A decision control method for autonomous driving based on multi-task reinforcement learning

Cai Yingfeng¹, Yang Shaoqing¹, Wang Hai², Teng Chenglong¹, Chen Long¹

¹Automotive Engineering Research Institute, Jiangsu University, Zhenjiang, Jiangsu, People’s Republic of China
²School of Automotive and Traffic Engineering, Jiangsu University, Zhenjiang, Jiangsu, People’s Republic of China

Corresponding author: Wang Hai (e-mail: wanghai1019@163.com).

This work was supported in part by the National Natural Science Foundation of China (U20A20333, 51875255), Key Research and Development Program of Jiangsu Province (BE2019010-2, BE20200083-3), Jiangsu Province’s six talent peaks (TD-GDZB-022).

ABSTRACT Following man-made rules in the traditional control method of autonomous driving causes limitations for intelligent vehicles under various traffic conditions that need to be overcome by incorporating machine learning-based method. The latter is inherently suitable for simple tasks of autonomous driving according to its limited characteristic under complex multi-lane traffic conditions. In this paper, a decision control method is proposed based on multi-task reinforcement learning to address the shortcomings of autonomous driving control under complex traffic conditions. Herein, the autonomous driving task is divided into several subtasks utilizing the proposed method to reduce the training time and improve traffic efficiency under complex multi-lane traffic condition. To ensure the efficiency and robustness of agent convergence to the optimal action space, an adaptive noise exploration method is designed for the subtasks with convex characteristics. Five-lane driving tasks scenarios embedded in Carla simulator have been conducted to verify the proposed method. The results of the simulation draw the conclusion that the proposed method increases the driving efficiency of intelligent vehicles under complex traffic conditions.

INDEX TERMS Autonomous driving, reinforcement learning, multi-task, exploration method

I. INTRODUCTION

Autonomous driving technology has been rapidly developed over the past few decades [1]. It has enormous potential not only economically but also in improving traffic efficiency and driving safety. The majority of autonomous driving community, including the academy and the industry, is focusing on more intelligent, safe and reliable autonomous driving technology. Despite the uncertainty as to when the autonomous driving technologies will be publicly available, they attracted enormous attentions in different sectors in the upcoming decades [2-3].

Up to now, the rule-based method that is a closed-loop system has been the most popular autonomous driving control method [1]. Systems based on man-made rules need to have a comprehensive consideration of traffic information such as pedestrians and vehicles to control intelligent vehicles appropriately according to the decision made by the results. The intelligent control method is a free man-made rules system based on neural networks to direct use the traffic information to control. To extract the features of traffic environment located in sensor bords of intelligent vehicle, the intelligent control method using the machine learning method to train a control model that has good effect on controlling nonlinear systems. With the support of sufficient data, this method can better adapt to complex traffic, but how to train such neural networks in a short time is a problem that needs to be solved urgently. The good results in the DARPA urban challenge brought researchers’ attention to the implementation of rule-based method into autonomous driving introduced by Montemerlo et al [4]. Ziegler j et al. [5] developed the rule-based method by designing a hierarchical finite state machine for autonomous driving tasks. In 2018, the team of Baidu Apollo proposed the classical EM motion planner based on double-stream states machine [6], in which upstream is the scenario manager and downstream is the combination of a series of small tasks.
Although the rule-based method is popular, the characteristic of finite state exposes its cognitive limitations which causes difficulties for this method to address the complex traffic tasks with enormous scenarios.

The intelligent control method has been proposed with some researchers [7,8,24,25] to deal with the limitations of the rule-based methods in complex dynamic scenarios using machine learning. Machine learning obtain the deep representation of the complex dynamic scenarios. Different from the rule-based method, this excellent ability of machine learning can avoid the complexity of manually selecting features and the dimension disaster of high-dimensional data. An intelligent control method is imitation learning using examples provided by experience drivers. When experienced drivers are driving, the surrounding environment information is recorded as input and the corresponding actions are used as labels for classification or regression. The trained model can approximate the behavior of experienced drivers. As early as 1988, the Navlab project used a simple fully connected network to map the front camera image and the steering angle [9]. In 2016, NVIDIA perfected this solution by using the forward-looking camera and CNN network [10]. Although imitation learning has achieved great success [9-13], it cannot deal with environments that experienced drivers have not encountered. It is inseparable from the sample data and cannot surpass an experienced driver. As a fashionable machine learning method, reinforcement learning obtains the optimal solution by constantly exploring the unknown environment [14]. By setting reward, a reinforcement learning agent can learn from the previous reward information to achieve a higher reward for the future behavior. With the continuous exploration and interaction with the environment, the agent can establish the optimal strategy model which suit for various scenarios [15].

The policy gradient algorithm has been widely used in the continuous action space in the reinforcement learning field. It constructs the policy network $\pi(a|s, \theta)$ with parameters $\theta$, in which input is the state $s$ and output is the action $a$. The direct policy search method parameterizes the strategy and optimizes network parameters for maximizing the expected cumulative reward. It can more efficiently solve the continuous action selection problem compare to the value-based method [15-17]. The policy search method designs an objective function about the policy and optimizes the parameters of the agent using the gradient descent algorithm to maximize the cumulative reward. The optimal policy is obtained by convergence after many iterations. As the classical policy search method, the TRPO algorithm proposed by John Schulman et al. [18] solves the selection of step size of the other search method. The Actor-Critic algorithm proposed by Deep mind [16,17] combines the policy search and the value-based methods to solve slow learning efficiency of the basic policy search method and improving training efficiency by updating the gradient at each step. In many studies, the strategy search method was applied to the autonomous driving task to realize the autonomous driving task in specific scenarios [19-23]. The particular car following task was studied in researches proposed by M Zhu et al. in [19,20], and the Actor-Critic algorithm was used to complete the lane keeping task in a short time [24,25]. However, the design of reward function is complicated due to the complex traffic conditions. The reinforcement learning method based on the value function evaluates the value function, selecting the most valuable action to execute according to the action value. When the model is unknown, the value function is estimated by the Monte Carlo method with random samples. Sutton et al. [26] proposed a temporal difference method to solve the low efficiency caused by updating the agent parameters at every episode. Before the policy search method, Deep mind proposed the value-based DQN algorithm which introduced neural networks into Q-learning [27]. The combination of reinforcement learning and deep learning solves the problem that Q-learning is difficult to solve in large state space. Although the value-based method does not deal with continuous action space well, it converges faster in finite discrete action space. An application of reinforcement learning in autonomous driving is to use photos and radar point cloud as the input of neural network [28-30], and the steering angle, brake, and throttle as the output. Although this type of end-to-end reinforcement learning method can complete specific autonomous driving tasks, it is challenging to meet them under complex traffic conditions. Moreover, there are some other problems in the application of reinforcement learning in autonomous driving, such as intricate design of reward function and long training time.

This paper focuses on solving the shortcoming of previous autonomous driving controllers under complex traffic conditions. The complex autonomous driving task is simplified by multiple sub tasks which applied with different algorithms. The main contributions of this work can be summarized as follows:

1. Proposing a multi-task reinforcement learning method for autonomous driving controller to overcome the shortcoming of the previous methods under complex traffic conditions. In experiments, we demonstrate that this method can improve traffic efficiency under complex multi-lane traffic conditions.
2. For different task characteristics, the targeted reward function and reinforcement learning algorithm are proposed for different task to reduce the training difficulty and time.
3. An adaptive noise exploration paradigm for convex tasks is designed, which balance the opposition relationship between exploration and utilization by using multi heads actor network.

II. Problem Formulation
The main objective of this paper is to solve the decision control task of autonomous driving under multi-lane conditions. An autonomous driving task is approximated as a Markov decision process (MDP) defined by the state space \( S \), action space \( A \), reward function \( r(s, a, r, s\_{t+1}) \), and state transfer equation \( P(s_{t+1} | s_t, a_t) \). The state transition equation \( P(s_{t+1} | s_t, a_t) \) is unknown in the model-free reinforcement learning algorithm, and the next state \( s_{t+1} \) is obtained from the interaction with the environment. In each interaction with the environment, an agent gets a tuple \( \langle s_t, a_t, r_t, s_{t+1} \rangle \). The autonomous driving task can be described as finding an optimal policy to maximize the cumulative reward expressed by \( R_\gamma = \sum_{t=0}^{T} \gamma^t r(s_t, a_t) \), where \( \gamma \) denotes the discount factor \( \gamma \in (0, 1) \). In the learning process, a vehicle continuously interacts with the traffic environment to obtain experience information and motivate the agent network to update.

For a complex task, the task can be artificially decomposed into multiple subtasks denoted as \( \{M_1, M_2, ..., M_k\} \) according to the task characteristics. This paper assumes that agent complete subtasks only through the action of brake, throttle, and steering angle at the bottom layer without conducting the decision-making task. The brake, throttle, and steering angle of the bottom layer are regarded as continuous actions. The value-based reinforcement learning algorithm discretizes similar continuous actions. But rough discretization of constant actions of the brake throttle and steering angle is not conducive to driving safety and comfort. In contrast, fine discretization results in huge action space, which is not conducive to the convergence of a value network. Therefore, the policy gradient algorithm is a better choice to solve the subtasks than the valued-based algorithm. The Actor-Critic algorithm is a mature policy gradient reinforcement learning algorithm. The predecessor of the Actor is the PG algorithm, which can be updated only in every episode. After using a value-based algorithm as the Critic, the Actor-Critic algorithm can use the temporal difference method to realize the step update.

As a type of Actor-Critic algorithm, the DDPG has the advantages of fast convergence and good stability after introducing a double-network structure [15]. In this work, the DDPG algorithm is to train the agent of a subtask. The DDPG algorithm framework includes four sub networks: main policy network \( \mu(s | \theta) \), target policy network \( \mu'(s | \theta') \), main value network \( Q(s, a | \omega) \), and target value network \( Q'(s, a | \omega') \). The N-step temporal difference method is used instead of the cumulative reward, the action value is estimated by the target value network \( Q'(s, a | \omega) \) and target policy network \( \mu'(s | \theta') \) as follows:

\[
y_t = \sum_{k=0}^{N-1} \gamma^k r_{t+k} + \gamma^N Q'(s_{t+1}, \mu'(s_{t+1} | \theta'))
\]

The loss function of the main value network \( Q(s, a | \omega) \) is given by:

\[
L = \frac{1}{n} \sum (y_t - Q(s_t, a_t | \omega))^2
\]  

The update gradient of the main policy network \( \mu(s | \theta) \) is chosen as:

\[
\nabla_{\theta} J \approx \frac{1}{n} \sum_{i=0}^{n} \nabla_s Q(s_t, a_t | \omega)_{t=0, \ldots, t=K(s_i, a_i)} \nabla_{\theta} \mu(s | \theta)_{i=0, \ldots, t=K(s_i, a_i)}
\]

The parameters of the main network are updated iteratively to the target network in real-time by:

\[
\begin{align*}
\theta' &= \theta + (1 - \tau) \theta' \\
\omega' &= \omega + (1 - \tau) \omega'
\end{align*}
\]

where \( \tau \) denotes the update rate, and \( \tau \in (0, 1) \).

The purpose of the target network is to improve the stability of the training.

In the process of interaction with the environment, the main policy network outputs the desired action \( \mu(s_i | \theta) \) according to the current environment state \( s_i \). In order to improve the agent’s ability to explore the environment, the agent executes the actual action \( a_i \) by adding noise \( N_i \) to the expected action \( \mu(s_i | \theta) \). The noise \( N_i \) is expected to tend to zero with the convergence of the network, which is expressed as:

\[
a_i = \mu(s_i | \theta) + N_i
\]
method, this paper designs an actor network.

The output of the main network connects multi head networks. To reflect the differences of each head network, the experience pool sampled by the training of each head network is independent. Due to the different initialization and sampling of each head network, the way to converge to the optimal solution is also different. Therefore, we use the variance of the output action of the head network to estimate this uncertainty measure:

\[ N_i = \begin{cases} 
  k \cdot \text{var}(\alpha(s)) & \text{if } k \cdot \text{var}(\alpha(s)) < N_{\text{threshold}} \\
  N_{\text{threshold}} & \text{if } k \cdot \text{var}(\alpha(s)) \geq N_{\text{threshold}} 
\end{cases} \quad (6) \]

where \( k \) is the scale factor. In the early stage of training, the action \( \alpha \) output from each head network varies greatly leading to large noise \( N_i \). The agent frequently will explore the edge value of the action space. Because of this situation, we set the threshold \( N_{\text{threshold}} \) to ensure the exploration in the early stage. Correspondingly, each episode randomly selects ahead action \( \mu_{\text{random}} \) to add noise \( N_i \) and output:

\[ \alpha_i = \mu_{\text{random}}(s_i | \theta) + N_i \quad (7) \]

The subtasks with a parallel relationship, such as lane keeping and left or right lane changing tasks, are summarized into the same set \( \{M_1, M_2, \ldots, M_m\} \). To achieve better completion of subtasks, a behavior agent needs to be trained to select an optimal subtask \( M \) according to the current environmental state \( s \). Therefore, the training of behavioral agent represents a classical reinforcement learning task with a limited action space. The DQN series algorithm introduces a neural network into the value-based reinforcement learning algorithm to solve a too-large state space with a limited action space [23]. The value of a subtask \( M \) in the environment state \( s \) is defined as \( Q(s, M | \omega) \). The behavior agent judges the most valuable subtask \( M_i = \arg \max(Q(s, M | \omega)) \) and forwards it to the subtask agent to realize the actual action. In the same way, the N-step temporal difference method is used to estimate the target value \( y \) of each subtask as follows:

\[ y_i = \begin{cases} 
  R_i & \text{if } s_i \text{ is the end state} \\
  \sum_{k=0}^{N-1} \gamma^k R_{i+k} + \gamma^n Q(s_{i+1}, M_i | \omega) & \text{else} 
\end{cases} \quad (8) \]

Also, \( n' \) groups of samples are randomly selected from the experience pool to train agents in real-time. All parameters \( \omega \) of the value network \( Q(s, M | \omega) \) are updated by the gradient back propagation method of the neural network using the mean square error loss \( L' \), which is defined as:

\[ L' = \frac{1}{n'} \sum_{i=1}^{n'} (y_i - Q(s_i, M_i | \omega))^2 \quad (9) \]

Similarly, the soft update was used to update the parameters of the main value network to the target value network in real-time to stabilize the training process, which can be expressed as follows:

\[ \omega' = \tau \omega + (1 - \tau) \omega' \quad (10) \]

III. Design of multi-task training model

This section described the proposed method in detail. As shown in Figure 2, the decomposed subtasks include lateral and longitudinal control tasks and decision-making tasks. The lateral control tasks are divided into lane keeping and left-right lane changing tasks, and the longitudinal task denotes the adaptive cruise task. The lateral and longitudinal control tasks are performed by steering wheel angle, brake, and throttle, which corresponds to the continuous action space. The Actor-Critic algorithm is used for subtask training, and it combines the policy gradient method and value-based method. The split lateral and longitudinal subtasks are considered that the optimal solution is unique. The adaptive noise exploration method is realized by using multi head actor, which reduces the invalid exploration of agent as much as possible and increases the utilization of the action. The characteristic of decision-making task is the discreteness of action space, in which each behavior can be regarded as an action. The purpose of the decision-making network is to select the most valuable behavior according to the current environmental information.

\[ \omega' = \tau \omega + (1 - \tau) \omega' \quad (10) \]

During the training process in subtasks, the task training has a dependency relationship with each other. The training of the lane changing task and longitudinal control task is based on the execution of the lane keeping task. Similarly, the training of the decision-making task is based on the longitudinal and lateral control tasks that can be executed. The specific training process is shown in Figure 3.
The cumulative reward for reinforcement learning enables the agent to consider potential threats. According to the environmental state, the agent chooses the most valuable action for the current and future. But it is difficult for the agent to understand the current environment by the current state, especially an intelligent vehicle with strong dynamic characteristics. The dynamic characteristics of intelligent vehicles are often hidden in the past few moments [31,32]. The states of the current moment and several past moments are taken as the network’s input to help the agent understand the dynamic characteristics better. The experimental results show that the continuous states of four moments are enough to demonstrate the dynamic characteristics of an intelligent vehicle.

A. LATERAL CONTROL TASK

1) LANE KEEPING TASK

To stabilize the intelligent vehicle driving in lane without deviation, the objective of lane keeping task is to minimize the lateral error and heading deviation under various curvature roads and vehicle speeds. Therefore, both vehicle speed and road information need to be considered when training lane keeping task [33, 34]. The road information only is considered in the lane keeping task, while the other vehicle information is neglected. The lateral error and heading angle deviation at the preview point should be considered to make the agent pay attention to the road information. And the vehicle speed is also a key factor to be considered to keep the vehicle on the line at different speeds. The lateral error and heading angle deviation at the preview point denote the measure of vehicle lateral control.

As shown in Figure 4, the coordinates of the navigation points \((x_i, y_i)\), heading deviation \(\varphi\), vehicle speed \(v\), and acceleration \(\dot{v}\) are taken as parameters of the current states, which is expressed as:

\[
s_{\text{lateral keep}} = \langle x_0, y_0, x_1, y_1, \ldots \varphi, v, \dot{v} \rangle
\]  

(11)

The action of lane keeping task is the steering angle \(a_{\text{steer}} \in [-1,1]\). In the design of the reward function of the lane keeping task, the lateral error and heading angle deviation are used as evaluation indexes of the lane keeping task, which is expressed as:

\[
 r_{\text{lateral keep}} = -k_1 \cdot \text{abs}(x_0) - k_2 \cdot \sin \varphi
\]  

(12)

If the lateral error of the current position \(x_0\) is larger than the predefined maximum lateral error \(x_{0\text{max}}\), the episode of iterative training is ended, and the next episode of training is performed.

2) LANE CHANGING TASK

Like the lane keeping task, lane information and vehicle speed need to be considered when training the lane changing task. Additionally, lane width needs to be considered due to lane changing trajectories and should be different for lanes of different widths. The current state of the lane changing task is shown in Figure 5, and it is defined as:

\[
s_{\text{lateral change}} = \langle x_0, y_0, x_1, y_1, \ldots \varphi, v, \dot{v}, \omega \rangle
\]  

(13)

The steering angle \(a_{\text{steer}} \in [-1,1]\) is considered as the action output.

The reward function designed for the lane keeping task is expressed as follows:
\[ r_{\text{lane\_change}} = -\frac{k_3}{\text{abs}(d_{\text{change}})} \sin\varphi - k_4 \cdot \text{abs}(d_{\text{change}}) \]  
(14)

where \( \varphi \) denotes the heading deviation between the lane and the vehicle heading direction, and \( d_{\text{change}} \) is the distance between the left (right) lane and the vehicle centroid.

The lane changing task is initiated as follow:

In the early stage of the lane changing, the current lane and the left (right) lane are parallel, \( \varphi \) is about zero, and \( d_{\text{change}} \) is maximum. At this time, \( \frac{k_3}{\text{abs}(d_{\text{change}})} \sin\varphi \) is about zero, and has little effect on \( r_{\text{lane\_change}} \), \( \text{abs}(d_{\text{change}}) \) has a greater effect on \( r_{\text{lane\_change}} \). So the reward \( r_{\text{lane\_change}} \) will promote the agent to make the corresponding action focusing on how to reduce \( d_{\text{change}} \).

In the later stage of the lane changing, \( \frac{k_3}{\text{abs}(d_{\text{change}})} \sin\varphi \) will increase with the gradual decrease with \( d_{\text{change}} \) and \( \varphi \), which will increase its effect on \( r_{\text{lane\_change}} \). At this stage, reward \( r_{\text{lane\_change}} \) will promote the agent to reduce both \( \varphi \) and \( d_{\text{change}} \) when making action.

### B. LONGITUDINAL CONTROL TASK
The longitudinal control task is defined as follows. The task objectives are divided into two parts for the two cases. In the first case, there is a vehicle within a certain distance in front of the controlled vehicle. Keeping an ideal distance from the vehicle in front without collision as the objective for the first case. Unlike the first case, there are no other vehicles and obstacles in front of the controlled vehicle in the second stage. Driving with a constant speed as the objective for the second case. In order to design a reward function for the above cases, it is necessary to assume that there is a virtual vehicle driving far ahead with a constant speed in the second case. According to the above description, the speed and distance information of the controlled vehicle and the front vehicle should be considered as the state for the longitudinal control task. As shown in Figure 6, the longitudinal control task takes the controlled vehicle’s speed \( v \), the controlled vehicle’s acceleration \( \dot{v} \), the front vehicle’s speed \( \dot{v}_f \), the front vehicle’s acceleration \( \ddot{v}_f \), the distance \( d \) from the vehicle in front, and the expected speed \( v_{\text{des}} \) of the controlled vehicle as the current state, which is expressed as:

\[ s_{\text{acc}} = \langle v, \dot{v}, v_f, \dot{v}_f, d, v_{\text{des}} \rangle \]  
(15)

The action \( a_{\text{acc}} \in [-1,1] \) of the agent includes the throttle action \( a_{\text{throttle}} \) and the brake action \( a_{\text{brake}} \), which are respectively given by:

\[
\begin{align*}
    a_{\text{throttle}} &= a_{\text{acc}}, a_{\text{brake}} = 0 \quad \text{if} \quad a_{\text{acc}} \geq 0 \\
    a_{\text{throttle}} &= 0, a_{\text{brake}} = a_{\text{acc}} \quad \text{if} \quad a_{\text{acc}} < 0
\end{align*}
\]  
(16)

For the longitudinal control task, the reward function is designed as follows:

\[
    r_{\text{acc}} = \begin{cases} 
    -k_4 \text{abs}(v - v_{\text{des}}) - k_4 \text{abs}(d - d_{\text{safe}}) & \text{if} \quad d > d_{\text{safe}} \\
    -100 & \text{if} \quad d \leq d_{\text{safe}}
    \end{cases}
\]  
(17)

where \( d_{\text{des}} \) and \( d_{\text{safe}} \) denote the expected distance and safe distance from the vehicle in front respectively. When the distance between the controlled vehicle and the vehicle in front is less than the safe distance \( d_{\text{safe}} \), the reward is -100, and the current episode is ended. In the training of the longitudinal control task, the speed of the front vehicle \( v_f \) and the expected speed of the controlled vehicle \( v_{\text{des}} \) are set randomly in each episode so that the trained agent can be generalized to more conditions.

### C. DECISION-MAKING TASK UNDER COMPLEX TRAFFIC CONDITIONS

The decision-making task under complex traffic conditions is based on the fact that lateral and longitudinal control tasks can be well completed. The decision behaviors include adaptive cruise, left lane change, and right lane change. For the case of complex traffic conditions, the agent needs to focus on the location information of the surrounding vehicles, especially in the five-lane decision-making task. The state of the decision-making task is divided into two parts: one is the relative position information of surrounding vehicles, and another is the independent hot coding of the controlled vehicle on the relative lane. The illustration of the state of the decision-making task is shown in Figure 7:

\[ s_{\text{decision}} = \langle d_{f_1}, d_{f_2}, d_{f_3}, d_{f_4}, d_{f_5}, d_{d_1}, d_{d_2}, d_{d_3}, d_{d_4}, d_{d_5}, O \rangle \]  
(18)
In this work, only vehicles at distances of up to 200 m from the controlled vehicle are considered. In (18), \( d_f \) represents the distance between the nearest front vehicle and the controlled vehicle in lane \( i \), \( d_r \) represents the distance between the nearest rear vehicle and the controlled vehicle in lane \( i \); \( O \) is the one-hot encode of the lane where the controlled vehicle is located, which is a 1 \( \times \) 5 vector. The dimension code of the lane where the controlled vehicle is located is one, and the code of other lanes is zero. For instance, the one-hot encode of the controlled vehicle in the second lane is \( O = [0, 1, 0, 0, 0] \).

Compared with the lateral and longitudinal control tasks, the action space of the decision-making task is discrete, and it is divided into adaptive cruise task, left lane changing task, and right lane changing task as follows:

\[
d_{\text{decision}} = \begin{cases} 
0 & \text{if decision is lane keeping} \\
1 & \text{if decision is turning left} \\
2 & \text{if decision is turning right}
\end{cases} \tag{19}
\]

The design of the reward function for decision-making task is as follows:

\[
r_{\text{decision}} = \begin{cases} 
-1 & \text{if lane change} \\
-100 & \text{if collision} \\
f(d) & \text{else}
\end{cases} \tag{20}
\]

In the normal driving process, an intelligent vehicle avoids frequent lane changes. When the agent performs the lane changing task, the agent is given a slightly smaller punishment to promote the agent to prevent the agent from making frequent lane changes. To avoid sparse reward, the reward function of the decision-making task is shaped as a function of the distance from the front vehicle \( d \) [35]. The closer the controlled vehicle is to the front vehicle, leads to a smaller reward. The agent is motivated to execute lane changing task with close distance to the front vehicle. In term of collision influenced by wrong decision, a larger penalty is given.

### IV. SIMULATION RESULT

The proposed method was verified using the open-source autonomous driving simulation software CALRA [36-38]. A series of training simulations were conducted in the environment of TOWN04 and TOWN06 in CARLA 0.9.6. The Carla simulation scenario and the multi-lane simulation environment are presented in Figure 8.

![FIGURE 7. Illustration of the state of the decision-making task.](image_url)

![FIGURE 8. Carla simulation scenario.](image_url)

| Task          | Content          | Range     | Type      | State                  | Action        | Algo type |
|---------------|------------------|-----------|-----------|------------------------|---------------|-----------|
| Lane keep     | Steer angle      | [-1,1]    | Continuous| \(< x_o, x_y, y_1, y_2, \phi_1, v_o, v >\) | \(d_{\text{decision}}\) | AC        |
| Lane change   | Steer angle      | [-1,1]    | Continuous| \(< x_o, x_y, y_1, y_2, \phi_1, v_o, v, \dot{\phi} >\) | \(d_{\text{decision}}\) | AC        |
| Adaptive      | Brake and throttle| [0,1]    | Continuous| \(< v, v_1, \dot{v}, d, v_o >\) | \(d_{\text{decision}}\) | AC        |
| Decision-making | behavior       | [0,1,2]   | Discrete  | \(< d, \ldots, d_o, d_4, \ldots, d_n, \phi, \dot{\phi}, v >\) | \(d_{\text{decision}}\) | DQN       |

Before training, the network parameters were initialized randomly. The algorithm type, network structure, input state, and output action of task agents are shown in Table 1. Many experiments were conducted on all training tasks.

#### A. ANALYSIS OF TRAINING RESULTS OF LATERAL CONTROL TASK

Due to the randomness of the number of steps that an intelligent vehicle can take in each training episode, using the cumulative reward as an evaluation metric of the agent’s training effect in the current episode is not suitable. Using an average reward of each time step of the current episode as an evaluation metric of the agent’s training effect is reasonable. The change in the lane-keeping task training effect with the number of training episodes is presented in Figure 9, where the abscissa and the ordinate represent the number of training episodes, and the average reward obtained in an episode respectively.
The change in the average reward of the lane keeping task with the number of training episodes.

The change in the average reward of the lane changing task with the number of training episodes.

In Figure 9, the blue curve represents the reward change curve of agent with Gaussian noise, the red curve is the reward change curve of agent with the adaptive noise exploration designed by multi head actor networks, and the shadow part is the standard error of five experiments. As shown in Figure 9, the agent achieved good performance for each type of lateral control task. The average reward of the lane changing task fluctuated significantly due to the randomness of the vehicle speed in each episode. Speed randomness of each episode of the lane changing task was greater than that of the lane keeping task. In Figure 9(a), the obvious advantages of adaptive noise in lane keeping tasks can be seen from the perspective of network convergence and the final height of reward. This mainly depends on that the adaptive noise can adjust the action noise according to the strangeness of the environment. This advantage avoids the problem of insufficient exploration of the environment by ordinary attenuation noise.

The agent of the lateral control task and the PID feedback control method were compared. The controlled vehicle passed the U-shaped curve at a constant speed of 60 km/h. The shape of the U-shaped curve is shown in Figure 10 and the lateral errors of the two methods of the U-shaped curve are presented in Figure 11. In the early episodes, the proposed method is affected by the change of road curvature to a certain extent. But after 30 episodes, it is almost not affected by the change of road curvature, and the large lateral error is also very small. Compared with the PID feedback control based on the single point preview, the proposed method fully considered the road curvature and thus reduced the lateral error caused by the change in the road curvature [39].

The change in the average reward of the longitudinal control task with the number of training episodes is presented in Figure 12. The blue curve and the red curve illustrate the reward change curve for the adaptive noise and the Gaussian noise respectively. Obviously, the training effect of the adaptive noise exploration method was better than that of the attenuation noise.
C. ANALYSIS OF TRAINING RESULTS OF DECISION-MAKING TASK

In the training decision-making task, the position and speed of other obstacle vehicles were generated randomly in each training episode. When the controlled vehicle arrived at the destination or followed the vehicle at a lower speed for too long, the current episode ended, and the next episode started. The decision-making task was tested with two value-based reinforcement learning algorithms, and the results are shown in Figure 13. The red curve and the blue curve depict the training result of the DDQN [40], and the training result of the DQN. As presented in Figure 13, the DDQN was slightly superior over the DQN. Regardless of the used algorithm, the value-based reinforcement learning algorithm had good performance in the decision-making task.

For the multi-lane driving task, effective lane changing behavior was found to be a key component for improving driving efficiency. A certain number of vehicles was randomly generated on the map, and their behaviors were also set randomly. The controlled vehicle needed to reach the destination under a certain traffic density. The traffic efficiency comparison of the agent with the DDQN as a decision-making network and the agent without the decision-making behavior for the multi-lane driving task is presented in Figure 14. The light part in Figure 14 indicates high traffic efficiency, and as the color becomes deeper, the traffic efficiency of the controlled vehicle becomes lower. In the early stage of the decision-making task training, aimless decision-making behavior could reduce the traffic efficiency of the controlled vehicle. However, when the decision-making agent was trained for a certain number of episodes, the effectiveness of the controlled vehicle’s decision-making behavior and traffic efficiency of the greatly improved, especially in the case of a large traffic density.

V. CONCLUSION

This paper divides an autonomous driving task into lateral control, longitudinal control, and decision-making tasks. The Actor-Critic algorithm is used in lateral and longitudinal control tasks, and the DQN series algorithms are adopted in the decision-making control task. The reward designed separately for each subtask which realizes the accurate control and efficient decision-making. For specific subtask, the noise adaptive exploration method is proposed to reduce the invalid exploration and improve the adaptability of agent in strange environment. This exploration method improves the efficiency and robustness for the learning of the agent. We evaluated our methods on CARLA simulator with five-lane driving tasks scenario, the learned agent showed its superior performance under different density traffic. Future work will pay more attention to traffic rules and autonomous driving tasks in urban traffic.

REFERENCES

[1] C. Badue, R. Guidolini, R. Vivaquaa Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, et al., "Self-driving cars: A survey" in arXiv:1901.04407, 2019. [online] Available: http://arxiv.org/abs/1901.04407.
[2] A. Jafaripournimchahi, W. Hu, and L. Sun, "Nonlinear stability analysis for an anticipation-memory car following model in the era of autonomous and connected vehicles," in 2020 International Conference on Urban Engineering and Management Science (ICUEMS), 2020.

[3] L. Sun, A. Jafaripournimchahi, A. L. Kornhauser, and W. Hu, "A new higher-order viscous continuum traffic flow model considering driver memory in the era of autonomous and connected vehicles," Physica A-statistical Mechanics and Its Applications, vol. 547, p. 123829, 2020.

[4] M. Montemero et al., "Junior: The Stanford entry in the urban challenge," J. Field Robot., vol. 25, no. 9, pp. 569-597, Sep. 2008.

[5] J. Ziegler et al., "Making Bertha Drive—An Autonomous Journey on a Historic Route," IEEE Intelligent Transportation Systems Magazine, vol. 6, no. 2, pp. 8-20, 2014.

[6] H. Fan et al., "Baidu Apollo EM Motion Planner," arXiv preprint arXiv:1807.08048, 2018.

[7] P. Wolf, C. Hubschneider, M. Weber, A. Bauer, and J. M. Zilner, "Learning How to Drive in a Real World Simulation with Deep Q-Nets," in 2017 IEEE Intelligent Vehicles Symposium, pp. 244-249, 2017.

[8] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in Conference on Robot Learning. PMLR, 2020, pp. 66-75.

[9] A. Jafaripournimchahi, W. Hu, and L. Sun, "An Asymmetric-Anticipation Car-following Model in the Era of Autonomous-Connected and Human-Driving Vehicles," Journal of Advanced Transportation, vol. 2020, 2020.

[10] M. Bojarski et al., End to end learning for self-driving cars, 2016, [online] Available: http://arxiv.org/abs/1604.07316.

[11] A. Kendall et al., "Learning to drive in a day," in 2019 International Conference on Robotics and Automation (ICRA), pp. 8248-8254, 2019.

[12] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in International conference on machine learning, pp. 387-395, 2014.

[13] T. P. Lillicrap et al., Continuous control with deep reinforcement learning, 2015, [online] Available: https://arxiv.org/abs/1509.02971.

[14] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in International conference on machine learning, pp. 1889-1897, 2015.

[15] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," Transportation Research Part C: Emerging Technologies, vol. 117, pp. 102662, 2020.

[16] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," Transportation research part C: emerging technologies, vol. 97, pp. 348-368, 2018.

[17] X. Ma, K. Driggs-Campbell, and M. J. Kochenderfer, "Improved robustness and safety for autonomous vehicle control with adversarial reinforcement learning," in 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1665-1671, 2018.

[18] C. J. H. Watkins, "Learning from delayed rewards," 1989.

[19] V. Mnih et al., Playing Atari with deep reinforcement learning, 2013, [online] Available: https://arxiv.org/abs/1312.5602.

[20] K. Min, H. Kim and K. Huh, "Deep q learning based high level driving policy determination," Proc. IEEE Intell. Vehicles Symp. (IV), pp. 226-231, Jun. 2018.

[21] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9329-9338.

[22] F. Codevilla, M. Müller, A. Dosovitskiy, A. M. López, and V. Koltun, "End-to-End Driving Via Conditional Imitation Learning," 2019.

[23] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," arXiv:2001.11231, 2020, [online] Available: http://arxiv.org/abs/2001.11231.