A Review on Reinforcement Learning-based Highway Autonomous Vehicle Control

***Abstract*—Autonomous driving is an active area of research in artificial intelligence and robotics. Recent advances in deep reinforcement learning (DRL) show promise for training autonomous vehicles to handle complex real-world driving tasks. This paper reviews recent advancement on the application of DRL to highway lane change, ramp merge, and platoon coordination. In particular, similarities, differences, limitations, and best practices regarding the DRL formulations, DRL training algorithms, simulations, and metrics are reviewed and discussed. The paper starts by reviewing different traffic scenarios that are discussed by the literature, followed by a thorough review on the DRL technology such as the state representation methods that capture interactive dynamics critical for safe and efficient merging and the reward formulations that manage key metrics like safety, efficiency, comfort, and adaptability. Insights from this review can guide future research toward realizing the potential of DRL for automated driving in complex traffic under uncertainty.**

***Index Terms*—Autonomous vehicles, connected and automated vehicles, deep reinforcement learning, platoon.**

I. Introduction

Autonomous vehicles (AVs) have the potential to transform transportation systems by improving safety, efficiency, accessibility, and comfort. According to [1], autonomous driving (AD) has the potential to revolutionize the way consumers experience mobility and make driving safer, more convenient, and more enjoyable. According to the Environmental and Energy Study Institute, when incorporated into shared mass transit systems, AVs are more accessible and sustainable, which could improve accessibility for people who may not have access to traditional transportation options [2]. However, developing reliable control policies for AVs to handle the complexity of real-world driving remains an immense challenge [3]. The AVs technology can save lives, reduce crashes, congestion, fuel consumption, pollution, and parking space [4]–[12]; but it can also increase traffic, disrupt transit and insurance sectors, and pose ethical and legal issues [13]–[15].

One challenge area in AD is the traffic merging control, especially in mixed-traffic environments. AVs must interact with human drivers in a safe and efficient manner. The merging scenario involves an ego AV attempting to merge from a ramp or lane onto a highway lane already occupied by host vehicles as shown in Fig. 1. Host vehicles could be AVs, human-driven vehicles (HDVs), or both. For this maneuver, the ego vehicle needs to negotiate right-of-way with the host vehicles while avoiding collisions and minimizing disruption to traffic flow. Merging control presents several key difficulties for AVs. First, human drivers have various driving styles, which can range from being aggressive to being cautious [16] and their behaviors are inherently ambiguous and uncertain [17]. According to [18], even the same driver may react differently to the same

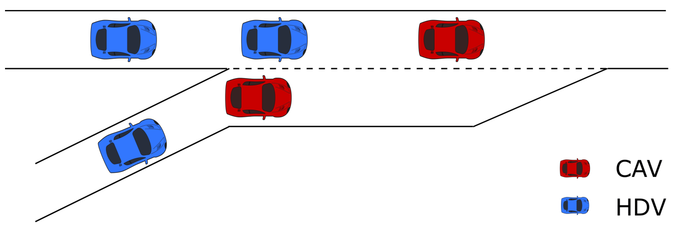

Fig. 1. On-ramp traffic scenario in the presence of HDVs.

circumstances. An AV needs to be able to quickly infer the cooperativeness of host drivers based on limited observations and react accordingly. Second, merging requires navigating a temporary conflict zone where collisions can easily occur if improperly timed, making an accurate real-time control necessary [19]. The AV must accurately judge acceptable gaps in host vehicle traffic and select appropriate merge points. Third, efficiency and comfort concerns during merging must be balanced with safety. Conservatively waiting for large gaps reduces disruption but can significantly slow down traffic flow. On the other hand, aggressively forcing merges into small gaps risks collisions or requires uncomfortable acceleration/braking.

AVs require control strategies to operate effectively. These strategies can be categorized as rule-based, optimization-based, or learning-based methodologies. Rule-based approaches rely on predefined models and hard-coded rules to determine AV behaviors, such as car following and lane changing rules [20], [21]. In [22], the authors suggested various rules and used a gap acceptance algorithm to determine vehicle behavior before it merges into the target lane. Similarly, In [23], authors tested rule-based baseline that predict the behavior of surrounding vehicle which would demonstrate social behavior in AVs. According to this approach, vehicles are less inclined to yield when accelerating, whereas they are more likely to yield when decelerating. Rule-based approaches lack adaptability to unexpected situations and becomes impractical in a complex scenarios.

In an optimization-based approach, vehicle interactions are modeled as a dynamic system with vehicles' actions as inputs such as acceleration or steering angle [5]. In reference [24], the authors studied cooperative merging algorithms for autonomous vehicles interacting with human-driven vehicles. They used a model predictive control (MPC) approach which breaks down the problem into handling groups of three vehicles (triplets) at a time. The MPC controller generates smooth motion trajectories for different compositions of vehicle triplets. In reference [25], the authors looked specifically at lane change scenarios where a human-driven vehicle cuts into a platoon of connected and autonomous vehicles. Although the MPC-based methods show promising results, they heavily depend on precise dynamic merging models (even for human driving behavior). Moreover, these methods often require significant computer resources due to solving online optimizations at every time step. On the other hand, learning-based methods utilizing deep neural networks can extract complex features and patterns to better imitate human driving. In AVs control, the learning-based approaches can be classified into supervised learning (SL) and Reinforcement learning (RL) [26]. SL techniques train models on labeled real-world or simulated driving data to reproduce expert driving performance. However, SL relies heavily on the quality and coverage of this training data [27]. Models may be biased if not exposed to rare but critical events like near-collisions. On the other hand, RL does not need human labeled-data but can generate the data directly through interacting with the environment. This paper focuses specifically on reinforcement learning (RL) approaches for autonomous vehicle merging control.

RL and especially deep reinforcement learning (DRL) have achieved remarkable success in recent years [28]–[31]. Neural networks serve as powerful function approximators that can learn to map complex, high-dimensional observations to optimal actions. For example, Deep Q-networks (DQN) developed in [28] used CNNs and experience replay to master Atari games from pixel inputs. Policy gradient methods like A3C apply actor-critic architectures with parallel workers to learn policies mapping images to controls [29]. AlphaGo combined deep neural networks with Monte Carlo tree searching to defeat world champions in the game of Go [30]. Deep RL with recurrent policies and attention has shown promise for partially observed tasks like navigating from first-person views [32]. End-to-end training of deep networks has enabled learning of visuomotor policies directly from raw sensory inputs. The ability of deep RL to handle high-dimensional state spaces provides wider applicability to real-world problems. However, challenges like sample efficiency, hyperparameter tuning, and training stability remain active areas of research. Careful architecture design and algorithms tailored to the problem dynamics are crucial for successful deep RL applications.

In recent years, DRL has shown increasing promise for AVs, being applied to behaviors like lane changing, merging, car following, and intersection handling [33]–[36]. Several key considerations arise when applying RL to AV control. Firstly, an appropriate reward function that balances safety, efficiency, and comfort must be carefully designed [37]. Common objectives include avoiding collisions, minimizing disruptions to traffic flow, and preferring smooth accelerations/decelerations. Secondly, the action space must also be chosen carefully as well, where options include continuous acceleration control [38], discrete maneuver choices [39], and high-level decisions [40]. Another challenge lies in the handling of the partial observability of the environment. RL agents are typically only provided local observations from onboard sensors. In this regard, recurrent policies [41] or attention mechanisms [42] have been investigated to help infer unobserved information like driver intent from observation histories. Moreover, multi-agent RL can also enable collaborative sensing between networked vehicles [43].

However, identifying effective RL formulations and algorithms tailored to the dynamics and constraints of on-road driving remains an active research problem. Careful design of the action space, state representation, reward, and training methodology is necessary to ensure that training can lead to optimal policies that meet safety and comfort requirements. Many surveys exist on the topic of Reinforcement learning in the context of autonomous driving, such as the paper by Kiran et al. [44]. They focus specifically on deep reinforcement learning for autonomous driving and provides background on RL, review applications to tasks like motion planning and control, and outlines challenges. Similarly, in [45], authors present an extensive survey on the application of deep learning (DL) and reinforcement learning (RL) across various functionalities in autonomous vehicles (AVs). This comprehensive survey focuses on key areas such as scene understanding, motion planning, decision-making, vehicle control, and social behavior of AVs. Yadav et al. [46] have also narrowed down the scope in a similar way. They provide a comprehensive survey on multi-agent reinforcement learning (MARL) for connected and automated vehicles (CAVs). They discuss various MARL techniques, algorithms for motion planning under uncertainty, learning paradigms, simulators, datasets, applications, and future research directions. However, there is a lack of extensive reviews on RL based autonomous vehicles that is focused on a specific scenario. Johnson et al. [47] examine the specific context of ramp merging strategies for connected and autonomous vehicles on freeways. It examines the impact of CAVs on traffic flow efficiency and safety at critical freeway junctions. The paper categorizes existing strategies based on application contexts, such as single-lane freeways with varying CAV penetration and multi-lane freeway scenarios.

In contrast, this paper aims to provide a comprehensive survey of deep RL (DRL) techniques for longitudinal and lateral vehicle control behaviors. In particular, we analyze various RL formulations, algorithms, simulations, and metrics used in a diverse set of studies related specifically to highway lane change, highway ramp merging, and platoon coordination. The primary objective is to describe recent advancements in this challenging area, highlight key trends, gaps, and best practices, and identify research directions. Our work adds to and expands upon the broader discussions found in the works of Elallid et al. [45] and Johnson et al. [47], making a unique contribution to the field of autonomous vehicle research. The insights from this literature review can help guide future research toward realizing the full potential of RL for AV control in complex real-world conditions. The papers reviewed in this paper are summarized in Tables I and II.

The remainder of this paper is organized as follows. Section II provides necessary preliminary on reinforcement learning and discusses key concepts. The scenarios studied are discussed in Section III, while Section IV provides an overview of the RL algorithms used and discusses sophisticated extensions added to the RL algorithms. Section V reviews relevant literature according to how action space is setup and Section VI reviews the key themes adopted when designing

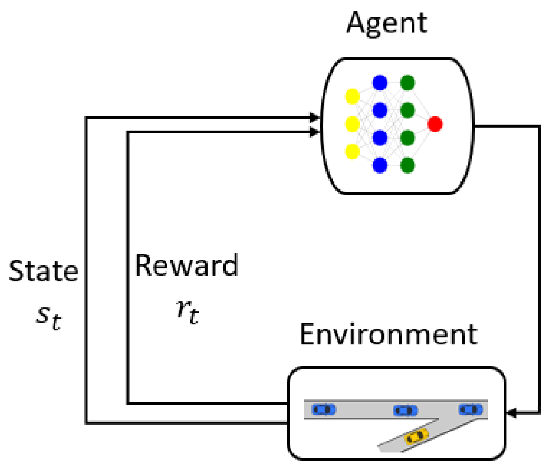

Fig. 2. General flow of reinforcement learning algorithm.

the state space. Section VII categorizes the literature according to the rewards function design, Section VIII discusses the challenges and provides future work directions, and the paper is concluded in Section IX.

II. Preliminary on Reinforcement Learning

Reinforcement learning (RL) is a feedback-based machine learning approach in which an agent learns how to operate in a given environment by choosing actions and observing their consequences [79]. The agent receives positive feedback for choosing a good action and negative feedback or a penalty for choosing a poor action. The agent interacts with its environment, and its primary objective is to enhance performance by maximizing positive reinforcements. As illustrated in Fig. 2, the key elements of RL are:

- Environment: The external world that the agent interacts with. It can be modeled as a Markov decision process (MDP) consisting of states s, actions a, transition probabilities $P(s'|s,a)$, and a reward function $R(s,a,s')$ [79]–[82].
- Agent: The learner that interacts with the environment by taking actions and observing results. Its goal is to find an optimal policy $\pi^*(a|s)$ that maximizes the expected cumulative reward. Simple agents use linear policies, while more complex agents use non-linear function approximators like neural networks.
- States: A state represents the current situation of the agent in its environment. It serves as input for the agent to determine the next action. States need to contain all relevant information while being represented efficiently.
- Actions: Actions are the moves the agent can make in response to the environment. Choosing the best action given the current state is the core challenge in RL.
- Reward: A reward is feedback from the environment evaluating the agent's previous action. The agent seeks to maximize the cumulative reward over time.

Particularly, at time step t, let $r(s_t,a_t)$ be a scalar reward function that indicates the immediate reward received by the agent from the environment after applying action a_t at state s_t and define the expected cumulative future rewards from time t as

$$G = E_\pi\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right], \quad (1)$$

where the scalar $\gamma \in [0,1]$ is the discount factor. Then the goal of RL training is to find an optimal policy $\pi^*(a|s)$ that maximizes G. Different RL algorithms employ different assumption about the value function approximation. The major categories of RL algorithms include: value-based methods like Q-learning [83], [84] and SARSA [85] that estimate value functions to derive a policy, policy gradient methods that directly learn and optimize parametric policies [86], actor-critic methods that learn both a policy and value function [87], and model-based methods that also learn a model of the MDP for planning [30], [88].

III. Scenarios for RL-based Control

This section discusses three major scenarios studied in the relevant literature, namely highway lane change, highway ramp merging, and platoon coordination, as shown in Table III.

A. *Highway Lane Change*

Lane changing on highways is an important and challenging task for AVs. Several studies have investigated using DRL to train AV agents to perform safe and efficient lane changes in highway environments. A common scenario studied is an agent controlling a single AV that needs to execute a lane change maneuver on a multi-lane highway segment in the presence of surrounding traffic [48], [58]. The highway environment is typically simulated using tools like SUMO (Simulation of Urban MObility) [48], VISSIM or custom cellular automaton models [34], with the RL agent controlling actions like lateral movement, acceleration, and deceleration. For instance, authors in [48] trained a policy using proximal policy optimization (PPO) to have an AV perform discretionary lane changes on a congested highway simulated in SUMO. Another example is [58], where the authors also studied an agent making a single lane change on a highway but used a deep Q-network (DQN) algorithm. Similarly, [78] used a double deep Q-network (DDQN) algorithm incorporated with a model of human lane-changing decisions as a safety supervisor to train AVs to drive safely in a 2-lane highway. The agent waited for a suitable gap before changing lanes and maintained lane keeping after the maneuver.

More complex highway scenarios with continuous lanes and dynamic surrounding traffic have also been explored as illustrated in Fig. 3. Wang et al. [34] used deep deterministic policy gradients (DDPG) to train an agent to make lane changes on a 3-lane highway with entering and exiting traffic modeled with an intelligent driver model (IDM). Wang et al. [36] also studied an agent controlling a vehicle that needed to change lanes to pass slower cars on a 3-lane highway simulated using an open-source self-driving car simulator provided by Udacity. Additionally, Zhou et al. [56] considered multiple AVs and

TABLE I
STATE OF THE ART ARTICLES ON AUTONOMOUS VEHICLE MANEUVERS CONTROL USING REINFORCEMENT LEARNING

References	Scenario	State Space	Action Space	Software	RL Method	Rewards
[48]	Lane changing	Raw sensors data (acc., pos., and speed)	Three discrete actions	SUMO	PPO	Comfort, efficiency, and safety
[49]	Navigate congested highway	Ego vehicle and surrounding raw sensors data	Discrete combinations of acceleration and steering angle	SUMO	Policy Gradient	Fast and safe driving
[39]	Platoons at non signalized intersection	AV and its surrounding motion values	Discrete set of the acceleration	SUMO	PPO	Average speed
[38]	Platoon in freeway	Leader, ego vehicle, and its preceding vehicle	Acceleration (continuous)	SUMO	CommPPO	Fuel consumption
[50]	Highway on-ramp	Relative speed and position of surrounding vehicles	Discrete high level decisions	Highway-env [51]	Novel MARL	Collision, merging cost, speed, and time headway
[52]	Highway on-ramp	Relative speed and position of surrounding vehicles and the Road layout	Discrete high level decisions	Highway-env [51]	MARL	Safety
[53]	Highway on-ramp	Traffic state and vehicle motion information	Three discrete actions	SUMO	STDQN	Efficiency, goal, driving comfort, and safety
[54]	Highway navigation	Self, connected agents and infrastructure observations (location, velocity and acceleration)	Discrete high level decisions	CARLA [55]	Constrained MARL	Maximize speed of every agent
[56]	Lane changing	Relative speed and position of surrounding vehicles	Discrete high level decisions	TORCS	Discrete high level decisions	Collision, comfort, speed, and time headway
[57]	Enter/Exit highway merge	The relative speed of surrounding vehicles, the position of the ego vehicle, and the road layout.	Discrete high level decisions	Highway-env	MADDQN	Safe altruistic behavior
[58]	Lane change	The relative distance to the surrounding vehicles and detected lane.	Continuous steering and longitudinal speed.	Python	DDPG	Security, comfort, and efficiency
[35]	Lane change	The relative distance and absolute speed of front vehicles.	To change lane or to stay	MATLAB & VISSIM [59]	DQN	Safety and efficiency
	Car-following	The relative distance and absolute speed of the front and back vehicles and the ego vehicle speed and maximum speed	Six different speeds to choose from	MATLAB & VISSIM	DQN	Safety, comfort, and efficiency
[60]	Highway on-ramp	Speed, position, heading angle, and lateral offset to the lines	Continuous acceleration and steering angle	Simulation of real world data	DQN	Safety, Smoothness, and timeliness
[34]	Lane changing	Ego and surrounding vehicles dynamics, and road curvature	Continuous yaw acceleration	Not mentioned	DDPG	Large action changes, Maneuvering time, and lateral deviation
[36]	Lane changing	Speed and position of the ego vehicle and its surrounding vehicles	Discrete high level decisions	Udacity simulator	DQN	Safety and Speed
[61]	Highway work zone	Speed and acceleration grid maps and neighboring vehicles information	Acceleration and deceleration	VISSIM	SAC	Safety, comfort, and traffic flow
[62]	Platoon maneuvers	Speed, acceleration and position of ego, front and rear vehicles	Brake, throttle and steering	CARLA	SAC	Safety, comfort, smoothness and headway
[63]	Platoon in a stop-and-go traffic	Headway, acceleration and speed of ego and front vehicles	Continuous acceleration	SUMO	SAC	Safety, efficiency, and oscillation dampening
[64]	Platoon forming	Speed, acceleration and headway	Full-speed headway	Not mentioned	DDPG	Velocity and headway
[65]	Platoon Longitudinal Control	Relative distance, ego vehicle speed, preceding vehicle speed, and the distance gap error	Continuous acceleration	SUMO	DDPG	Gap regulation, comfort speed consensus,and safety
[66]	Highway-on ramp	Speed and position of AVs and HVs, merging vehicle, and road layou	Continuous acceleration and steering angle	OpenAI Gym	Multi-agent A2C	driving safety, traffic flow efficiency, and socially desirable behaviors
[67]	Highway-on ramp	Speed and position of AVs and HVs, merging vehicle, and road layout	Discrete high level actions	OpenAI Gym	Multi-agent A2C	traffic throughput, safety, and individual driving comfort

TABLE II
STATE OF THE ART ARTICLES ON AUTONOMOUS VEHICLE MANEUVERS CONTROL USING REINFORCEMENT LEARNING - CONTINUED

References	Scenario	State Space	Action Space	Software	RL Method	Rewards
[68]	Highway merging	Egos speed & acceleration, surrounding vehicles relative speed & distance, and distance to both conflict zone and goal	Discrete high level actions	Not mentioned	DQN	Efficiency and comfort
[37]	Highway merging	Egos kinematics, surrounding occupancy and speed map, priority, and driver type	Discrete actions	Custom simulator	A-C MARL	Safety, flow, efficiency, and success
[40]	Highway merging	Egos local information, position, motion, & Gaps between surrounding vehicles	Discrete actions	NGSIM	A2C	Speed and successful merge
[69]	Highway merging	Egos kinematics and position and speed of surrounding vehicles	Continuous acceleration	SUMO	A2C	Collisions, braking, and successful merge
[70]	Lane changing	Egos kinematics and position ,speed, heading angle, and size of surrounding vehicles	Continuous acceleration and steering angle	Not mentioned	FSM and SAC	Risk penalty, approach reward, and comfort
[71]	Lane changing	Egos kinematics and position & speed of surrounding vehicles	Discrete actions	Julia [72]	DQN	Safety, efficiency, and lane changing
[73]	Highway merging	Ego vehicle, road geometry, and surrounding traffic	Discrete actions	Custom simulator	DQN	Average speed
[74]	Highway merging	Distance to the merge point, velocity, acceleration, and cooperation parameter	Discrete actions	Julia [72]	DQN	Collision, goal, and time penalty
[33]	Platoons facing Road reduction	Distance to the merge point, position, and state	Discrete actions	Python	MPPO	Energy, time, speed, and jerk
[75]	Highway on-ramp	Closing gap and speed, time to position, and position of the ego	Discrete actions	Python	Q-learning	Safety, comfort, and energy
[76]	Platoon coordination management	Platoon state, states of AVs in the platoon, and environment state.	Discrete actions	PLEXE [77]	DRG-SP	Driving strategy
[78]	Lane changing	Velocity and relative distance of surrounding vehicles, and ego kinematics.	Discrete actions	CARLA	DDQN	Safety, speed, and lane centering

TABLE III
STUDIES ORGANIZED BY AUTOMATED DRIVING SCENARIO

Scenario		References
Highway Lane Change	Simple surrounding	[48], [58], [78]
	Complex surrounding	[34], [36], [56]
	Lane merging	[61], [70], [71], [74]
Highway Ramp Merging	Multi-agent merging	[50], [52], [53], [57], [75]
	Cooperative merging	[37], [66]–[69]
	Using real-world traffic data	[40], [60], [89]
Platooning	Cooperative speed control	[39], [62]–[64]
	Safe coordination	[38], [54], [76]
	Platoon joint	[33], [65]

human-driven vehicles (HDVs) in a two-lane highway. The scenario starts with the AVs and HDVs randomly spawned on the highway with different initial speeds. As the vehicles drive on the highway, the AVs will try to make lane changes to overtake slower HDVs, while cooperating with each other and reacting to the HDVs. Similarly, Hwang et al. [70] proposed a hybrid finite state machine (FSM) and RL approach for AV merging. The FSM handled gap selection while the RL policy executed the final merge maneuver.

In addition to discretionary lane changes, mandatory lane changes such as merging have been studied. In Ref. [61], a highway work zone merge scenario is simulated where the

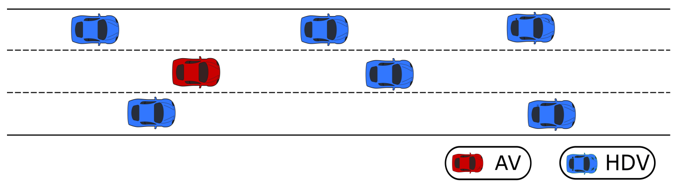

Fig. 3. Illustration of the Lane-changing highway scenario.

right lane is closed and vehicles in the right lane have to move into the left lane. In the scenario, there are multiple independent agents, each of which is a vehicle in the closed right lane. The highway consists of three zones: two metering zones and one merging zone. In metering zones, vehicles in the right lane use RL to optimize their longitudinal positions but do not change lanes. In the merging zone, vehicles merges based on their longitudinal positions. Bouton et al. [71] specifically examined dense traffic merging using a level-k reasoning model for surrounding vehicles. A common evaluation approach is to initialize episodes with different traffic densities and vehicle configurations and assess the trained policy based on metrics such as success rate, collision rate, and trip time [34], [39], [69]. On the other hand, Bouton et al. proposed a Cooperative Intelligent Driver Model (C-IDM) to simulate the longitudinal driving behavior of the main lane vehicles [74]. It adds a cooperation level parameter to the basic Intelligent Driver Model (IDM). The scenario is initialized by simulating the main lane vehicles for 10–20 seconds to create a realistic dense traffic situation. The main lane has 10-14 vehicles initially, with a randomized initial velocity of 4-6 m/s. The ego vehicle then starts driving in the merge lane, observing the main lane vehicles.

While the core task is similar, studies have explored various unique scenario variants. The studies incorporated different elements in their highway lane change scenarios to evaluate their RL policies under varying conditions. Ye et al. [48] specifically focused on congested highways to test discretionary lane changes in dense traffic. In [34], the simulated scenario involves one reinforcement learning (RL) ego vehicle agent that is learning to perform lane change behaviors, interacting with other vehicles on a three-lane highway. In each episode, the ego RL vehicle starts randomly in the middle lane and travels approximately 80 meters before a lane change command to either the left or right is issued. A gap selection module picks a target gap for the RL vehicle to merge into. The RL agent then attempts to execute a smooth lane change maneuver into this target gap. There are two conditions that will terminate the episode: if the RL vehicle deviates more than one full lane width from the center of the target lane, or if the lane change maneuver exceeds 10 seconds. When an episode ends, the next one begins with the ego RL vehicle respawned in a new random position on the middle lane. Through this training approach of placing the learning vehicle in diverse situations across many episodes, the goal is for the RL agent to learn an optimized policy for completing commanded lane changes properly despite realistic highway traffic. Wang et al. [36] leveraged an open source simulator to evaluate lane changes for passing slower vehicles. Hwang et al. [70] included a finite state machine for higher-level planning in their hybrid approach. Bouton et al. [71] is unique in using level-k reasoning for surrounding vehicles to model imperfect human drivers. The diversity of conditions and approaches highlights the complexity of the lane change problem and the need for adaptable RL solutions.

B. Highway Ramp Merging

Various approaches have been proposed for AVs to safely and efficiently merge onto highways amidst surrounding traffic using RL. A common scenario studied is an AV starting on a highway on-ramp that must merge onto an adjacent multi-lane highway before a predefined merge point. The highway has surrounding HDVs or AVs that the merging vehicle must coordinate its actions with. Fig. 1 depicts such a scenario.

Several studies have employed multi-agent reinforcement learning (MARL) algorithms for this merging task. Schester et al. [75] presents a multi-agent reinforcement learning approach where tabular Q-learning is used, comparing single-agent and multi-agent formulations, to learn policies that capture the complex interactions between vehicles during merging. Authors demonstrate that a multi-agent approach considering joint actions can achieve lower collision rates compared to single-agent policies. Additionally, Chen et al. [50] developed a MARL approach with discrete actions, safety measures like action masking, and a reward function considering metrics like collisions. The proposed MARL algorithm incorporates two safety measures. Firstly, it employs an action masking technique to eliminate any invalid actions. Secondly, a novel priority-based safety supervisor is introduced to evaluate safety by predicting the future movement of vehicles for a defined number of steps. Mahatthanajatuphat et al. [52] extended Chen et al.'s problem where multiple agents (AVs) are randomly placed on the merge lane.

Other work has focused on DRL that utilize spatiotemporal information. Due to the limited information that the current step sensor observations provide, using only the current observations as inputs to the learning agent is not enough to accurately determine surrounding vehicles' intent. Researchers have looked into using a temporal series of data as inputs to enrich the agent's representation and reasoning of the dynamic traffic environment. This is intended to incorporate more contextual information regarding the past behaviors and interactions of relevant participants. By utilizing these additional spatiotemporal insights, the agent can achieve greater awareness and foresight regarding the potential trajectories and actions of other vehicles. Wang et al. [53] proposed a spatiotemporal deep Q network (STDQN) that processes current and prior observations to find the optimal action. However, the raw sensor data collected from the traffic environment is likely very high-dimensional and contains a lot of redundant or irrelevant information. Hence, instead of having the observations fed directly to the network, a set of the current and prior observations will go through a spatiotemporal information extraction module to extract only valuable information, which will be sent to the network. The information extraction model consists of a long short-term memory neural network with an attention mechanism (AttenLSTMNN) and a graph convolution network (GCN) to encode spatial and temporal structure in the traffic data. Valiente et al. [57] also used DRL but focused on the robustness of AVs in a mixed-traffic environment to different human-driven vehicle behaviors. Furthermore, a decentralized reward function that can promote different social value orientations (SVO), such as altruistic or egoistic behaviors, is used. As a result, altruistic AVs learn to account for other vehicles' interests safely, such as accelerating or decelerating to allow human drivers to exit or merge on the highway.

Enabling altruistic and cooperative actions has been another area of focus. Toghi et al. [66], [67] simulated AVs learning to coordinate with each other and yield to a human-driven merging vehicle. The scenario involves a highway with multiple lanes and a merging ramp. The scenario starts with all vehicles initialized at random positions on the highway and a single HV on the merging ramp trying to merge into the highway traffic. The goal is for the AVs to learn altruistic behaviors to coordinate with each other and allow the merging vehicle to safely merge into traffic without collisions. In contrast, Kamran et al., [68], focuses on the problem of AV merging in environments with multiple interacting agents such as highways or unsignalized intersections. The key element studied is the uncertainty in predicting whether other vehicles will cooperatively create gaps or not. The authors simulate an ego vehicle approaching a merge point and interacting with up to 16 randomly behaving surrounding vehicles. Some agents are set as cooperative, slowing down to allow merging, while others are non-cooperative.

Some studies have concentrated on the development of decentralized policies and interaction-aware decision making for merging vehicles. Lin et al. [69] focused on learning fully decentralized policies for smooth and safe merging based on local observations. The merging problem encompasses a rich set of scenarios, challenges, and approaches using RL for AV control. Continued progress in areas like interaction-aware MARL, handling complex dynamics, and testing in realistic traffic conditions will further advance capabilities in this critical domain. Hu et al. [37] proposed a model named IDAS that handles negotiating and leveraging cooperation from surrounding human drivers. The primary goals are to enable an AV to strategically leverage human drivers' cooperation and negotiate smooth merging maneuvers via multi-agent interactions. The scenario involves the interaction of a total of eight autonomous and human-driven vehicles. To teach the agent general policy, various driver behaviors are used at random - some drivers are more cooperative when merging than others.

Finally, approaches using real traffic data have been explored. Triest et al. [40] extracted over 400 real highway merging scenarios from the NGSIM dataset [89] to train and test an ego vehicle. However, this mean that the surrounding vehicles follow their recorded trajectories from the NGSIM data without responding to the ego agent. The ego agent must learn to safely merge into gaps between host vehicles based on local observations of surrounding vehicles. On the other hand, Wang et al. [60] combined LSTM (Long Short-Term Memory) and DQN to learn optimal policies while addressing challenges like balancing exploration/exploitation and avoiding local optima. In particular, the authors used a LSTM architecture to model the interactive environment and incorporate historical driving information. The LSTM is pre-trained in a supervised manner on real-world driving data to represent the interactive environment. The DQN is then trained via deep Q-learning using the simulated scenarios. This method recognizes the importance of temporal context in decision-making and leverages real-world driving data to train the model. However, it should be noted that while this approach is effective, it may introduce complexity and computational overhead.

C. Platooning

Vehicular platooning has emerged as an essential research topic for enabling cooperative and automated driving behaviors. A vehicle platoon consists of a company of coordinated vehicles traveling together. The vehicles maintain close proximity to one another in order to decrease aerodynamic friction and increase roadway throughput. There may be both AVs and HDVs in the platoon. Maintaining safe longitudinal and lateral control of platoon vehicles, responding to perturbations, and performing split/join maneuvers to modify platoon composition are crucial technical challenges. If vehicles can be controlled in a safe, seamless, and coordinated manner, platooning has the potential to increase traffic flow stability, improve mobility, and decrease energy consumption. This encouraged research into RL techniques for training vehicle controllers capable of handling the unique difficulties of platoon coordination in mixed traffic.

A common scenario is a mix of AVs, controlled by a centralized RL agent, and HDVs following simple car-following models. A major focus has been on using RL to enable cooperative acceleration and speed control in platoons. The human unpredictability stresses the RL agent's ability to handle unknown dynamics. In [64], the platoon consists of 8 vehicles, with vehicles 1, 3, 5, 7 being autonomous and 2, 4, 6, 8 being human-driven. HDVs cause randomness and unknown dynamics for the DRL algorithm to handle. The scenario starts with the platoon trying to catch up to the lead vehicle, which starts much farther ahead. The goal for the DRL agent is to learn to coordinate the AV accelerations to help the whole platoon steadily catch up to the lead vehicle. Lu et al. [62] developed a novel DRL algorithm, platoon sharing deep deterministic policy gradient algorithm (PSDDPG), which outperformed traditional methods in smoothing traffic flow and robustness. The PSDDPG is used to train three different networks: the lane-changing, car-following, and decision-making networks. So, for different networks, the authors designed different reward functions to achieve good cruising, overtaking, and obstacle avoidance strategies. Jiang et al. [63] applied RL to dampen stop-and-go oscillations in vehicle platoons by training cooperative longitudinal control policies. The authors train RL agents using real driving data, collected from German highways using drones, demonstrating the applicability of RL to improve existing adaptive cruise control systems. Quang et al. [39] investigated using RL for autonomous lead vehicles to improve flow at intersections in mixed traffic.

Ensuring safe and efficient coordination is another key challenge. Zhang et al. [54] incorporated safety constraints and spatial-temporal modeling of the environment for CAV coordination. The authors proposed using multi-agent reinforcement learning (MARL) for CAVs facing problematic driving scenarios in mixed traffic, such as vehicles running red lights and sudden brakes on the highway. The authors designed a safety shield module that uses control barrier functions and quadratic programming to loop through all candidate actions to check the safety of each action and mask unsafe actions. Li et al. [38] proposed a multi-agent algorithm, CommPPO, that uses a specialized communication protocol to avoid common multi-agent RL issues and improve platoon energy efficiency. Therefore, the proposed communication protocol only transmits valuable information explicitly designed for the leader-follower platoon topology. Also, a more explicit and representative reward for each agent is used to avoid lazy agent issues.

Recent work has also focused on integrated longitudinal control combining speed regulation, spacing, and platoon joining/leaving. Berahman et al. [65] developed a unified DRL solution using DDPG that jointly handles speed control, gap regulation, and split/join maneuvers in a single framework. Precisely controlling vehicle platoons involves balancing multiple objectives such as maintaining a constant speed, fixing gaps between vehicles, and smoothly joining/leaving the platoon. This paper formulates a multi-task DRL framework to jointly learn all these platoon behaviors, which is first trained using only two vehicles: the ego vehicle being controlled and the lead vehicle in front of it. The two vehicles are spawned with random initial speeds and inter-vehicle gaps. This small scenario allows the agent to learn longitudinal control behaviors like gap regulation and speed tracking through trial-and-error experience. After the training is complete, the controller is then tested in a more complex situation with one lead car and seven follower vehicles driving behind it. This larger platoon more realistically evaluates the scalability and performance of the trained control policy on factors like string stability, robustness to perturbations, speed consensus among followers, and inter-vehicle gap errors. Moving from logitudinal control to wider platoon management, [76] proposes a hybrid deep reinforcement learning and genetic algorithm called DRG-SP for smart platooning AVs. The key objectives are to intelligently control the leader AV to make optimal platooning decisions, effectively form platoons, and maintain balanced platoon structures. The environment is a four-lane highway populated with AVs. Each platoon has one lead AV (captain AV) and a number of follower AVs. Initially, some AVs are on the highway driving individually but as the simulation progresses, these individual AVs send join requests to the captain AV of a platoon. The captain AV decides whether to accept or reject these requests based on the platoons current state. The simulation ends after sufficient time has elapsed to evaluate the platoon management strategies.

More complex scenarios are being studied as well. Irshayyid et al. [33] applied RL to optimize merge coordination between platoons in a lane reduction scenario, utilizing gap generation maneuvers. Two leader-follower formation platoons facing a two-lane road reduction are simulated. The destination platoon consists of ten vehicles, and the merging platoon consists of four. The two-lane highway is to be reduced to one with the merging platoon lane closing. The goal of the RL is to find the optimal distance from the merging point at which

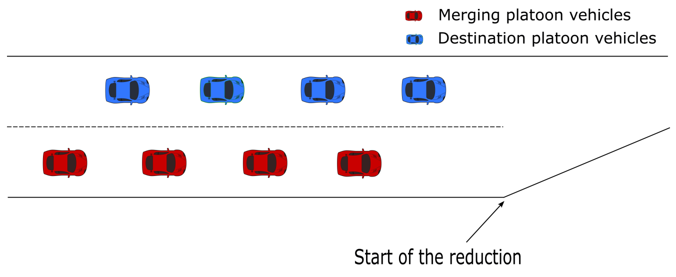

Fig. 4. Two platoons facing a road reduction.

the merging vehicles should ask to perform the lane change maneuver. When a vehicle asks to change lanes, the destination platoon performs a gap generation operation, which will be cooperatively performed to make a safe distance for the merging vehicle. See Fig. 4 for more details.

IV. RL Algorithms

RL has become a dominant technique for training autonomous agents to optimize behaviors and policies through trial-and-error interactions with an environment [44], [90]. The past decade has witnessed remarkable advances in RL algorithms, enabling agents to achieve superhuman performance across complex domains like games, robotics, and autonomous driving. This section synthesizes key developments in modern RL algorithms based on recent research papers in this growing field, as summarized in Table IV.

A. Deep Reinforcement Learning Algorithms

Recent research in RL for autonomous driving has explored both single agent and multi-agent approaches. Within single agent methods, deep Q-networks (DQN) have emerged as a widely adopted algorithm, leveraged for tactical decision making tasks like lane changing and merging [35], [36], [68], [71], [73], [74]. In [35], Deep Q-Network (DQN) is used to train two different policies: the lane-changing policy and the car-following policy in mixed traffic. First, Q-learning is used to estimate the Q values (expected rewards) for each state-action pair, which are stored in a table. Then use the Q values from the table to train neural networks to approximate the Q function. The DQN agent in [36] learns to map traffic states to optimal lane change actions. Actions are selected using an ϵ-greedy, [79], approach to balance exploration and exploitation.

While adopting DQN as the core algorithm, differences exist in the specific DQN extensions used, with papers selecting double DQN [91], dueling DQN [92], prioritized experience replay [93], etc. based on their needs. Some works integrate DQN with safety mechanisms like action masking or trajectory checks to override unsafe decisions made by the DQN policy. As an example, in [36], the DQN outputs high-level lane change decisions, but before executing the action, rule-based safety checks are performed. If the DQN decision is unsafe (will cause a collision), it is overridden. The car is forced to stay in the current lane instead. This prevents the DQN from choosing actions that lead to crashes which means that

TABLE IV
STUDIES ORGANIZED BY RL ALGORITHMS

RL Algorithms		References
Single Agent RL	DQN and DDQN	[35], [36], [60], [68], [71], [73], [74], [78]
	PPO	[33], [39], [48]
	DDPG	[34], [58], [64], [65], [69]
	Others (SAC, A2C, etc.)	[40], [49], [63], [70], [75], [76]
Multi-Agent RL	Centralized training	[37], [54], [62]
	Decentralized training	[50], [56], [57], [66], [67]
Curriculum learning		[34], [37], [38], [71]
Representation learning		[37], [53], [54], [60], [73]

the agent will not learn to not select unsafe actions because these unsafe actions will not be executed. Similarly, Wang et al. [60] proposed dividing the reinforcement learning into two components - an LSTM network to model the interactive driving environment's history, and a DQN for estimating Q-values and action selection. The LSTM handles the non-Markovian aspect and history modeling, while the DQN provides a way to learn an optimal policy from scratch. In [73], the authors uses DQN enhancements such as Double DQN and Dueling DQN to improve the merging agents performance and stability. additionally, Convolutional Neural Network (CNN) is used for handling the image-based state space.

Other papers have explored alternative single agent RL algorithms such as, PPO [33], [39], [48], DDPG, [34], [58], [64], [65], [69], SAC [63], Vanilla policy gradient [49], and A2C [40]. For instance, Szoke et al. [49] adopted a vanilla policy gradient approach with a neural network policy representation to learn optimal acceleration and steering on highways. Policy gradient methods can directly optimize policies, unlike value-based techniques like DQN. Meanwhile, Jiang et al. [63] applied Soft Actor-Critic (SAC), an off-policy actor-critic algorithm well-suited for continuous action spaces like vehicle acceleration control. Compared to DQN, SAC provides increased stability and sample efficiency stemming from its focus on entropy maximization. In [39], authors propose using PPO with an adaptive KL penalty to optimize the policy for controlling multiple AVs at a non-signalized intersection.

B. Multi-agent reinforcement learning

Instead of having all the agents controlled by a centralized RL policy, a multi-agent technique is used. In multi-agent reinforcement learning (MARL), each agent, i.e., each AV, is controlled by its own policy. There are multiple themes to build MARL algorithms based on what the agents can share between each other and how this shared information is used [94]. Centralized training and a decentralized execution scheme is often preferred in multi-agent reinforcement learning (MARL) because agents communicate and coordinate their actions during training, but when it comes to execution, they act independently. This can be seen as a more realistic scenario in many real-world applications where perfect communication is not realistic. Several papers utilize decentralized execution with a centralized critic to enable training across agents' experiences. Lu et al. [62] propose a novel deep reinforcement learning algorithm named platoon sharing deep deterministic policy gradient algorithm (PSDDPG) to overcome the problem of low efficiency of continuous action space exploration. It allows connected vehicles to jointly learn a shared policy network through a centralized training process. Instead of taking the local observations of all the vehicles as an input and generating multiple outputs, the PSDDPG receives the observations of the ego vehicle and its preceding vehicle and generates only one action. simultaneously, all vehicles use the network to get actions, and all vehicles experiences are used to train the network. Hu et al. [37] use a centralized action-value critic alongside decentralized value critics. Other works focus on fully decentralized approaches which is scalable to varying fleet sizes in contrast to decentralized execution with a centralized critic. For instance, Chen et al. [50] and Zhou et al. [56] employ a multi-agent advantage actor-critic (MA2C) with shared parameters and multi-objective rewards. Similarly, [57], the main RL method used is a multi-agent version of the Double Deep Q-Network (DDQN). The problem is formulated as a partially observable stochastic game (POSG) with decentralized agents (AVs) that receive local observations and rewards. On the other hand, Zhang et al. [54] adapt constrained MARL with decentralized training. Toghi et al. [66], [67] propose a decentralized A2C variant with independent actors and critics. Li et al. [38] present CommPPO which adapts PPO for platoon-based multi-agent coordination. In the CommPPO algorithm, each vehicle in the platoon is treated as an agent that can learn and interact with the environment and other agents. The agents share the same neural network parameters and update them based on the collective reward and the policy gradient method. The communication protocol of the CommPPO consists of two parts:

- State transmission part: shares state information between agents based on a predecessor-leader follower topology of the platoon.
- Reward transmission part: A new reward communication channel is proposed propagates rewards to avoid issues like spurious rewards and lazy agents.

A key challenge in MARL is the credit assignment problem—how to allocate rewards or blame to each agent when there are joint rewards or complex interactions. Most papers use simple techniques like local rewards [57], [66], but this can fail in complex cooperative settings. Hu et al. [37] employed a counterfactual baseline in the centralized critic to address this issue [95]. The counterfactual compares the joint Q-value to the Q-value obtained if the agent did not take the action which helps calculate the agent's contribution. Another limitation of current MARL algorithms is the assumption of a fixed number of agents (AVs). All of the papers discussed have a fixed number of agents during the whole episodes for training and evaluation. However, real-world traffic scenarios involve varying numbers of vehicles entering or leaving the environment. The ability to handle a dynamic number of vehicles during training and execution remains an open challenge for MARL scaling.

C. Curriculum Learning

Curriculum learning is an important technique used in several papers to improve the training process and performance of RL agents for autonomous driving. For instance, in Bouton et al. [71], curriculum learning is implemented by gradually exposing the DQN agent to more sophisticated and diverse driving behaviors. Training begins against simple rule-based drivers at lower levels of reasoned decision making. As training progresses, higher level-k opponents with more complex learned policies are introduced to increase task difficulty and variability. This continuation method provides a smoother task progression for the agent to master. Li et al., [38], also leverage curriculum learning for their multi-agent platoon coordination task. They begin training with small platoon sizes of just 2-3 vehicles and incrementally increase the platoon length as training advances. This prevents initial collisions that could occur with large platoons and allows for an easier-to-harder task curriculum. In contrast, in [37], the curriculum learning is employed when designing the reward function. The agent started with individual rewards before adding multi-agent joint rewards. Additionally, Wang et al., [34], take a similar approach by splitting training into two stages. First, an easy lane keeping task is learned. Once the agent can reliably perform lane following, the harder lane changing skills are trained on top of those basics. The platform skills learned in early curricula lead to faster training on more complex tasks. Curriculum learning has several benefits for RL-based autonomous driving:

- Smoother task progression prevents the agent from getting overwhelmed early on
- Platform skills in early curricula accelerate later training stages
- Gradual exposure allows for unsafe exploratory actions to be filtered
- Shaping the task difficulty aids credit assignment in multi-agent settings

The schedule and implementation of curriculum learning remains an open research area. Adaptive curricula based on agent progress seem promising. Overall, curriculum learning is an effective technique for managing the training process of RL driving agents.

D. Representation Learning

Existing state-of-the-art RL algorithms still require millions of training examples in order to learn a decent or near-optimal policy for completing a given task. This plays a notably critical role in real-world implementations in industries, whether in robotics or other complex optimization problems related to decision-making or optimal control. In DRL, the agent's policy directly maps sensory input to action, requiring simultaneous learning of representation and control. Learning useful representations is key for AD, as raw sensor inputs like camera images are high-dimensional and unstructured [96].

A common approach is to manually extract features like distances to lane markings, surrounding vehicles, speed, etc. based on domain knowledge of useful driving information [33], [35], [38], [39], [48]–[50], [52], [75]. Using engineered features can make it easier for the RL agent to learn by reducing the state dimensionality and simplifying the manually specified domain knowledge. However, this requires significant feature engineering effort and may miss useful patterns that could be automatically learned from raw data. It limits the system to what humans can manually identify as relevant. Once the features are extracted, conventional fully connected neural networks are commonly used to represent the policy and value functions [49], [64].

On the other hand, papers like [54] propose using graph neural networks (GNNs) to encode spatial relationships between vehicles and extract structured state representations. GNNs can capture complex dependencies better than fully-connected layers [97]. Also, convolutional neural networks (CNNs) are widely adopted to process visual inputs and extract hierarchical spatial features [37], [73]. This takes advantage of CNNs abilities for translation invariance and local connectivity. Attention mechanisms are being increasingly incorporated [53] to focus neural networks on relevant parts of the observation for decision making, as opposed to treating all inputs uniformly.

Lastly, recent works propose decoupling representation learning completely from policy learning for reinforcement learning agents by using a trained external module to extract key features that will then be fed to the RL agent. For instance, [60], uses real-world driving data (video of a bird's eye view of the highway merge) to train the LSTM module, in a supervised learning fashion, to process sequential observations and extract relevant historical driving context.

V. Action Space

The research papers examined in this review employed a variety of action spaces, as summarized in Table V, to train RL agents for AD tasks like lane changing, merging, car following, and platoon coordination. In particular, the RL actions can be either continuous or discrete, or both, and can include safety modules to ensure the safety during RL training.

The choice of action space plays an important role in determining the flexibility and performance of the trained agent. A key distinction is between continuous and discrete action spaces. Continuous action spaces allow fine-grained control over vehicle actuators like steering angle and acceleration/deceleration [99]. This supports smoother driving behavior. However, continuous spaces can be more challenging to learn due to their high dimensionality [100]. Discrete action spaces, on the other hand, simplify the learning problem but reduce control flexibility [101]. Hybrid approaches combining

TABLE V
STUDIES ORGANIZED BY THE ACTION SPACE USED

Reference	Action Type	Number of Actions	Safety Mechanism
[48]	Discrete	6	Safety Intervention Module [98]
[49]	Discrete	9	None
[39]	Continuous	1	None
[38]	Continuous	1	DRAC
[50]	Discrete	5	Novel priority-based safety supervisor
[52]	Discrete	5	None
[53]	Discrete	3	None
[54]	Discrete	5	Control barrier function
[56]	Discrete	5	None
[57]	Discrete	5	Time-to-collision (TTC)
[58]	Continuous	2	None
[35]	Binary	1	None
	Discrete	6	None
[60]	Continuous	2	None
[34]	Continuous	1	None
[36]	Discrete	3	Rule-based constraints
[61]	Continuous	1	VISSIM basic safety constraints
[62]	Continuous	2	None
	Discrete	3	None
[63]	Continuous	1	None
[64]	Continuous	1	None
[65]	Continuous	1	None
[66]	Discrete	5	None
[67]	Discrete	5	None
[68]	Discrete	3	MPC constraints
[37]	Discrete	7	Masking
[40]	Discrete	4	Low-level controller constraints
[69]	Continuous	1	None
[70]	Continuous	2	Finite state machine (FSM)
[71]	Discrete	5	Intelligent Driver Model (IDM)
[73]	Discrete	3	None
[74]	Discrete	7	None
[33]	Binary	4	None
[75]	Discrete	5	None
	Discrete	25	None
[78]	Discrete	5	Human decision-making model

both continuous and discrete actions are a useful compromise, with discrete actions for high-level decisions and continuous actions for lower-level control.

A. Continuous Action Spaces

Several studies utilized continuous action spaces directly controlling longitudinal acceleration and lateral steering angle [60], [69], [70]. This enables precise vehicle control for maneuvers like merging and lane changing. However, directly outputting raw control values from the policy network can result in jerky trajectories. Constraining the maximum per-timestep change in acceleration and steering helps smooth the motion [70]. In contrast, [34] used a continuous action of the derivative of the yaw rate only, which is referred to as the yaw acceleration, to perform lane change maneuvers. But directly controlling accelerations and steering angles requires extensive training data and tuning to ensure safety. In [62], continuous actions are used for throttle/braking and steering control, while discrete actions are used for high-level decision making like lane changes. For all networks, a hyperbolic tangent activation is used to map the output to the interval of $[-1, 1]$. For the car-following network, to get independent control of acceleration and deceleration, the output interval is divided into two subintervals [-1, 0] and [0, 1], where throttle values range from 0 to 1 and brake values range from -1 to 0. For the lane-changing network, the network output directly mapps to the steering angle using the $tanh$ activation, ranging from -1 to 1 (full left to full right steering). For the decision making network, the output is divided into 3 discrete actions: $[-1, -0.5]$ for left lane change, $[-0.5, 0.5]$ for no lane change, and $[0.5, 1]$ for right lane change.

In [39], the RL agent only controls the acceleration of AVs to be within a specific range of maximum deceleration and maximum acceleration values. In contrast, [38] used a continuous action space between -3 m/s^2 and 3 m/s^2 for acceleration. Similarly, in [58], a continuous action space was employed that enables control over both the steering wheel angle and the longitudinal speed of the vehicle, which are essential for executing lane-changing maneuvers. The use of a continuous action space allows for more delicate actions to ensure safety and comfort while driving. In [63], [65], the action taken by the RL agents is the (continuous) acceleration and deceleration of the ego vehicle, with bounds on the maximum acceleration and deceleration. For example, [63] uses $[-3\ m/s^2, 2\ m/s^2]$ while [65] uses $[3.5\ m/s^2, -3.5\ m/s^2]$. These range reflects the realistic acceleration capabilities of regular passenger vehicles. The objective of the RL agent is to smoothly regulate the ego vehicle's speed and dampen any oscillations propagated from the lead vehicle(s) by choosing appropriate accelerations.

B. Discrete Action Space

Many works selected discrete speed control actions that are converted to smooth accelerations by lower-level controllers [33], [73], [74], [78]. This simplifies learning compared to directly outputting accelerations, while maintaining adequate control over speed changes for tasks like cooperative merging. However, having only a few discrete speed options reduces flexibility. In the research conducted by [48], the action space is composed of six discrete actions categorized into longitudinal and lateral control. The longitudinal control includes following the current lane leader or the target lane leader, while the lateral control includes lane keeping, changing lane, and aborting the lane change. Each action affects the ego vehicle's speed and direction, resulting in six possible actions in the action space. In a similar study conducted by [49], the action space that controls the ego vehicle's acceleration and steering is described. Each action at every time step is defined by two values: steering angle and acceleration, both of which are discretized into three possible values each, resulting in nine possible discrete combinations of steering and acceleration.

In many scenarios, the actions naturally correspond to high-level tactical maneuvers like lane changes, while relying on lower-level motion planners to execute the actions [33], [40], [50], [67]. This simplifies the policy learning problem. However, performance depends heavily on the capabilities of the downstream planner. If the planner cannot closely follow the high-level policy, the end-to-end behavior will suffer. In [67], the paper defines a high-level, discrete action space for each AV. The action space consists of the following:

- Lane change left: Move the AV left to the adjacent lane
- Lane change right: Move the AV right to the adjacent lane
- Accelerate: Increase the AVs speed
- Decelerate: Decrease the AVs speed
- Cruise: Do not change the AVs lane or speed

See Fig. 5 for more details. In [50], [56], the action space consists of high-level discrete actions, such as turn left, turn right, cruising, speed up, and slow down, that control both lateral and longitudinal vehicle dynamics. Similarly, in [53], high-level discrete actions, such as keep straight, turn left, and turn right, are used. In a similar manner, Wang et al., [35], employed high-level discrete actions for the lane change scenario, allowing the vehicle to choose between merging or staying in the same lane. In [54], the authors also employ high-level discrete actions, such as keep lane speed, change lane left, change lane right, brake, and a specified number of discretized throttle intervals. High level action space is also used for platoon coordination management, [76] used a high level action space taken by the captain AV which include accepting/rejecting requests from AVs to join the platoon. Lastly, in [57], [78], high-level discrete actions such as change to the right lane, change to the left lane, accelerate, decelerate, and idle are used. The agent in [36], has a relatively simple action space, with only three discrete actions: stay in the current lane, change lanes to the left, and change lanes to the right. Low-level steering and acceleration are not directly controllable. The agent only makes a high-level decision about whether to stay or change lanes to the left or right. Similarly, [75] set the action space to be a set of accelerations for the each controlled vehicle. Which is discretized into five values: high-decelerate, decelerate, no-decelerate, accelerate, high accelerate. For the multi-agent setting, the Cartesian product of the ego and traffic vehicle accelerations. The simple discrete action space reduces the complexity of the control problem for the DQN. In [68], the action space for the RL agent consists of high-level maneuver decisions that specify the operating mode for the lower-level motion planner. The available actions are: progressive give-way, defensive give-way, and cooperative give-way These actions correspond to different parameterized versions of the safe give-way maneuver that the model predictive control (MPC) planner executes. The progressive option drives faster to increase chances of merging, defensive stops more conservatively, and cooperative cruises to gather more information. In [71], the action space consists of discrete longitudinal and lateral actions, making six total discrete actions. Longitudinally, the action space consists of three different speeds $\{0\ m/s, 3\ m/s, 5\ m/s\}$. The longitudinal action is then converted into a continuous acceleration using the Intelligent Driver Model (IDM). For the lateral part, the agent chooses between staying in the lane and changing lanes. The lateral action triggers a proportional-derivative (PD) controller to execute the lane change.

For platoon coordination, high-level actions like gap generation and role re-assignment allow training cooperative merging policies [33], but require efficient platoon maneuvers from the lower-level controllers. Some works avoided low-level vehicle control entirely, instead using actions to set parameters of car-following models that generate controls autonomously [64]. In [64], the action taken by the DRL agent is setting the full-speed headway parameter for each AV. In other words, rather than directly controlling accelerations, the DRL agent sets the headway distance parameter in the optimal velocity model (OVM) of each AV. Lower headway parameters lead to higher accelerations, and vice versa. The OVM model then converts the headway parameter to a corresponding acceleration control signal for that vehicle. So the action is a vector of continuous headway parameter values, one for each AV. The headway parameter values are bounded between 10-60m based on reasonable driving headways.

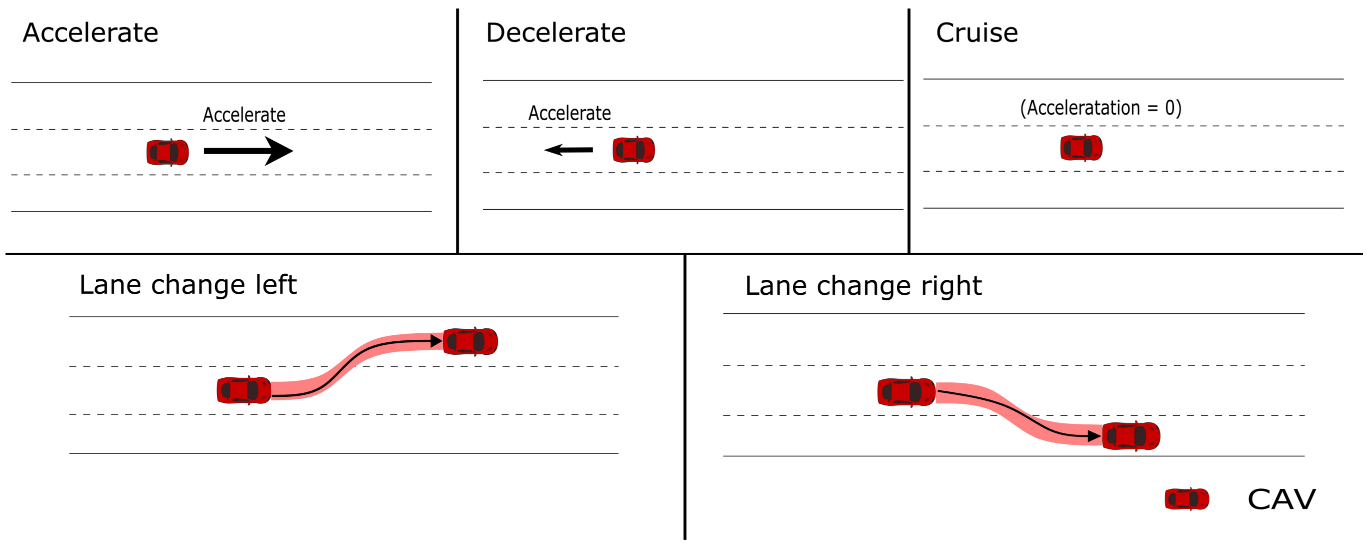

Fig. 5. High level action space.

Overall, direct continuous control enables highly flexible policies at the cost of complex training. Discrete and high-level actions simplify learning but constrain policies. Hybrid action spaces help balance flexibility and tractability.

C. Safety Modules

Most of the papers focus on incorporating safety through careful design of the reward function without using a separate safety module. That is, when the agent chooses an unsafe action, a negative reward is applied to decrease the likelihood of choosing this action at the current state, discussed in Section VII. Other studies handle safety through basic constraints enforcement by the simulating the environment during training [61], [71], or by designing and employing a separate module to classify and override unsafe actions [36], [38], [50], [54], [56], [58], [68], [70]. While the specific implementations differ, some key themes emerge in how these modules shape the action spaces of the RL agents. A common technique is using the safety module as a filter on the policy's outputs before execution. Ye et al. [48] and Hwang et al. [70] evaluate candidate actions and replace unsafe ones with safer alternatives. Hwang et al. uses a finite-state machine (FSM) that determines the high-level driving phase of the AV based on the risks associated with nearby vehicles. The FSM consists of four phases: Ready, Approach, Negotiation, and Lane-change. Each phase has a rule-based controller that ensures safety by calculating the minimum safe distances and times-to-collision with the surrounding vehicles. This restricts the action space by only allowing the subset of actions deemed safe by the module. This study, [50], proposed a priority-based safety supervisor approach to improve the safety of the MARL algorithm for CAVs during highway merging operations. The primary concept is the allocation of a priority index, denoted as p_i, to each CAV. This index is determined based on factors such as the CAV's position, distance from the merging point, and headway time. The priority index of the i_{th} CAV can be expressed as follows:

$$p_i = \alpha_1 p_m + \alpha_2 p_d + \alpha_3 p_h + w_i, \tag{2}$$

where, p_m, p_d,and p_h represent the merging lane priority, distance priority, and time headway priority, respectively. α_1, α_2, and α_3 are the tuning weights of each priority metric. The variable w_i is assigned a modest random value in order to prevent the occurrence of identical priority indices. During each time step, the CAVs are arranged in a list denoted as P_t based on their priority index. Beginning with the top CAV denoted as $P_t[0]$, its exploratory action is evaluated by making predictions about the future movements of the CAV itself as well as the surrounding vehicles over the subsequent T_n time steps. In the event that a collision is detected, the risky action is substituted with the action that ensures the highest level of safety, determined by maximizing the minimum anticipated safety distance. The process continues sequentially along P_t by checking lower priority CAVs while using updated motions for higher priority ones. By prioritizing vehicles with lower safety margins, this scheme significantly improves safety and learning efficiency. The prediction horizon T_n allows foresighted decisions but should be tuned to balance efficiency and uncertainty.

In contrast, Hu et al. [58] embed safety directly into the policy network through masking or priority-based coordination. This paper uses masking mechanisms in the policy network to prevent the RL agent from taking unsafe actions that would violate kinematics constraints, speed limits, or safe distances. Similarly, [38] uses the deceleration rate required to avoid a crash (DRAC) to calculate the Maximum Conflict Acceleration (MCA). If the DRAC between a CAV and its predecessor exceeds a threshold, indicating a crash risk, the CAV's acceleration is limited to MCA to avoid the crash. While more scalable without a separate execution module, directly altering policy outputs may distort learning and constrain the original task. The action space representation also impacts integration of safety. Zhou et al. [56] and Kamran et al. [68] use discrete action spaces of high-level maneuvers, simplifying safety evaluations but reducing control precision. The safety module is composed of two parts: a trajectory planning module with hard constraints and a safety supervisor. The trajectory planning module ensures that the AVs follow a safe and feasible trajectory that satisfies the physical and environmental constraints. The safety supervisor monitors the actions of the AVs and intervenes if they violate the safety rules or cause collisions. The safety module works together with the policy, which is learned by reinforcement learning, to achieve safe and comfortable driving behaviors.

Authors in [36] propose a rule-based constraint module to ensure the safety of the lane change decisions. Specifically, after the DQN agent chooses a high-level lane change action, the controller predicts the trajectories of the ego vehicle and surrounding vehicles based on this action. If the predicted distance between the ego vehicle and a surrounding vehicle drops below a predefined safe threshold at any time, the lane change decision is deemed unsafe and overruled - the ego vehicle stays in the current lane. One of the limitations is that it relies on accurate trajectory prediction, which may be difficult with complex real-world dynamics. In the same vein, [78] use regret theory to model human drivers lane-changing behavior. This model is integrated with the RL agent to assess safety implications of the predicted actions of the agent. A key limitation is that only one drivers behavior is used to train and validate the model. In contrast, [54] uses a parallel safety module based on control barrier functions (CBFs), [102], to constrain their multi-agent A2C policy learning. For each candidate action, the CBF safety check solves a quadratic program to find a control input that keeps the system safe with respect to inter-vehicle distance thresholds. If no feasible solution exists, the action is classified unsafe. This provides formal safety guarantees based on control theory, avoiding reliance on accurate modeling. However, constructing appropriate CBFs can be nontrivial for complex systems.

VI. State Space

This section discusses the state space setups adopted in literature, which are also summarized in Table VI.

In RL, the state space plays a crucial role in determining the behavior of agents. The state space determines what information the agent receives about the environment to determine its actions. The state space should be able to describe the important properties of the environment at the current time step. A well-designed state space in RL should be compact, expressive, and general [103], [104]. It should be minimal and only include the most important information needed for the task to enable efficient learning while still containing enough expressive detail to capture key dynamics and constraints for good decision-making. The state representation should focus on general features of the environment (rather than specifics) to generalize learned behaviors to new situations. In various studies in the field of vehicle automation, different approaches have been taken to define the state space. The most common design choice is to include the ego vehicle kinematics, such as longitudinal and lateral position, velocity, acceleration, and heading angle [38], [39], [48], [49], [64], [65]. This provides

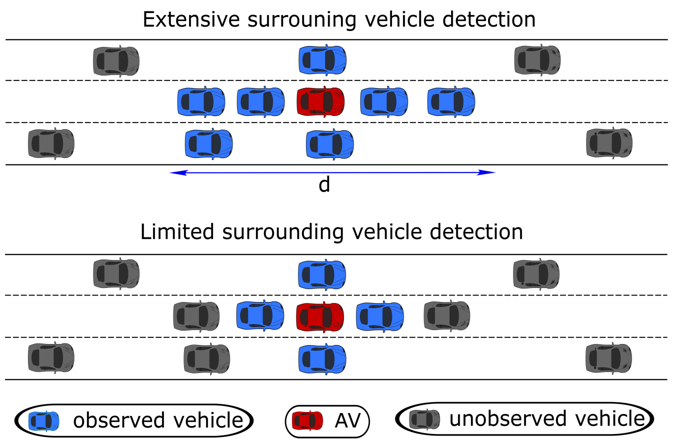
Extensive surrouning vehicle detection
d
Limited surrounding vehicle detection
observed vehicle
AV
unobserved vehicle

Fig. 6. Limited vs extensive surrounding vehicle detection ranges. Where d represents the detection range of the AV.

the most important information needed for motion planning and control. The majority of papers also incorporate data on surrounding vehicles, ranging from just the immediately adjacent vehicles [39], [65], [75] to more extensive context including multiple lead and follow vehicles [48], [49], [53], [63], [64]. See Fig. 6. More information about the environment makes it easier to predict and respond to other vehicles behavior, but it also increases the complexity of the state space.

A. State Information from Surrounding Vehicles

The number of surrounding vehicles that are included into state space differs significantly, ranging from just lead and follow vehicles to up to eight neighbors in [39], [65], [78] and [64], [71], respectively. More information on surrounding vehicles enhances the environmental context for decision-making, but exponentially expands the state space and increases the training difficulty. Some studies explicitly examine trade-offs between state vector dimensionality and learning efficacy [63]. Another difference is the incorporation of road geometry details, like lane curvature, to improve generalization [34], [76].

Most of the papers reviewed here have explored state space representations based on ego vehicle kinematics and surrounding vehicle information, seeking to balance compactness and expressiveness. An emerging alternative approach encodes a local overhead perspective of the environment surrounding the ego vehicle in grid or image form. This representation provides useful traffic context for decision-making while leveraging the power of deep learning methods like convolutional neural networks. Wang et al., [36], designed a grid encoding positions and speeds of nearby vehicles to support lane change behaviors. Additionally, in [53], an occupancy matrix that captures the spatial relationships among the vehicles and traffic information such as volume and density are included in the network. Meanwhile, Nishitani et al., [73], used time-series grayscale images to capture vehicle dynamics and road geometry from a first-person visual perspective. In [50], [56], the authors took a hybrid approach, combining binary occupancy maps with relative vehicle positions and speeds processed

TABLE VI
STUDIES ORGANIZED BY STATE SPACE SCHEME USED

Reference	Current time step	Temporal	Real world data	Extraction module	Row sensor data	Surrounding vehicles
[48]	✓					4 vehicles (leaders and followers at current and target lanes)
[49]	✓					8 (all adjacent) vehicles
[39]	✓					2 (immediately preceding and following) vehicles
[38]	✓					2 (predecessor, and platoon leader) vehicles
[50]	✓					Nearest five Vehicles within 150 m
[52]	✓					Vehicles that are on the highway and on the merge ramp
[53]				✓		All surrounding vehicles within 20-60 m
[54]				✓	✓	3-5 CAV vehicles
[56]	✓					All detected vehicles
[57]		✓				All detected vehicles
[58]	✓					All detected vehicles
[35]	✓					3 (left, right, front)
[60]			✓	✓		2 (front, rear)
[34]	✓					immediate surrounding vehicles
[36]	✓					Vehicles within 60 m in front and 30 m rear
[61]	✓					3 (lead vehicle on current lane, and (lead and lag) vehicles in target lane)
[62]		✓			✓	Front vehicle
[63]	✓					2 (front, rear) vehicles
[64]	✓					All vehicles within its V2V communication range
[65]	✓					Immediate preceding vehicle
[66]		✓				All vehicles within its V2V communication range
[67]		✓				All detected vehicles
[68]		✓		✓		16 vehicles
[37]	✓					All vehicles within 100m in front and behind
[40]	✓		✓			4 (2 front and 2 rear) vehicles
[69]	✓					5 (2 leading, 2 following, and merging vehicle) vehicles
[70]		✓				4 (leading, lagging, front-of-leading, and front) vehicles
[71]	✓					The 8 closest vehicles within 30m
[73]	✓				✓	All vehicles on the on-ramp
[74]	✓				✓	4 (front of ego, behind merge point, front and rear of ego vehicle's projection on main lane)
[33]	✓					10 (destination platoon vehicles)
[75]	✓					1 vehicle
[76]	✓					All AVs
[76]	✓					4 (two front, two rear) vehicles

separately. Though distinct from the previous egocentric representations, these local overhead view techniques share the goal of supplying key environmental state information to guide the learning agent. Both egocentric and allocentric representations have respective strengths in enabling efficient reinforcement learning in AVs. The state space consists of four key observations that provide the necessary context for the ego vehicle to learn effective driving actions in a platoon system:

- $d_{(i-1,i),k}$: The actual longitudinal distance between the ego vehicle i and its preceding vehicle $i-1$ at time step k.
- $e_{i,k}$: The gap error between the desired gap distance d_d and the actual gap distance $d_{(i-1,i),k}$. This lets the agent know how far off it is from the target gap distance:

$$e_{i,k} = d_{(i-1,i),k} - L - d_d, \tag{3}$$

  where L is the length of each vehicle.
- $v_{i,k}$: The current speed of the ego vehicle i.
- $v_{i-1,k}$: The current speed of the preceding vehicle $i-1$. This allows the agent to observe the velocity of the lead vehicle it is following.

Together, these four components give the core state space that provides the necessary observational context about gaps, speeds and gap errors for the agent to learn effective acceleration actions. The gap error $e_{i,k}$ in particular gives a clear feedback signal to the agent about how well it is tracking the desired gap. The use of the preceding vehicle's speed $v_{i-1,k}$ helps the follower learn to match velocities with its leader.

While most papers choose for an egocentric representation from the perspective of the ego vehicle, a few adopt a more global view encoding the entire system state [33], [64] or the relative positions of all agents [37]. This facilitates the modeling of multi-agent interactions, but it may restrict scalability. To explicitly model inter-agent cooperation, some consider augmenting the state variables with non-physical data such as priority levels [37] or collaboration indications [74]. In [74], two distinct observation areas were used. In the first area, the ego vehicle can only detect the longitudinal position and speed of four adjacent vehicles. In the second experiment, the ego vehicle, on the other hand, can completely perceive the state of the surrounding vehicles, the distance to the merging point, longitudinal velocity, acceleration, and cooperation level all correspond to a vehicle's state. The essential distinction here is the collaboration level, denoted as c, which is represented by a binary value. When $c = 1$, the driver cooperates fully and yields to allow the merging vehicle to enter. When $c = 0$, the driver fully ignores the merging vehicle and follows conventional IDM [105].

B. End-to-end State Space

The direct use of raw sensor streams like cameras, LiDAR, and radar as the RL state representation provides both benefits and challenges. On the positive side, this retains maximum real-world details about the driving environment. Sensor data captures rich information on road users, geometry, and dynamics that is otherwise lost with engineered abstractions. For instance, image frames can encode semantic entities like road signs, lane markings, pedestrians, etc. that are not present in simplified state variables [106].

However, using raw data for RL in AVs presents a number of challenges. First, the volume of high-dimensional signals poses a risk of overwhelming the RL algorithm with excessive noise and irrelevant details, which may slow down the learning process [107]. For instance, the majority of pixel values in LIDAR point clouds or camera frames contain no valuable information. Second, the native low-level sensor representations require additional steps to derive the meaningful features and abstractions required for decision making [108]. This increases engineering time and computing costs. Thirdly, interactions and dynamics are not explicitly modeled, so the agent must infer them from observations [109]. Lastly, platform-specific sensor differences complicate the transmission of policies. Several papers, such as Nishitani et al. [73] and Zhang et al. [54], have chosen to represent the state space using raw sensor data. Nishitani et al. utilize grayscale images to provide information about the road environment and the dynamics of surrounding vehicles. These images encode the positions, dynamics, and road shape, offering a rich state representation. In contrast, Hu et al. [58] employ a more minimalist approach by defining the state space using relative distances obtained from a single LiDAR sensor. This reduced representation still proves effective for various driving tasks, emphasizing the adaptability of state-space design in autonomous driving.

Overall, end-to-end RL from raw data remains an open challenge. While methods like deep CNNs show promise for processing high-dimensional inputs [110]–[112], more research is needed on efficiently learning core abstractions from sensor streams for sample-efficient RL. Hybrid approaches that combine learned feature extraction with structured representations may provide a promising direction.

C. Temporal Information

Some recent papers have explored encoding temporal context and history into the state space for reinforcement learning in autonomous driving. Kamran et al. [68] used a k-Markov approximation to include previous observations over several seconds, finding performance benefits with 2.4s of historical data. The authors of [62], [70] tailor their state representation to different control tasks by using multiple frames of states, providing valuable time sequence information. Other studies [57], [66], [67] use Velocity-maps history that capture successive observations to incorporate temporal information into the state space. Meanwhile, Wang et al., [53], incorporate both spatial and temporal information, including the speed and position of CAVs and HDVs over multiple time steps. In [73], image-based state representation was chosen to provide the agent with sufficient information. The state input is three grayscale images, 80 x 256 pixels, showing the road area, including the on-ramp and first lane of the main-lane. The images cover the current time as well as 0.5 and 1.0 seconds in the past, which provides information on the dynamics of the vehicles.

However, determining the optimal history length is challenging - too short loses valuable context while too long risks overwhelming the model. Careful engineering or architecture search is needed to balance efficiency and performance [113]. To address the complexity of temporal data, extraction modules have been proposed to distill historical information into useful state representations. For instance, Wang et al. [60] used a LSTM encoder trained in a supervised manner to summarize interactive driving dynamics from raw trajectory data. Similarly, in [54], a GCN-Transformer module is used to utilize ego vehicle observation, shared observations, and infrastructure observations to generate a spatial-temporal representation of the environment. These learned extraction models aim to automatically determine the most relevant temporal signals, reducing manual feature engineering [114].

However, challenges remain in constrained training of extractors and ensuring the distilled states sufficiently capture all critical environmental details [115]. Architectural choices introduce implicit biases that may overlook important signals. More research is needed into unsupervised state extraction directly optimized for downstream policy performance. Overall, temporal representations and extraction modules show promise but require further analysis on their impact to sample efficiency and generalizability.

VII. Reward

The design of the reward function is a critical component in RL for AV applications. The summary of research papers highlights the variety of approaches taken to formulate rewards that balance key objectives such as safety, efficiency, comfort, and goal achievement. This section compares and contrasts the reward design of several papers that apply RL to autonomous cooperative driving tasks, such as lane changing, merging, intersection crossing, and traffic oscillation, with special attention assigned to the following four aspects: safety, efficiency, comfort, and adaptability, as summarized in Table VII.

A. Safety

Safety is a paramount concern for AV, and it involves avoiding collisions or near-collisions with other vehicles or

TABLE VII
STUDIES ORGANIZED BY REWARD FUNCTIONS

Rewards		References
Safety	Speed-based penalty	[38], [65], [70]
	Distance penalty	[33], [35], [50], [54], [58], [60], [62], [78]
	Action-based penalty	[39], [48], [50], [53], [54], [56], [67]
Efficiency	Speed-based	[33], [36], [38], [40], [48]–[50], [56], [58], [60], [63], [70], [78]
	Social utility	[37], [39], [53], [54], [57], [67], [74]
	Position-based or maneuver-based	[37], [40], [68], [69], [71], [73], [73]
Comfort	Jerk minimization	[33], [38], [48], [58], [65], [68]–[70], [75]
	Control inputs smoothing	[34], [53], [70]
	Speed/distance tracking	[38], [49], [62], [78]
	Supplementary techniques	[68], [70]
Adaptability		[57]

obstacles. Most of the papers reviewed here include some form of safety reward or penalty in their reward functions. However, the scale of these penalties varies. Some impose only minor penalties for collisions [40], [69], [71], [74], while others treat any collision as a terminating state with a large negative reward [63]–[65]. Similarly, [49] uses a negative terminating reward for encountering terminating events such as collisions or leaving the road. The safety reward can be represented as follows:

$$r_t(a, s_t) = -100 \times (1 - \mathbb{I}(E)), \tag{4}$$

where $\mathbb{I}(E)$ represents the indicator function of the event E that the agent has reached at the end. $\mathbb{I}(E)$ of reaching terminating event (collision) is 0 and the reward is -100. Otherwise, if E has reached the end of the episode (without terminating earlier), $\mathbb{I}(E)$ is 0 and the reward is 0. Conversely, in [33], a large penalty is applied if the merging platoon gets too close to the start of the road reduction but with no termination. A more sophisticated way of calculating the safety penalty is used in Ref. [38]. Authors uses a risk measure based on the Deceleration Rate to Avoid a Crash (DRAC), which penalizes the agent if it exceeds a maximum available deceleration rate (MADR), as well as a penalty for exceeding acceleration limits. The safety reward is represented as follows:

$$r = \begin{cases} -\left(\frac{a}{a_{\text{max}}}\right)^2 - 1, & \text{if gap} > L_{\text{threshold}} \text{ or } a > a_{\text{conflict}} \\ -\left(\frac{a}{a_{\text{max}}}\right)^2, & \text{else} \end{cases}, \tag{5}$$

where gap is the headway distance of the ego vehicle, and a_{max} is the acceleration bound.

On the other hand, [70] uses time to collision with a four seconds threshold to determine what is considered unsafe. Similarly, in [35], the reward design encourages CAVs to change lanes safely. It takes into account two important factors: the presence of other vehicles in the adjacent lane and the availability of future driving space. If there are no other vehicles in the same position on the adjacent lane, the reward function assigns a positive reward, encouraging the CAV to change lanes safely. If other vehicles are present, a higher penalty is imposed to discourage lane changes that would cause significant interference. The reward function also takes into account the longitudinal distance between the target vehicle and the vehicle in front of the selected lane, as well as the distance traveled by the vehicle in a future time step.

Similarly, [65] penalizes unsafe speed differences between vehicles to prevent collisions and oscillations. In [48], the safety objective is also evaluated by the risk of collisions or near-collisions. Ref. [39] penalizes collisions with other vehicles or pedestrians, while [50], [56] penalize collisions with other vehicles or lane boundaries. Frequent lane changes is penalized in [53], [67], as they are associated with higher risks. Ref. [54] penalizes getting too close to other vehicles, while [58] incorporates collision avoidance and different expected lane-changing distances in its reward function. Ref. [60] considers actions such as large acceleration/deceleration, small distance to surrounding vehicles, and low speed under free flow conditions as unsafe actions and therefore incur large negative rewards. Similarly, [50], [54], [56] penalize collisions and reward safe headway time between vehicles. Specifically, a large weighting factor w_c is applied for the collision evaluation and a logarithmic function of the time headway is used to measure the safety margin between vehicles, where a predefined time headway threshold is used to avoid penalizing vehicles that maintain a safe distance. Ref. [62] employs a nonlinear function that heavily penalizes collisions but provides an incentive for maintaining sufficient yet not excessive headway distance. This balances the trade-off between safety and traffic flow efficiency. Finally, in [68], safety is handled implicitly by the lower-level motion planning layer.

B. Efficiency

Efficiency is another important objective that is incorporated in many reward functions for autonomous driving. Efficiency generally refers to making progress towards the driving goal in a timely manner without excessive delays. Some papers directly reward higher speeds to encourage efficiency. For example, [53], [54], [58], [60] include components in their reward function that reward higher speeds of the ego vehicle. [38] penalizes the agent for having low speeds to avoid inefficient low-speed driving states. Similarly, [49], [63] uses an immediate reward based on the difference between the agent's current and desired speeds.

Other papers focus on making progress towards a predefined goal position or completing the maneuver itself. For example, [37], [68], [69], [71], [73] provide positive rewards for successfully completing the merging maneuver. Ref. [40] rewards forward progress along the road, while [33] investigates rewards based on minimizing time to complete the merge. In [73], the reward is only provided when the ego vehicle completes the merge onto the main lane. In other words, no intermediate reward is given during the merging process. Some papers also consider efficiency more holistically, looking at the traffic flow overall [37].

For instance, [39] aims to maximize the average speed of all vehicles at the intersection, while [54] rewards maximizing the average speed of all CAVs. A social reward is included in [67] that accumulates the progress of all vehicles, while [74] uses a time penalty factor to incentivize reaching the goal position quickly. Similarly, [53] rewards maximizing the velocity of both CAVs and HDVs. Ref. [57] rewards optimizing social utility, which involves cooperation among AVs to achieve socially desirable outcomes. The reward of vehicle, i, can be defined as

$$R_i(s,a) = \cos(\phi) \times r_i^{\text{ego}} + \sin(\phi) \times r_i^{\text{social}}, \tag{6}$$

where r_i^{ego} is the specific reward of the AV (egoistic) and r_i^{social} is the overall reward of other vehicles (social) in relation to the i^{th}. The distance traveled or time taken to reach the goal position is also used as a measure of efficiency in some works. Ref. [48] uses a reward function based on travel time and distance to the target lane, and [36] rewards driving as fast as possible down the highway. Finally, [70] rewards reducing the time taken to approach the target lane center, while [50], [56] reward reaching the target lane within a given time horizon.

By incorporating various efficiency-related rewards and penalties, each formulation above aims to obtain the right balance between making timely progress and other objectives like safety and comfort. The weights given to the efficiency components allow tuning this trade-off as per the needs of the specific scenario.

C. Comfort

Comfort is a key criterion that needs to be optimized in AV to provide a smooth and pleasant riding experience for passengers. Several papers approach this in different ways through their reward function formulation, as detailed below.

a) Jerk minimization: A common technique is to penalize large jerks and sudden changes in accelerations/decelerations. This helps avoid abrupt starts and stops that reduce comfort. For example, [33], [38], [48], [65], [68]–[70] impose absolute or squared penalties on the magnitude of jerk. In particular, [68] filters out jerks above a threshold so only large uncomfortable jerks are penalized, while [38] penalizes exceeding acceleration limits. [48] uses a comfort objective evaluated by the jerk in lateral and longitudinal directions, while, Ref. [58] penalizes minimizing the angular velocity of the steering wheel and jerk, represented as:

$$R_{\text{comfort}} = k_w \times \dot{\theta}_w + k_a \times j, \tag{7}$$

where j represents the jerk, $\dot{\theta}_w$ is the angular velocity of the steering wheel of the agent, and k_w and k_a are the weight coefficients.

b) Smooth control inputs: Maintaining smooth steering and throttle/braking control is also important. In this regard, [34] punishes large yaw rate and yaw acceleration for smooth lane changes,

$$r_s = -w_1 \times |\omega_{\text{yaw}}| - w_2 \times |a_{\text{lat}}|, \tag{8}$$

where ω_{yaw} represents the yaw rate, the a_{lat} is the yaw acceleration, and w_1 and w_2 are the weight coefficients. while [70] penalizes large angular speeds of the steering wheel. Finally, [53] also penalizes acceleration, deceleration, and jerk that exceed a threshold.

c) Speed and distance regulation: Comfortable speed control and maintaining safe distances from other vehicles helps avoid sudden braking scenarios. To achieve this goal, [49] rewards speed tracking accuracy, while [38] penalizes going slower than acceptable speeds. Similarly, [62] rewards matching lead vehicle speed when headway is large. It also uses a nonlinear headway reward that incentivizes sufficient but not excessive distance between vehicles. The reward can be described as follows:

$$\text{Reward}_{\text{headway}} = \begin{cases} -100, & \text{if x} \leq 0, \\ -100(1 - \sqrt{(1-(x-1)^2)}), & 0 < \text{x} \leq 1, \\ 0, & \text{x} > 1. \end{cases} \tag{9}$$

where x represent the time gap between the front and ego vehicle. It is limited to a maximum of 100 to prevent very large headway values from degrading the training.

d) Supplementary techniques: Some papers use supplementary techniques in addition to reward design to improve comfort. For instance, [68] handles collisions at a lower level so comfort can be prioritized in the reward, while [70] adapts weights based on proximity to the mandated lane change point.

D. Adaptability

Adaptability is another aspect of AV that involves adjusting to different behaviors and traffic conditions. It involves learning from experience and generalizing to new situations. Among all the papers reviewed here, only one paper [57] explicitly includes an adaptability component in its reward function. In particular, rewards AVs for adjusting to different behaviors and traffic conditions using an implicit learning approach. Valiente et al. define an adaptation error (A_{error}) that explicitly rewards adaptability of the trained AVs to new scenarios. This is calculated as:

$$A_{\text{error}} = w_s \times (C) + w_e \times \big(1 - \frac{DT}{DT_{\text{max}}}\big), \tag{10}$$

where:

- C is the percentage of episodes with a crash when tested in the new scenario
- DT is the average distance traveled by AVs in the new scenario
- DT_{max} is the maximum possible distance in that scenario
- w_s and w_e are weights for the safety and efficiency terms

Lower adaptation error indicates the AVs are adjusting well to the new conditions. The safety term penalizes crashes more heavily with a higher weight w_s. The AVs are trained using decentralized multi-agent reinforcement learning, with each AV, i, optimizing its own reward function:

$$R_i(s,a) = R_{\text{ego}} + R_{\text{social}}, \tag{11}$$

where the ego reward is defined as:

$$R_{\text{ego}}(s,a) = \cos(\phi_i)\, r_i(s,a), \tag{12}$$

where the ego reward $r_i(s,a)$ encourages progress of the ego vehicle based on traffic metrics like speed. The angle ϕ_i controls the weight on the ego vs social reward. The social reward contains terms for both cooperation with other AVs and sympathy for the human drivers:

$$\begin{aligned} R_{\text{social}} = \sin(\phi_i) \Bigg[\sum_j r_{i,j}^{\text{AV}}(s,a) + \sum_k r_{i,k}^{\text{HV}}(s,a) \\ + \sum_k r_{i,k}^{M}(s,a) \Bigg] \end{aligned} \tag{13}$$

The r^M term accounts for completing the assigned mission (merging, exiting), while r^{AV} and r^{HV} reward altruistic consideration of other AVs and humans respectively.

E. Summary

In summary, we have compared and contrasted the reward design of several papers that apply RL to autonomous driving tasks. We have focused on four aspects: safety, efficiency, comfort, and adaptability. We have found that most of the papers include some form of safety and efficiency reward or penalty in their reward functions, while fewer papers include comfort and adaptability components. We have also found that different papers use different metrics and methods to evaluate and optimize these aspects. This shows that reward design is a challenging and diverse problem that requires careful consideration of the goals and constraints of the task at hand.

VIII. Additional Discussion and Future Work

A. Intersection Control

This review has focused on RL techniques for automated driving behaviors related to highways, such as lane changing, merging, and platooning. However, its worth noting that similar approaches can also be extended to intersection scenarios. Efficient intersection management remains an open challenge for autonomous vehicles [116], [117], though initial RL research shows promise [118]–[120]. Similar to highway ramp merging, intersections require safe coordination between multiple vehicles with potential conflicts. However, intersections introduce additional complexities such as complying with traffic priorities and avoiding deadlocks. Additionally, state representations must also capture different road features like stop lines and crossing routes.

Despite these differences, core RL components analyzed in this review like reward design, action spaces, and safety mechanisms remain highly relevant.

B. Real World Data

The incorporation of real-world driving data provides greater realism and captures interactive dynamics that are hard to simulate [40], [60]. Real-world trajectories better represent complex road user behaviors and edge cases that agents may encounter in deployment [121], [122]. This approach helps overcome the reality gap compared to purely simulated training data. However, significant challenges exist with leveraging real-world data for RL. First, collecting and annotating large-scale driving datasets is hugely expensive and time-consuming, restricting availability [123]. Second, there is a risk of overfitting to the distributions and scenarios present in the finite dataset. This could harm the model's generalizability [124]. Third, static offline datasets cannot cover all possible events, lacking richness and adaptivity [125]. Finally, data balancing and biases in collection impact the learned policies [126]. Overall, further research is needed to unlock the full potential of real-world data for more robust RL policies.

C. Future Directions

Based on the review of the relevant papers in the field of RL-based highway AV control, the following key directions for future research include:

- Multi-agent reinforcement learning (MARL) algorithms that can handle varying numbers of agents. Current MARL implementations assume a fixed number of AVs during training and execution. Developing more flexible approaches that can train agents for scenarios with dynamic vehicles entering and exiting could improve scalability.
- Increase testing in diverse and realistic traffic conditions. Most studies use simulated environments with limited realism. Leveraging large-scale real-world driving datasets and traffic simulators with sophisticated driver models would help evaluate generalization.
- Development of standardized benchmarks for comparative evaluation. Currently, there is a lack of common benchmarks that can systematically validate and compare different RL driving policies. Each work uses distinct simulation setups and metrics, preventing standardized assessments. Constructing benchmark suites with scenarios sampled from real-world distributions could better analyze generalization. Shared benchmarks would also facilitate reproducible comparisons between algorithmic innovations and measure progress.
- More sophisticated human driver models that capture complex interactive behaviors. Most papers assume human-driven vehicles will follow simple car-following rules along fixed trajectories, rather than responding realistically to the actions of learning agents. Developing better human driver models that cover the diversity of human behaviors could significantly enhance RL learning and policy evaluation.
- Handling imperfect state observations from real-world sensors. Most works assume perfect knowledge of surrounding vehicle states used in the RL input representation. However, accurately perceiving complex noisy

traffic scenes poses major challenges for real AV sensor suites. Testing how learned policies degrade with noisy detections and exploring sensor fusion to improve state estimates are vital.

IX. Conclusion

This paper reviewed the state-of-the-art reinforcement learning (RL) techniques for autonomous vehicle (AV) control in various scenarios, such as lane changing, ramp merging, and platooning. Existing problem formulations, RL algorithms, simulations, and metrics studies have been analyzed in terms of their design choices, benefits, and challenges. RL-based AV control, especially in highway conditions, has significant benefits to improve our society, such as enabling cooperative and altruistic behaviors, handling complicated dynamics and uncertainties. In addition, the limitations and gaps of current methods are discussed, including balancing state-space dimensionality and expressiveness, assuring safety and robustness, and testing under realistic traffic conditions. This survey's findings can guide future research toward the development of more effective and generalizable RL solutions for automated driving in complex environments.

References

[1] J. Deichmann, *Autonomous Driving's Future: Convenient and Connected*. McKinsey, 2023.

[2] Environmental and E. S. I. (EESI), "Issue brief: Autonomous vehicles: State of the technology and potential role as a climate solution," Jun 2021. [Online]. Available: https://www.eesi.org/papers/view/issue-brief-autonomous-vehicles-state-of-the-technology-and-potential-role-as-a-climate-solution

[3] Y. Fan, N. Wexler, F. Douma, G. Ryan, C. Hong, Y. Li, and Z.-L. Zhang, "Advancing social equity with shared autonomous vehicles: Literature review, practitioner interviews, and stated preference surveys," 2022.

[4] T. Qie, W. Wang, C. Yang, Y. Li, W. Liu, and C. Xiang, "A path planning algorithm for autonomous flying vehicles in cross-country environments with a novel TF-RRT* method," *Green Energy and Intelligent Transportation*, vol. 1, no. 3, p. 100026, 2022.

[5] Z. Zhou, C. Rother, and J. Chen, "Event-triggered model predictive control for autonomous vehicle path tracking: Validation using CARLA simulator," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 6, pp. 3547–3555, June 2023.

[6] Z. Zhou, J. Chen, M. Tao, P. Zhang, and M. Xu, "Experimental validation of event-triggered model predictive control for autonomous vehicle path tracking," in *2023 IEEE International Conference on Electro Information Technology*, Romeoville, IL, May 18–20, 2023.

[7] M. R. Hajidavalloo, J. Chen, Q. Hu, and Z. Li, "Study on the benefits of integrated battery and cabin thermal management in cold weather conditions," in *American Control Conference*, San Diego, CA, May 31–June 2, 2023.

[8] L. Yang, C. Lu, G. Xiong, Y. Xing, and J. Gong, "A hybrid motion planning framework for autonomous driving in mixed traffic flow," *Green Energy and Intelligent Transportation*, vol. 1, no. 3, p. 100022, 2022.

[9] J. Chen and Z. Yi, "Comparison of event-triggered model predictive control for autonomous vehicle path tracking," in *IEEE Conference on Control Technology and Applications*, San Diego, CA, August 8–11, 2021.

[10] F. Poinsignon, L. Chen, S. Jiang, K. Gao, H. Badia, and E. Jenelius, "Autonomous vehicle fleets for public transport: scenarios and comparisons," *Green Energy and Intelligent Transportation*, vol. 1, no. 3, p. 100019, 2022.

[11] C. Rother, Z. Zhou, and J. Chen, "Development of a four-wheel steering scale vehicle for research and education on autonomous vehicle motion control," *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 5015–5022, August 2023.

[12] J. Chen, M. Liang, and X. Ma, "Probabilistic analysis of electric vehicle energy consumption using MPC speed control and nonlinear battery model," in *2021 IEEE Green Technologies Conference*, Denver, CO, April 7–9, 2021.

[13] J. M. Anderson, K. Nidhi, K. D. Stanley, P. Sorensen, C. Samaras, and O. A. Oluwatola, *Autonomous vehicle technology: A guide for policymakers*. Rand Corporation, 2014.

[14] L. Gomes, "Hidden obstacles for google's self-driving cars," *MIT Technology Review. Retrieved November*, vol. 13, p. 2020, 2014.

[15] G. De La Torre, P. Rad, and K.-K. R. Choo, "Driverless vehicle security: Challenges and future research opportunities," *Future Generation Computer Systems*, vol. 108, pp. 1092–1111, 2020.

[16] D. Sadigh, K. Driggs-Campbell, A. Puggelli, W. Li, V. Shia, R. Bajcsy, A. L. Sangiovanni-Vincentelli, S. S. Sastry, and S. A. Seshia, "Data-driven probabilistic modeling and verification of human driver behavior," in *AAAI Spring Symposium-Technical Report*, 2014, pp. 56–61.

[17] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[18] Y. Chen, N. Sohani, and H. Peng, "Modelling of uncertain reactive human driving behavior: a classification approach," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3615–3621.

[19] J. Wu, Y. Wang, Z. Shen, L. Wang, H. Du, and C. Yin, "Distributed multilane merging for connected autonomous vehicle platooning," *Science China Information Sciences*, vol. 64, no. 11, pp. 1–16, 2021.

[20] J. Hourdakis and P. G. Michalopoulos, "Evaluation of ramp control effectiveness in two twin cities freeways," *Transportation Research Record*, vol. 1811, no. 1, pp. 21–29, 2002.

[21] R. Scarinci and B. Heydecker, "Control concepts for facilitating motorway on-ramp merging using intelligent vehicles," *Transport reviews*, vol. 34, no. 6, pp. 775–797, 2014.

[22] X.-m. Chen, M. Jin, C.-Y. Chan, Y.-s. Miao, and J.-w. Gong, "Bionic decision-making analysis during urban expressway ramp merging for autonomous vehicle," Tech. Rep., 2017.

[23] J. Wei, J. M. Dolan, and B. Litkouhi, "Autonomous vehicle social behavior for highway entrance ramp management," in *2013 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2013, pp. 201–207.

[24] M. Karimi, C. Roncoli, C. Alecsandru, and M. Papageorgiou, "Cooperative merging control via trajectory optimization in mixed vehicular traffic," *Transportation Research Part C: Emerging Technologies*, vol. 116, p. 102663, 2020.

[25] M. Aramrattana, T. Larsson, C. Englund, J. Jansson, and A. Nåbo, "A simulation study on effects of platooning gaps on drivers of conventional vehicles in highway merging situations," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 4, pp. 3790–3796, 2020.

[26] M. I. Pavel, S. Y. Tan, and A. Abdullah, "Vision-based autonomous vehicle systems based on deep learning: A systematic literature review," *Applied Sciences*, vol. 12, no. 14, p. 6831, 2022.

[27] L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance," *arXiv preprint arXiv:2207.14529*, 2022.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[30] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[31] F. Dang, D. Chen, J. Chen, and Z. Li, "Event-triggered model predictive control with deep reinforcement learning," *IEEE Transactions on Intelligent Vehicles*, accepted for Publication, October 2023.

[32] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[33] A. Irshayyid and J. Chen, "Comparative study of cooperative platoon merging control based on reinforcement learning," *Sensors*, vol. 23, no. 2, p. 990, 2023.

[34] P. Wang, H. Li, and C.-Y. Chan, "Continuous control for automated lane change behavior based on deep deterministic policy gradient algorithm," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1454–1460.

[35] J. Wang, C. Hu, J. Zhao, S. Zhang, and Y. Han, "Deep q-network-based efficient driving strategy for mixed traffic flow with connected and autonomous vehicles on urban expressways," *Transportation Research Record*, p. 03611981231161355, 2023.

[36] 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.

[37] Y. Hu, A. Nakhaei, M. Tomizuka, and K. Fujimura, "Interaction-aware decision making with adaptive strategies under merging scenarios," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 151–158.

[38] M. Li, Z. Cao, and Z. Li, "A reinforcement learning-based vehicle platoon control strategy for reducing energy consumption in traffic oscillations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5309–5322, 2021.

[39] D. Quang Tran and S.-H. Bae, "Proximal policy optimization through a deep reinforcement learning framework for multiple autonomous vehicles at a non-signalized intersection," *Applied Sciences*, vol. 10, no. 16, p. 5722, 2020.

[40] S. Triest, A. Villaflor, and J. M. Dolan, "Learning highway ramp merging via reinforcement learning with temporally-extended actions," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1595–1600.

[41] M. Bouton, K. Julian, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Utility decomposition with deep corrections for scalable planning under uncertainty," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 462–469.

[42] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[43] M. Kaushik, K. M. Krishna *et al.*, "Parameter sharing reinforcement learning architecture for multi agent driving behaviors," *arXiv preprint arXiv:1811.07214*, 2018.

[44] 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*, vol. 23, no. 6, pp. 4909–4926, 2021.

[45] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 9, pp. 7366–7390, 2022.

[46] P. Yadav, A. Mishra, and S. Kim, "A comprehensive survey on multi-agent reinforcement learning for connected and automated vehicles," *Sensors*, vol. 23, no. 10, p. 4710, 2023.

[47] J. Zhu, S. Easa, and K. Gao, "Merging control strategies of connected and autonomous vehicles at freeway on-ramps: a comprehensive review," *Journal of Intelligent and Connected Vehicles*, vol. 5, no. 2, pp. 99–111, 2022.

[48] F. Ye, X. Cheng, P. Wang, C.-Y. Chan, and J. Zhang, "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1746–1752.

[49] L. Szoke, S. Aradi, T. Bécsi, and P. Gaspar, "Vehicle control in highway traffic by using reinforcement learning and microscopic traffic simulation," in *2020 IEEE 18th International Symposium on Intelligent Systems and Informatics (SISY)*. IEEE, 2020, pp. 21–26.

[50] D. Chen, Z. Li, Y. Wang, L. Jiang, and Y. Wang, "Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic," *arXiv preprint arXiv:2105.05701*, 2021.

[51] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[52] C. Mahatthanajatuphat, K. Srisomboon, W. Lee, P. Samothai, and A. Kheaksong, "Investigation of multi-agent reinforcement learning on merge ramp for avoiding car crash on highway," in *2022 37th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. IEEE, 2022, pp. 1050–1053.

[53] S. Wang, H. Fujii, and S. Yoshimura, "Generating merging strategies for connected autonomous vehicles based on spatiotemporal information extraction module and deep reinforcement learning," *Physica A: Statistical Mechanics and its Applications*, vol. 607, p. 128172, 2022.

[54] Z. Zhang, S. Han, J. Wang, and F. Miao, "Spatial-temporal-aware safe multi-agent reinforcement learning of connected autonomous vehicles in challenging scenarios," *arXiv preprint arXiv:2210.02300*, 2022.

[55] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[56] W. Zhou, D. Chen, J. Yan, Z. Li, H. Yin, and W. Ge, "Multi-agent reinforcement learning for cooperative lane changing of connected and autonomous vehicles in mixed traffic," *Autonomous Intelligent Systems*, vol. 2, no. 1, p. 5, 2022.

[57] R. Valiente, B. Toghi, R. Pedarsani, and Y. P. Fallah, "Robustness and adaptability of reinforcement learning-based cooperative autonomous driving in mixed-autonomy traffic," *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 397–410, 2022.

[58] H. Hu, Z. Lu, Q. Wang, and C. Zheng, "End-to-end automated lane-change maneuvering considering driving style using a deep deterministic policy gradient algorithm," *Sensors*, vol. 20, no. 18, p. 5443, 2020.

[59] M. Fellendorf and P. Vortisch, "Microscopic traffic flow simulator vissim," *Fundamentals of traffic simulation*, pp. 63–93, 2010.

[60] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–6.

[61] T. Ren, Y. Xie, and L. Jiang, "Cooperative highway work zone merge control based on reinforcement learning in a connected and automated environment," *Transportation research record*, vol. 2674, no. 10, pp. 363–374, 2020.

[62] S. Lu, Y. Cai, L. Chen, H. Wang, X. Sun, and Y. Jia, "A sharing deep reinforcement learning method for efficient vehicle platooning control," *IET Intelligent Transport Systems*, vol. 16, no. 12, pp. 1697–1709, 2022.

[63] L. Jiang, Y. Xie, N. G. Evans, X. Wen, T. Li, and D. Chen, "Reinforcement learning based cooperative longitudinal control for reducing traffic oscillations and improving platoon stability," *Transportation Research Part C: Emerging Technologies*, vol. 141, p. 103744, 2022.

[64] T. Chu and U. Kalabić, "Model-based deep reinforcement learning for cacc in mixed-autonomy vehicle platoon," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 4079–4084.

[65] M. Berahman, M. Rostami-Shahrbabaki, and K. Bogenberger, "Multi-task vehicle platoon control: A deep deterministic policy gradient approach," *Future transportation*, vol. 2, no. 4, pp. 1028–1046, 2022.

[66] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Cooperative autonomous vehicles that sympathize with human drivers," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4517–4524.

[67] ——, "Altruistic maneuver planning for cooperative autonomous vehicles using multi-agent advantage actor-critic," *arXiv preprint arXiv:2107.05664*, 2021.

[68] D. Kamran, Y. Ren, and M. Lauer, "High-level decisions from a safe maneuver catalog with reinforcement learning for safe and cooperative automated merging," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 804–811.

[69] Y. Lin, J. McPhee, and N. L. Azad, "Anti-jerk on-ramp merging using deep reinforcement learning," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 7–14.

[70] S. Hwang, K. Lee, H. Jeon, and D. Kum, "Autonomous vehicle cut-in algorithm for lane-merging scenarios via policy-based reinforcement learning nested within finite-state machine," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 594–17 606, 2022.

[71] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement learning with iterative reasoning for merging in dense traffic," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[72] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[73] I. Nishitani, H. Yang, R. Guo, S. Keshavamurthy, and K. Oguchi, "Deep merging: Vehicle merging controller based on deep reinforcement learning with embedding network," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 216–221.

[74] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Cooperation-aware reinforcement learning for merging in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3441–3447.

[75] L. Schester and L. E. Ortiz, "Longitudinal position control for highway on-ramp merging: A multi-agent approach to automated driving," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3461–3468.

[76] S. B. Prathiba, G. Raja, K. Dev, N. Kumar, and M. Guizani, "A hybrid deep reinforcement learning for autonomous vehicles smart-platooning," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 12, pp. 13 340–13 350, 2021.

[77] F. De Rango, P. Raimondo, and D. Amendola, "Extending sumo and plexe simulator modules to consider energy consumption in platooning management in vanet," in *2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*. IEEE, 2019, pp. 1–9.

[78] D. Chen, L. Jiang, Y. Wang, and Z. Li, "Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 4355–4361.

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

[80] H. Dong, H. Dong, Z. Ding, S. Zhang, and Chang, *Deep Reinforcement Learning*. Springer, 2020.

[81] W. Gao, M. Mynuddin, D. C. Wunsch, and Z.-P. Jiang, "Reinforcement learning-based cooperative optimal output regulation via distributed adaptive internal model," *IEEE transactions on neural networks and learning systems*, 2021.

[82] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 4, no. 1, pp. 1–103, 2010.

[83] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[84] J. Chen, X. Meng, and Z. Li, "Reinforcement learning-based event-triggered model predictive control for autonomous vehicle path following," in *2022 American Control Conference*, Atlanta, GA, June 8–10, 2022.

[85] H. Jiang, R. Gui, Z. Chen, L. Wu, J. Dang, and J. Zhou, "An improved Sarsa (λ) reinforcement learning algorithm for wireless communication systems," *IEEE Access*, vol. 7, pp. 115 418–115 427, 2019.

[86] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, p. 229–256, may 1992. [Online]. Available: https://doi.org/10.1007/BF00992696

[87] V. R. Konda and J. N. Tsitsiklis, "Onactor-critic algorithms," *SIAM journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.

[88] S. Levine and V. Koltun, "Guided policy search," in *International conference on machine learning*. PMLR, 2013, pp. 1–9.

[89] "Next generation simulation (ngsim)," 2020, [Onlne]. Available:, https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

[90] A. Sharma, K. Xu, N. Sardana, A. Gupta, K. Hausman, S. Levine, and C. Finn, "Autonomous reinforcement learning: Formalism and benchmarking," *arXiv preprint arXiv:2112.09605*, 2021.

[91] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[92] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[93] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[94] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2022.

[95] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[96] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[97] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing graph neural networks: A survey from algorithms to accelerators," *ACM Computing Surveys (CSUR)*, vol. 54, no. 9, pp. 1–38, 2021.

[98] W. Saunders, G. Sastry, A. Stuhlmueller, and O. Evans, "Trial without error: Towards safe reinforcement learning via human intervention," *arXiv preprint arXiv:1707.05173*, 2017.

[99] S. Wang, D. Jia, and X. Weng, "Deep reinforcement learning for autonomous driving," *arXiv preprint arXiv:1811.11329*, 2018.

[100] 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.

[101] N. Müller and T. Glasmachers, "Challenges in high-dimensional reinforcement learning with evolution strategies," in *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part II 15*. Springer, 2018, pp. 411–423.

[102] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.

[103] C. L. Lan, S. Tu, A. Oberman, R. Agarwal, and M. G. Bellemare, "On the generalization of representations in reinforcement learning," *arXiv preprint arXiv:2203.00543*, 2022.

[104] A. Merckling, N. Perrin-Gilbert, A. Coninx, and S. Doncieux, "Exploratory state representation learning," *Frontiers in Robotics and AI*, vol. 9, p. 762051, 2022.

[105] 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.

[106] I. Papadeas, L. Tsochatzidis, A. Amanatiadis, and I. Pratikakis, "Real-time semantic image segmentation with deep learning for autonomous driving: A survey," *Applied Sciences*, vol. 11, no. 19, p. 8802, 2021.

[107] A. Remonda, S. Krebs, E. Veas, G. Luzhnica, and R. Kern, "Formula rl: Deep reinforcement learning for autonomous racing using telemetry data," *arXiv preprint arXiv:2104.11106*, 2021.

[108] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2722–2730.

[109] Y. Zhang, A. Carballo, H. Yang, and K. Takeda, "Perception and sensing for autonomous vehicles under adverse weather conditions: A survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 196, pp. 146–177, 2023.

[110] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[111] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions (2014)," *arXiv preprint arXiv:1409.4842*, vol. 10, 2014.

[112] I. Kerenidis, J. Landman, and A. Prakash, "Quantum algorithms for deep convolutional neural networks," *arXiv preprint arXiv:1911.01117*, 2019.

[113] D. Zhang, J. Han, L. Zhao, and D. Meng, "Leveraging prior-knowledge for weakly supervised object detection under a collaborative self-paced curriculum learning framework," *International Journal of Computer Vision*, vol. 127, pp. 363–380, 2019.

[114] I. Rida, "Feature extraction for temporal signal recognition: An overview," *arXiv preprint arXiv:1812.01780*, 2018.

[115] T. Li, J. P. Higgins, and J. J. Deeks, "Collecting data," *Cochrane handbook for systematic reviews of interventions*, pp. 109–141, 2019.

[116] A. Abbas-Turki, Y. Mualla, N. Gaud, D. Calvaresi, W. Du, A. Lombard, M. Dridi, and A. Koukam, "Autonomous intersection management: Optimal trajectories and efficient scheduling," *Sensors*, vol. 23, no. 3, p. 1509, 2023.

[117] L. Wei, Z. Li, J. Gong, C. Gong, and J. Li, "Autonomous driving strategies at intersections: Scenarios, state-of-the-art, and future outlooks," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 44–51.

[118] Y. Liu, Q. Zhang, and D. Zhao, "A reinforcement learning benchmark for autonomous driving in intersection scenarios," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 1–8.

[119] B. B. Elallid, H. E. Alaoui, and N. Benamar, "Deep reinforcement learning for autonomous vehicle intersection navigation," *arXiv preprint arXiv:2310.08595*, 2023.

[120] W.-L. Chen, K.-H. Lee, and P.-A. Hsiung, "Intersection crossing for autonomous vehicles based on deep reinforcement learning," in *2019 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. IEEE, 2019, pp. 1–2.

[121] D. Xu, Y. Chen, B. Ivanovic, and M. Pavone, "Bits: Bi-level imitation for traffic simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 2929–2936.

[122] S. Schestakov, P. Heinemeyer, and E. Demidova, "Road network representation learning with vehicle trajectories," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2023, pp. 57–69.

[123] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2636–2645.

[124] D. Isele and A. Cosgun, "Selective experience replay for lifelong learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[125] R. Raileanu and T. Rocktäschel, "Ride: Rewarding impact-driven exploration for procedurally-generated environments," *arXiv preprint arXiv:2002.12292*, 2020.

[126] Z. Yu, "Fair balance: Mitigating machine learning bias against multiple protected attributes with data balancing," 2021.