Parameterized Decision-making with Multi-modality Perception for Autonomous Driving

Yuyang Xia[1], Shuncheng Liu[1], Quanlin Yu[2], Liwei Deng[1], You Zhang[3], Han Su[2,1]✉, Kai Zheng[1]✉
[1]University of Electronic Science and Technology of China, China
[2]Yangtze Delta Region Institute(Quzhou), University of Electronic Science and Technology of China
[3]University of Michigan, USA
{xiayuyang,liushuncheng,quanlin.yu,deng_liwei}@std.uestc.edu.cn,
youzh@umich.edu, {hansu,zhengkai}@uestc.edu.cn

***Abstract*—Autonomous driving is an emerging technology that has advanced rapidly over the last decade. Modern transportation is expected to benefit greatly from a wise decision-making framework of autonomous vehicles, including the improvement of mobility and the minimization of risks and travel time. However, existing methods either ignore the complexity of environments only fitting straight roads, or ignore the impact on surrounding vehicles during optimization phases, leading to weak environmental adaptability and incomplete optimization objectives. To address these limitations, we propose a pArameterized decision-making framework with mUlti-modality percepTiOn based on deep reinforcement learning, called AUTO. We conduct a comprehensive perception to capture the state features of various traffic participants around the autonomous vehicle, based on which we design a graph-based model to learn a state representation of the multi-modal semantic features. To distinguish between lane-following and lane-changing, we decompose an action of the autonomous vehicle into a parameterized action structure that first decides whether to change lanes and then computes an exact action to execute. A hybrid reward function takes into account aspects of safety, traffic efficiency, passenger comfort, and impact to guide the framework to generate optimal actions. In addition, we design a regularization term and a multi-worker paradigm to enhance the training. Extensive experiments offer evidence that AUTO can advance state-of-the-art in terms of both macroscopic and microscopic effectiveness.**

***Index Terms*—Decision-making, Autonomous Vehicle, Reinforcement Learning**

I. Introduction

With the rapid growth in urbanization and vehicle ownership, most major cities around the world suffer from traffic congestion, resulting in serious traveling inefficiency, fuel waste, and air pollution [1]. Typically, road environments or drivers are to blame for traffic congestion [2]. Environment variables like road construction and a reduction in the number of lanes (bottleneck) can give rise to traffic congestion. Additionally, a driver's poor driving behaviors (e.g., hard braking and abrupt lane-changing) may result in traffic congestion or even accidents. However, the latter is more frequent due to some limitations of human drivers (limited field of view and long reaction time) and heterogeneity (different driving habits among drivers) [3]. It is highly challenging for human drivers to make optimal decisions all the time.

✉Corresponding author: Kai Zheng is with Shenzhen Institute for Advanced Study, University of Electronic Science and Technology of China, Shenzhen, China

In the last decade, autonomous driving has gained broad attention from the public, which aims to assist or even replace a human driver with a robot that constantly receives environmental information via various sensor technologies (as compared to human eyes) and consequently determines vehicle behaviors with proper algorithms (as compared to human brains) [4]. Considering a mechanism that perceives the surrounding traffic environment and makes decisions, it fits well within the realm of reinforcement learning [5]. There have been many works utilizing reinforcement learning-based methods [6]–[8] to accomplish autonomous driving and outperform traditional rule-based methods [9]–[11] and deep learning-based methods [12]–[14]. However, the existing works share some common limitations, including *weak environmental adaptability* and *incomplete optimization objectives*. For the first limitation, the aforementioned algorithms are implemented in simple traffic scenarios (e.g., highways). They assume that the environment is constant and vehicle information can be accurately acquired in advance, leading to low applicability and high uncertainty in complex traffics. Recently, some methods utilize cameras and/or LiDARs to perceive complex environmental information to adapt to autonomous driving on urban roads [15]–[17]. However, they over-simplify the environments only using images and/or point cloud data that ignore road maps and varying states of the autonomous vehicle, e.g., orientation and offset from lanes. Although multi-sensor fusion is the trend of autonomous driving, few studies have fully exploited multi-modal information from sensors and road networks that can improve adaptability. For the second limitation, existing approaches [15], [18], [19] mainly optimize the driving safety, traffic efficiency, and passenger comfort of autonomous vehicles, leaving the impact on other conventional vehicles largely uninvestigated. Generally, hard braking and forced lane-changing behaviors of a vehicle can cause a negative impact on its rear vehicles, which eventually causes traffic delays and even congestion [20], [21]. The former framework [16] attempts to make coarse-grained decisions with consideration of multiple impact situations. Its lane-changing behaviors are discrete rather than continuous steering angle control. Therefore, this method are not practical for real-world driving situations and cannot deal with the impact

of fine-grained steering angle control. Further, the discrete manner cannot roll back a lane-changing decision halfway, which may cause safety issues in some corner cases.

To enhance environmental adaptability, we can consider a comprehensive traffic environment including roads, vehicles, traffic signs, and traffic lights. The autonomous vehicle is able to collect information from different data sources including an offline high-definition map (HD map) and multiple onboard sensors [22]–[24], i.e., Camera, Light Detection and Ranging (LiDAR), Global Navigation Satellite System (GNSS), and Inertial Measurement Unit (IMU). Specifically, HD maps provide lane information (e.g., topology and curvature) and static traffic regulatory elements (e.g., traffic signs) along a specific route, and Cameras capture time-varying traffic light states, and LiDARs obtain the information (e.g., velocity, position, and size) of surrounding vehicles, and GNSS and IMU provide the position and orientation of the autonomous vehicle. In this context, the autonomous vehicle gets multi-modal inputs from HD maps and sensors, and a decision-maker outputs velocity and steering angle at each time step. To optimize decision-making, we can enable the autonomous vehicle to complete a specific route with driving safety, traffic efficiency, passenger comfort, and minimal impact on surrounding vehicles following the paradigm of reinforcement learning. However, the intuitions will face three main challenges: (1) Environmental information is complex and multi-modal. The aggregation methods in previous works [15], [16] are time-consuming and fail to exploit road semantic information since they model the multi-modal traffic participants as a complete graph and ignore the distinctions in traffic conditions on different lanes. (2) Both lane-following and lane-changing require calculating steering angles. Only generating a steering angle usually causes the autonomous vehicle to deviate far from the lane centerline since it cannot distinguish between lane-changing behaviors and lane-following behaviors on curved multi-lane roads. (3) It is common to optimize the safety, traffic efficiency, and passenger comfort of the autonomous vehicle [15], [18], [19], but minimizing the impact on surrounding vehicles has been less studied. How to design reward functions and measure the impact factor is challenging.

To tackle these challenges, we propose a pArameterized decision-making framework with mUlti-modality percepTiOn based on deep reinforcement learning, called AUTO. To address the first challenge, we propose a state representation model LCA that aims to learn a comprehensive representation of the multi-modal features. It organizes the multi-modal features as multiple agent-centric star graphs to efficiently represent their interaction relationship, and proposes a novel lane-wise fuse paradigm to effectively exploit road semantic information. To address the second challenge, we propose a reinforcement learning model RBP-DQN to optimize a parameterized action structure, which calculates three different sets of action values (i.e., steering angle and acceleration brake rate) to respectively represent left lane-changing, lane-following, and right lane-changing behaviors and then selects the one with the greatest reward value to execute. This design allows the autonomous vehicle to adjust the steering angle accurately, thus achieving better driving performance. In addition, due to the complexity of the traffic environment and action space, we design a regularization term to prevent large policy shifts during policy optimization. To address the third challenge, we design a hybrid reward function to guide our framework to learn the state representation and optimize the parameterized action calculation. It incorporates safety, traffic efficiency, passenger comfort, and impact terms. In particular, the impact term is used to penalize the autonomous vehicle when it executes harsh actions that force other vehicles to decelerate, thus reducing the impact of the autonomous vehicle on traffic flow.

To the best of our knowledge, this work provides the first data-driven solution to make decisions in continuous action space that considers multi-modal information inputs and the impact of the autonomous vehicle. Overall, our main contributions can be summarized as follows:

- We develop a decision-making framework that enables the autonomous vehicle to complete a route in comprehensive traffic with safety, traffic efficiency, passenger comfort, and minimal impact on surrounding vehicles.
- We propose an efficient graph-based model to exploit valuable features from multi-modal environmental data and design a parameterized action paradigm to calculate fine-grained actions based on coarse-grained decisions.
- We propose a hybrid reward function to guide the optimization of reinforcement learning and use a regularization term and a multi-worker setting to enhance the training.
- We conduct extensive experiments to evaluate the superiority of our framework on an open-source simulator from both macroscopic and microscopic metrics.

II. Problem Definition

A. Preliminary Concepts

We consider an interactive traffic environment where there is one autonomous vehicle A and a set of conventional vehicles $\mathbb{C}$ traveling on complex multi-lane roads. The autonomous vehicle is equipped with an HD map and onboard sensors, i.e., Camera, LiDAR, GNSS, and IMU. Benefiting from the development of feature extraction techniques [22]–[24] for map and sensor data, we can obtain the real-time multi-modal features of lanes, vehicles, and traffic lights along a specific route. Based on these features, the autonomous vehicle makes an action decision (i.e., a steering angle and an acceleration brake rate) at each time step t within a time duration $\mathbb{T}$ of interest. Next, we will explain some definitions and notations used in the rest of this paper.

Route. A route rou is a sequence of roads that the autonomous vehicle needs to follow in succession, which indicates a planned path from an origin to a destination. It can be denoted as $rou = \langle rd_1, rd_2, \ldots, rd_g \rangle$, where g is the number of roads in the route.

Lane. A lane l is part of a road used to guide drivers and reduce traffic conflicts. Usually, a road has multiple lanes, and they are numbered incrementally from the leftmost side

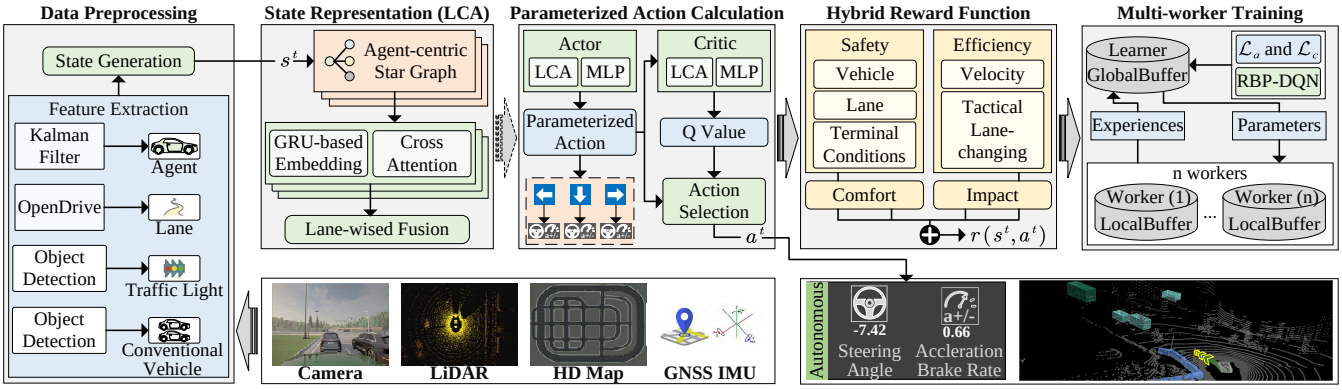


Fig. 1: Framework Overview

to the rightmost side, i.e., $l_1, l_2, \ldots, l_{\mathcal{K}}$, where l_1 and $l_{\mathcal{K}}$ indicate the leftmost lane and rightmost lane, respectively. Each lane is represented as a sequence of waypoints sampled equidistantly on the centerline at one-meter intervals [25], and the feature vector f_{wp} of each waypoint wp is defined as $f_{wp} = (alt, k, \theta, attr)$ that consists of the altitude, lane ID, orientation, and attribute of the waypoint. The altitude and orientation provide the slope and curvature of the waypoint, and the attribute refers to the static traffic sign information at the waypoint. We consider speed limit signs for now and set $attr$ of each waypoint as the speed limit.

Vehicle. We represent the features of the autonomous vehicle as $f_A = (vel, \theta, len, wid, off)$, where $f_A.vel$ is the velocity, $f_A.\theta$ is the orientation, $f_A.len$ is the length, $f_A.wid$ is the width, and $f_A.off$ denotes the offset from the current lane centerline [26]. Then for the conventional vehicles $\mathbb{C}$, we calculate their relative features with respect to the autonomous vehicle A. For example, the features of C_i are denoted as $f(C_i, A) = (dis, vel, \theta, len, wid, off)$, where $f(C_i, A).dis$ refers to the relative distance along the lane, $f(C_i, A).vel$ and $f(C_i, A).\theta$ refer to the relative velocity and orientation, $f(C_i, A).len$ and $f(C_i, A).wid$ denote the length and width, and $f(C_i, A).off$ is the offset from the current lane centerline.

Traffic Light. A traffic light tl consists of three signals (i.e., red, yellow, and green) that transmit real-time traffic regulation information to drivers, and vehicles are allowed to pass when the light is green. Formally, The features f_{tl} of a traffic light tl can be represented as $f_{tl} = (status, dis)$, where $status$ denotes the color of tl, and dis denotes the relative distance between the traffic light tl and the autonomous vehicle A. Specifically, we use one-hot vectors to represent the status of traffic lights. $[1, 0, 0]$, $[0, 1, 0]$ and $[0, 0, 1]$ denote the red light, yellow light, and green light, respectively.

Time Step. In order to model the problem more concisely, we treat the continuous time duration $\mathbb{T}$ as a discretized set of time steps, i.e., $\mathbb{T} = \langle 1, 2, \ldots, t, t+1, \ldots \rangle$. We denote Δt as the time interval between two consecutive time steps, which is set to 0.1 seconds following the previous works [18], [27].

Steering Angle. The steering angle deg determines the orientation of a vehicle, which is defined as the angle between the front of the vehicle and the steered wheel direction. The angle varies from $-60°$ to $60°$ [28]. The negative values mean the autonomous vehicle is turning left or changing lanes to the left, and vice versa.

Acceleration Brake Rate. The acceleration brake rate abr determines the velocity of a vehicle, which is defined as the magnitude of the accelerator and brake [28]. It is in the range of $[-1, 1]$, where positive values $(0, 1]$ mean accelerating, negative values $[-1, 0)$ mean braking, and 0 means a neutral state with no braking and accelerating.

Objective. Our objective is that the autonomous vehicle can output an optimal steering angle deg and an acceleration brake rate abr to follow a specific route while achieving safety, driving efficiency, passenger comfort, and minimal impact on the surrounding conventional vehicles.

B. DRL-based Decision-making

Considering the mechanism that an autonomous vehicle perceives the surrounding traffic environment and makes a decision, it fits well within the realm of Partially Observable Markov Decision Process (POMDP) [29] and deep reinforcement learning [30]. Then we present some important definitions under POMDP as follows: - *Agent*: the autonomous vehicle A. - *Environment*: the surrounding environment. - *State* s^t: current and historical multi-modal inputs extracted from the environment, including the features of the agent and its surrounding lane waypoints, vehicles, and traffic lights along the route. - *Action* a^t: a steering angle deg^t and an acceleration brake rate abr^t of the agent. - *Reward* $r(s^t, a^t)$: a feedback value after the agent performs a^t at s^t, which is the aggregation of multiple aspects: safety, traffic efficiency, passenger comfort, and impact on rear conventional vehicles. As the agent explores the environment, the deep reinforcement learning paradigm aims to exploit valuable features from states and form a policy that can generate actions with large reward values.

III. Framework and Methodology

A. Framework Overview

Figure 1 shows the architecture of our framework, which consists of five components: data preprocessing, state representation, parameterized action calculation, hybrid reward function, and multi-worker training. For the *data preprocessing*, we take the data from HD maps and multiple sensors (i.e.,

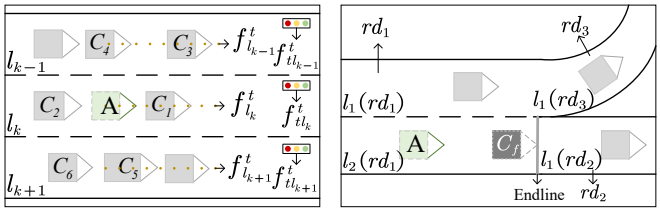


Fig. 2: Example of State Generation

Camera, LiDAR, GNSS, and IMU) as input, based on which we respectively extract the feature vectors of lane waypoints, vehicles, and traffic lights and generate a multi-modal state s^t for the agent. For the *state representation*, we propose a lane-wised cross attention model (LCA) to learn a latent representation of the state s^t. It organizes s^t as multiple agent-centric star graphs and uses a GRU network to embed the node features. Then, it uses a cross attention mechanism to aggregate each star graph and introduces a lane-wise paradigm to fuse these aggregated results as a state representation. For the *parameterized action calculation*, we first integrate LCA into an actor network and a critic network, respectively. Then, we compute an action a^t using a parameterized action structure that first decides whether to perform a lane-changing decision (high level) and then compute an exact action to execute (low level). For the *hybrid reward function*, we calculate a reward value $r(s^t, a^t)$ for the action a^t and state s^t, which serves as a signal to guide the agent to learn an optimal action policy. For the *multi-worker training*, we use a regularization technique to improve the convergence performance of our reinforcement learning model RBP-DQN and speed up the training speed using distributed computation following the previous studies [31].

B. Data Preprocessing

This component extracts multi-modal features (i.e., lane waypoints, vehicles, and traffic lights) from the raw data of HD maps and sensors. Then based on these features and a specific route, we generate a fixed-dimensional state s^t at time step t for the agent.

Feature Extraction. We conduct the experiments in a widely-used simulator Carla that mimics real-world traffic scenarios. It provides high-fidelity sensor models that can replicate the characteristics (e.g., field of view and resolution) of real-world sensors and generate synthetic data based on the virtual environment and the positions of objects. To obtain the surrounding environment features, we first take advantage of the extended Kalman filter [32] to process the data from GNSS and IMU to get the position and orientation of the agent, based on which we can locate the agent and the route in an OpenDrive-based HD map [33] and acquire the surrounding stationary lane features. Then with the aid of object detection technology [34], [35], we extract the features of surrounding conventional vehicles from the point cloud data by LiDAR and extract the features of traffic lights from the image data by Camera. In addition, the detection distances of Camera and LiDAR always have an impact on the decision-making performance of autonomous driving. To test the impact, we conduct a hyper-parameter sensitivity analysis in Section IV-J.

State Generation. Based on the feature extraction, at time step t, we can collect the feature vector f^t_A of the agent and f^t_k, f^t_{k-1}, f^t_{k+1} that respectively consists of the multi-modal features on the current lane l_k, the left lane l_{k-1}, and the right lane l_{k+1}. As shown in the left side of Figure 2, we select six conventional vehicles on three lanes, which cover six key areas centered on the agent, i.e., 1) front (C_1), 2) rear (C_2), 3) front left (C_3), 4) rear left (C_4), 5) front right (C_5), 6) rear right (C_6) areas [36], [37]. Then taking f^t_k as an example, f^t_k can be represented as $f^t_k = (f^t_{l_k}, f^t_{v_k}, f^t_{tl_k})$. Specifically, $f^t_{l_k}$ is the concatenation of the feature vectors of waypoints on the lane l_k, i.e., $f^t_{l_k} = f^t_{wp_{k,1}} \oplus f^t_{wp_{k,2}} \oplus \cdots \oplus f^t_{wp_{k,m}}$, where m is the number of waypoints. $f^t_{v_k}$ is the concatenation of the feature vectors of vehicle C_1 and C_2 on l_k, i.e., $f^t_{v_k} = f^t(C_1, A) \oplus f^t(C_2, A)$. $f^t_{tl_k}$ refers to the feature vector of traffic light on l_k.

In summary, the state s^t consists of F^t_A and $X^t_{k-1} = [F^t_{l_{k-1}}, F^t_{v_{k-1}}, F^t_{tl_{k-1}}]$, $X^t_k = [F^t_{l_k}, F^t_{v_k}, F^t_{tl_k}]$, and $X^t_{k+1} = [F^t_{l_{k+1}}, F^t_{v_{k+1}}, F^t_{tl_{k+1}}]$. Note that F means that it contains historical feature vectors in the past z time steps rather than just the current time step, e.g., $F^t_A = [f^{t-z+1}_A, f^{t-z+2}_A, \ldots, f^t_A]$.

C. State Representation

Existing studies on the representation learning of traffic context can be divided into two categories: raster-based methods [38], [39] and vector-based methods [40]–[42]. The raster-based methods rasterize the traffic scene of the agent into a grided image and apply a series of standard convolution layers to process it. However, these approaches require significantly more computation to train and the rasterization process inevitably results in information loss [43]. Recently, the vector-based method as a more succinct representation has developed rapidly in many traffic tasks [40], [44], [45], which uses feature vectors to represent different traffic participants (i.e., lane waypoints, conventional vehicles, and traffic lights) and apply the graph neural network to capture the complex interaction relationship of them.

Therefore, there are many decision-making works [15], [16], [18], [19] increasingly applying vector-based methods to learn a representation vector of the traffic scene. But in this paper, these works usually lead to unsatisfactory results due to the following reasons: (1) Their aggregation manners are redundant since they model the autonomous vehicle and other traffic participants as a complete graph. Although this design can fully exploit the interaction relationship in the traffic, some interactions are irrelevant to the decision-making of our autonomous vehicle, which can introduce additional calculations and degrade the driving performance. (2) They fail to exploit road semantic information in the traffic since they only focus on the aggregation of different traffic participants while ignoring the distinctions in traffic conditions on different lanes. Evidently, an effective road semantic representation can aid the autonomous vehicle to improve its driving decisions, especially lane-changing decisions. (3) Their robustness is

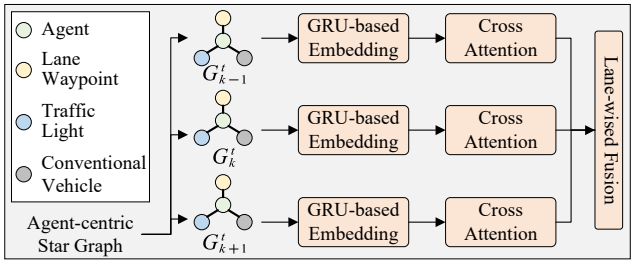


Fig. 3: Network Structure of LCA

poor. In practice, the features collected by the autonomous vehicle are inevitably subject to noises, e.g., the traffic light detection in our experiments has an error rate of up to 18% compared to the ground truth. Solely relying on the current state features inherently leaves decision-making vulnerable to observation noises. In contrast, incorporating historical features into states empowers the model to comprehend changes in the environment and thus improve the robustness to noises [46].

In this work, we propose a lane-wised cross attention model, called LCA, to learn a representation of the state s^t as shown in Figure 3. To efficiently exploit the interaction relationship, we organize the state s^t into three agent-centric star graphs (i.e., G^t_{k-1}, G^t_k, and G^t_{k+1}) to enable the autonomous vehicle to focus primarily on the influence of different traffic participants on its future decisions. Then, we apply a GRU-based embedding network to embed the historical state features in each node and apply a cross attention layer to aggregate each graph. This step has embedded the road's waypoint features (e.g., position, orientation, and traffic sign features). Finally, we fuse them lane by lane to augment road semantic information.

Network Structure. In this section, we will detail the workflow of LCA as follow:

(1) GRU-based Embedding. Taking G^t_k as an example, G^t_k is an undirected graph $G^t_k = (\mathbb{V}, \mathbb{E})$, where the node features of $\mathbb{V}$ are F^t_A (agent), $F^t_{l_k}$ (lane waypoint), $F^t_{v_k}$ (conventional vehicle), and $F^t_{tl_k}$ (traffic light), and $\mathbb{E}$ includes the edges between the agent node and other nodes. Each node contains the corresponding feature vectors of z time steps, e.g., $F^t_A = [f^{t-z+1}_A, f^{t-z+2}_A, \ldots, f^t_A]$. Then, we encode these node features by the Gated Recurrent Unit (GRU) [47] as follows:

$$\begin{aligned} h^\tau_A &= GRU(f^\tau_A, h^{\tau-1}_A; W_1), \\ h^\tau_{l_k} &= GRU(f^\tau_{l_k}, h^{\tau-1}_{l_k}; W_2), \\ h^\tau_{v_k} &= GRU(f^\tau_{v_k}, h^{\tau-1}_{v_k}; W_3), \\ h^\tau_{tl_k} &= GRU(f^\tau_{tl_k}, h^{\tau-1}_{tl_k}; W_4), \end{aligned} \tag{1}$$

where $\tau \in \{t-z+1, t-z+2, \ldots, t\}$, $W_1 \in \mathbb{R}^{5\times D_1}$, $W_2 \in \mathbb{R}^{4m\times D_1}$, $W_3 \in \mathbb{R}^{12\times D_1}$, and $W_4 \in \mathbb{R}^{4\times D_1}$ are the weight matrices. The hidden vectors of GRU default to zero vectors when τ is $t-z+1$. Finally, we take the hidden vectors (i.e., $h^t_A, h^t_{l_k}, h^t_{v_k}, h^t_{tl_k}$) at time step t as the node embedding.

(2) Cross Attention. After embedding each node in G^t_k, we utilize a cross attention machanism [48] to aggregate G^t_k into a feature vector E^t_k based on its edge connection as follows:

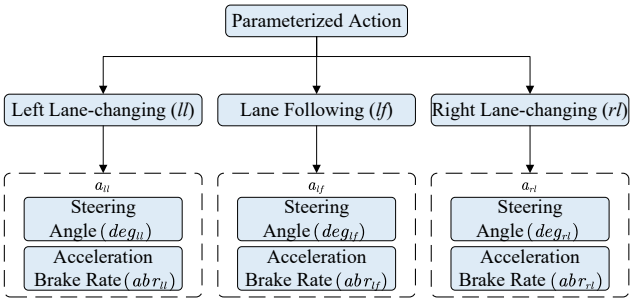


Fig. 4: Parameterized Action

$$E^t_k = Softmax(\frac{W_5 h^t_A (W_6 H^t_k)}{\sqrt{D_2}})(W_7 H^t_k), \tag{2}$$

where $H^t_k = [h^t_{l_k}, h^t_{v_k}, h^t_{tl_k}]$, $W_5 \in \mathbb{R}^{D_1\times D_2}$, $W_6 \in \mathbb{R}^{D_1\times D_2}$, and $W_7 \in \mathbb{R}^{D_1\times D_2}$ are the weight matrices for computing a query matrix, a key matrix, and a value matrix based on the attention mechanism [48], and D_2 is the embedding dimension of these matrices.

(3) Lane-wised Fusion. We execute the embedding and aggregation on all star graphs, i.e., $G^t_{k-1}, G^t_k, G^t_{k+1}$. Afterwards, we can get three feature vectors E^t_{k-1}, E^t_k, and E^t_{k+1}, which have respectively aggregated the multi-modal features on the lane l_{k-1}, l_k, l_{k+1}. Finally, we use a multi-layer perceptron network to fuse the concatenation of these vectors into a state representation E^t_s as follows:

$$E^t_s = W_8(E^t_{k-1} \oplus E^t_k \oplus E^t_{k+1}), \tag{3}$$

where $W_8 \in \mathbb{R}^{3D_2\times D_3}$ is the weight matrix.

D. Parameterized Action Calculation

In order to form an optimal action policy with large reward values, previous researchers propose the Q-learning method [49] to construct a Q-table to store the expected reward value Q. The idea of this method is to query the Q-table for the action with the largest Q value according to the current state s^t. However, the query and update of the Q-table are time-consuming, and the dimension of the Q-table will increase dramatically if the number of state features and alternative actions increases. Therefore, considering the complexity and continuity of states and actions in our work, we use neural networks instead of the Q-table, which uses an actor network and a critic network to respectively compute actions and estimate Q values. Next, we will introduce the action structure and calculation of our framework.

Parameterized Action. Intuitively, an action of the agent in multi-lane scenarios requires a high-level decision (whether to perform lane-changing) and a low-level action (how to implement the high-level decision). A single steering angle is only suitable for open spaces with no lane division but not for multi-lane scenarios since it cannot distinguish between lane-changing behaviors and lane-following behaviors on curved roads [50], [51]. Therefore, as shown in Figure 4, we introduce a parameterized action structure to separate different lane-changing decisions. Specifically, we define three mutually exclusive high-level decisions P comprising left lane-changing

(ll), lane following (lf), and right lane-changing (rl), i.e., $P = (ll, lf, rl)$. Each decision p in P is coupled with a low-level action a_p that consists of a steering angle deg_p and an acceleration brake rate abr_p, i.e., $a_p = (deg_p, abr_p)$.

Actor. Given the state s^t, the actor network is used to compute actions. For the network structure, it includes an LCA model in Section III-C to learn a state representation E_s^t, based on which we use a multi-layer perceptron network (MLP) and an activation function $Tanh$ to compute an action vector $\mathcal{A}^t$ as follows:

$$\mathcal{A}^t = Tanh(W_9 E_s^t), \tag{4}$$

where $W_9 \in \mathbb{R}^{D_3 \times 6}$ is the weight matrix. $\mathcal{A}^t$ consists of all actions, i.e., $\mathcal{A}^t = [a_{ll}^t, a_{lf}^t, a_{rl}^t]$, where the steering angle and acceleration brake rate in each action will be scaled into the defined ranges in Section II-A.

Q Value. In this work, the goal of deep reinforcement learning is to select actions with maximal expected γ-discounted cumulative reward, i.e., the state-action value function Q [49]. A better action is expected to have a larger Q value. When the agent performs an action a_p^t of decision p at the state s^t, the Q value $Q(s^t, p, a_p^t)$ is calculated using the Bellman Equation [52], as follows:

$$\begin{aligned} &Q(s^t, p, a_p^t) := Q(s^t, deg_p^t, abr_p^t) = \mathbb{E}_{s^{t+1}}\big[r(s^t, a^t) + \\ &\gamma \max(\sup Q(s^{t+1}, \hat{p}, a_{\hat{p}}^{t+1}) | a_{\hat{p}}^{t+1} = (deg_{\hat{p}}^{t+1}, abr_{\hat{p}}^{t+1}))\big], \end{aligned} \tag{5}$$

where $\gamma \in [0, 1)$ is a discount factor, the max operation is used to find the best decision $\hat{p}$ from $P = (ll, lf, rl)$, and the sup is used to find the best action $a_{\hat{p}}^{t+1}$ under $\hat{p}$.

Critic. Given the state s^t and the action vector $\mathcal{A}^t$, we utilize a critic network to approximate a vector $Q(s^t, P, \mathcal{A}^t)$ that contains three Q values of the actions in $\mathcal{A}^t$, i.e., $Q(s^t, P, \mathcal{A}^t) = [Q(s^t, ll, a_{ll}^t), Q(s^t, lf, a_{lf}^t), Q(s^t, rl, a_{rl}^t)]$. For the network structure, it first includes an LCA model in Section III-C to learn a state representation E_s^t and then uses a multi-layer perceptron network (MLP) and an activation function $ReLU$ to learn an action representation E_a^t of $\mathcal{A}^t$ as follows:

$$E_a^t = ReLU(W_{10} \mathcal{A}^t), \tag{6}$$

where $W_{10} \in \mathbb{R}^{6 \times D_4}$ is the weight matrix and D_4 is the dimension of E_a^t. Then, we apply another MLP to compute $Q(s^t, P, \mathcal{A}^t)$ with the concatenation of E_s^t and E_a^t as input, as follows:

$$Q(s^t, P, \mathcal{A}^t) = W_{11}(E_s^t \oplus E_a^t), \tag{7}$$

where $W_{11} \in \mathbb{R}^{(D_3 + D_4) \times 3}$ is the weight matrix of the MLP.

Action Selection. After acquiring $\mathcal{A}^t$ and $Q(s^t, P, \mathcal{A}^t)$ from the actor network and critic network respectively, this component is utilized to select an action for the agent A. Based on a specific route of A, we first need to exclude actions that violate the route. It can be divided into two scenarios: (1) Mandatory lane-changing scenario. Taking the scenario (Figure 2) mentioned in the state generation of Section III-B as an example, the agent must change from the current lane to the left lane before the intersection. In this scenario, we will exclude the action of right lane-changing when the agent is less than 100 meters away from the endline of the current lane following [53], [54]. The fake stationary vehicle on the endline will force the agent to perform left lane-changing in advance or stop and wait. (2) Non-mandatory lane-changing scenario. If the target lane of a lane-changing decision is not in the route, the action of this decision will be excluded.

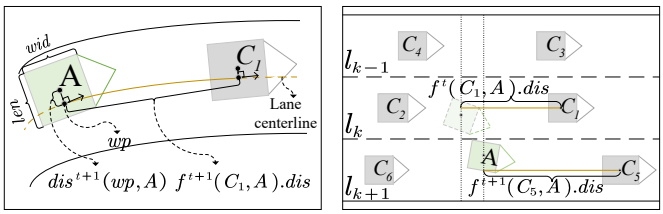


Fig. 5: Example of Reward Calculation

We exclude an action by setting its corresponding Q value to negative infinity. Then, we choose an action that has maximal Q value from all actions as follows:

$$\begin{aligned} \hat{p} &= argmax Q(s^t, P, \mathcal{A}^t), \\ a^t &:= a_{\hat{p}}^t = (deg_{\hat{p}}^t, abr_{\hat{p}}^t), \end{aligned} \tag{8}$$

where $a_{\hat{p}}^t$ is the action that has maximal Q value. Therefore at time step t, the steering angle $deg_{\hat{p}}^t$ and acceleration brake rate $abr_{\hat{p}}^t$ is the action a^t performed by the agent.

E. Hybrid Reward Function

The reward function serves as an exploration signal to teach the agent to learn an optimal action policy. In this work, a good action of the agent A should achieve safety, driving efficient, passenger comfort, and minimal impacts on other vehicles. Therefore, we construct a hybrid reward function considering four aspects as follows:

$$r(s^t, a^t) = w_1 r_1^t + w_2 r_2^t + w_3 r_3^t + w_4 r_4^t, \tag{9}$$

where w_1, w_2, w_3, w_4 are four tunable coefficients to adjust the importance of safety, traffic efficiency, passenger comfort, and impact, respectively. Next, we define the reward values for safety (r_1^t), traffic efficiency (r_2^t), passenger comfort (r_3^t), and impact (r_4^t).

Safety. We set up three safety reward terms to penalize the agent A for unsafe behaviors and traffic violations.

(**1**) Safety reward for vehicle. This reward is set to avoid the agent from colliding with other vehicles and we use time-to-collision (TTC) [55], [56] as the safety indicator of the agent. As shown in the left of Figure 5, after A performs an action a^t, TTC^{t+1} is calculated as $TTC^{t+1} = \frac{f^{t+1}(C_1, A).dis - len_{AC_l}}{f^{t+1}(C_1, A).vel}$, where $f^{t+1}(C_1, A).dis$ denotes the relative distance between C_1 and A along the lane, len_{AC_l} equals to half the sum of the length of A and C_1, and $f^{t+1}(C_1, A).vel$ denotes their relative velocity. Following [18], if TTC^{t+1} is between 0 and a threshold $\mathcal{G}$, the agent will receive a negative reward $r_{1,1}^t$ that is calculated as $r_{1,1}^t = max(-3, log(\frac{TTC^{t+1}}{\mathcal{G}}))$, otherwise $r_{1,1}^t = 0$.

(**2**) Safety reward for lane. This reward is set to avoid the agent from a large lane offset. As shown in the left of Figure 5, we first find the nearest waypoint wp of the agent and then calculate the Euclidean distance $dis^{t+1}(wp, A)$ between wp

and A. Then, the safety reward $r_{1,2}^t$ for lane is calculated as $r_{1,2}^t = -\frac{dis^{t+1}(wp,A)}{d}$, where d is the lane width.

(**3**) Terminal conditions. We consider four terminal conditions for the agent: 1) collision with conventional vehicles, 2) out of the road, 3) route deviations, and 4) violation of traffic rules (i.e., traffic lights and signs). Once they happened, the current episode will be terminated and the agent receives a negative reward $r_{1,3}^t$ of -20.

Traffic Efficiency. We set up two traffic efficiency reward terms to reward high velocities and tactical lane-changing behaviors of the agent.

(**1**) Traffic Efficiency reward for velocity. The velocity $f_A^{t+1}.vel$ of the agent directly reflects its driving efficiency. Thus, the efficiency reward value $r_{2,1}^t$ for velocity is defined as $r_{2,1}^t = \frac{f_A^{t+1}.vel}{vel_{max}}$, where vel_{max} denotes the speed limit.

(**2**) Traffic Efficiency reward for tactical lane-changing. A tactical lane-changing behavior refers to a maneuver where the agent attempts to avoid following a crowded lane [10]. As shown in the right of Figure 5, if the agent A changes from a lane to an adjacent lane, we record the relative distance $f^t(C_1, A).dis$ between A and C_1 at time step t and the relative distance $f^{t+1}(C_5, A).dis$ between A and C_5 at time step $t+1$. Then, the efficiency reward $r_{2,2}^t$ for tactical lane-changing is as $r_{2,2}^t = (\frac{f^{t+1}(C_5,A).dis - len_{AC_5}}{f^t(C_1,A).dis - len_{AC_1}} - 1) * 20$. In this way, the agent will receive a positive reward if it changes from a crowded lane to a freer lane, otherwise a negative reward.

To this end, we aggregate $r_{2,1}^t$ and $r_{2,2}^t$ to get the traffic efficiency reward r_2^t, i.e., $r_2^t = \sum_{i=1}^{2} r_{2,i}^t$.

Passenger Comfort. Jerk, defined as the change rate of acceleration, is used to measure comfort since it has a strong influence on the comfort of passengers [18]. The passenger comfort reward value r_3^t is defined as $r_3^t = -\frac{|acc^t - acc^{t-1}|}{2acc_{thr}}$, where acc^t and acc^{t-1} are two accelerations of agent A at time step t and $t-1$, acc_{thr} is an acceleration threshold.

Impact. The velocity change or lane-changing behavior of the agent could affect its rear vehicles, reducing their driving efficiency and comfort, especially for some harsh behaviors, e.g., hard braking and abrupt lane-changing [36]. To reduce the impact of the agent, we design an impact reward value to measure the degree that the agent forces the rear vehicles to decelerate. Based on the analysis of existing studies [36], [56], whether a vehicle decelerates or changes lanes, it only affects the vehicle behind it. Thus, after the agent performs an action a^t, we record the deceleration of C_2 to calculate the impact reward r_4^t as follows:

$$r_4^t = \begin{cases} \frac{f_{C_2}^{t+1}.vel - f_{C_2}^t.vel}{2acc_{thr} * \Delta t}, & f_{C_2}^t.vel - f_{C_2}^{t+1}.vel > v_{thr} \\ 0, & otherwise, \end{cases} \quad (10)$$

where C_2 is the first vehicle behind the agent, v_{thr} is a deceleration threshold used to measure whether the agent affects C_2, and $2acc_{thr} * \Delta t$ refers to the possible maximum deceleration between two consecutive time steps. The impact reward r_4^t only works when C_2 decelerates greater than v_{thr} at time step $t+1$.

F. Multi-worker Training

In this section, we will introduce the training process of our reinforcement learning model and a multi-worker setting that is used to accelerate the training.

Model Training. In our decision-making task, the network model aims to optimize the parameterized action calculation in Section III-D, which consists of a discrete lane-changing decision and continuous action commands (i.e., steering angle and accelerate brake rate). Recently, a BP-DQN model [16] is proposed to directly learn the action policy in the parameterized action space, which has achieved the state-of-the-art performance. It concurrently outputs three actions corresponding to three different lane-changing decisions and then automatically selects the best action to execute, However, this method suffers from the overestimation problem of Q values, leading to suboptimal action policy and even training failure in our experiments. Therefore, we propose a RBP-DQN model that introduces a behavior cloning regularization technique [57] to solve the overestimation problem and thus improve the policy optimization. In RBP-DQN, the loss function of the actor network in Section III-D is formulated as follows:

$$\mathcal{L}_a = -\sum_{p \in P} (\frac{\alpha Q(s^t, p, u_p(s^t; W_a))}{AvgQ} + (u_p(s^t; W_a) - a^t)^2), \quad (11)$$

where W_a denotes all the learnable parameters in the actor network, u_p refers to the learned action policy, α is a tunable coefficient and $AvgQ$ is the average Q value in a batch of training experiences. Instead of just using the negative sum of Q values as the loss function, RBP-DQN first uses α to adjust the weight of the Q value $Q(s^t, p, u_p(s^t; W_a))$ and then add $(u_p(s^t; W_a) - a^t)^2$ into the loss function. It can reduce the shift between the learning policy and historical experiences and thus avoid the overestimation of Q values [57]. Compared to other Q regularization methods [58], [59], this method fits the parameterized action structure perfectly and achieves stable optimization in our task.

In addition, the critic network remains the loss function in BP-DQN, which is formulated as follows:

$$\mathcal{L}_c = \frac{1}{2}(y - Q(s^t, p, a_p^t; W_c))^2, \quad (12)$$

where W_c denotes all the learnable parameters in the critic network, a_p^t is the action performed by the agent at state s^t, and y is a target Q value calculated by a target actor network and a target critic network following [60].

Multi-worker Setting. In our experiments, we find that the environment interaction and update in the simulation take much longer time than the model update. Therefore, in order to make full use of computing resources and improve the training efficiency of our model, we employ multiple agents to interact with the traffic environment following the standard multi-worker setting in [31], [61]. Specifically, there are n workers and a learner in our framework. Each worker initializes an agent to asynchronously explore the environment with different random noises and maintains a separate replay buffer called *LocalBuffer* to collect experiences. The learner is

responsible for updating the network parameters (i.e., W_c and W_a) and manages all experiences with a replay buffer called *GlobalBuffer*.

Algorithm 1: Worker

Input: The parameters W_a of the actor network and W_c of the critic network from the learner, the buffer size B_l of *LocalBuffer*

repeat
  $s^t \leftarrow$ State generation (Section III-B);
  $\mathcal{A}^t \leftarrow$ Actor(W_a) (Equation 4);
  $Q(s^t, P, \mathcal{A}^t) \leftarrow$ Critic(W_c) (Equation 6 and 7);
  $\hat{p}, a^t \leftarrow$ Action selection (Equation 8);
  $r(s^t, a^t) \leftarrow$ Reward calculation (Equation 9);
  $s^{t+1} \leftarrow$ Environment.step();
  LocalBuffer.add($(s^t, \hat{p}, \mathcal{A}^t, r(s^t, a^t), s^{t+1})$);
  if *LocalBuffer*.size() $\geq B_l$ **then**
    $exp \leftarrow$ *LocalBuffer*.get();
    GlobalBuffer.add(exp);
  end
until *convergence*;

Algorithm 2: Learner

Input: The size b of mini-batches, the buffer size B_g of *GlobalBuffer*

repeat
  $EXP \leftarrow$ Sample b experiences;
  $\mathcal{L}_c$, $\mathcal{L}_a \leftarrow$ Compute loss (Equation 11 and 12);
  W_c, $W_a \leftarrow$ Update parameters;
until *convergence*;

IV. Experiments

A. Experimental Settings

We conduct our experiments on the CARLA simulator [1], which is a widely-used project focused on creating a publicly available virtual environment for the autonomous driving industry [62]. It has 8 available towns and supports almost all sensors with the goal of flexibility and realism in high-fidelity simulations. We use 7 towns for training and hold out Town05 for evaluation following the previous work TransFuser [12]. The training benchmark has 35 routes and the evaluation benchmark has 20 routes, each with various road structures, e.g., intersections. In each episode, the autonomous vehicle is required to follow the predefined routes without colliding, driving off the road, or violating red lights and speed limits. Further, we vary the traffic density in each episode from $60 vehicles/km$ to $180 vehicles/km$ and vary the weather condition (e.g., normal, rain, fog, and night) to demonstrate the scalability and robustness of our framework. Based on these environment variables, there are $16,800$ environment settings for training and $9,600$ environment settings for evaluation.

Implementation Details. All of the experiments are run on an Ubuntu Server with an Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz and 4 NVIDIA GeForce RTX 3080 GPUs. The detection range D_{tl} of traffic lights by the camera is set to $60m$, and the detection range D_v of conventional vehicles by LiDAR is set to $90m$. The time interval Δt between two decisions is set to $0.1s$ [56]. For the network structure in the state representation and parameterized action calculation, we set the corresponding dimensions as $D_1 = 32$, $D_2 = 32$, $D_3 = 32$, and $D_4 = 32$. For the hybrid reward function, the TTC threshold $\mathcal{G}$ in the safety reward is set as $4s$, and the acceleration threshold acc_{thr} in the comfort reward is set to $3m/s^2$ following the settings in the previous work [18]. In addition, the velocity change threshold v_{thr} in the impact reward is set to $0.1m/s$, which is used to determine if driving behaviors of the autonomous vehicle affect its rear vehicle [3]. To the end, we train our model using the Adam optimizer [63] with a scheduled learning rate of 0.001 and a batch size of 128. The reward discount factor γ in the Bellman function is 0.9 and the target actor network and critic network use a soft update mechanism with an updated ratio of 0.01 similar to DDPG [60]. In the setting of multi-worker training, We employ 11 (i.e., $n = 11$) workers to collect experiences and one learner to update parameters, which can make full use of the computing resource.

Baselines. We compare our framework with the following methods.

(**1**) ACC-LC [9], [10]. Rule-based method that includes an adaptive cruise algorithm and a traditional lane-changing model to make decisions.

(**2**) TransFuser [12]. End-to-End imitation learning-based method that takes the raw sensor data as input and uses a transformer-based network structure to imitate labeled human driving behaviors.

(**3**) DRL-MF [15]. Deep reinforcement learning-based method that exploits useful features from image and point cloud data. It cannot perceive road maps and pose information (e.g., orientation) of the autonomous vehicle and ignore its impacts on the rear traffic flow.

(**4**) HEAD [16]. Deep reinforcement learning-based method that proposes a BP-DQN model to make decisions. It only computes discrete lane-changing decisions without continuous steering angle control. For a fair comparison, we integrate our state representation module into HEAD to learn the multi-modal features since it only utilizes vehicle features to make decisions.

Variants. In addition to these baselines, we also perform ablation experiments with some variants of our framework.

To evaluate the impact reward in the hybrid reward function in Section III-E, we set a variant to test its effectiveness in reducing the impact on the rear vehicle.

(**1**) AUTO-NoIMP. We remove the impact reward value and only consider the safety, traffic efficiency, and passenger comfort reward values.

To evaluate our state representation module III-C, we first set three variants to test its effectiveness and efficiency in exploiting useful features from multi-modal inputs.

(**2**) AUTO-NoVEC. Instead of vector representations of the multi-modal features in the state representation module, it ras-

[1] https://carla.readthedocs.io/en/0.9.14/

TABLE I: Macroscopic Evaluation

Methods	Avg DT-A (s)	Avg AT-C (s)	Avg DI-C
ACC-LC	57.08	7.03	1.46
TransFuser	56.45	6.73	1.43
DRL-MF	55.02	7.22	1.49
HEAD	55.57	6.55	1.41
AUTO-NoIMP	**53.33**	6.67	1.42
AUTO	53.61	**5.48**	**1.35**

terizes these features in a bird-eye view and uses convolutional layers to exploit useful features following [64].

(**3**) AUTO-NoSTAR. Instead of the agent-centric star graph in the state representation module, it constructs a complete graph to model the relationship between the agent and other traffic participants, i.e., lane waypoints, conventional vehicles, and traffic lights.

(**4**) AUTO-NoLANE. Instead of the lane-wised fusion paradigm in the state representation module, it directly fuses all feature vectors on different lanes by the cross attention layer.

Then, we set a variant to evaluate the robustness.

(**5**) AUTO-NoHIS. Instead of using historical features in the state generation module in Section III-B, it only uses features at the current time step as the state. Afterwards, we use MLP to embed node features rather than GRU.

To evaluate the multi-worker training setting in Section III-F, we set a variant to test its effectiveness in improving the convergence performance of our framework.

(**6**) AUTO-NoMUL. Instead of multiple workers, it only uses one worker to explore the environment and store experiences.

Compared Methods of Reinforcement Learning. For our reinforcement learning model RBP-DQN, we compare it against three methods for optimizing our parameterized action calculation, as follows:

(**1**) Q-PAMDP [65]. Parameterized Q-learning method that alternates learning discrete decisions and continuous action commands.

(**2**) P-DDPG [66]. Parameterized deep deterministic policy gradients method that collapse the parameterized action into a continuous one.

(**3**) BP-DQN [16]. Branched parameterized deep Q-network that directly learns the parameterized action structure without Q regularization.

B. Macroscopic Evaluation

In this section, we conduct a macroscopic evaluation of the autonomous vehicle regarding its traffic efficiency and impact on other vehicles. The compared methods consist of our framework AUTO, five baselines, and a variant AUTO-NoIMP.

Evaluation Metrics At each evaluation episode, all methods enable the autonomous vehicle to complete a route safely and without violation of traffic rules. Then, we utilize three metrics in each evaluation episode to evaluate the macroscopic effectiveness of our framework.

(**1**) Average driving time of the autonomous vehicle (AvgDT-A). We record the driving time of the autonomous vehicle for every $1km$ road. A smaller AvgDT-A indicates that the autonomous vehicle has higher driving efficiency.

(**2**) Average affected time of the rear vehicle by the autonomous vehicle (AvgAT-C). We record the total time that the rear vehicle decelerates greater than $0.1m/s$ within a time step (i.e., $0.1s$). A smaller AvgAT-C indicates that the rear vehicle is less affected by the autonomous vehicle.

(**3**) Average delay index of the rear vehicle of the autonomous vehicle (AvgDI-C). Following [56], the delay index (DI) refers to the ratio of the actual travel time to the ideal travel time in free state.

We report AvgDT-A, AvgAT-C, and AvgDI-C in Table I. As depicted, our framework AUTO achieves a shorter AvgDT-A and AvgAT-C and has a smaller AvgDI-C compared to these baselines. These results demonstrate that AUTO not only enables the autonomous vehicle to have high driving efficiency but also reduces its impact on the rear vehicle. The first reason lies in that our framework AUTO can learn a comprehensive state representation that exploits useful features from multi-modal inputs. Secondly, based on the learned representation, the parameterized action structure and hybrid reward function in AUTO enable the autonomous vehicle to learn tactical acceleration and lane-changing behaviors to achieve high traffic efficiency while minimizing the impact on other vehicles. To study the effectiveness of the impact reward term in reducing the impact caused by the autonomous vehicle, we compare our framework AUTO against a variant AUTO-NoIMP that removes the impact reward term. As shown in Table I, compared to AUTO, AUTO-NoIMP has a slightly better AvgDT-A but has a noticeably worse AvgAT-C and AvgDI-C. This is because the autonomous vehicle under AUTO-NoIMP performs harsh actions in pursuit of high traffic efficiency but is prone to trigger a large impact and delay to the rear vehicle. Therefore, the impact reward term is critical in lowering the impact caused by our autonomous vehicle.

TABLE II: Microscopic Evaluation

Methods	Min TTC-A (s)	Avg OFF-A (m)	Avg VEL-A (m/s)	Avg JERK-A (m/s^2)	Avg DEC-C (m/s)
ACC-LC	2.94	0.39	17.99	1.32	0.55
TransFuser	2.55	0.51	18.35	1.17	0.52
DRL-MF	2.78	0.57	18.81	1.40	0.59
HEAD	2.85	0.40	18.48	1.05	0.48
AUTO-NoIMP	2.23	0.42	**19.25**	1.07	0.50
AUTO	**3.15**	**0.37**	19.12	**0.97**	**0.44**

C. Microscopic Evaluation

In this section, we conduct a microscopic evaluation of our framework AUTO, five baselines, and a variant AUTO-NoIMP for each action of the autonomous vehicle.

Evaluation Metrics We utilize five metrics to evaluate the microscopic effectiveness based on our hybrid reward function (Section III-E).

(**1**) Minimal time-to-collision of the autonomous vehicle (MinTTC-A). We record the time-to-collision (TTC) of the

autonomous vehicle. A larger MinTTC-A indicates that the autonomous vehicle is safer.

(**2**) Average lane offset of the autonomous vehicle (AvgOFF-A). We record the offset related to the lane centerline when the autonomous vehicle follows a curved lane. A smaller AvgOFF-A indicates that the vehicle performs better in lane following.

(**3**) Average velocity of the autonomous vehicle (AvgVEL-A). We record the velocity of the autonomous vehicle. A larger AvgVEL-A indicates higher traffic efficiency.

(**4**) Average jerk of the autonomous vehicle (AvgJERK-A). We record the acceleration change of the autonomous vehicle. A smaller AvgJERK-A indicates a more comfortable driving experience.

(**5**) Average deceleration of the rear conventional vehicle of the autonomous vehicle (AvgDEC-C). After the autonomous vehicle performs braking or lane-changing behaviors, we record the deceleration of the vehicle behind it. A smaller AvgDEC-C means the autonomous vehicle has less impact on the vehicle behind it.

We report MinTTC-A, AvgOFF-A, AvgVEL-A, AvgJERK-A, and AvgDEC-C in Table II. Compared to four baselines, we can see that our framework AUTO has the longest MinTTC-A, the highest AvgVEL-A, and the smallest AvgOFF-A, AvgJERK-A, and AvgDEC-C, demonstrating that AUTO enables the autonomous vehicle to generate acceleration brake rates and steering angles with safety, traffic efficiency, passenger comfort, and minimal impact on the rear vehicle. In these baselines, both DRL-MF and HEAD outperform the rule-based method ACC-LC and the imitation learning-based methods TransFuser due to the superior optimization effect of reinforcement learning. For DRL-MF, it lacks consideration of the road semantic information and pose information (e.g., position and orientation) of the autonomous vehicle and thus cannot accurately estimate the relative pose information between the autonomous vehicle and the specific route, leading to a poorer performance of DRL-MF compared to AUTO in all metrics, especially the lane offset. In addition, the experimental results of HEAD and AUTO prove that our parameterized action structure and reinforcement learning model RBP-DQN facilitate the decision-making of the autonomous vehicle. Further, the experimental results of AUTO-NoIMP prove that removing the impact reward item will lower all metrics except traffic efficiency from a microscopic point of view.

TABLE III: Effect of State Representation

Metric	AUTO-NoVEC	AUTO-NoSTAR	AUTO-NoLANE	AUTO
AvgR	0.31	0.46	0.45	**0.48**
AvgIT (ms)	51.83	17.08	**8.62**	8.96

D. Evaluation of State Representation

The lane-wised cross attention model (LCA) in our state representation module is proposed to exploit valuable features from multi-modal inputs. We calculate the average reward value (AvgR) of each action in the evaluation phase to compare

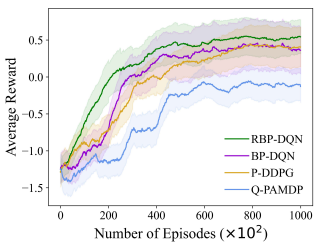


Fig. 6: Effect of Reinforcement Learning

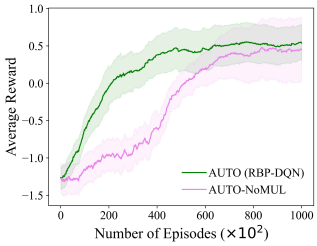


Fig. 7: Effect of Multi-worker Setting

the effectiveness of AUTO against three variants AUTO-NoVEC, AUTO-NoSTAR, and AUTO-NoLANE in Table III. As depicted, AUTO-NoVEC has the smallest AvgR among these methods, which proves that vector-based methods are more effective than raster-based methods in representing the multi-modal environment. Then, We can see that AUTO has a higher AvgR compared to both AUTO-NoSTAR and AUTO-NoLANE, which demonstrates that both the agent-centric star graph structure and lane-wised fusion paradigm in AUTO are beneficial in learning useful semantic features from the multi-modal traffic and thus improving the decision-making performance of the autonomous vehicle.

Inference time. In our decision-making framework, the average inference time (AvgIT) to calculate each action depends mainly on the time required to compute a representation of observed state features. We present the AvgIT in table III. As shown, AUTO has a higher decision-making speed than AUTO-NoVEC and AUTO-NoSTAR and a close speed to AUTO-NoLANE. Further, compared to the time interval ($\Delta t = 0.1s = 100ms$) between two decisions, the inference time of $8.96ms$ can meet the requirement of real-time decision-making.

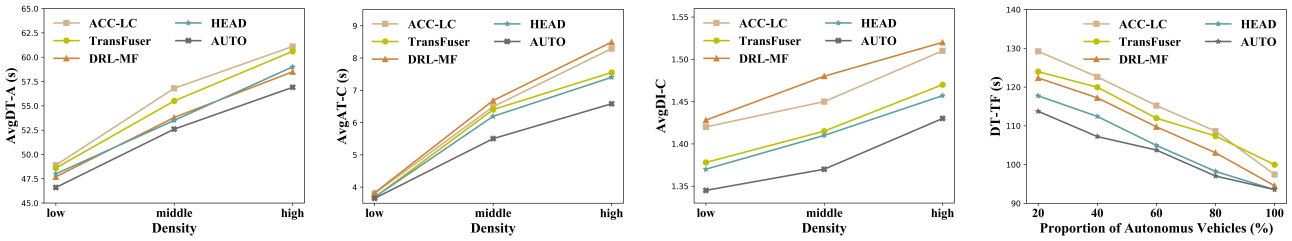
nk

E. Evaluation of Reinforcement Learning

To evaluate the proposed reinforcement learning model RBP-DQN in our framework, we compare it with Q-PAMDP, P-DDPG, and BP-DQN. As shown in Figure 6, we provide the learning curves of these methods. We can see that RBP-DQN has the highest convergence reward and converges faster than other methods. The reason for its superior optimization performance is that it can effectively learn the parameterized action structure and can avoid the overestimation problem of Q values via a regularization technique. For the regularization term, we find that our framework AUTO reaches the peak performance when α equals 0.6. Unless otherwise specified, other experiments are done when α equals 0.6.

F. Evaluation of Multi-worker Training

The multi-worker setting in our framework AUTO is utilized to speed up the training process and increase the diversity of learnable experiences. As shown in Figure 7, to evaluate the effectiveness, we compare the learning curves of AUTO against the variant AUTO-NoMUL that removes the multi-worker setting. We can see both the convergence speed and



(a) Driving Time. (b) Affected Time. (c) Delay Index. (d) Driving Time of Traffic Flow.

Fig. 8: Scalability Analysis

TABLE IV: Robustness under Different Weather Conditions

PT-RT (%)	Weather Conditions			
Methods	Normal	Rain	Fog	Night
AUTO-NoHIS	2.2	9.8	12.1	2.9
AUTO	1.8	6.3	10.3	2.7

convergence reward of AUTO are better than AUTO-NoMUL. This is because the joint exploration by multiple agents can collect diverse experiences and explore different regions of the solution space, thus improving the learning efficiency of AUTO and avoiding getting stuck in local optima.

G. Robustness Analysis

To evaluate the robustness of our framework AUTO to observation noises, we provide the safety performance of the autonomous vehicle in four weather conditions, i.e., normal, rain, fog, and night. Under these conditions, the observed features undergo a 3% to 18% perturbation compared to the ground truth. For example, the traffic light detection in our experiment has an average error rate of 13% in rainy conditions. To evaluate the robustness, we record the proportion of time (PT-RT) that the autonomous vehicle is under a risky time-to-collision (TTC). Based on the previous study [56], a vehicle can be considered to be at risk of collision when its TTC value is less than $4s$. As shown in Table IV, we report the PT-RT of our framework AUTO and a variant AUTO-NoHIS that only takes the current environmental features as states but disregards the historical features. As shown, AUTO always has a smaller PT-RT than AUTO-NoHIS under all challenging weather conditions, which proves that incorporating the historical environmental features into states is beneficial to the robustness of our framework.

H. Scalability Analysis

In this work, we train our framework AUTO on a training benchmark and evaluate it on an evaluation benchmark. From the macroscopic evaluation in Section IV-B and the microscopic evaluation in Section IV-C, we can see that AUTO can achieve a good performance in the evaluation scenarios, demonstrating a great potential of AUTO in terms of scaling to unseen scenarios. Furthermore, we conduct two experiments to evaluate our framework when the density of conventional vehicles changes or the proportion of autonomous vehicles increases.

Increase Conventional Vehicles. In Figure 8(a), (b), and (c), we use three macroscopic metrics (i.e., average driving time AvgDT-A, average affected time AvgAT-C, and average delay index AvgDI-C that are defined in Section IV-B) to evaluate the driving performance of the autonomous vehicle under three different densities (low: 60 vehicles/km, middle: 120 vehicles/km, and high: 180 vehicles/km) of conventional vehicles. Evidently, as the density increases, the autonomous vehicle has a decreasing traffic efficiency and is more prone to trigger large negative impacts on the traffic flow. As shown, based on the comparison of our framework AUTO with the baselines, we can see that AUTO always performs better, which proves its superiority in adaptation to different traffic densities.

Increase Autonomous Vehicles. In addition to the driving performance of an autonomous vehicle, we further explore the traffic efficiency of an entire traffic flow when the proportion of autonomous vehicles increases. Specifically, as the proportion scales from 20% to 100%, we record the driving time (DT-TF) of the entire traffic flow of $1km$. As shown in Figure 8(d), our framework AUTO always has a smaller DT-TF compared to the baselines. Moreover, as the proportion increases, the DT-TF of AUTO progressively reduces, demonstrating that the increasing proportion of autonomous vehicles is beneficial to traffic flow efficiency.

TABLE V: Effect of Coefficients in Hybrid Reward Function

Coefficient	Min	Max	Step	Best
w_1	0.5	1	0.1	**0.9**
w_2	0	1	0.2	**0.8**
w_3	0	1	0.2	**0.6**
w_4	0	0.5	0.1	**0.2**

I. Reward Shaping

For the hybrid reward function in Section III-E, w_1, w_2, w_3, and w_4 correspond to the coefficients of r_1 (safety), r_2 (traffic efficiency), r_3 (passenger comfort), and r_4 (impact), respectively. To achieve better performance, we adopt the grid search strategy [67] on a small number of training scenarios to determine them, which is shown in Table IV-I. Unless otherwise specified, we set the coefficients as $w_1 = 0.9$, $w_2 = 0.8$, $w_3 = 0.6$, $w_4 = 0.2$ in all experiments.

J. Hyper-Parameter Sensitivity

In this section, we study the effect of different detection distances of LiDAR and Camera since they can impact the decision-making performance of the autonomous vehicle. Intuitively, Further detection distances can benefit autonomous vehicle decision-making. However, in real autonomous vehicle systems, further detection results are always accompanied by greater detection noise. Moreover, distant detection results usually have little impact on decision-making. As shown in Table VI and VII, we use the average reward of all actions to represent model performance. The best detection distances of LiDAR and Camera in our experiments are 90 meters and 60 meters respectively.

TABLE VI: Impact of Detection Range of LiDAR

Detection Range	50m	60m	70m	80m	90m	100m	110m
Average Reward	0.454	0.472	0.481	0.482	0.483	0.482	0.482

TABLE VII: Impact of Detection Range of Camera

Detection Range	50m	60m	70m	80m	90m	100m	110m
Average Reward	0.481	0.483	0.481	0.481	0.480	0.480	0.479

V. Related Work

Autonomous driving is considered to be one of those technologies that could herald a major shift in transportation [1]. In the following, we will introduce some related works on feature extraction and decision-making in autonomous driving.

Feature Extraction. In this paper, we have built a fully-autonomous driving perception stack that can obtain the real-time multi-modal features of lanes, vehicles, and traffic lights from an HD map and onboard sensors. We conduct the experiments in a widely-used simulator Carla that mimics real-world traffic scenarios. It provides high-fidelity sensor models that can replicate the characteristics (e.g., field of view and resolution) of real sensors and generate synthetic data based on the virtual environment and the positions of objects. Specifically, we use an open-source standard OpenDrive [33] as the format of HD maps and estimate the pose of autonomous vehicles by fusing the geographic position provided by GNSS and the orientation provided by IMU [32]. Then, we extract the features of surrounding vehicles from the point cloud data by LiDAR and extract the features of traffic lights from the image data by Camera.

For real-time analysis, with the development of feature extraction technologies, researchers have developed lightweight models that maintain high accuracy but require less computational power. For example, the methods in [34], [35] can obtain the position and velocity information of surrounding vehicles at greater than 30 fps. In order to improve the robustness, our framework incorporates the detection results at multiple time steps to alleviate the impacts of detection missing and errors.

Decision-making. Automatic driving decision-making methods can be categorized into end-to-end [12], [68]–[70], and modular approaches [6]–[8], [71]–[73] based on their underlying architecture and design principles. End-to-end methods directly map raw sensor data to control commands, typically using deep neural networks. For example, TransFuser [12] proposes a transformer-based network structure to imitate labeled human driving behaviors. To improve safety, ThinkTwice [68] injects spatial-temporal prior knowledge and dense supervision into the imitation learning process. In contrast, modular methods firstly extract vehicle and traffic light features from raw sensor data and then calculate control commands. Compared with training directly based on raw sensor data, modular methods have better interpretability and decision-making performance since they start training from refined detection results rather than redundant sensor data. In this paper, we focus on the modular methods. However, existing modular methods either ignore the complexity of environments only fitting straight roads, or ignore the impact on surrounding vehicles during the optimization phase. To address these limitations, we aim to design a decision-making framework that can not only achieve safety, traffic efficiency, passenger comfort, and minimal impact on traffic flow, but also adapt to complex traffic scenarios with multi-modal features.

Although the above learning-based methods can perform well in fusing multi-modal features and multi-objective optimization, there is always a safety module to guarantee safety in industrial autonomous driving systems [74]–[76]. Specifically, the safety module is used to check whether the action output by the learning-based decision-making module is safe. If it is unsafe, the autonomous vehicle enters safe mode and uses a rule-based method to calculate a driving action. This usually occurs in emergency situations, such as a person suddenly appearing in front of the vehicle.

VI. Conclusion

In this paper, we propose AUTO to enable the autonomous vehicle to complete a specific route with safety, traffic efficiency, passenger comfort, and minimal impact on the rear vehicle. We first conduct a comprehensive perception of the traffic environment to obtain multi-modal inputs for the autonomous vehicle. Then, we design a graph-based model to exploit useful semantic features from the input and use a parameterized action paradigm to calculate fine-grained actions based on coarse-grained decisions. Finally, we propose a hybrid reward function to guide the optimization of reinforcement learning and use a regularization term and a multi-worker setting to enhance the training. Extensive experiments confirm the superiority of AUTO over state-of-the-art approaches in terms of both macroscopic and microscopic effectiveness.

Acknowledgment

This work is partially supported by NSFC (No. 62272086, 61972069, 61836007 and 61832017), Shenzhen Municipal Science and Technology R&D Funding Basic Research Program (JCYJ20210324133607021), and Municipal Government of Quzhou under Grant (No. 2022D037, 2023D004, 2023D044), and Key Laboratory of Data Intelligence and Cognitive Computing, Longhua District, Shenzhen.

REFERENCES

[1] D. Schrank, L. Albert, B. Eisele, and T. Lomax, "2021 urban mobility report," Texas A&M Transportation Institute, Tech. Rep., 2021.

[2] M. Won, T. Park, and S. H. Son, "Toward mitigating phantom jam using vehicle-to-vehicle communication," *IEEE transactions on intelligent transportation systems*, pp. 1313–1324, 2016.

[3] X. Qu, Y. Yu, M. Zhou, C.-T. Lin, and X. Wang, "Jointly dampening traffic oscillations and improving energy consumption with electric, connected and automated vehicles: a reinforcement learning based approach," *Applied Energy*, p. 114030, 2020.

[4] S. E. Merriman, K. L. Plant, K. M. Revell, and N. A. Stanton, "Challenges for automated vehicle driver training: a thematic analysis from manual and automated driving," *Transportation research part F: traffic psychology and behaviour*, pp. 238–268, 2021.

[5] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[6] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," *arXiv preprint arXiv:1911.12250*, 2019.

[7] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[8] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *IEEE SMC*, 2019, pp. 2326–2331.

[9] L. Xiao, M. Wang, and B. Van Arem, "Realistic car-following models for microscopic simulation of adaptive and cooperative adaptive cruise control vehicles," *Transportation Research Record*, pp. 1–9, 2017.

[10] J. Erdmann, "Sumo's lane-changing model," in *Modeling Mobility with Open Data*, 2015, pp. 105–123.

[11] V. Milanés and S. E. Shladover, "Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data," *Transportation Research Part C: Emerging Technologies*, pp. 285–300, 2014.

[12] A. Prakash, K. Chitta, and A. Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7077–7087.

[13] J. Chen, B. Yuan, and M. Tomizuka, "Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 2884–2890.

[14] D. A. Tedjopurnomo, Z. Bao, B. Zheng, F. M. Choudhury, and A. K. Qin, "A survey on modern deep neural network for traffic prediction: Trends, methods and challenges," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, pp. 1544–1561, 2020.

[15] W. Fu, Y. Li, Z. Ye, and Q. Liu, "Decision making for autonomous driving via multimodal transformer and deep reinforcement learning," in *2022 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2022, pp. 481–486.

[16] S. Liu, Y. Xia, C. Xu, J. Xie, H. Su, and K. Zheng, "Impact-aware maneuver decision with enhanced perception for autonomous vehicle," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, 2023.

[17] Ó. Pérez-Gil, R. Barea, E. López-Guillén, L. M. Bergasa, C. Gómez-Huélamo, R. Gutiérrez, and A. Díaz-Díaz, "Deep reinforcement learning based control for autonomous vehicles in carla," *Multimedia Tools and Applications*, pp. 3553–3576, 2022.

[18] 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*, p. 102662, 2020.

[19] Z. Xu, S. Liu, Z. Wu, X. Chen, K. Zeng, K. Zheng, and H. Su, "Patrol: A velocity control framework for autonomous vehicle via spatial-temporal reinforcement learning," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, 2021, pp. 2271–2280.

[20] A. Talebpour and H. S. Mahmassani, "Influence of connected and autonomous vehicles on traffic flow stability and throughput," *Transportation Research Part C: Emerging Technologies*, pp. 143–163, 2016.

[21] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, pp. 205–221, 2018.

[22] C. Gómez-Huélamo, A. Diaz-Diaz, J. Araluce, M. E. Ortiz, R. Gutiérrez, F. Arango, Á. Llamazares, and L. M. Bergasa, "How to build and validate a safe and reliable autonomous driving stack? a ros based software modular architecture baseline," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1282–1289.

[23] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1341–1360, 2020.

[24] C. Gómez-Huélamo, J. D. Egido, L. M. Bergasa, R. Barea, E. López-Guillén, F. Arango, J. Araluce, and J. López, "Train here, drive there: Simulating real-world use cases with fully-autonomous driving architecture in carla simulator," in *Workshop of Physical Agents*. Springer, 2020, pp. 44–59.

[25] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[26] R. Gutiérrez, E. López-Guillén, L. M. Bergasa, R. Barea, Ó. Pérez, C. Gómez-Huélamo, F. Arango, J. Del Egido, and J. López-Fernández, "A waypoint tracking controller for autonomous road vehicles using ros framework," *Sensors*, p. 4062, 2020.

[27] Y. Liu, Y. Gao, Q. Zhang, D. Ding, and D. Zhao, "Multi-task safe reinforcement learning for navigating intersections in dense traffic," *Journal of the Franklin Institute*, 2022.

[28] Y. Chen, C. Dong, P. Palanisamy, P. Mudalige, K. Muelling, and J. M. Dolan, "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving," in *CVPR Workshops*, 2019, pp. 0–0.

[29] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, pp. 674–691, 2021.

[30] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[31] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," *arXiv preprint arXiv:1803.00933*, 2018.

[32] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, p. 46, 2004.

[33] A. Diaz-Diaz, M. Ocaña, Á. Llamazares, C. Gómez-Huélamo, P. Revenga, and L. M. Bergasa, "Hd maps: Exploiting opendrive potential for path planning and map monitoring," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, 2022, pp. 1211–1217.

[34] Z. Lari, A. Habib, and E. Kwak, "An adaptive approach for segmentation of 3d laser point cloud," *International archives of the photogrammetry, remote sensing and spatial information sciences*, vol. 38, no. 5, p. W12, 2011.

[35] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 7464–7475.

[36] S. Liu, H. Su, Y. Zhao, K. Zeng, and K. Zheng, "Lane change scheduling for autonomous vehicle: A prediction-and-search framework," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3343–3353.

[37] M. Fu, T. Zhang, W. Song, Y. Yang, and M. Wang, "Trajectory prediction-based local spatio-temporal navigation map for autonomous driving in dynamic highway environments," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[38] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2090–2096.

[39] W. Zeng, C. Lin, K. Liu, J. Lin, and A. K. Tung, "Modeling spatial nonstationarity via deformable convolutions for deep traffic flow prediction," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2021.

[40] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized rep-

resentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 525–11 533.
[41] M. Chen, Y. Zhao, Y. Liu, X. Yu, and K. Zheng, "Modeling spatial trajectories with attribute representation learning," *TKDE*, 2020.
[42] W. Lian, Y. Wang, X. Zeng, S. Liu, Y. Xia, H. Tang, and H. Su, "Egl: Efficient graph learning with safety constrains for heterogeneous trajectory prediction," in *International Conference on Database Systems for Advanced Applications*. Springer, 2023, pp. 67–78.
[43] M. Liang, B. Yang, R. Hu, Y. Chen, R. Liao, S. Feng, and R. Urtasun, "Learning lane graph representations for motion forecasting," in *European Conference on Computer Vision*, 2020, pp. 541–556.
[44] K. Zheng, Y. Zhao, D. Lian, B. Zheng, G. Liu, and X. Zhou, "Reference-based framework for spatio-temporal trajectory compression and query processing," *TKDE*, vol. 32, no. 11, pp. 12 227–2240, 2020.
[45] Y. Zhao, S. Shang, Y. Wang, B. Zheng, Q. V. H. Nguyen, and K. Zheng, "Rest: A reference-based framework for spatio-temporal trajectory compression," in *SIGKDD*, 2018, pp. 2797–2806.
[46] S. Morad, R. Kortvelesy, M. Bettini, S. Liwicki, and A. Prorok, "Popgym: Benchmarking partially observable reinforcement learning," in *The Eleventh International Conference on Learning Representations*.
[47] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.
[48] X. Chen, H. Zhang, F. Zhao, Y. Cai, H. Wang, and Q. Ye, "Vehicle trajectory prediction based on intention-aware non-autoregressive transformer with multi-attention learning for internet of vehicles," *IEEE Transactions on Instrumentation and Measurement*, pp. 1–12, 2022.
[49] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, pp. 529–533, 2015.
[50] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *IEEE ICRA*, 2021, pp. 1140–1146.
[51] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, pp. 569–597, 2022.
[52] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, pp. 279–292, 1992.
[53] T. Shang, G. Lian, Y. Zhao, X. Liu, and W. Wang, "Off-ramp vehicle mandatory lane-changing duration in small spacing section of tunnel-interchange section based on survival analysis," *Journal of Advanced Transportation*, 2022.
[54] Z. Huang, Z. Zhang, H. Li, L. Qin, and J. Rong, "Determining appropriate lane-changing spacing for off-ramp areas of urban expressways," *Sustainability*, p. 2087, 2019.
[55] L. Evans, *Traffic safety and the driver*. Science Serving Society, 1991.
[56] Y. Xia, S. Liu, X. Chen, Z. Xu, K. Zheng, and H. Su, "Rise: A velocity control framework with minimal impacts based on reinforcement learning," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*, 2022, pp. 2210–2219.
[57] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *Advances in neural information processing systems*, vol. 34, pp. 20 132–20 145, 2021.
[58] D. Brandfonbrener, W. Whitney, R. Ranganath, and J. Bruna, "Offline rl without off-policy evaluation," *Advances in neural information processing systems*, vol. 34, pp. 4933–4946, 2021.
[59] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *arXiv preprint arXiv:2110.06169*, 2021.
[60] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[61] D. Bertsekas, *Rollout, policy iteration, and distributed reinforcement learning*, 2021.
[62] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.
[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[64] N. Djuric, V. Radosavljevic, H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin, and J. Schneider, "Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks," *arXiv preprint arXiv:1808.05819*, p. 6, 2018.
[65] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
[66] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv preprint arXiv:1511.04143*, 2015.
[67] I. Syarif, A. Prugel-Bennett, and G. Wills, "Svm parameter optimization using grid search and genetic algorithm to improve classification performance," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 14, no. 4, pp. 1502–1509, 2016.
[68] X. Jia, P. Wu, L. Chen, J. Xie, C. He, J. Yan, and H. Li, "Think twice before driving: Towards scalable decoders for end-to-end autonomous driving," in *CVPR*, 2023.
[69] D. Coelho and M. Oliveira, "A review of end-to-end autonomous driving in urban environments," *IEEE Access*, vol. 10, pp. 75 296–75 311, 2022.
[70] P. Wu, X. Jia, L. Chen, J. Yan, H. Li, and Y. Qiao, "Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline," *Advances in Neural Information Processing Systems*, vol. 35, pp. 6119–6132, 2022.
[71] X. Zeng, Q. Yu, S. Liu, Y. Xia, H. Su, and K. Zheng, "Target-oriented maneuver decision for autonomous vehicle: A rule-aided reinforcement learning framework," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 3124–3133.
[72] Y. Fu, S. Liu, Y. Xia, F. Guo, and K. Zheng, "Cross-scenario maneuver decision with adaptive perception for autonomous driving," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 535–544.
[73] Y. Xia, S. Liu, R. Hu, Q. Yu, X. Feng, K. Zheng, and H. Su, "Smart: A decision-making framework with multi-modality fusion for autonomous driving based on reinforcement learning," in *Database Systems for Advanced Applications: 28th International Conference, DASFAA 2023, Tianjin, China, April 17–20, 2023, Proceedings, Part IV*. Springer, 2023, pp. 447–462.
[74] J. Wang, Q. Zhang, D. Zhao, and Y. Chen, "Lane change decision-making through deep reinforcement learning with rule-based constraints," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.
[75] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *2020 IEEE 23rd international conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.
[76] K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2020.