward to implement and offer clear interpretability, they demand manual rule-based configurations and regular updates. These types of methods, however, may falter in real-world environments with scenarios too intricate for rule-based descriptions, leading to compromised robustness and adaptability. Moreover, real-world driving often presents "gray areas", where a single scene can offer multiple logical decision paths or involve conflicts between several rules.

There are also many methods for the trajectory planning of autonomous vehicles, including the A* algorithm, sampling-based fast-exploring random number (RRT) algorithm, model predictive control (MPC) algorithm, and B-spline-based trajectory planning algorithm. Reference [7] proposes a criterion-based A* algorithm that uses the criteria generated by human programming or global programming to develop heuristic functions. However, the A* algorithm still has some problems, such as the inaccurate detection of a road edge and inapplicability during vehicle turning. In reference [8], the MPC algorithm is used to solve the problem of lane-change decision and control, and a decision method based on model predictive control is proposed. In this method, the control of a vehicle running on the expressway is divided into two parts: lane-change decision and lane-change control, which are solved by the MPC method, respectively. A new collaborative trajectory planning strategy is proposed in reference [9], which is particularly concerned with column stability during vehicle formation. Unlike traditional methods, this strategy advances the goal of distance control to the planning stage, rather than adjusting it only in the feedback control stage. By integrating the concept of string stability, the strategy enables a smooth transition between autonomous driving and collaborative formation driving. In addition, B-spline curves are used to design trajectories of cooperative vehicles, and a series of piecewise polynomial functions are generated.

In [10], a network structure and hyperparameter model, pretrained in a simulator, were adapted to a car equipped with a camera. Using monocular images as an input enabled the vehicle to learn lane-following in real-world scenarios for the first time. In [11], a hybrid approach blending imitation learning with the deep deterministic policy gradient (DDPG) algorithm was trained within the Carla simulator [12]. The model introduced a controllable gating mechanism to address the inefficiencies in exploring continuous space. Under the influence of varying steering angle rewards, specialized strategies were cultivated for each control signal.

Currently, reinforcement-based decision algorithms excel in exploring the depth of scenes. They can comprehensively address all operational conditions by leveraging extensive datasets, autonomously extracting features and decision attributes, which facilitates algorithm iterations. However, the interpretability of such algorithm models is limited. The decision-making efficacy is contingent on the sample dataset's quality, model network's structure, and adequacy of sample sizes; any mismatches in these can lead to the overfitting or underfitting of the model. A fully end-to-end approach demands considerable hardware computing capabilities in intelligent vehicles. These types of systems tend to be complex, opaque, and difficult to interpret. Consequently, the decision and planning modules have evolved independently, reserving reinforcement learning algorithms for high-level decision-making. The planning module, in turn, formulates a coherent trajectory

aligned with vehicle dynamics, enhancing the system's interpretability and adaptability. However, a majority of these reinforcement learning algorithms, when employed for high-level decisions, operate on static environmental analysis. They rely on present-time environmental data, neglecting potential future state changes. Hence, the algorithm's decision output might achieve only local optimization, lacking a globally optimal decision across the entire environment.

There are some problems in the decision-making algorithm of intelligent driving vehicles, such as the difficulty of describing the real environment with rules, poor algorithm interpretation, and lack of learning ability. Because the LSTM [13] trajectory prediction algorithm can help the model adapt to the changes of a dynamic environment, it also can help the DDPG algorithm to better process time-series information in the environment and improve the agent's generalization ability to the environment. Therefore, a DDPG intelligent vehicle decision algorithm based on LSTM trajectory prediction is introduced in this paper. By integrating the anticipated trajectory of interactive vehicles as part of the input, we constructed the LSTM-DDPG model on the MATLAB/Simulink platform. When contrasted with the Conv-DDPG decision algorithm, which solely accounts for the current state information, the proposed algorithm's superiority was clearly evident.

2. LSTM and DDPG

2.1. LSTM

Long short-term memory (LSTM) leverages extensive sequence information to create a learning model owing to its inherent memory function. By introducing a memory unit, LSTM addresses the gradient vanishing issue encountered in long sequence prediction. Moreover, the control gate structure facilitates the coordination of information transmission between memory cells, effectively tackling the long-term dependency challenges in time-series prediction. This makes LSTM particularly suitable for time-series forecasting.

LSTM served as the foundational predictor for the reinforcement learning algorithm [14]. Utilizing the LSTM decoder structure, we predicted the trajectory of obstacle vehicles, established the connection between the anterior and posterior trajectory coordinates of the vehicle, and conducted a more in-depth analysis of the microscopic characteristics of these vehicle trajectory coordinates [15].

2.2. DDPG

Reinforcement learning [16] is a process in which the agent learns continuously in the environment through different strategies. A policy maps the state space to the action space. Guided by this policy, the agent selects actions as input to interact with the environment. In response, the environment provides real-time feedback in the form of rewards and transitions to the next state based on state transition probabilities. Through ongoing interactions and trial-and-error between the agent and environment, an optimal policy is ultimately learned, aiming to maximize the cumulative reward from environmental actions.

The policy gradient (PG) algorithm can parameterize the policy, compute the gradient function relative to the action, and refine the policy using gradient descent, ultimately yielding the optimal policy action. Although the output of the stochastic policy gradient presents a probability distribution, and hence, a specific output action value remains indeterminate, the deterministic policy gradient algorithm operates differently. In the case of a deterministic policy, a specific action value corresponds to a definitive action; thus, for the same state input into the network, the output action remains consistent.

The deep deterministic policy gradient (DDPG) [17] algorithm, which employs the actor–critic framework [18], addresses the limitation of the DQN algorithm being confined to discrete actions, enabling its application to tasks with continuous action spaces. DDPG synergizes the deterministic policy gradient algorithm with techniques inherent to the DQN algorithm.

3. DDPG Lane-Changing Decision Algorithm Combined with LSTM Trajectory Prediction

Vehicle acceleration and steering wheel angles are continuous variables. In this study, we selected the DDPG algorithm, recognized for its superior performance in continuous states, to drive vehicle decision-making. We gleaned the decision-making behavior of real drivers from experience, establishing a mapping [19] from road state information to acceleration and steering wheel angles. Concurrently, lane-changing is a nuanced and intricate decision-making process, requiring a vehicle to assimilate both its current data and the interactive vehicular information from the road environment. During real-world driving, drivers often anticipate the future trajectories of surrounding vehicles before deciding on a maneuver. Traditional reinforcement learning models for lane-changing decisions tend to focus solely on the static information of the current moment as captured by vehicle sensors. This approach offers a limited overall understanding of the environment and lacks foresight into the future actions of interactive vehicles. This type of approach falls short in accurately capturing the dynamics of real-world traffic scenarios.

Therefore, this paper proposes an LSTM-DDPG intelligent lane-changing decision algorithm with an interactive vehicle-prediction trajectory function. The integrated LSTM module considers the influence of future interactive vehicles on the lane-changing behavior of local vehicles and adds observation and status modules to the LSTM to improve the adaptability of the traditional algorithm in complex traffic scenarios. This decision is more closely related to real road scenarios.

3.1. Subsection

In this study, we integrated LSTM into the DDPG framework, resulting in LSTM-DDPG algorithm models, as illustrated in Figure 1. The LSTM is split into two branches. Each LSTM unit comprises an observation and a status module. The status module translates the observed values from the observation module into distinct status information. The second LSTM ingests the position data of the obstacle vehicle and, through the encoder, convolutional layer, and decoder, derives the anticipated trajectory information. The reinforcement learning component of LSTM-DDPG consists of both an actor network and a critic network, as depicted in Figure 1. Both the actor and critic networks employ a four-tiered structure, encompassing an input layer, an output layer, and two intermediate hidden layers. The hidden layers utilize the ReLU activation function to model the relationship between input and output signals.

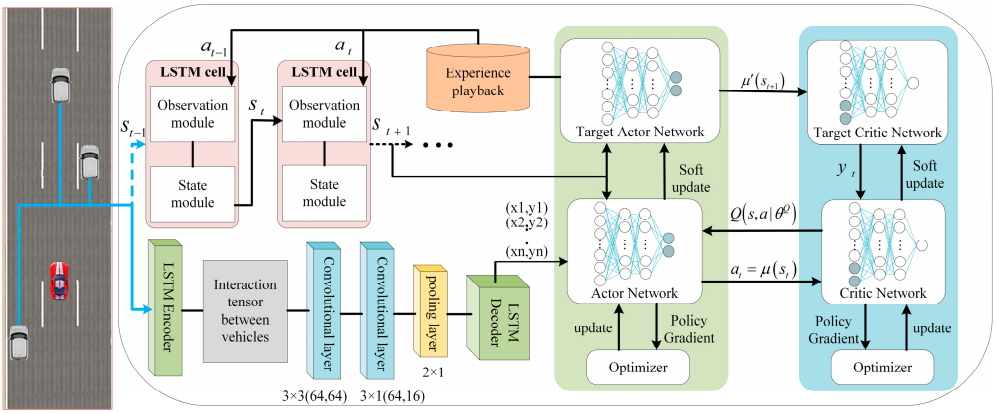


Figure 1. Framework of the LSTM-DDPG algorithm.

The LSTM prediction model processes signals from various vehicle sensors, capturing the present state of the host vehicle and its surroundings. This model predicts the future trajectories of nearby vehicles. Using the forecasted trajectory combined with the current vehicle position, data are fed into the DDPG action network. The actor network then generates continuous action values for acceleration and the front wheel angle, based on an action strategy. Meanwhile, the critic network processes the state transformed by the LSTM

along with the action output from the actor network, producing a return value. This return value is used to assess and continually refine the strategy of the actor network. Ultimately, the quintet of the current state, action, updated state, return value, and termination status is saved in the experience pool. Both actor and critic networks are refined using samples drawn from this pool.

The LSTM-DDPG algorithm pseudo-code is shown in Table 1.

Table 1. LSTM-DDPG algorithm pseudo-code.

Algorithm: LSTM-DDPG	
1	Initialize experience pool D
2	Initialize critic network θ^{Q}, actor network θ^{μ}
3	Initialize target network θ^{Q^-}, θ^{μ^-}
4	for episode in (1, M) do
5	set initial state S_0
6	for t in (1, T) do
7	get the historical track X_0 and Map to vector μ^t
8	encode μ^t as an implicit state vector by encoder
9	predict the surrounding vehicle track Y_0 by decoder
10	convert Y_0 to location information and update s_t
11	select the optimal action a_t according to step 10 and action strategy
12	execute a_t and obtain current reward r_t and observe next state s_{t+1}
13	store the element (s_i, a_i, r_i, s_{t+1}) in D
14	sample mini-batch (s_i, a_i, r_i, s_{t+1}) from D
15	update critic network using loss function L
16	update actor network using policy gradient $\nabla_{\theta^{\mu}}\mu$
17	update the two target networks by soft update mode
18	end for
19	end for

3.2. Markov Process Modeling

3.2.1. Set State Space

During driving, the vehicle consistently interacts with its environment, maintaining a flow of state information. When making lane-changing decisions, it is crucial to consider the vehicle's driving state and dynamics between the vehicle and its surroundings. In this study, the chosen state set incorporated information from both the host vehicle and road environment.

1. The ego vehicle information on the road at time t, $M[t] = \{v, d, x, \mathrm{y}\}$;

Ego vehicle information: The scalar input is chosen to represent the ego vehicle information using speed v of the ego vehicle, distance d between the ego vehicle and preceding vehicle, and ordinate (x, y) of the ego vehicle trajectory.

2. Predicted trajectories of surrounding vehicles on the road at time t;

LSTM was utilized to forecast the trajectories of neighboring vehicles. To uniformly represent the vehicle trajectory's high-dimensional features, the historical trajectory coordinates of the surrounding vehicles at the given input time were transformed into word-embedding vectors using the fully connected layer μ^t:

$$\mu^t = FC\left(X^t;\ W_{fc}\right), \tag{1}$$

where $FC()$ denotes a fully connected layer function, and W_{fc} denotes the weight parameter of the fully connected layer.

The word-embedding vector of the corresponding vehicle historical trajectory and hidden state vector h^{t-1} of the historical trajectory at the last moment were passed via

the LSTM encoder to obtain the current hidden state vector h^t containing the contextual information of the vehicle motion features:

$$h^t = encoder\left(\mu^t, h^{t-1}; W_{enc}\right), \tag{2}$$

where *encoder*() is responsible for encoding the word-embedding vector μ^t of the vehicle trajectory into an implicit state vector, and W_{enc} denotes the weight parameter of the encoder.

Finally, the trajectory encoding the hidden state vector of all the vehicles around the current time is obtained as follows:

$$h_0^t, h_s^t (s \in 1, 2, \ldots, n), \tag{3}$$

Predicted trajectory: At any time t, the inputs of the trajectory prediction model are the trajectory coordinates of all vehicles v_0 around the host vehicle in the historical observation domain length *his*:

$$\begin{array}{c} X_0 = \left[\left(x_0^{t-his},\ \mathrm{y}_0^{t-his}\right) \ldots \left(x_0^{t-1},\ \mathrm{y}_0^{t-1}\right), \left(x_0^t,\ \mathrm{y}_0^t\right)\right] \\ X_s = \left[\left(x_s^{t-his},\ \mathrm{y}_s^{t-his}\right) \ldots \left(x_s^{t-1},\ \mathrm{y}_s^{t-1}\right), \left(x_s^t,\ \mathrm{y}_s^t\right)\right], \end{array} \tag{4}$$

The model output comprises the coordinates of the driving trajectory of the target vehicle in the future prediction domain length *pred*:

$$Y_0 = \left[\left(x_0^{t+1}, \mathrm{y}_0^{t+1}\right) \ldots \left(x_0^{t+2}, \mathrm{y}_0^{t+2}\right), \left(x_0^{t+pred}, \mathrm{y}_0^{t+pred}\right)\right], \tag{5}$$

3.2.2. Set Action Space

In the decision-making lane-changing problem of intelligent vehicles in this study, longitudinal velocity and lateral lane change need to be considered simultaneously. Therefore, we define the acceleration and steering wheel angle with continuous values as actor–network output $A[t]$: $\{a[t], \delta[t]\}$ [20].

Considering the longitudinal comfort of passengers, an acceleration output range within [–5, 5], and the tanh activation function were used in the output layer of the actor network to make the acceleration map output within [–1, 1]. Considering the actual situation of vehicle steering and lateral comfort of passengers, the wheel angle was limited in the range of [–30°, 30°], and finally, a tanh activation function was used to keep it within [–1, 1], which was convenient for model convergence. OU random processes with $\theta = 0.25$ and $\sigma = 0.2$ are added to the output action to avoid a decrease in generalization ability due to sensing errors.

3.2.3. Design Reward Function

The actor network of the agent extracts the current status information from the LSTM module. It then selects an action from the action space according to its strategy and receives the corresponding reward (or penalty) based on this action. This interactive process continues until a termination condition is met. The agent's goal is to achieve the maximum cumulative reward. Typically, the agent's actions are assessed using a reward function. Hence, designing an apt reward function is crucial to the efficacy of the LSTM-DDPG algorithm. In this study, we designed a modular reward function that prioritizes safety, efficiency, and comfort.

1. Safety

In terms of safety, autonomous vehicles must prioritize avoiding collisions with other vehicles on the road. It's essential to guide the vehicle in selecting the appropriate driving lane. Specifically, if the vehicle chooses a positive steering wheel angle while in the left lane (indicating a lane change to the left) or a negative steering wheel angle in the right lane (indicating a lane change to the right), then these scenarios indicate an aberrant steering

wheel angle. Consequently, a penalty value of −50 should be imposed in such instances. If a collision occurs during a lane change or while following another vehicle, a penalty of −200 should be assigned, ending the training. If the distance between vehicles falls below the safe margin given the current speed, then a penalty of −50 is imposed. In all other scenarios, a reward of +5 is given.

$$D_{safe} = vt + D_{-default}, \tag{6}$$

where D_{safe} denotes the safe distance for the vehicle at a given speed, v denotes the current vehicle speed, t denotes the velocity coefficient, and $D_{\text{-}default}$ denotes the initial safe distance.

$$r_1 = \begin{cases} -200 & if\ d < L_{vehicle} \\ -50 & if\ d < D_{safe} \\ -50 & if\ y > L_{lane},\ \delta \geq 0, \\ -50 & if\ y < L_{lane},\ \delta \leq 0 \\ +5 & else \end{cases} \tag{7}$$

where d denotes the distance between the ego and preceding vehicles, y denotes the lateral coordinate of the ego vehicle, δ denotes the front steering wheel angle of the ego vehicle, $L_{vehicle}$ denotes the length of the ego vehicle, and L_{lane} denotes the length of the ego vehicle.

2. Efficiency of Traffic

To ensure safety during lane changes, autonomous vehicles need to drive at an efficient pace without exceeding speed limits or changing lanes too frequently. Prolonged lane-changing maneuvers can reduce the efficiency of road usage, which would warrant a greater penalty.

$$r_2 = -\mathrm{dt}, \tag{8}$$

where dt denotes the simulation step size.

3. Comfort

In the real driving process, frequent speed changes affect the comfort of passengers in a vehicle. Therefore, a reward function for lane-change comfort was designed according to vehicle acceleration and jerk.

$$r_3 = \frac{1}{1 + |a| + |\Delta a|}, \tag{9}$$

where a denotes the acceleration of the vehicle, and Δa denotes its jerk.

4. Reward Function Ensemble

During the lane-change process, safety, lane-change efficiency, and comfort must be balanced. The total reward agent obtained for each time step is as follows:

$$R = \omega_1 r_1 + \omega_2 r_2 + \omega_3 r_3\ , \tag{10}$$

where $\omega_1, \omega_2, \omega_3$ denote the respective weight of the reward function involved in safety, lane-changing efficiency, and comfort. The safety weight mainly considers whether the vehicle complies with the traffic rules and whether there is a collision in the decision-making process. The weight of lane-change efficiency mainly considers the reasonable acceleration and deceleration in the process of vehicle running. The comfort weight mainly considers the acceleration and jerk of the vehicle. The greater the weight, the more the trained model emphasizes that particular factor. However, an excessively high weight can prevent the model from converging. The impact of the reward function on the policy network is intricate, and finding the optimal weight coefficient requires careful parameter tuning. Finally, weights for the reward function were designated as 5.0, 2.0, and 1.0.

3.3. Network Parameter Updating

DDPG consists of four networks: the online policy network, target policy network, online Q network, and target Q network, and the parameters of each network are updated alternately.

The critic network fits the action state value function according to the current state information s_t and expected action μ_t generated by the actor network. To enhance the accuracy of the action value evaluation by the critic network, the online Q-network parameter value θ^Q is updated by minimizing the loss function, which can be defined as:

$$L = \frac{1}{n}\sum_{i}^{n}\left(Q\left(s_i,\ a_i\middle|\theta^Q\right) - \mathrm{y}_i\right)^2, \tag{11}$$

$$\mathrm{y}_i = R_i + \gamma Q^-\left(s_i^{'}, \mu^-\left(s_i^{'}\middle|\theta^{\mu^-}\right)\middle|\theta^{Q^-}\right), \tag{12}$$

where n denotes the sample number of the batch sampling experience, R_i denotes the reward value of the experience sample i; γ denotes the discount factor, θ^Q denotes the parameter of the online Q-network, θ^{Q^-} denotes the parameter of the target Q-network, $Q\left(s_i,\ a_i\middle|\theta^Q\right)$ denotes the action value estimated using the online critic network, and $Q^-\left(s_i^{'}, \mu^-\left(s_i^{'}\middle|\theta^{\mu^-}\right)\middle|\theta^{Q^-}\right)$ denotes the future action value estimated using the target actor network and target critic network.

The actor network fits the policy function to generate the desired action μ_t based on the current state information s_t of the model input, and the online policy network parameter θ^μ updates the policy gradient expression as follows:

$$\nabla_{\theta^\mu}\mu \approx \frac{1}{n}\sum_{i}\ \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}\nabla_a Q\left(s, a\middle|\theta^Q\right)|_{s_i,\mu(s_i)}, \tag{13}$$

where $\mu(s|\theta^\mu)$ denotes a deterministic policy, and θ^μ denotes the online policy network parameter.

After each training iteration, gradients are utilized to update both online network parameters, while the target network parameters are updated using a soft update method. This method efficiently mitigates abrupt shifts and divergence in network gradient calculations, reduces large variations in network parameter updates, and promotes swift convergence during model training.

$$\begin{aligned}\theta^{\mu^-} &= \tau\theta^\mu + (1-\tau)\theta^{\mu^-}\\ \theta^{Q^-} &= \tau\theta^Q + (1-\tau)\theta^{Q^-},\end{aligned} \tag{14}$$

where θ^μ, θ^{μ^-} denote actor and target actor network parameters, respectively, θ^Q, θ^{Q^-} denote critic and target critic network parameters, respectively, and τ denotes the soft update coefficient.

4. Verification of LSTM-DDPG Algorithm

In this study, we utilized the MATLAB R2021a simulation platform to establish simulation scenarios and algorithmic models. A representative two-lane highway setting was chosen to compare the LSTM-DDPG algorithm, which incorporates trajectory prediction, against the conventional Conv-DDPG algorithm that lacks this feature.

4.1. Training Scene Construction

When training the LSTM-DDPG algorithm, we should pay attention to the computational complexity of the algorithm. LSTM is a special recurrent neural network (RNN) whose computational complexity depends mainly on the depth and width of the network. Although the computational complexity of the LSTM in time steps is constant (O(1)), the overall computational burden increases as the number of network layers increases. The

computational complexity of the DDPG mainly comes from updating the value function and optimizing the strategy. Therefore, the computational complexity of LSTM-DDPG will be the sum of the two, which can lead to challenges in real-time simulations. Considering the computational complexity of the LSTM-DDPG algorithm, the scenario shown in Figure 2 is set up.

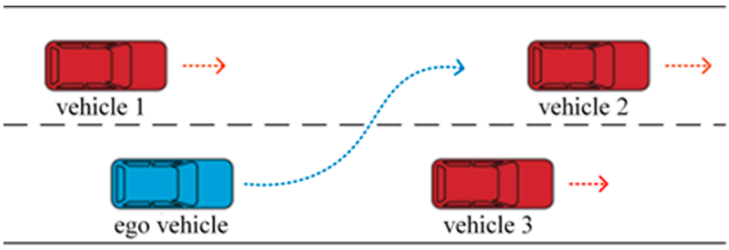


Figure 2. Scene feature description.

Two training scenarios are selected to train the LSTM-DDPG algorithm. In scenario 1, there are two obstacle cars, whose positions are left behind and right in front. Scenario 2 has three obstacle vehicles, whose positions are left rear, left front, and right front. In the chosen scenario, the ego vehicle starts in the right lane. The position of the three obstacle cars is left rear, left front, and right ahead. However, the initial conditions of the obstacle vehicles (such as relative distance and speed in relation to the ego vehicle) adhere to specific constraints. The ego vehicle initiates at a speed of 65 km/h, with a maximum allowable speed of 100 km/h. The speed of obstacle car 1 is randomly generated in the range of 60–70 km/h, the speed of obstacle car 2 is randomly generated in the range of 70–80 km/h, and the speed of obstacle car 3 is randomly generated in the range of 60–70 km/h, and the initial distance from the car is 25 m.

During training, the initial parameters for the network training were set as follows: a learning rate of 0.005, batch size of 128, maximum of 500 iterations, and gradient threshold of 1. The Adam optimizer was chosen for the training process. The actor and critic networks in DDPG utilize a multi-layered fully connected structure. The actor network had neuron counts of 8, 256, 256, and 2, while the critic network comprised 10, 256, 256, and 1. The learning rates for the actor and critic networks were established at 0.001, and a discount factor of 0.99 was applied. Minibatch gradient descent was employed for training, with a batch size of 128 and cap of 1400 training epochs. The simulation operated at timesteps of 0.1 s.

4.2. LSTM Trajectory Prediction

4.2.1. Trajectory Preprocessing

In this study, we utilized MATLAB to design traffic scenarios on a straight roadway with varying densities: three cars and four cars. For each condition, the sequence of vehicle trajectory coordinates was captured at every time step to facilitate model training. The original data was captured at a frequency of 10 Hz, but was subsequently resampled at 5 Hz to ensure the retention of critical features in the dataset. For the processed data, an 8 s sliding window was employed to segment and generate data samples. Within this 8 s span, the initial 3 s served as historical data input to the model, while the succeeding 5 s acted as the reference for future trajectory predictions. The dataset was then categorized into training, validation, and testing subsets. Specifically, the initial 70% of samples were earmarked for training, the following 10% (from 70% to 80%) for validation, and the final 20% (from 80% to 100%) for testing.

Before employing the LSTM for trajectory prediction, it is essential to normalize the trajectory coordinates. This involves calculating the mean and standard deviation of the dataset and transforming it into a standardized dataset with a mean of one and variance of

one. To better support the subsequent phased training of the LSTM network, datasets are stored using the cell-type data format.

To assess the accuracy of the predictions, the root−mean−square error (RMSE) was utilized to measure the deviation between the predicted and actual trajectories.

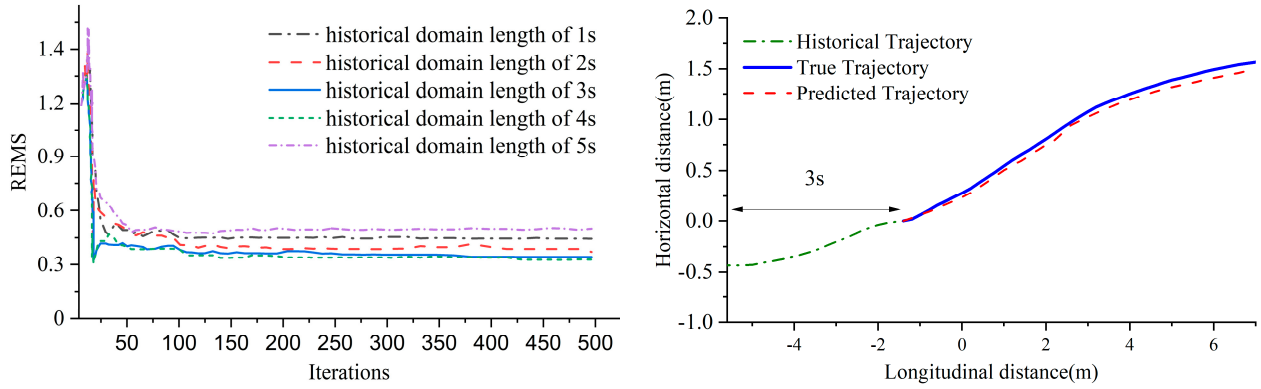
$$RMSE = \sqrt{\frac{1}{pred}\sum_{t=t_n+1}^{t_n+pred} \delta_t^T \delta_t}, \tag{15}$$

$$\delta = \left(x - \mu_x, \mathrm{y} - \mu_\mathrm{y}\right)^T, \tag{16}$$

where t_n denotes the prediction start time, (x, y) denote the coordinates of the real trajectory points, and the actual predicted trajectory coordinates are represented by the mean of the future trajectory distribution output (μ_x, μ_y) of the model.

4.2.2. Effect of Historical Duration

The appropriate input length for the LSTM network is crucial. If the input length is too brief, then it may not adequately capture the inherent mathematical characteristics of the data, leading to a significant decline in trajectory prediction accuracy. To delve deeper into how the duration of historical input trajectory impacts the extraction of vehicle interaction features by the trajectory prediction model, both qualitative and quantitative analyses of the model's performance under varying input durations were conducted. The LSTM model was fed historical trajectory inputs of 1 s, 2 s, 3 s, 4 s, and 5 s. The RMSE values recorded during the training process are illustrated in Figure 3a. Upon analysis, the average RMSE values for the durations from 1 s to 5 s were 0.05457, 0.05324, 0.05115, 0.05067, and 0.05575, respectively.



(**a**) root−mean−square error (**b**) vehicle trajectory

Figure 3. Curve of root−mean−square error input for different historical track lengths and comparison between the predicted trajectory and true trajectory.

Figure 3a demonstrates that the model begins to show signs of convergence around 50 steps for various input history lengths. As the input history field length ranged from 1 s to 4 s, there was a noticeable reduction in the deviation between the predicted and actual trajectories as the input historical trajectory of the model increased. This suggests that extending the historical trajectory can enhance the extraction of vehicle interaction features. However, when the length of the history field extended to 5 s, the predictive trajectory's error began to increase, even surpassing the prediction inaccuracies observed at history field lengths of 1 s and 2 s. This indicates that an excessively long historical trajectory input may distort the interaction features the model seeks to extract. Analyzing the average RMSE value, the smallest error occurred at a model length of 4 s, indicating high predictive accuracy at this input length. Nevertheless, the prediction error for a 4 s input length was only marginally better than that for a 3 s input length, but it demanded

more time and computational resources. When balanced against these factors, a history length of 3 s proved to be the most suitable model input.

As depicted in Figure 3b, the predicted vehicle trajectory coordinates align closely with the actual trajectory. The minimal prediction error suggests that the LSTM module developed in this study is effective in forecasting the genuine trajectory. Moreover, the trained encoder is well-suited for integration into the subsequent DDPG reinforcement learning decision-making model.

4.3. Comparison and Analysis

Based on the LSTM trajectory prediction module of the previous step, the representative states in the predicted trajectory and status module of the surrounding vehicles were considered as part of the input data of the LSTM-DDPG. During the simulation, the ongoing training episode was halted upon a collision with or overtaking of the lead vehicle. Subsequently, the virtual driving environment was reinitialized, and a fresh training episode commenced until the predefined training epochs were fulfilled. Tables 2 and 3 provide detailed information on the DDPG's network architecture and hyperparameter settings.

Table 2. DDPG network structure.

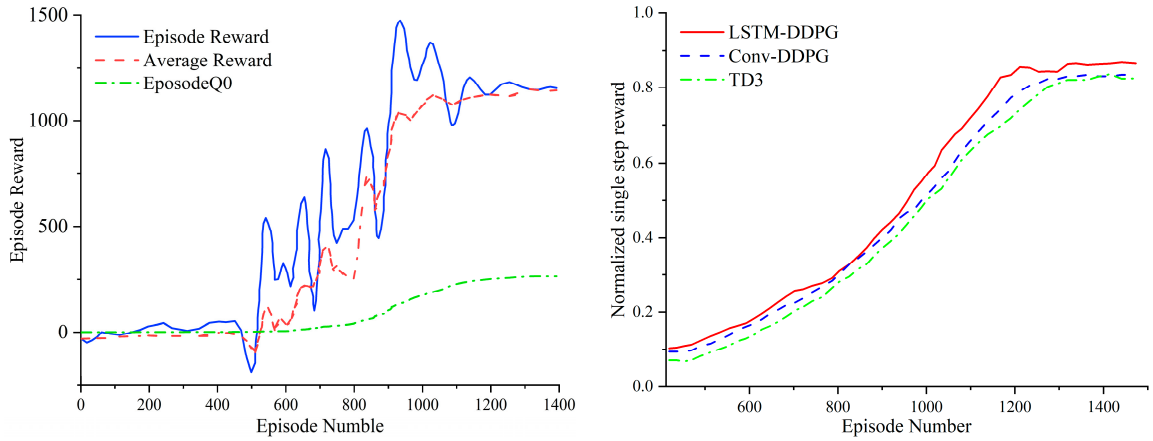
Parameters	Layer	Dimension	Activation Function
Critic network	fully connected layer 1	(10, 256)	ReLU
	fully connected layer 2	(256, 256)	ReLU
	fully connected layer 3	(256, 1)	ReLU
Actor network	fully connected layer 1	(8, 256)	ReLU
	fully connected layer 1	(256, 256)	ReLU
	fully connected layer 1	(256, 2)	ReLU

Table 3. Training hyperparameter settings for DDPG.

Parameter	Parameter Values	Parameter	Parameter Values
Network learning rate	0.001	Sample size	64
Discount factor	0.99	Experience pool size	10,000
Gradient threshold	0.001	Maximum training period	1400

In the DDPG algorithm's learning journey, both episode reward and average reward serve as indicators of training convergence and the overall efficacy of learning. Figure 4a illustrates the training progression of the enhanced LSTM-DDPG. During the initial stages of training, the agent struggled to make appropriate lane-changing decisions. However, a significant surge in the average reward was observed between the 600th and 1000th training periods. Furthermore, this average reward largely remained consistent in the latter phases, suggesting that over time, the vehicle became adept at opting for actions with more favorable reward values, thus making better lane-changing choices.

It is necessary to balance safety, traffic efficiency, and comfort in lane-change decisions of autonomous vehicles. The curve of the average reward value normalized by a single step in the training process is smoothed, and the change curve is shown in Figure 4b. It can be observed that the reward value of the LSTM-DDPG algorithm gradually converges to 0.89 after about 1200 rounds of training. After about 1260 training rounds, the reward value of the Conv-DDPG algorithm gradually converges to 0.83. After about 1300 training rounds, the reward value of the TD3 algorithm gradually converges to 0.82. By comparison, it is found that the LSTM-DDPG algorithm has the fastest convergence speed, and the reward return is also improved.



(**a**) reward value (**b**) normalization

Figure 4. Total reward value during training and comparison of normalized single-step reward returns.

As can be seen from Figure 4, with the continuous increase in the number of training rounds, the total reward value and the single-step reward value during training gradually become stable, which also reflects some indicators in the reward and punishment function. The reward and punishment functions are designed according to safety, traffic efficiency, and comfort, respectively. When the distance between the car and the front car, lane-changing situation of the car, speed, and acceleration of the car change greatly, the final reward value will be affected. Therefore, the stabilization of the reward value can reflect that the speed, acceleration, and traffic efficiency of the vehicle are also stable.

The average speed, average jerk value, and maximum steering wheel angle of the three algorithms in the process of lane change are shown in Table 4. It can be seen from the table that the LSTM-DDPG strategy algorithm performs better than the Conv-DDPG and TD3 strategy algorithm. The LSTM-DDPG strategy algorithm can change lanes with a smaller steering wheel angle, smaller jerk value, and higher speed, which not only ensures lane-change efficiency but also ensures comfort. The LSTM-DDPG strategy algorithm can run stably in different scenarios.

Table 4. Comparison of algorithms.

	Average Velocity Speed (km/h)	**Average Jerk Value ($m \cdot s^{-3}$)**	**Maximum Steering Wheel Angle (°)**
LSTM-DDPG (Scenario1)	76.3	0.6	18.1
LSTM-DDPG (Scenario2)	77.5	0.7	18.1
Conv-DDPG	72.4	1.1	21.2
TD3	72.6	1.0	20.6

5. Conclusions

A lane-changing model was conceptualized using a Markov decision process. This model integrated LSTM-based trajectory predictions and pertinent state information into the actor–critic framework of the DDPG. Additionally, a multifaceted reward function specifically tailored for autonomous lane-changing was devised. Through iterative training guided by this reward function, the system can discern the optimal decision for autonomous lane-changing, considering the future state of vehicles it interacts with.

1. Prediction of surrounding vehicles using LSTM: After the training of the data set, the LSTM module built can finally predict the relatively accurate future vehicle trajectory, and the prediction results are shown in Figure 3. At the same time as

the state input of the DDPG algorithm, the vehicle can make better decisions on the lane-change trajectory.

2. The LSTM-DDPG algorithm also performs well during training. As shown in Figure 4, both round reward and average reward increase significantly after the training cycle reaches 600 cycles. After the training cycle reaches 1000 cycles, the reward reaches an undetermined trend, and the vehicle can choose a better action to change lanes. The convergence speed of the LSTM-DDPG algorithm is faster than that of the Conv-DDPG algorithm.
3. As can be seen from the data in Table 3, the LSTM-DDPG algorithm proposed in this paper can perform lane changes with faster speed and smaller acceleration in different scenarios. The vehicle can be more efficient and more comfortable while completing the task of a lane change.
4. In the LSTM-DDPG algorithm, LSTM can help the DDPG algorithm capture and exploit this long-term dependency. When the driving scene changes, the LSTM-DDPG algorithm can update the strategy by learning the dynamic characteristics of the environment to provide continuous action space. The LSTM-DDPG algorithm may provide more robust decision strategies by learning long-term dependencies and continuous action spaces.

Author Contributions: Conceptualization, Z.Y. and H.W.; methodology, H.W. and Y.W.; software, H.W. and W.Z.; validation, Z.Y. and Y.W.; formal analysis, investigation, resources, Z.Y., H.W. and Y.W.; data curation, Z.W. and Y.W.; writing—original draft preparation, Z.W.; writing—review and editing, Z.Y., H.W. and Z.W.; visualization, H.W.; supervision, project administration, Z.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the following funders. Hubei Province key research and development project, grant number 2023BAB169. Major science and technology project in Wuhan, Hubei Province, grant number 2022013702025184. The central government guided special projects for local scientific and technological development, grant number 2022BGE248. Hubei University of Automotive Technology PhD Fund Project, grant number BK202215.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

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

References

1. Cesari, G.; Schildbach, G.; Carvalho, A.; Borrelli, F. Scenario model predictive control for lane change assistance and autonomous driving on highways. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 23–35. [CrossRef]
2. Yang, Z.; Gao, Z.; Gao, F.; Wu, X.; He, L. Lane changing assistance strategy based on an improved probabilistic model of dynamic occupancy grids. *Front. Inf. Technol. Electron. Eng.* **2021**, *22*, 1492–1504. [CrossRef]
3. He, Y.; Feng, J.; Wei, K.; Cao, J.; Chen, S.; Wan, Y. Modeling and simulation of lane-changing and collision avoiding autonomous vehicles on superhighways. *Phys. A Stat. Mech. Its Appl.* **2023**, *609*, 128328. [CrossRef]
4. Moon, H.; Kang, S.H.; Eom, J.; Hwang, M.J.; Kim, Y.; Wang, J.; Kim, B.; Kim, T.; Ga, T.; Choi, J.; et al. Autonomous Robot Racing Competitions: Truly Multivehicle Autonomous Racing Competitions [Competitions]. *IEEE Robot. Autom. Mag.* **2024**, *31*, 123–132. [CrossRef]
5. Selvi, V.T.; Dhurgadevi, M.; Prithiviraj, P.; Jackulin, T.; Baskaran, K. Extending the FSM Model for Critical Decision-Making and Safety Control in Autonomous Vehicles. *Int. J. Intell. Syst. Appl. Eng.* **2024**, *12*, 397–410.
6. Perez, J.; Milanes, V.; Onieva, E.; Godoy, J.; Alonso, J. Longitudinal fuzzy control for autonomous overtaking. In Proceedings of the IEEE International Conference on Mechatronics, Istanbul, Turkey, 13–15 April 2011; pp. 188–193.
7. Erke, S.; Bin, D.; Yiming, N.; Qi, Z.; Liang, X.; Dawei, Z. An improved A-Star based path planning algorithm for autonomous land vehicles. *Int. J. Adv. Robot. Syst.* **2020**, *17*, 1729881420962263. [CrossRef]
8. Moghadam, M.; Elkaim, G.H. A hierarchical architecture for sequential decision-making in autonomous driving using deep reinforcement learning. *arXiv* **2019**, arXiv:1906.08464.
9. Van Hoek, R.; Ploeg, J.; Nijmeijer, H. Cooperative driving of automated vehicles using B-splines for trajectory planning. *IEEE Trans. Intell. Veh.* **2021**, *6*, 594–604. [CrossRef]

10. Kendall, A.; Hawke, J.; Janz, D.; Mazur, P.; Reda, D.; Allen, J.M.; Lam, V.D.; Bewley, A.; Shah, A. Learning to drive in a day. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 8248–8254.
11. Liang, X.; Wang, T.; Yang, L.; Xing, E. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 584–599.
12. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An open urban driving simulator. In Proceedings of the Conference on Robot Learning, PMLR, Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.
13. Karimzadeh, M.; Aebi, R.; de Souza, A.M.; Zhao, Z.; Braun, T.; Sargento, S.; Villas, L. Reinforcement learning-designed LSTM for trajectory and traffic flow prediction. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
14. Meng, L.; Gorbet, R.; Kulic, D. Memory-based deep reinforcement learning for pomdps. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 5619–5626.
15. Cui, Y.; Huang, X.; He, P.; Wu, D.; Wang, R. A two-timescale resource allocation scheme in vehicular network slicing. In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–5.
16. Bertsekas, D. *Reinforcement Learning and Optimal Control*; Athena Scientific: Nashua, NH, USA, 2019.
17. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
18. Haarnoja, T.; Zhou, A.; Hartikainen, K.; Tucker, G.; Ha, S.; Tan, J.; Levine, S. Soft actor-critic algorithms and applications. *arXiv* **2018**, arXiv:1812.05905.
19. Perot, E.; Jaritz, M.; Toromanoff, M.; De Charette, R. End-to-end driving in a realistic racing game with deep reinforcement learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 3–4.
20. An, H.I.; Jung, J. Decision-making system for lane change using deep reinforcement learning in connected and automated driving. *Electronics* **2019**, *8*, 543. [CrossRef]