Tactical Decision-making for Autonomous Driving using Dueling Double Deep Q Network with Double Attention

SHUWEI ZHANG\(^1\), (Student Member, IEEE), YUTIAN WU\(^1\), (Student Member, IEEE), HARUTOSHI OGAI\(^1\), (Member, IEEE), HIROSHI INUJIMA\(^1\), and SHIGEYUKI TATENO\(^1\)

\(^1\)Graduate School of Information, Production and Systems, Waseda University, Kitakyushu 8080135, Japan

Corresponding author: Shuwei Zhang (ZHANGSHUWEI@toki.waseda.jp) and Yutian Wu (wuyutian@fuji.waseda.jp)

ABSTRACT

Decision-making is still a significant challenge to realize fully autonomous driving. Using deep reinforcement learning (DRL) to solve autonomous driving decision-making problems is a recent trend. A common method is to encode surrounding vehicles in a grid to describe the state space to help DRL network extract the scene features. However, in the process of human driving, surrounding vehicles at different positions have different contributions to decision-making. Meanwhile, the network is difficult to fully extract useful features in a sparse state, which can also lead to catastrophic actions. Therefore, this work proposes a spatial attention module to calculate different weights to represent different contributions to decision-making results. And a channel attention module is proposed to fully extract useful features in sparse state features. These two attention modules are integrated into dueling double deep Q network, named D3QN-DA, as a high-level decision-maker of a hierarchical hierarchical control structure based decision-making system. To improve agent performance, an emergency safe checker is introduced in this system. And the agent is trained and test with a designed reward function from safety and efficiency in simulation. The experimental results show that the proposed method increases the safety rate by 54\%, and the average explore distance by 30\%, which is safer and more intelligent for the decision-making process of automatic driving.

INDEX TERMS

autonomous driving, decision-making, dueling double deep Q network, spatial attention, channel attention

I. INTRODUCTION

In the last decades, autonomous driving has achieved a great development as the increasing of computing power and algorithmic innovation. It is expected to take many benefits, such as a reduction of accidents and better energy efficiency [1], [2]. But it is still a challenge to achieve fully autonomous driving because of the strict request of efficient and safe decision-making in the complex and uncertain environments. The goal of decision-making system in autonomous vehicle is to give safe and appropriate action instructions according to surrounding environments information [3]. The decision-making task is commonly divided into three types: strategic, tactical, and operational which are also named navigation, guidance and stabilization [1], [4], [5]. In this paper, we focus on high level tactical decisions-making system, which inputs high-dimensional state and outputs discrete actions, such as lane change and speed up during driving. The inputs of this system are the state of ego vehicle and surrounding vehicles, High Definition (HD) map, and target reference path and the outputs are the actions based on the current moment in order to plan a trajectory [6].

Traditionally, rule-based decision-making algorithms are common solutions for autonomous driving [7], [8], [4], [9], [10] have applied rule-based algorithms in DARPA Urban Challenge successfully. However, the drawback of rule-based decision-making is not flexible enough to deal with complex situations. When some new rules are added, the old rules will be influenced, such as undone or rewritten, which makes this system unrobust. Moreover, when different driving tasks are combined, rules have different combinations methods.
which makes the coupling between different rules and even contradictions between each other. These problems limit the generalization of the algorithm and makes it difficult to scale the algorithms to complex and uncertain environments. To solve these problem, machine learning based methods are developed for complex environment decision-making. Since the decision-making process does not have a certain standard, ground truth, to evaluate the performance of the current action, it can not adopt supervised machine learning which is usually used in environment perception. The problem which Deep Reinforcement Learning (DRL) solved is how the agent adopts actions in the environment to obtain the maximum reward, which coincides with the problem of autonomous driving decision-making [11]. Many studies have applied DRL in autonomous driving. [12] summarizes a DRL framework for autonomous driving which proves the feasibility of DRL in decision-making.

DRL has two main approaches to determine the optimal actions: value-based and policy-based algorithms [13]. The optimization goal of value-based DRL is to find an action with the maximum action value and usually outputs discrete action. While the target of policy-based methods is to find a policy which can achieve maximum reward and usually outputs probability distributions of actions to deal with the problem of continuous action space [14]. Value-based DRL happens be used to solve the autonomous driving tactical decision-making problems. Deep Q Network (DQN) is a widely used value-based DRL algorithm, where Q is action value [15], [16] is a pioneer to propose this original concept and the authors improve DQN powerful performance in games to control the agents playing as a human in [17]. But since DQN selects the maximum action value as the target Q value and employs the dynamic optimization target, over estimation and difficulty in convergence become the pivotal problems of DQN. [18] propose a Double DQN (DDQN) to solve this overestimation by decoupling selection of action and calculation of action value. And [19] uses a dueling network, named Dueling DQN, with action advantages function to reduce fluctuations in action value to improve the stability of DQN. [20] integrates DDQN and Dueling DQN as Dueling Double DQN (D3QN). In this paper, we adopt D3QN as the framework on tactical decision-making for autonomous driving.

A valid method of encoding state information and extract feature are especially important to help DRL-based network learn effective features. [21] uses a vector based on a fixed number of vehicles and characteristics to construct the state space and trains agents with double action spaces by Fully Connected Neural Network (FCNN). But this method did not consider the spatial position relationship between ego vehicle and other vehicles. [22] uses an 2-dimensional (2D) occupancy grid map to represent the environment state as observation space and trains agent to select the lane changing actions in highway scenarios. When the networks extract features from an occupancy grid, each vehicle is treated as the same importance to generate current decisions. But commonly in human driving, different surrounding vehicles contribute differently to make decision and we pay different attention to different vehicles. Attention mechanism that assigns different attention according to different attributes, has been used in the field of object detection [23], [24], object segmentation [25] and natural language processing for machine translation [26]. In object detection task, every feature point has different contributions to the final object detection result. Inspired by these algorithms, we present a spatial attention-based module to model this mechanism to imitate human driving process.

Besides, surrounding vehicles information, such as velocity, heading, positions, are encoded as the channel feature of occupancy grid map. [27] uses the position, speed and heading of a leading vehicle, the ego vehicle and a following vehicle in this lane as channel features to select the decision actions. [28] encodes the position and speed of all the surrounding vehicles as the input of network. However, different features also contribute differently for decision-making according to surrounding vehicles. If the network can not abundantly and effectively extract features in sparse state space, it will cause the inaccurate action prediction to produce catastrophic exploratory actions. Thence we present a channel attention module to extract the features effectively to make the output actions safer and more accurate by reweighting of objects features to the feature map. Finally, we integrate spatial attention module and channel attention module in D3QN, named D3QN with double attention (D3QN-DA), as a high-level decision maker in this research.

In this paper, we present a hierarchical control structure system to control the ego vehicle interacting in simulation environment. The system contains a high-level decision maker, a middle emergency safe checker and a low-level executive controller. To establish a strong baseline to better validate the modules we propose, a well-designed reward function according to safety and efficiency is introduced to guide networks learning.

The contributions of this study are as following:

1) We design a hierarchical control structure system to control the ego vehicle interacting during the simulation environment. The system contains a high-level decision maker, a middle emergency safe checker and a low-level executive controller.

2) An adaptive state encoder in hierarchical control structure system is used to effectively encode surrounding vehicles.

3) We model the human driving attention process, include a spatial attention models the spatial relationship among vehicles, and a channel attention to calculate channels weights to extract features abundantly and effectively. We integrate these two attention modules into a D3QN to make safer and more efficient decisions for autonomous driving.

4) We train and test the model in a simulation environment with a designed reward function according to safety and efficiency. Then we qualitatively and quanti-
titatively compare the results with the DRL methods using two kinds of metrics: reward metric and capability metric. The results prove the proposed method is more secure and smarter than baseline models.

The rest structure of this paper is as followed: The development and optimization goals of D3QN are introduced in Section II. In Section III, we introduce the hierarchical control structure system with a D3QN-DA based highway-level decision-maker and a emergency safe checker. In Section IV, we demonstrate the agent training environment, conduct a comparative experiment and evaluate the effectiveness of proposed method. Finally, the conclusions are arranged in Section V.

II. BACKGROUND

Reinforcement Learning is that an agent learns which action in a specific situation can get the most rewards during the trial process. The agent interacts with an environment and tries to learn a policy \( \pi \) which can get max potential future rewards [29]. The policy is a process that the agent chooses which action \( a \) in state \( s \). The environment will give next time state \( s' \) and return a reward \( r \) after executing the action \( a \). Almost all reinforcement learning problems can be modeled as Markov Decision Process (MDP) [30]. That is, a certain state information contains all relevant history. The state \( s \) at time \( t+1 \) is determined only by the state \( s \) and action \( a \) at time \( t \). MDP is defined as a 5-tuple \((S, A, P, R, \gamma)\), where \( S \) is a set of states, \( A \) is a set of actions, \( P \) is state transition probability function from \( s \) to \( s' \), \( R \) is reward function, and \( \gamma \in [0, 1] \) is discount factor. At every time step \( t \), the goal of the agent is to obtain the max return of future discount as Eq.(1).

\[
R_t = \sum_{i=0}^{n} \gamma^i r_{t+i}
\]  

where \( r_{t+i} \) is the reward at time step \( t+i \).

**Q Learning** is one of the classic reinforcement learning algorithms [15] that the agent is trained to find an action with the optimal action value function \( Q^*(s, a) \) defined in Eq.(2).

\[
Q^*(s, a) = \max_{\pi} E[R_t|s_t = s, a_t = a]
\]  

We use the Bellman optimality equation [15], [31] to express the optimal action value function, see Eq.(3).

\[
Q^*(s, a) = E[r + \gamma \max_{a'} Q^*(s', a')|s_t = s, a_t = a]
\]  

where \( a' \) is the next time step action, and its optimal action value is \( Q^*(s', a') \). When the state space is discrete and finite, we can easily formulate Q function. But the state space of most problems are infinite and continuous. DQN is proposed to solve this situation.

**DQN** is produced to handle the high dimension catastrophe by combining reinforcement learning and deep learning. In every time step \( t \), the interactive data with environment as a 4-tuple \( e_t = (s_t, a_t, r_t, s_{t+1}) \) of agent are collected in an experience replay dataset \( D(e_1, e_2, ..., e_n) \). DQN randomly samples a batch size \( B \) tuples from replay dataset \( D \) to guarantee the independence of training data. The batch tuples are input the network as the training data to train the agent to approximate the action value function [17]. The action value function is defined by \( Q(s, a; \theta_i) \) to approximate \( Q^*(s, a) \), where \( \theta_i \) are the training weights of the Q network in every iteration \( i \). DQN has two networks with the same structure, a target Q net \( F_i \) to calculate target Q value \( Q_t \) and an estimated Q net \( F_e \) to estimate Q value \( Q_e \) in current state. The goal of DQN is to train suitable weights to minimize the loss \( L_i(\theta_i) \) of the \( Q_t \) and \( Q_e \) as shown in Eq.(4):

\[
L_i(\theta_i) = E_N[(r + \gamma \max_{a'} Q^*(s', a'; \theta^-) - Q(s, a; \theta_i))^2]
\]  

The training parameters of \( F_t \) are updated by the parameters of an \( F_i \) every \( C \) episodes. Then DQN uses the stochastic gradient descent method to optimize the loss function. And the advantage of the double network structure is to give a fixed target to avoid the non-stationary target causing dimensional calculation explosion. DQN uses a \( \varepsilon \)-greedy policy to deal with the trade-off between exploration and exploitation. The agent selects a random action with probability \( \varepsilon \), otherwise chooses the action which has the highest action value.

**DDQN** is proposed to address the overestimation of action values in DQN algorithm due to the \( F_i \) selects Q value expectation after taking the maximum action value as the \( Q_t \). DDQN solves this overestimation by decoupling selection of the \( F_i \) action and calculation of the \( Q_t \). Instead of finding the maximum Q value directly in the \( F_i \), DDQN selects action with the maximum Q value from \( F_e \) and then calculates the \( Q_t \) in the \( F_i \), shown as Eq.(5)

\[
L_i(\theta_i) = E_N[(\gamma \max_{a} Q^*(s', a'|s; \theta) - a \max_{a'} Q(s', a'; \theta^-) + r - Q(s, a; \theta_i))^2]
\]  

**Dueling DQN** [19] is proposed to improve the DQN stability and performance from an alternative but complementary approach by reconstructing the network. Dueling DQN separates the estimation of state values and state-dependent action advantages. [32] puts forward to the advantage function which expresses the advantage of a certain action \( a \) relative to the average in state \( s \) to normalize Q value, shown as Eq.(6).

\[
Q^*(s, a) = V^*(s) + A^*(s, a)
\]  

[19] uses a Convolutional Neural Network (CNN) layer to extract the features and then uses two streams of FCNN to output the value function \( V(s; \theta, \beta) \) and advantage function, an \( |A| \)-dimensional vector \( A(s, a; \theta, \alpha) \). \( \theta \) denotes the parameters of the CNN layers, and \( \alpha \) and \( \beta \) are the parameters of the two streams of FCNN layers. Dueling DQN uses Eq.(7) to calculate the Q-value and shares the same input-output interface with standard DQN.

\[
Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s, a; \theta, \alpha) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta, \alpha))
\]
D3QN [33] is integrated DDQN and Dueling DQN to solve over estimation, unstable and the difficulty in convergence problems of DQN. In this work, we adopt D3QN as a strong baseline framework to help us verify the proposed new model.

III. METHODOLOGY

In this section, we will introduce the proposed hierarchical control structure based decision-making system as shown Fig.1. The system contains a high-level decision maker, a middle emergency safe checker and a low-level executive controller. The high-level decision-maker uses our proposed D3QN-DA to generate an action according to surrounding vehicles’ intentions and behaviours. Section III-A1 introduces the state space with an adaptive state encoder, action space and a designed reward function of D3QN-DA. Section III-A2 illustrates our proposed spatial and channel attention modules which integrated in D3QN to improve the performance of the agent. In Section III-B, a middle emergency safe checker is introduced to avoid some catastrophic exploratory actions. This module limits some endangered actions by the speed limit. Finally, in Section III-C, we introduce the low-level controller, which used to implement the actions to control the ego vehicle interacting in environment.

A. HIGH-LEVEL DECISION MAKER

The ego vehicle interacts with the environment to get observation information and each action reward to make optimal rational decision. We treat this decision-making process in autonomous driving as a MDP. And we use D3QN-DA algorithm to solve this MDP problem.

1) MDP

Since the state transition probability function $P$ is expressed by a neural network, MDP is defined with a $4-tuple$ $(S, A, R, \gamma)$, where $\gamma$ is a hyperparameter, $S$ is state representations, $A$ is action space and $R$ is reward function. This tuple contains the input and output of the network. And $S, A, R$ are demonstrated as following:

State Representation. In order to build the spatial relationship of the surrounding environment, a common method of state representation is encoding surrounding objects in a grid map according to the each object centroid [34]. But these methods did not consider the relationship between vehicle size and grid size, the lacking of the shape characteristics of the vehicles makes the network may not fully extract real features, resulting in producing wrong actions. In this work, an adaptive state encoder is proposed to express the shape characteristics of the vehicles. Fig.2(a) shows vehicles position, where $(x_0, y_0)$ is the ego vehicle global coordinates and $(x_1, y_1), (x_2, y_2)$ are surrounding vehicles global coordinates. Each vehicles size is $5\text{meters} \times 2\text{meters}$. And the grids size is $2\text{meters} \times 2\text{meters}$. Fig.2(b) shows that vehicles are adaptive encoded in three or four grid cells according to their size and centroid position. The spatial feature, $p$, whether exists vehicle or not, in a cell has been constructed. Since far away vehicles are not necessary for current time step decision, we only select vehicles within a Region Of Interest (ROI) area around the ego vehicle. In this simulation, we set the region is $128\text{meters} \times 128\text{meters}$ while all vehicles velocity are limited in $[V_l, V_h]$, set by $[30, 40]m/s$. In this work, we define the states of all vehicles are in Eq.(8):

$$s = (p, x_i, y_i, v_i^x, v_i^y, \cos \varphi_i, \sin \varphi_i)_{i \in [0, N]}$$

where $s_0$ is the ego vehicle status, $s_k$ are other vehicles status, $x, y$ are the relative position to the ego vehicle, $v_i^x, v_i^y$ are the vehicles speed, $\cos \varphi_i, \sin \varphi_i$ are the vehicles heading. All features are normalized values in $[0, 1]$.

![FIGURE 2. Original and grid encoder map vehicles position.](image)

### TABLE 1. Action space.

| action | content          |
|--------|------------------|
| $a1$   | lane change to the left |
| $a2$   | lane keep         |
| $a3$   | lane change to the right |
| $a4$   | faster            |
| $a5$   | slower            |

Reward Function. Reward function is significant to guide DRL learning. Different with the supervised learning, DRL
only has an optimization goal defined by reward and expectation of future discounted rewards without an exact ground truth. A good reward function can guide the agent to produce correct actions and avoid catastrophic actions. In this research, our reward function is designed to achieve higher efficiency while maintain safety. In terms of safety, the ego vehicle is capable of making appropriate choice to avoid the collisions. When the collisions occur, the agent gets collision punishment \( R_c = -20 \). Discrete sparse rewards are not enough for agent which may slow learning process or lead to ineffective learning. We design reward to solve the sparse reward problem by adding a safety guidelines in this safety guidelines, a Dangerous Approach Punishment (DAP) \( R_a = -20 \) is brought in evaluation mechanism, decided by time to collision \( T_{ttc} \), which is calculated by the distance between the ego and a leading front vehicle \( d_f \) on this lane and the relative velocity of front vehicle \( v_f \) and ego vehicle \( v \), shown as Eq.9.

\[
T_{ttc} = \frac{d_f}{v - v_f} \quad if \quad v \geq v_f \tag{9}
\]

The time to collision is limited within \([T_{max}, T_{min}]\), we set [3, 1] in this research.

The next safety reward index is Unsafe Lane Change Punishment (ULCP) \( R_u = -10 \). \( R_u \) is obtained when the distance between the ego vehicle and a front vehicle \( d_{lcf} \) in the target lane is less than a preset lane change front safe distance \( D_{lcsf} \), or when the distance between the ego vehicle and a rear vehicle \( d_{ler} \) in the target lane is less than a preset lane change safe rear distance \( D_{lcsr} \). To avoid random lane change action generation, we set a target lane \( l_t \) (the most right lane is the target lane in this work) and a cost of lane change actions \( R_l = -0.4 \). When the ego vehicle travels in the most right lane, it gets the reward \( R_r = 1.2 \) in total 4 lanes \( L_{max} \), linearly mapped to zero for left lane according the lane id \( l_{id} \). Besides, to avoid invalid lane change action, e.g. when the ego vehicle is on the side lane, such as \( l_{id} = 0 \), and the action is lane change left, we set an additional Illegal Lane Change Punishment (ILCP) \( R_i = -3 \). In terms of efficiency, when the agent drives at full speed, it will get the velocity reward \( R_v = 3.0 \) linearly mapped to zero and normalized in range \([V_l, V_h]\). The ego vehicle speed is 35\( m/s \) for initialization, and the maximum target speed is limit in 40\( m/s \) while other vehicles speed are limited in 40\( m/s \).

Overall, the reward function \( R \) can be expressed in Eq.(10)

\[
R = \begin{cases}
R_c & if \ collision \ occurs \\
\frac{T_{max} - T_{ttc}}{T_{max} - T_{min}} R_a & if \ T_{ttc} \leq T_{max} \\
R_u & if \ d_{lcf} \leq D_{lcsf} \ or \ d_{ler} \leq D_{lcsr} \\
\frac{l_{id}}{T_{max}} R_r & if \ l_{id} < target \ l_{id} \\
R_l & if \ a \ is \ lane \ change \\
R_i & if \ a \ is \ illegal \ lane \ change \\
\frac{v - V_i}{V_h - V_i} R_v & \end{cases}
\tag{10}
\]

2) D3QN-DA

In this study, we use D3QN-DA shown in Fig.3 to estimate the actions according to current status in simulation environment. The structure of D3QN-DA contains a feature extraction subnet and an action prediction subnet. The input of D3QN-DA network is the adaptive encoded grid map \( M \in \mathbb{R}^{H \times W \times C} \). The outputs are the Q of five actions and the action \( a \) which has maximum Q will be chosen by the agent. CNN [35] is widely used to extract features while keeping position invariance of inputs. We use three convolutional layers \( F_1, F_2, F_3 \) with channels \( C_1, C_2, C_3 \), which we set to 16, 32, 64 in this study, to extract features from \( M \). All layers have the same kernel size, 2 \( \times \) 2, stride 2 and a rectified linear unit (ReLU) activation function. Then, we integrate double attention (DA) module in \( M \) and the this subnet outputs a weighted feature map \( M_d \). The action prediction subnet is composed of two sequential FCNN layers \( F_c \) and a dueling network. The first FCNN inputs the flatten \( M_d \) and outputs 256 neurons. The second FCNN layer inputs the 256 neurons and output 128 neurons. Each layer has a ReLU activation function. Finally, via the dueling network to calculate advantage function and value function in Eq.(7), the network will output action state value \( Q(s, a) \).

Spatial Attention Module. For each surrounding vehicle, the ego vehicle should give different attentions like a human driver driving. We add a Spatial Attention (SA) module to model this attention as ‘weight’ to make decision actions. In SA module, we reweight the feature map \( M_3 \in \mathbb{R}^{H_3 \times W_3 \times C_3} \) obtained via three convolutional layers from input grid map \( M \). We use a slicing of tensor \( M_3 \) to express \( M_3 \) as shown in Eq.(11).

\[
M_3^i = [m_{1,1}^{i,j}, m_{1,2}^{i,j}, ..., m_{i,j}^{i,j}, ..., m_{H_3,W_3}^{i,j}] \tag{11}
\]

where \( m_{i,j}^{i,j} \in \mathbb{R}^{1 \times 1 \times C_3} \), is the element in spatial location \((i, j)\), \( i \in [1, H_3], j \in [1, W_3] \).

The SA module is calculated by a channel squeeze function \( F_{cs}() \) and a Sigmoid activation functions. \( F_{cs}() \) is achieved by CNN which has a convolutional layer with kernel size \( 1 \times 1 \), stride 1, and a \( C_3 \) channel. After these operations, we get a projection tensor \( P \in \mathbb{R}^{H_3 \times W_3 \times 1} \), that is the weight for each spatial element. Then, we use \( P \) to reweight each
spatial location. The spatial attention based feature map \( M'_s \) is shown in Eq.(12).

\[
M'_s = P \times M'_3
= [p^{1,1}, ..., p^{i,j}, ..., p^{H_3 \times W_3}] \tag{12}
\]

where \( p^{i,j} \in P \) represent each weight for spatial element calculated from spatial attention module.

**Channel Attention Module.** The surrounding vehicles have lots of information characteristics, such as velocity, position, orientation, lane, etc. When the networks predict the actions, different features contribute differently for decision-making. For example, a closer low-speed vehicle before the ego vehicle is more 'important' than a far away high-speed vehicle in this lane. If the ego vehicle treats all the features as the same importance, the features which really have an impact for action prediction can not be fully utilized. The network may not estimate a precise action which may cause catastrophic collisions. Same with the squeeze and excitation principle [36] which is used in object detection task, we introduce a Channel Attention (CA) module to recalibrate features by using global information \( Z \in \mathbb{R}^{1 \times 1 \times C_3} \) from to pick significant features and suppress less useful ones. We use a channel combination tensor \( M'_3 \) to express \( M_3 \), shown as Eq.(13).

\[
M'_3 = [m'_1, m'_2, ..., m'_i, ..., m'_C_3] \tag{13}
\]

where \( m'_i \in \mathbb{R}^{H_3 \times W_3} \) is the \( i \)th channel slicing elements.

We extract the global information through a spatial squeezing function \( F_{ss}() \), that is a global average pooling, shown as Eq.(14).

\[
Z = F_{ss}(M'_3) = \frac{1}{H_3 \times W_3} \sum_{i=0}^{H_3} \sum_{j=0}^{W_3} m'_i(i,j) \tag{14}
\]

Then, \( Z \) is transformed to a channel excitation function \( F_{ce}() \) to calculate channel weight \( U \in \mathbb{R}^{1 \times 1 \times C_3} \). \( F_{ce}() \) contains two FCNN layers with the parameters \( \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_c \), a ReLU layer \( \delta \) to increase the non-linear interaction and a sigmoid function \( \sigma \) shown as Eq.(15).

\[
U = F_{ce}(Z, \mathbf{W}) = \sigma(\mathbf{W}\delta(Z)) = [u_1, ..., u_i, ..., u_{C_3}] \tag{15}
\]

Finally, \( U \) is used to reweight all each channel of \( M'_3 \). The channel attention based feature map \( M''_3 \) is shown in Eq.(16).

\[
M''_3 = U \times M'_3 = [u_1m'_1, ..., u_i m'_i, ..., u_{C_3}m'_{C_3}] \tag{16}
\]

**Double Attention Module.** We integrate the above two modules, SA module and CA module, in the D3QN algorithm through the element-wise addition as a double attention (DA) module, shown as Eq.(17).

\[
M''_a = M''_s + M''_c = [m''_1^{1,1} + m''_c^{1,1}, ..., m''_i^{i,j,k} + m''_c^{i,j,k}, ...] \tag{17}
\]

where \( M''_c^{i,j,k} \) is the \((i,j,k)\)th element is \( M''_a, i \in [1, H_3], j \in [1, W_3], k \in [1, C_3] \).

In this work, we set DA module after the third convolutional layers. We also discuss the setting of DA module in Section IV-B.

**B. MIDDLE EMERGENCY SAFE CHECKER**

To improve the safety of the ego vehicle, we add a middle emergency safe checker in lateral and longitudinal. According to limit velocity \([V_0, V_l]\) and D3QN-DA decision time, we set a dangerous distance \( D_d = 10.0 \) meters shown in Fig.1, the red circle.

In longitudinal safe checker, when the distance between the closest front vehicle on this lane and the ego vehicle is less than \( D_d \) and the velocity of ego vehicle \( v \) is higher than the front vehicle velocity \( v_f \), the target speed will be reduced linearly according to the relative velocity.

In the meanwhile, when the action produced in high-level decision maker is a lane change action and there is a vehicle in dangerous region in target lane, the lane change action will be cancelled, and keep in current lane until the next decision action generated.
C. LOW-LEVEL EXECUTIVE CONTROLLER
In the low-level controller, the ego vehicle is controlled by a lateral steering angle controller and a longitudinal acceleration controller by a Proportional Integral Derivative (PID) controller. When the ego vehicle gets a lane change command, it will be controlled according to lateral position error and the ego vehicle orientation error. Lateral position error $\Delta l$ is converted to a heading reference $\psi_{ref}$ by a proportional controller, shown as Eq.(18).

$$
\psi_{ref} = \arcsin\left(\frac{v}{v_{tar}}\right) = \arcsin\left(\frac{K_{p,l} \Delta l}{v}\right) \quad (18)
$$

where $K_{p,l}$ is the lateral position proportional gain.

Then, a steering angle command $\theta$ is generated by the ego vehicle heading $\psi$, shown as Eq.(19)

$$
\theta = K_{p,h}(\psi_{ref} - \psi) \quad (19)
$$

where $K_{p,h}$ is the lateral heading proportional gain.

And the motions of the vehicles in the longitudinal is also controlled by a proportional controller with target velocity $v_{tar}$ and vehicle speed $v$, shown as Eq.(20)

$$
\theta = K_{p,v}(v_{tar} - v) \quad (20)
$$

where $K_{p,v}$ is the longitudinal speed proportional gain.

IV. EXPERIMENT AND RESULT
We verify our algorithm in a challenging environment, a devised dynamic dense highway traffic environment based on [37]. And training and testing details are introduced in Section IV-A. Finally, we evaluate the performance of D3QN-DA and compare the results with the relative works in Section IV-B.

A. SIMULATION ENVIRONMENT
Since free exploration can increase the diversity of DRL training data, we construct lots of other vehicles (maximum 25 vehicles) to build an complex and dynamic environment based on highway scene [37] with 4 lanes. In order to better simulate the real world, we model the acceleration of the other vehicles by Intelligent Driver Model (IDM) [38] and model the discrete lane change decisions by the MOBIL model [39]. The environment vehicles are also controlled by PID controller same as the ego vehicle.

We use D3QN-DA algorithm to train the ego vehicle in 2500 episodes to explore in the environment. For each episode, when the ego vehicle gets a collision or when its travelling time is more than 40 seconds, the episode will be terminated. After training, we evaluate the final model in 100 testing episodes.

**Training Parameters** We train the agent using D3QN algorithm as the framework described in Section II. To appropriately collect training samples, the agent should get balance between exploration and exploitation. We adopt epsilon greedy behaviour to handle the trade-off between exploration and exploitation. The epsilon greedy $\epsilon$ decreases from initial exploration constant $\epsilon_i$ to final exploration constant $\epsilon_f$ within the interval of final exploration iteration $I_e$. During testing episodes, all the exploration rate $\epsilon$ is set to 1, that means all the actions are from DRL algorithms without random generation. Discount factor $\gamma$ is used to discount future potential value to calculate the $Q_t$ in network. In every time step $t$, the agent interactive data with environment are collected in an experience replay dataset $D$ with capacity $D_{max}$ as mentioned in Section II. The network samples batch size $B$ training data in $D$. The goal of D3QN is to train suitable weights to minimize the error of $Q_t$ and $Q_e$. We make a summary of the training hyperparameters as shown in Table.2.

| Hyperparameters | Symbol | Value |
|-----------------|--------|-------|
| Discount factor  | $\gamma$ | 0.8   |
| Memory capacity  | $D_{max}$ | 15000 |
| Batch size       | $B$    | 64    |
| Learning rate    | $\eta$ | 0.001 |
| Target network update frequency | $f_t$ | 50 |
| Initial exploration constant | $\epsilon_i$ | 1 |
| Final exploration constant | $\epsilon_f$ | 0.05 |
| Final exploration iteration | $I_e$ | 6000 |

B. RESULTS AND EVALUATION
In our experiment, six models are built to make comparative experiments to demonstrate the effectiveness of hierarchical control structure system. Since our improvements are based on D3QN and other modules are not changed, we use basic D3QN as baseline. We add an adaptive grid encoder as Adaptive D3QN (A-D3QN) model. We add an emergency safe checker in A-D3QN as Emergency D3QN (AE-D3QN) model. We separately add SA module and CA module in AE-D3QN as Spatial D3QN (AE-D3QN-SA), Channel D3QN (AE-D3QN-CA) and integrate SA and CA in AE-D3QN as Double Attention D3QN (AE-D3QN-DA). After experiment, we use two kinds of metrics to qualitatively and quantitatively evaluate the property, reward metric and capability metric. Finally, an ablation study of different integrate methods of spatial attention and channel attention with D3QN are discussed.

**Reward Metric.** Since DRL is different from supervised learning without ground truth, reward is applied to guide the training process. Hence, the reward is a natural indicators to evaluate the performance of the agent. The higher the reward, the better the performance. Fig.4(a) shows the total rewards during 2500 training episodes with smoothing. And Fig.4(b) shows the average rewards during 100 testing episodes. We can see that A-D3QN improves 4.78% rewards comparing with D3QN, and it proves the necessity of adaptive encoder which encodes surrounding environment effectively and completely for network. AE-D3QN can also improve 22.97% average reward. We can also see that attention based D3QN algorithm, AE-D3QN-SA, AE-D3QN-CA, AE-D3QN-DA,
can further obtain more rewards during training process. In the test set, they improve 24.12\%, 24.41\%, 28.39\% rewards, respectively.

**Capability Metric.** The second metric is the capability metric to analyze the performance of models quantitatively in test set. This metric contains 4 indicators, success rate, average speed, average exploration time and average exploration distance. Success rate is proportion of the episodes without crash and the total episodes, shown as Fig.5(a). Average speed is shown as Fig.5(b). Average exploration time is the travelling time in each episode, shown as Fig.5(c), note that maximum travelling time is limited within 40 seconds. Average exploration distance is the ego vehicle travelling distance until this episode ended, shown as Fig.5(d). All the indicators are averaged by the number of testing episodes. The results of A-D3QN and AE-D3QN indicate an adaptive grid encoding module could accelerate to earn short-term rewards as shown by high speed. But it crashes often without safe limit, as shown that success rate only increases 10\% while other models improve more than 42\%. We will obtain a riskier driving policy if the network do not have a safe checker. AE-D3QN could drastically improved success rate, exploration time and exploration distance with 42\%, 27.5\%, 23.72\%, but it still have some corner case that it can not solve. The results shows that the agent fails to utilize the information of other vehicles, leading to short exploration.
To further evaluate attention modules, Table 3 counts the proportion of punishment actions, Safe Check (SC) ratio, Unsafe Lane Change Punishment (ULCP) ratio, Dangerous Approach Punishment (DAP) ratio and Illegal Lane Change Punishment (ILCP) ratio, in reward function during 4000 steps in 100 episodes test set. For each ratio in table, the ratio value lower, the performance better. AE-D3QN-SA improves success rate 49%, average exploration time 30.57%, average exploration distance 26.1% while reduce emergency safe check times 4.51%, unsafe lane change times 0.99%, closest approach to the front vehicle times 1.25%. AE-D3QN-CA could also improve success rate 50% while reduce unsafe actions. AE-D3QN-DA could improve success rate 54%, average explore time 35.71%, average explore distance 30.21% while maintain the same speed with AE-D3QN. And it also reduce 7.22% safe check ratio, 0.8% unsafe lane change actions, 3.27% close approach actions and 1.22% illegal lane change actions. Combining these two kinds of attention modules, we can obtain better results. The results spatial attention based module considers the spatial position relative relationship among ego vehicle and other vehicles and give different attentions for different vehicles. And for different features, the ego vehicle also makes full use of information with different characteristics. Through the different attentions for each surrounding vehicle, a much safer action is chosen to travel in the road. Owing to a significant performance improvement, the overall resulting manifest that it makes ego vehicle actions more intelligent.

Ablation Study. To further explore the influence of integration ways of SA module and CA module to D3QN algorithm, we make an ablation study to analyze different integration ways during 100 episodes test set. We set the double attention modules after each convolution layers, F1, F2, F3 in Fig.3, as model named Combine1, Combine2, Combine3, shown as Table 4. Comparing with D3QN, Combine1 could improve 23.23% rewards and 45% success rate, but the average speed drop more than other models. Combine2 could keep a higher speed than Combine1 and Combine3, but raising of other indicators is not as good as above two model. Combine3 could achieve a higher reward, significantly improve 54% success rate while maintains the average velocity. We assume this is because after three convolutional layer, the feature map has more advanced semantic features which is useful for decision-making. Thence, we choose Combine3 as D3QN-DA algorithm integration method.

| Models     | SC   | ULCP | DAP   | ILCP   |
|------------|------|------|-------|--------|
| AE-D3QN    | 14.54| 2.72 | 21.97 | 4.16   |
| AE-D3QN-SA | 10.03| 1.74 | 20.72 | 3.59   |
| AE-D3QN-CA | 13.35| 1.73 | 20.52 | 4.19   |
| AE-D3QN-DA | 7.32 | 1.92 | 18.7  | 2.94   |

V. CONCLUSION

Autonomous driving decision making is still an important technical problem for fully autonomous driving. This work puts forward a hierarchical control structure based AE-D3QN-DA algorithm through giving different attentions for different vehicles to improve efficiency and ensure safety in uncertain highway environment. We experiment and evaluate the proposed method with reward metric and capability metric. The results show AE-D3QN-DA performance in the experiment by improve average total reward, success rate, average explore time and average explore distance while maintains average velocity. For the future work topics, we intend to analyze the mobility of the proposed method in more scenes, such as Intersection scene, and consider the trajectory prediction of other cars intention recognition.

ACKNOWLEDGMENT

Portions of this manuscript were presented at the 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Chiang Mai, Thailand, September 25-26, 2020.

REFERENCES

[1] C.-J. Hoel, K. Driggs-Campbell, K. Wolff, L. Laine, and M. J. Kochenderfer, “Combining planning and deep reinforcement learning in tactical decision making for autonomous driving,” IEEE transactions on intelligent vehicles, vol. 5, no. 2, pp. 294–305, 2019.

[2] R. Hult, G. R. Campos, E. Steinmetz, L. Hammerstrand, P. Falcone, and H. Wymeersch, “Coordination of cooperative autonomous vehicles: Toward safer and more efficient road transportation,” IEEE Signal Processing Magazine, vol. 33, no. 6, pp. 74–84, 2016.

[3] W. Schwarting, J. Alonso-Mora, and D. Rus, “Planning and decision-making for autonomous vehicles,” Annual Review of Control, Robotics, and Autonomous Systems, vol. 1, pp. 187–210, 2018.

[4] C. Urmon, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer et al., “Autonomous driving in urban environments: Boss and the urban challenge," Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.

[5] S. Ulbrich, A. Reschka, J. Rieken, S. Ernst, G. Bagschik, F. Dierkes, M. Nolte, and M. Maurer, “Towards a functional system architecture for automated vehicles,” arXiv preprint arXiv:1703.08557, 2017. [Online]. Available: https://arxiv.org/abs/1703.08557

[6] S. Aradi, “Survey of deep reinforcement learning for motion planning of autonomous vehicles,” IEEE Transactions on Intelligent Transportation Systems, 2020.

[7] P. Gippes, “A model for the structure of lane-changing decisions,” Transportation Research Part B: Methodological, vol. 20, no. 5, pp. 403–414, 1986.

[8] S. P. Sethi and Q. Zhang, Hierarchical decision making in stochastic manufacturing systems, Springer Science & Business Media, 2012.

[9] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnhke et al., “Junior: The stanford entry in the urban challenge,” Journal of field Robotics, vol. 25, no. 9, pp. 569–597, 2008.
A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, M. Fleming, A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, M. Sewak, “Deep Q network (DQN), double DQN, and dueling DQN,” in Proceedings of the AAAI Conference on Artificial Intelligence, 2018.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning,” in Thirty-second AAAI conference on artificial intelligence, 2018.

E. Leurent and J. Mercat, “Social attention for autonomous decision-making in dense traffic,” arXiv preprint arXiv:1911.12250, 2019.

M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Eesden, A. A. S. Awwal, and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” arXiv preprint arXiv:1803.01164, 2018.

J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2018, pp. 7132–7141.

E. Leurent, “An environment for autonomous driving decision-making,” https://github.com/eleurent/highway-env, 2018.

M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” Physical Review E, vol. 62, no. 2, p. 1805, 2000.

A. Kesting, M. Treiber, and D. Helbing, “General lane-changing model for car-following models,” Transportation Research Record, vol. 1999, no. 1, pp. 86–94, 2007.

SHUWEI ZHANG (Student Member, IEEE) received the B.E. degree in process equipment and control engineering from the Beijing Institute of Technology, China, in 2016, and the M.E. degree from Waseda University, Kitakyushu, Japan, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include autonomous driving, planning and decision making, and deep reinforcement learning.

YUTIAN WU (Student Member, IEEE) received the B.E. degree in computer science and software engineering from Southeast University, China, in 2017, and the M.E. degree from Waseda University, Kitakyushu, Japan, in 2019, where she is currently pursuing the Ph.D. degree. Her research interests include computer vision, deep learning, and 3-D scene understanding.

HARUTOSHI OGAI (Member, IEEE) received the Ph.D. degree in engineering from the Tokyo Institute of Technology in 1997. He is currently a Professor with the Graduate School of Information, Production, and Systems, Waseda University. His research interests include process control, process modeling and simulation, process analysis and data mining, autonomous driving.
HIROSHI INUJIMA received the BE degree from Department of Electrical Engineering, Hokkaido University, Japan, in 1975, and he joined Mitsubishi Electric Corporation. In 1989, he received his Ph.D. degree from Waseda University. His research and development activities have focused on degradation detection and control in plants. Since April 2002, he has been a Professor in Graduate School of Information, Production and System of Waseda University. His research area includes signal processing, diagnosis system, monitoring system, and artificial intelligence.

SHIGEYUKI TATENO received his Ph.D. degree from Department of Chemical Engineering, Kyushu University, Japan, in 2002. In 2003, he was an associate professor in Graduate School of Information, Production and Systems of Waseda University. Since April 2021, he has been a professor at Waseda University. His research area includes fault diagnosis, monitoring system, wireless communication, and artificial intelligence.