HIGHWAY MERGING CONTROL USING MULTI-AGENT REINFORCEMENT LEARNING: EXPLORING CENTRALIZED AND DECENTRALIZED SCHEMES

by

ALI SAEED IRSHAYYID

A dissertation submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL AND COMPUTER ENGINEERING

2024

Oakland University
Rochester, Michigan

Doctoral Advisory Committee:

Jun Chen, Ph.D., Chair
KaC Cheok, Ph.D.
Micho Radovnikovich, Ph.D.
Darrell Schmidt, Ph.D.

To my father, my mother, my siblings Ameer, Mohammed,

Tabarak, Hussein and Zain,

Without your unconditional love and support, this would

have never been possible. This is for you.

ACKNOWLEDGMENTS

The author wishes to express his profound gratitude and heartfelt appreciation to Jun Chen, Ph.D., my esteemed advisor, for the invaluable mentorship, support, and insightful guidance provided throughout this academic journey. Your expertise and high standards have consistently challenged me to push beyond my limits and strive for excellence. Your mentorship has been instrumental in shaping both this dissertation and my growth as a researcher, instilling in me a rigorous work ethic and a passion for quality research. Thanks are also due to committee members KaC Cheok, Ph.D., Micho Radovnikovich, Ph.D., Darrell Schmidt, Ph.D. for their contributions to this dissertation. Likewise, acknowledgment should also be extended to peers Zhaodong Zhou, Christopher Rother, Hussein Alawsi, and Luke Nucalaj.

Last but not least, I want to thank the amazing friends I made while studying abroad. Their support and understanding during this journey meant the world to me.

Ali Saeed Irshayyid

ABSTRACT

HIGHWAY MERGING CONTROL USING MULTI-AGENT REINFORCEMENT LEARNING: EXPLORING CENTRALIZED AND DECENTRALIZED SCHEMES

by

Ali Saeed Irshayyid

Adviser: Jun Chen, Ph.D.

This dissertation addresses critical challenges in autonomous vehicle (AV) control, focusing on complex highway merging control during lane reduction using multi-agent reinforcement learning (MARL). Both centralized and decentralized approaches are presented. For the centralized approach, Proximal Policy Optimization based approach is employed to learn optimal merging policies for a fixed number of vehicles in two platoons. To scale up for the large number of AVs in a typical highway scenario, a decentralized approach is investigated, where each AV acts independently based on local observations. Furthermore, as the number of AVs in a traffic flow can vary, a self-attention network is used to handle the varying number of AVs. Several reward functions are explored and compared, including global speed, local speed, fuel consumption, and ride comfort. Novel quantitative metrics are introduced to evaluate the fairness and efficiency of the learned merging strategies. Both proposed MARL approaches consistently outperform benchmark RL method and a rule-based zipper merge strategy across various metrics, including up to 60.14% improvement in traffic flow at higher speeds, among many other advantages. Finally, the generalizability of the framework is demonstrated by training the MARL model using low speed scenario and testing the learned policy using high speed scenario.

TABLE OF CONTENTS

TABLE OF CONTENTS—Continued

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Autonomous vehicles (AVs) promise major benefits in terms of safety, efficiency, and accessibility [1–4]. However, developing reliable control policies for AVs remains an immense challenge [5]. A key difficulty involves handling scenarios, where AVs must interact safely and efficiently both among themselves and with human drivers having diverse driving styles [6]. Highway merging maneuvers, such as on-ramp merging, road reduction, and lane changes are among the most complex and safety-critical scenarios for AVs [7]. Road reduction scenarios, where multiple lanes merge into fewer lanes, present significant challenges for traffic flow and safety. These bottlenecks often lead to congestion, increased travel times, and a higher risk of accidents. Highway merging maneuvers is challenging even for human drivers [8] and is responsible for around 40-80% of congestion in the United States [9].

However, as AVs become more prevalent, there is an opportunity to address these challenges through intelligent coordination and control strategies. Improper merging can lead to collisions, traffic disruptions, and reduced throughput, undermining the potential benefits of AVs [10]. Developing effective merging control strategies for AVs poses several key challenges. One of which is the inherent uncertainty in predicting the actions of other vehicles, particularly those driven by humans [11–13]. Human drivers exhibit a wide range of behaviors and decision-making styles, which can be influenced by factors such as experience, age, and cultural background [14]. AVs must be able to anticipate and respond to these diverse behaviors, while also adhering to traffic rules and ensuring the safety of all road users [15–17]. Another challenge lies in the complexity of the merging task itself. Merging scenarios often involve coordinating multiple AVs

and human-driven vehicles simultaneously, requiring sophisticated multi-agent control algorithms [18, 19]. Cooperation and communication between vehicles can improve coordination with the cost of additional complexities. Additionally, merging decisions must be made in real-time, taking into account the dynamic traffic conditions, vehicle kinematics, and safety constraints [20].

Several approaches have been proposed in the literature to address the challenge of AV highway merging [21, 22]. These methods can be broadly categorized into rule-based approaches, intent modeling of the host vehicle, and optimization-based techniques. Rule-based approaches for AV merging depend on predetermined sets of if-then conditions or heuristics to make decisions. These decisions are usually based on factors such as vehicle positions, speeds, and gap sizes. For example, the authors in [23] introduced a cooperative merging strategy that relies on mainline vehicles slowing down to create sufficient gap space for on-ramp vehicles to merge. In [24], the researchers employed gap-acceptance theory and driving rules to model the decision-making process for on-ramp merging in urban expressway scenarios. Furthermore, work conducted by [25] developed a slot-based merging algorithm, which determines a slot's occupancy status based on the speed, position, and acceleration or deceleration behavior of mainline vehicles. Similarly, [26] used a slot-based approach, identifying potential merge slots between mainline vehicles. The approach evaluates the feasibility of each slot using a rule-based planner, and the ego-vehicle attempts to merge into the most suitable slot using a motion planner.

Rule-based techniques can manage simple traffic issues, but they may not be able to handle complex ones. This limitation has motivated intent modeling approaches, which aim to predict the behavior of the host vehicle to make merge decisions. Authors of [27] proposed a probabilistic graphical model to estimate the host vehicle's intent to yield based on features including its velocity and time-to-arrival at the merge point, assuming a fixed merge point. Intent modeling can provide insight into vehicle behaviors,

but it cannot capture the complexity of multi-vehicle interactions. Researchers have also examined optimization-based approaches that formulate the merging problem as an optimal planning and control planning problem. For example, [28] proposed a centralized optimal control scheme for coordinating merging vehicles, assuming the availability of vehicle-to-vehicle communication, and researchers in [29] used model predictive control (MPC) for decentralized merging. While optimization-based approaches can generate optimal merging trajectories, they face challenges in real-world implementation because of model accuracy and real-time computation issues as the number of interacting vehicles increases [30, 31].

To address these limitations, recent research has explored the application of deep reinforcement learning (DRL) to learning AV merging policies that map observations to actions [7, 32]. DRL has emerged as a robust learning-based approach that combines the strengths of RL with deep neural networks, and it has attracted significant attention in recent years due to its ability to achieve remarkable performance in complex tasks, e.g., surpassing human performance in playing board games such as Go, Chess, and Shogi [33, 34]. By leveraging deep multi-layer neural networks as policy approximators, DRL can handle large state and action spaces [35], making it well-suited for applications such as automated merging in AVs. The learning capacity of DRL is enhanced by its ability to learn from interactions with the environment over millions of time steps during training, eliminating the need to explicitly model the environment's complex dynamics. Such a model-free approach allows DRL agents to learn optimal actions that maximize long-term rewards, even in highly complex environments. With the increasing availability of affordable high-performance computing resources for training, DRL has found successful applications across various domains, demonstrating near-optimal control performance compared to traditional methods like model predictive control (MPC) [36] while requiring minimal computation time during deployment.

Building upon the challenges and potential presented by AVs in highway merging scenarios, this dissertation addresses critical challenges AVs control, focusing on complex highway merging scenarios, particularly road reduction. The research aims to develop and evaluate DRL approaches for AV merging control, investigating both centralized and decentralized multi-agent reinforcement learning (MARL) methods. The primary contributions of this dissertation are:

- Applying RL to highway merging control due to lane reduction, presenting all of Centralized Training and Centralized Execution (CTCE), Decentralized Training and Decentralized Execution (DTDE), and Centralized Training and Decentralized Execution (CTDE) Multi-agent RL frameworks.

- Developing a MARL CTDE framework capable of handling a variable number of agents, addressing the limitations of current approaches.

- Exploring multiple reward models, including global speed optimization, local speed optimization, fuel efficiency, and ride comfort, to comprehensively evaluate merging strategies.

- Introducing quantitative metrics to assess the fairness and efficiency of learned merging behaviors.

The remainder of this dissertation is organized as follows. Chapter 2 provides background on RL, MARL, and formulates the AV merging problem. Chapter 3 presents a comprehensive literature review of related work on RL for autonomous vehicle control, focusing on different merging scenarios, algorithms, action spaces, state spaces, and reward function designs. Chapter 4 presents a case study on cooperative platoon merging control using a CTCE approach. Chapter 5 explores a CTDE multi-agent RL framework for highway merging scenarios with a dynamic number of agents. Chapter 6, further evaluates

CTDE approach by presenting the DTDE approach for the highway merging scenarios with a dynamic number of agents. Finally, Chapter 7 concludes the dissertation and discusses directions for future research.

CHAPTER TWO

PRELIMINARIES AND PROBLEM FORMULATION

2.1 Preliminaries on Reinforcement Learning

This section briefly reviews the fundamentals of RL. For more details, please refer to [35]. At its core, RL algorithms are designed to solve problems that can be modeled as Markov decision processes (MDPs), a mathematical framework that describes how agents interact with an environment to select the most suitable actions based on observations for maximizing a long term return [37].

In an MDP, the environment is represented by a tuple $(\mathbf{S}, \mathbf{A}, p, r, \gamma)$, where $\mathbf{S}$ is the state space, $\mathbf{A}$ is the action space, p is the transition probability matrix, r is the immediate reward, and γ is the discount factor [35]. The agent interacts with the environment over multiple time steps, aiming to learn a policy, π, that maximizes the cumulative discounted rewards, i.e.,

$$\max_{\pi} J(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{2.1}$$

RL algorithms generally fall into two main categories: value-based methods and policy-based methods. Value-based methods focus on learning the value function, which estimates the expected return from a given state, and deriving a policy from this function. Policy-based methods, on the other hand, directly learn the policy without explicitly computing/estimating the value function [35].

One effective approach that combines the strengths of both value-based and policy-based methods is the actor-critic methods [38]. This approach combines two key components: an actor, which learns the policy, and a critic, which evaluates the policy's performance. The actor-critic method aims to simultaneously improve the policy and its value estimation, leading to more efficient learning.

Actor-critic algorithms can be implemented using various approaches. One prominent method is the policy gradient technique, which directly optimizes the policy by estimating the gradient of expected cumulative rewards with respect to the policy parameters [39]. In policy gradient methods, the actor objective function, J, can be defined as:

$$J(\pi_\theta) = E_{\pi_\theta}\left[\sum_t \log \pi_\theta(a_t|s_t) * A_t\right], \tag{2.2}$$

where $\pi_\theta(a_t|s_t)$ represents the probability of taking action a_t in state s_t under the policy parameterized by θ. The objective is to change θ to increase the probability of choosing actions that lead to higher advantages. The advantage function, A_t, is defined as:

$$A_t = Q_\pi(s_t, a_t) - V_\pi(s_t), \tag{2.3}$$

where $V_\pi(s_t)$ is the expected return from s_t and acting according to policy π and $Q_\pi(s_t, a_t)$ is the expected return starting from s_t and taking action a_t (also termed as Q-value).

The advantage function (2.3) quantifies the benefit of choosing action a_t at state s_t compared to the expected return of following the policy π from state s_t. The advantage A_t can be approximated using the temporal difference (TD) error:

$$A_t = r_t + \gamma V_\pi(s_{t+1}) - V_\pi(s_t), \tag{2.4}$$

where r_t is the immediate reward received. The critic network, parameterized by weights ϕ, is used to estimate $V_\pi(s_t)$, where ϕ is updated to minimize the following loss function:

$$L(\phi) = E\left[\left(r_t + \gamma V_\pi(s_{t+1}) - V_\phi(s_t)\right)^2\right], \tag{2.5}$$

Both the actor parameter θ and critic parameter ϕ are updated after collecting the experiences or trajectories, training episodes, from the environment.

2.2 Preliminaries on Multi-Agent Reinforcement Learning

The application of MARL to multiple AV highway merging has evolved significantly in recent years, with various approaches exploring different levels of centralization and decentralization among agents. These approaches can be broadly categorized into three main schemes: Centralized Training Centralized Execution (CTCE), Decentralized Training Decentralized Execution (DTDE), and Centralized Training Decentralized Execution (CTDE).

CTCE involves a central learner that gathers information from all agents to learn a global policy, which helps mitigate issues of partial observability and non-stationarity. When designing a CTCE algorithm, a fixed maximum number of agents must be defined, and if the environment has a fewer number of agents, a padding vector is used, as shown in Fig. 2.1 where the input to the policy network contains n blocks and the output is a vector of length n, with n being the maximum number of AVs . In [40] a centralized RL approach is introduced to control multiple AVs in a simple merging scenario. While the authors did not explicitly state a maximum number of controlled vehicles, their approach used a centralized policy to control a small fraction of AVs. This approach is expanded in [41], where a centralized RL is applied to both small and large-scale scenarios. For example, the centralized RL approach was scaled up for the I-696 highway merge scenario to control up to 30 AVs. This test demonstrated the potential for centralized RL to handle larger, more complex traffic situations. However, challenges with the growing state and action spaces was noted. Specifically, the authors identified three main challenges. First, the state and action spaces grow exponentially with the number of controlled vehicles, significantly increasing computational demands. Second, there is a significant time lag between an AV's action and its effect on the system's average speed and outflow, resulting in delayed reward feedback for the learning algorithm. Finally, in larger networks, the relationship between the centralized agent's actions and the overall system performance becomes less direct,

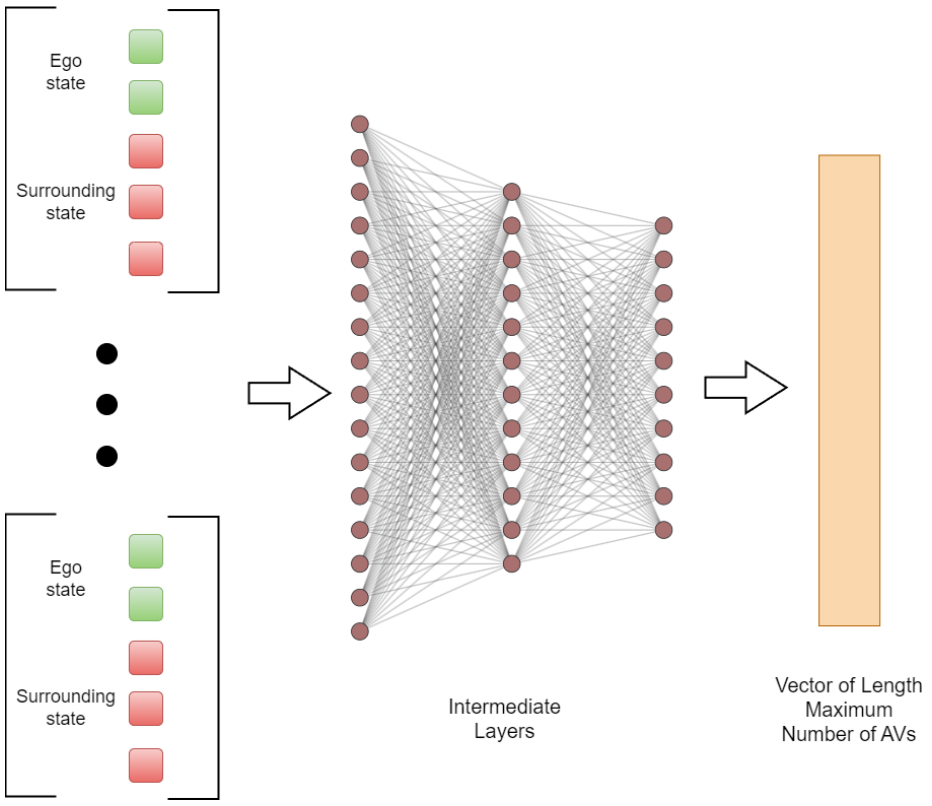

Figure 2.1: CTCE RL: Centralized policy processes all vehicles' observations and outputs all vehicles' actions. The input block is repeated by n times, where n is the maximum number of AVs, and the output is a vector of the length same as the maximum number of AVs.

resulting in noisier reward signals. Which is because many vehicles in the network are more influenced by their own local conditions than by the centralized controller. These centralized approaches, as mentioned above, allow for coordination between controlled AVs, potentially leading to cooperative agents. However, they often lack of scalability in environments with continuous state and action spaces, which are common in driving scenarios. Due to the "curse of dimensionality," the learned policy tends to be sub-optimal [42].

To address the scalability limitation of CTCE, researchers have proposed the DTDE approach, which is a fully decentralized framework where both the training and execution phases are independent. Each agent independently learns its policy using only its local observations and rewards, without access to global state information or the actions of other agents. See Fig. 2.2. This approach differs from CTCE in that it does not rely on a central controller, allowing for greater scalability and flexibility in the number of agents. However, DTDE may struggle with coordination among agents and can lead to suboptimal solutions in scenarios requiring cooperative behavior. Several studies have explored the use of DTDE in highway merging scenarios. In [43], a multi-agent Q-learning was employed in a simplified two-vehicle merging scenario. In particular, a taper-type highway on-ramp with only two vehicles was considered, where one vehicle tries to merge and the other one stays in-lane. The authors proposed a simplified mathematical model to simulate fundamental interactions between the merging vehicle and the in-lane traffic vehicle, essentially creating a two-player grid-world with finitely discretized state and action spaces. A subsequent study extended this approach to continuous state and action spaces, using a decentralized multi-agent Deep Deterministic Policy Gradient (DDPG) algorithm to control vehicle acceleration [44].

DTDE-based merging control, as discussed above, offer improved scalability compared to CTCE methods, as they don't require a central controller and can adapt to varying numbers of agents. However, they face several challenges: (1) DTDE approach may not achieve the optimal coordination among agents, particularly in scenarios requiring highly cooperative behavior; (2) The scalability of DTDE approach, though better than CTCE, may still be an issue when the number of agents is large [45]. This is primarily due to the lack of experience sharing among agents, which introduces additional sample complexity during training; and (3) As each agent treats all other agents as part of

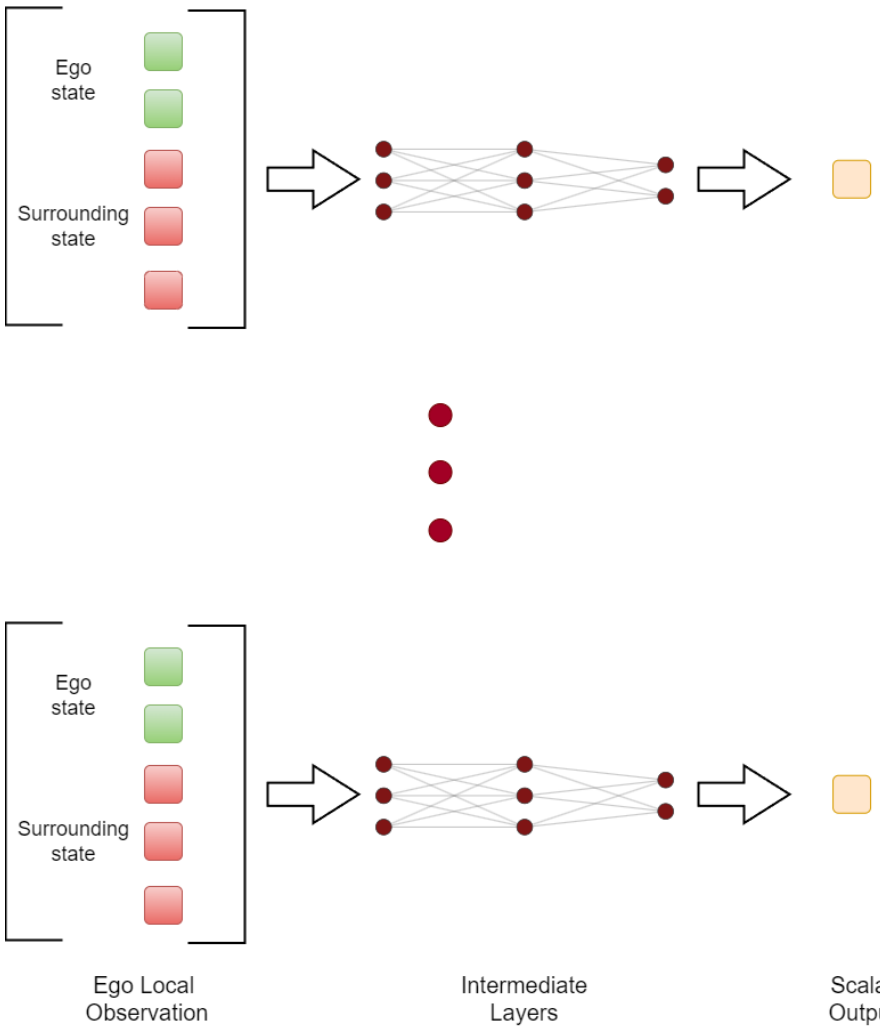

Figure 2.2: DTDE RL: Each ego AV processes only local observations and outputs single action.

the environment, whose dynamics are constantly changing since all agents are learning concurrently, Markov stationary property can be violated, leading to instability [46].

The third major approach in MARL for AV highway merging is CTDE. This approach is considered more practical as it attempts to balance the advantages of both CTCE and DTDE approaches. CTDE is particularly suited to real-world applications, leveraging the benefits of centralized learning while maintaining the scalability and

flexibility of decentralized execution. Typically, CTDE algorithms utilize an actor-critic architecture, which separates policy and value estimations into distinct actor and critic networks. The critic network is used exclusively during training to enhance the gradient updates of policy networks, and multiple agents (i.e., actor network) are trained collectively with a shared critic. After training, only the independent actors are necessary for execution, enabling the agents to operate without further communication [47]. CTDE addresses some of the limitations of both CTCE and DTDE. For example, it doesn't require a central controller during execution, which improves scalability and reduces communication overhead in real-world applications compared to CTCE. At the same time, CTDE mitigates the coordination issues often faced by DTDE approaches by incorporating global information during the training phase.

2.3 AV Merging Problem Formulation

The AV merging problem in road reduction scenarios presents a critical challenge for traffic management and safety. This dissertation explores two distinct yet complementary merging scenarios: platoon merging and individual vehicle merging. Both scenarios share key elements that must be addressed. First, state representation is crucial for capturing relevant traffic information, including vehicle positions, speeds. Second, the action space must define possible decisions for AVs. Third, reward design is essential for incentivizing desired merging behavior that balances safety, efficiency, and passenger comfort. Lastly, the learning approach must focus on developing effective merging strategies that can adapt to dynamic traffic conditions. The first case study (Chapter 4) examines platoon merging under CTCE control. This scenario represents a more structured, cooperative environment with a fixed number of vehicles. In contrast, the second case study (Chapter 5) investigates individual vehicle merging with CTDE control, simulating a dynamic environment with a variable number of independent agents. This latter

scenario more closely resembles the complex, realistic highway merging situations. These scenarios differ significantly in their problem formulations. The control scheme shifts from centralized in the platoon scenario to decentralized in the individual vehicle scenario. The number of agents changes from fixed to variable. The state and action spaces evolve from platoon-level considerations to individual vehicle-level decisions. By exploring both scenarios, this research aims to develop effective reinforcement learning-based strategies across a spectrum of road reduction situations. This progressive development allows for the extraction and understanding of critical information about merge scenes, enabling more intelligent control strategies.

CHAPTER THREE

LITERATURE REVIEW ON RL-BASED AV MERGING AND PLATOONING

This chapter provides a comprehensive review of reinforcement learning (RL) approaches for AV control, with a particular focus on highway scenarios. The review is structured to systematically examine key aspects of RL applications in this domain.

3.1 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 3.3.

3.1.1 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, 60]. The highway environment is typically simulated using tools like SUMO (Simulation of Urban MObility) [48], VISSIM or custom cellular automaton models [64], 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 [60], where the authors also studied an agent making a single lane change on a highway but used a deep Q-network (DQN) algorithm. Similarly, [84] used a double deep Q-network (DDQN) algorithm incorporated with a model of human lane-changing decisions as a safety

Table 3.1: 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
[50]	Platoons at non signalized intersection	AV and its surrounding motion values	Discrete set of the acceleration	SUMO	PPO	Average speed
[51]	Platoon in freeway	Leader, ego vehicle, and its preceding vehicle	Acceleration (continuous)	SUMO	CommPPO	Fuel consumption
[52]	Highway on-ramp	Relative speed and position of surrounding vehicles	Discrete high level decisions	Highway-env [53]	Novel MARL	Collision, merging cost, speed, and time headway
[54]	Highway on-ramp	Relative speed and position of surrounding vehicles and the Road layout	Discrete high level decisions	Highway-env	MARL	Safety
[55]	Highway on-ramp	Traffic state and vehicle motion information	Three discrete actions	SUMO	STDQN	Efficiency, goal, driving comfort, and safety
[56]	Highway navigation	Self, connected agents and infrastructure observations (location, velocity and acceleration)	Discrete high level decisions	CARLA [57]	Constrained MARL	Maximize speed of every agent
[58]	Lane changing	Relative speed and position of surrounding vehicles	Discrete high level decisions	TORCS	Discrete high level decisions	Collision, comfort, speed, and time headway
[59]	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
[60]	Lane change	The relative distance to the surrounding vehicles and detected lane.	Continuous steering and longitudinal speed.	Python	DDPG	Security, comfort, and efficiency
[61]	Lane change	The relative distance and absolute speed of front vehicles.	To change lane or to stay	MATLAB & VISSIM [62]	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
[63]	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
[64]	Lane changing	Ego and surrounding vehicles dynamics, and road curvature	Continuous yaw acceleration	Not mentioned	DDPG	Large action changes, Maneuvering time, and lateral deviation
[65]	Lane changing	Speed and position of the ego vehicle and its surrounding vehicles	Discrete high level decisions	Udacity simulator	DQN	Safety and Speed
[66]	Highway work zone	Speed and acceleration grid maps and neighboring vehicles information	Acceleration and deceleration	VISSIM	SAC	Safety, comfort, and traffic flow
[67]	Platoon maneuvers	Speed, acceleration and position of ego, front and rear vehicles	Brake, throttle and steering	CARLA	SAC	Safety, comfort, smoothness and headway

Table 3.2: 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]	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
[69]	Platoon forming	Speed, acceleration and headway	Full-speed headway	Not mentioned	DDPG	Velocity and headway
[70]	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
[71]	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
[72]	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
[73]	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
[74]	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
[75]	Highway merging	Egos local information, position, motion, & Gaps between surrounding vehicles	Discrete actions	NGSIM	A2C	Speed and successful merge
[76]	Highway merging	Egos kinematics and position and speed of surrounding vehicles	Continuous acceleration	SUMO	A2C	Collisions, braking, and successful merge
[77]	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
[78]	Lane changing	Egos kinematics and position & speed of surrounding vehicles	Discrete actions	Julia [79]	DQN	Safety, efficiency, and lane changing
[80]	Highway merging	Ego vehicle, road geometry, and surrounding traffic	Discrete actions	Custom simulator	DQN	Average speed
[81]	Highway merging	Distance to the merge point, velocity, acceleration, and cooperation parameter	Discrete actions	Julia	DQN	Collision, goal, and time penalty
[43]	Highway on-ramp	Closing gap and speed, time to position, and position of the ego	Discrete actions	Python	Q-learning	Safety, comfort, and energy
[82]	Platoon coordination management	Platoon state, states of AVs in the platoon, and environment state.	Discrete actions	PLEXE [83]	DRG-SP	Driving strategy
[84]	Lane changing	Velocity and relative distance of surrounding vehicles, and ego kinematics.	Discrete actions	CARLA	DDQN	Safety, speed, and lane centering

Table 3.3: Studies organized by automated driving scenario

Scenario		References
Highway Lane Change	Simple surrounding	[48, 60, 84]
	Complex surrounding	[58, 64, 65]
	Lane merging	[66, 77, 78, 81]
Highway Ramp Merging	Multi-agent merging	[43, 52, 54, 55, 59]
	Cooperative merging	[71–74, 76]
	Using real-world traffic data	[63, 75, 85]
Platooning	Cooperative speed control	[50, 67–69]
	Safe coordination	[51, 56, 82]
	Platoon joint	[70]

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. 3.1. Ref. [64] 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). Same authors [65] 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, the study [58] considered multiple AVs and 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, the authors in [77] 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.

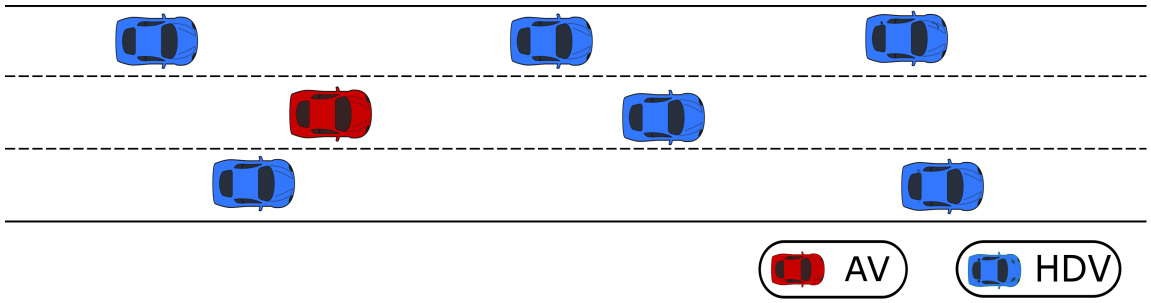

Figure 3.1: Illustration of the Lane-changing highway scenario.

In addition to discretionary lane changes, mandatory lane changes such as merging have been studied. In Ref. [66], a highway work zone merge scenario is simulated where the 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. Research presented in [78] 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 [50,64,76]. On the other hand, study presented in [81] proposed a Cooperative Intelligent Driver Model (C-IDM) to simulate the longitudinal driving behavior of the main lane vehicles . 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. The work in [48] specifically focused on congested highways to test discretionary lane changes in dense traffic. In [64], the simulated scenario involves one 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. The methodology outlined in [65] leveraged an open source simulator to evaluate lane changes for passing slower vehicles. Ref. [77] included a finite state machine for higher-level planning in their hybrid approach. The work in [78] 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.

3.1.2 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

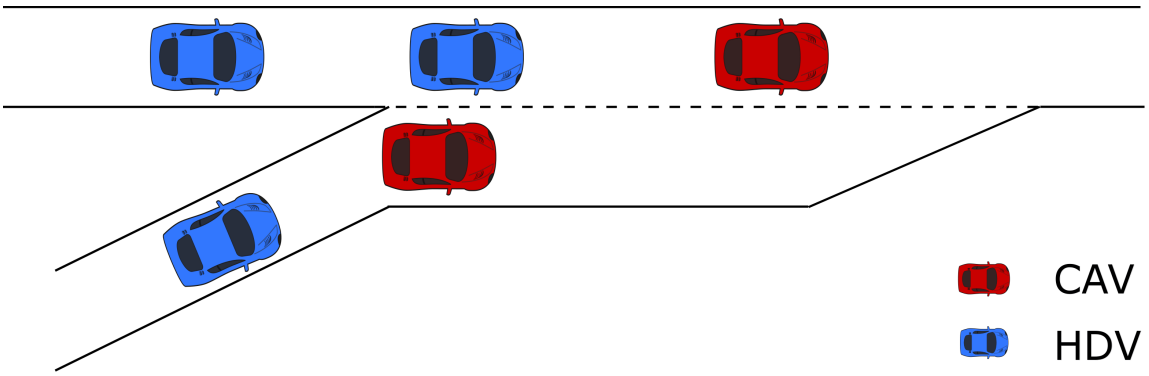

Figure 3.2: On-ramp traffic scenario in the presence of HDVs.

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. 3.2 depicts such a scenario.

Several studies have employed multi-agent reinforcement learning (MARL) algorithms for this merging task. Research presented in [43] presents a MARL 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, paper [52] 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. Work in [54] extended Chen et al.'s, [52], 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. In [55], authors 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. In [59] DRL is also adopted 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. Investigation from [71, 72] 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, study in [73], 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. In [76], researchers 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. The following work, [74], 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. Analysis conducted in [75] extracted over 400 real highway merging scenarios from the NGSIM dataset [85] 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, Ref. [63] 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.

3.1.3 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 [69], 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. Authors of [67] 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. In [68] RL is applied 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. Using RL for autonomous lead vehicles to enhance intersection flow in mixed traffic was studied by [50].

Ensuring safe and efficient coordination is another key challenge. Authors in [56] 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. Research

presented in [51] 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. Using DDPG, the authors of [70] created a unified DRL solution that manages split/join maneuvers, gap regulation, and speed control all within the same 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, [82] 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

Table 3.4: Studies organized by RL algorithms

RL Algorithms		References
Single Agent RL	DQN and DDQN	[61, 63, 65, 73, 78, 80, 81, 84]
	PPO	[48, 50]
	DDPG	[60, 64, 69, 70, 76]
	Others (SAC, A2C, etc.)	[43, 49, 68, 75, 77, 82]
Multi-Agent RL	Centralized training	[56, 67, 74]
	Decentralized training	[52, 58, 59, 71, 72]
Curriculum learning		[51, 64, 74, 78]
Representation learning		[55, 56, 63, 74, 80]

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.

3.2 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 [86, 87]. 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 3.4.

3.2.1 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 [61, 65, 73, 78, 80, 81]. In [61], 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 [65] learns to map traffic states to optimal lane change actions. Actions are selected using an ε-greedy, [35], 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 [88], dueling DQN [89], prioritized experience replay [90], 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 [65], 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 the agent will not learn to not select unsafe actions because these unsafe actions will not be executed. Similarly, reference [63] 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 [80], 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 [48, 50], DDPG, [60, 64, 69, 70, 76], SAC [68], Vanilla policy gradient [49], and A2C [75]. For instance, the framework proposed by [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, the methodology outlined in [68] 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 [50], authors propose using PPO with an adaptive KL penalty to optimize the policy for controlling multiple AVs at a non-signalized intersection.

3.2.2 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 [91]. 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. In [67], a novel DRL algorithm named platoon sharing deep deterministic policy gradient algorithm (PSDDPG) is proposed 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. In addition to decentralized value critics, the framework proposed by [74] employs a centralized action-value critic. 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, employ a multi-agent advantage actor-critic (MA2C) with shared parameters and multi-objective rewards. As an example, two studies [52, 58] employ a multi-agent advantage actor-critic (MA2C) with shared parameters and multi-objective rewards. Similarly, [59], 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, the study in [56] adapt constrained MARL with decentralized training. A decentralized A2C variant with independent actors and critics is proposed in [71, 72]. While authors in [51] presented 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 [59, 71], but this can fail in complex cooperative settings. Research in [74] employed a counterfactual baseline in the centralized critic to address this issue [92]. 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.

3.2.3 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 [78], 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.In [51], the proposed framework 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 [74], the curriculum learning is employed

when designing the reward function. The agent started with individual rewards before adding multi-agent joint rewards. Additionally, the study in [64] took 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

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.

3.2.4 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 [93].

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 [43, 48–52, 54, 61]. 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, 69].

On the other hand, papers like [56] 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 [94]. Also, convolutional neural networks (CNNs) are widely adopted to process visual inputs and extract hierarchical spatial features [74, 80]. This takes advantage of CNNs abilities for translation invariance and local connectivity. Attention mechanisms are being increasingly incorporated [55] 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, [63], 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.

3.3 Action Space

The research papers examined in this review employed a variety of action spaces, as summarized in Table 3.5, 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 [95]. This supports smoother driving behavior. However, continuous spaces can be more challenging to learn due to their high dimensionality [96]. Discrete action spaces, on the other hand, simplify the learning problem but reduce control flexibility [97]. Hybrid approaches combining both continuous and discrete actions are a useful compromise, with discrete actions for high-level decisions and continuous actions for lower-level control.

3.3.1 Continuous Action Spaces

Several studies utilized continuous action spaces directly controlling longitudinal acceleration and lateral steering angle [63, 76, 77]. 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 [77]. In contrast, [64] 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 [67], 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

Table 3.5: 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
[50]	Continuous	1	None
[51]	Continuous	1	DRAC
[52]	Discrete	5	Novel priority-based safety supervisor
[54]	Discrete	5	None
[55]	Discrete	3	None
[56]	Discrete	5	Control barrier function
[58]	Discrete	5	None
[59]	Discrete	5	Time-to-collision (TTC)
[60]	Continuous	2	None
[61]	Binary	1	None
	Discrete	6	None
[63]	Continuous	2	None
[64]	Continuous	1	None
[65]	Discrete	3	Rule-based constraints
[66]	Continuous	1	VISSIM basic safety constraints
[67]	Continuous	2	None
	Discrete	3	None
[68]	Continuous	1	None
[69]	Continuous	1	None
[70]	Continuous	1	None
[71]	Discrete	5	None
[72]	Discrete	5	None
[73]	Discrete	3	MPC constraints
[74]	Discrete	7	Masking
[75]	Discrete	4	Low-level controller constraints
[76]	Continuous	1	None
[77]	Continuous	2	Finite state machine (FSM)
[78]	Discrete	5	Intelligent Driver Model (IDM)
[80]	Discrete	3	None
[81]	Discrete	7	None
[43]	Discrete	5	None
	Discrete	25	None
[84]	Discrete	5	Human decision-making model

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 [50], the RL agent only controls the acceleration of AVs to be within a specific range of maximum deceleration and maximum acceleration values. In contrast, [51] used a continuous action space between -3 m/s^2 and 3 m/s^2 for acceleration. Similarly, in [60], 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 [68, 70], 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, [68] uses $[-3\ m/s^2, 2\ m/s^2]$ while [70] 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.

3.3.2 Discrete Action Space

Many works selected discrete speed control actions that are converted to smooth accelerations by lower-level controllers [80, 81, 84]. 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 [52, 72, 75]. 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 [72], 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. 3.3 for more details. In [52, 58], 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 [55], high-level discrete actions, such as keep straight, turn left, and turn right, are used. In a similar manner, researchers employed high-level discrete actions for the lane change scenario, allowing the vehicle to choose between merging or staying in the same lane [61]. In [56], 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

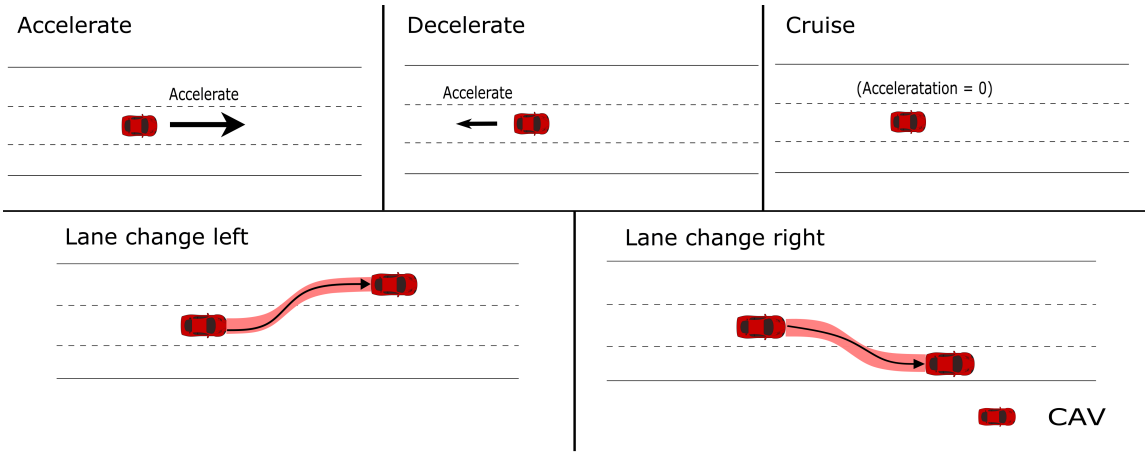

Figure 3.3: High level action space.

level action space is also used for platoon coordination management, [82] used a high level action space taken by the captain AV which include accepting/rejecting requests from AVs to join the platoon. Lastly, in [59, 84], 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 [65], 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, [43] 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 [73], 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 [78], 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, 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 [69]. In [69], 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.

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.

3.3.3 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 5.2. Other studies handle safety through basic constraints enforcement by the simulating the environment during training [66, 78], or by designing and employing a separate module to classify and override unsafe actions [51, 52, 56, 58, 60, 65, 73, 77]. 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. Two studies [48, 77] evaluate candidate actions and replace unsafe ones with safer alternatives. Ref. [**?**] used 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, [52], 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:

$$p_i = \alpha_1 p_m + \alpha_2 p_d + \alpha_3 p_h + w_i, \tag{3.1}$$

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, the study in [60] 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, [51] 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. Two studies [58,73] 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 [65] 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, [84] 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, [56] uses a parallel safety module based on control barrier functions (CBFs), [99], 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.

3.4 State Space

This section discusses the state space setups adopted in literature, which are also summarized in Table 3.6. 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

Table 3.6: 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
[50]	✓					2 (immediately preceding and following) vehicles
[51]	✓					2 (predecessor, and platoon leader) vehicles
[52]	✓					Nearest five Vehicles within 150 m
[54]	✓					Vehicles that are on the highway and on the merge ramp
[55]				✓		All surrounding vehicles within 20-60 m
[56]				✓	✓	3-5 CAV vehicles
[58]	✓					All detected vehicles
[59]		✓				All detected vehicles
[60]	✓					All detected vehicles
[61]	✓					3 (left, right, front)
[63]			✓	✓		2 (front, rear)
[64]	✓					immediate surrounding vehicles
[65]	✓					Vehicles within 60 m in front and 30 m rear
[66]	✓					3 (lead vehicle on current lane, and (lead and lag) vehicles in target lane)
[67]		✓			✓	Front vehicle
[68]	✓					2 (front, rear) vehicles
[69]	✓					All vehicles within its V2V communication range
[70]	✓					Immediate preceding vehicle
[71]		✓				All vehicles within its V2V communication range
[72]		✓				All detected vehicles
[73]		✓		✓		16 vehicles
[74]	✓					All vehicles within 100m in front and behind
[75]	✓		✓			4 (2 front and 2 rear) vehicles
[76]	✓					5 (2 leading, 2 following, and merging vehicle) vehicles
[77]		✓				4 (leading, lagging, front-of-leading, and front) vehicles
[78]	✓					The 8 closest vehicles within 30m
[80]	✓				✓	All vehicles on the on-ramp
[81]	✓				✓	4 (front of ego, behind merge point, front and rear of ego vehicle's projection on main lane)
[43]	✓					1 vehicle
[82]	✓					All AVs
[82]	✓					4 (two front, two rear) vehicles

the important properties of the environment at the current time step. A well-designed state space in RL should be compact, expressive, and general [100, 101]. 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 [48–51, 69, 70]. This provides 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 [43,50,70] to more extensive context including multiple lead and follow vehicles [48, 49, 55, 68, 69]. See Fig. 3.4. 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.

3.4.1 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 [50, 70, 84] and [69, 78], 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 [68]. Another difference is the incorporation of road geometry details, like lane curvature, to improve generalization [64, 82].

Most of the papers reviewed here have explored state space representations based on ego vehicle kinematics and surrounding vehicle information, seeking to balance

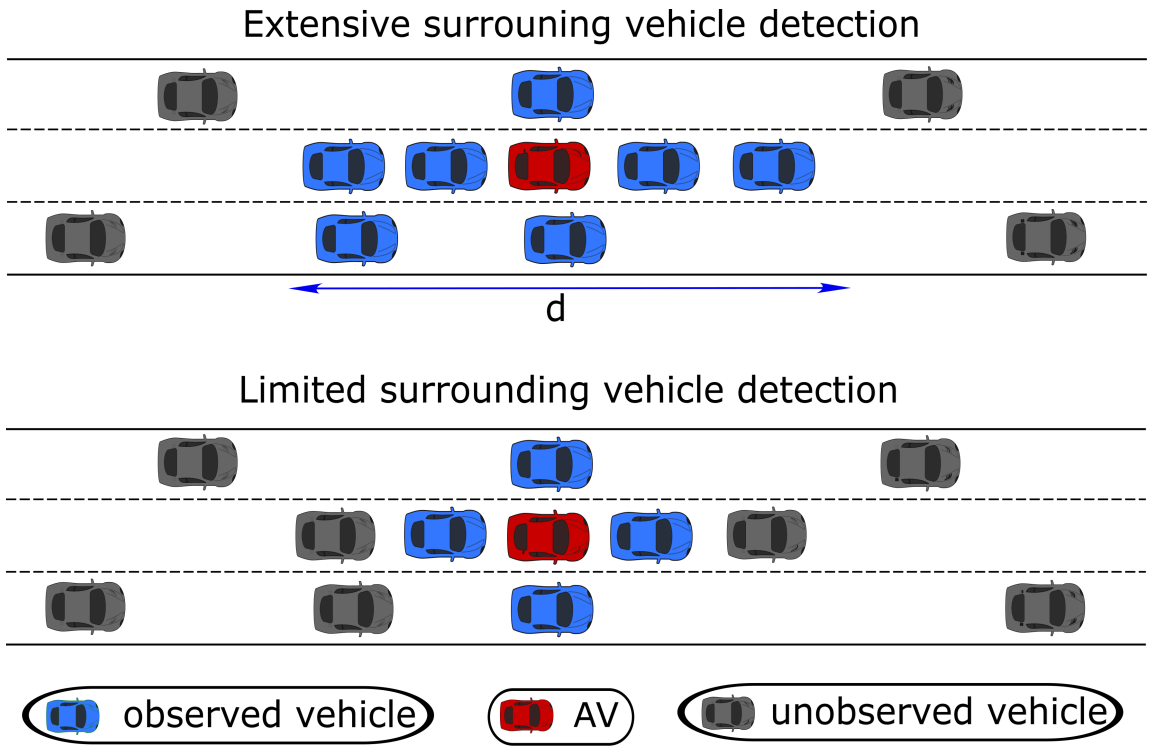

Figure 3.4: Limited vs extensive surrounding vehicle detection ranges. Where *d* represents the detection range of the AV.

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. Authors in [65] designed a grid encoding positions and speeds of nearby vehicles to support lane change behaviors. Additionally, in [55], 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, a study [80] used time-series grayscale images to capture vehicle dynamics and road geometry from a first-person visual perspective. In [52, 58], the authors took a hybrid approach, combining binary occupancy maps with relative vehicle positions and speeds processed 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:

$$e_{i,k} = d_{(i-1,i),k} - L - d_d, \tag{3.2}$$

  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 [69] or the relative positions of all agents [74]. 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 [74] or collaboration indications [81]. In [81], 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 [102].

3.4.2 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 [103].

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 [104]. 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 [105]. This increases engineering time and computing costs. Thirdly, interactions and dynamics are not explicitly modeled, so the agent must infer them from observations [106]. Lastly, platform-specific sensor differences complicate the transmission of policies. Several papers, such as [80] and [56], have chosen to represent the state space using raw sensor data. This study [80] 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, Research presented in [60] 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 [107–109], 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.

3.4.3 Temporal Information

Some recent papers have explored encoding temporal context and history into the state space for reinforcement learning in autonomous driving. An investigation [73] explored the use of a k-Markov approximation for incorporating historical observations. The findings revealed performance improvements when utilizing 2.4 seconds of prior data. The authors of [67, 77] tailor their state representation to different control tasks by using multiple frames of states, providing valuable time sequence information. Other studies [59, 71, 72] use Velocity-maps history that capture successive observations to incorporate temporal information into the state space. Meanwhile, work in [55] incorporate both spatial and temporal information, including the speed and position of CAVs and HDVs over multiple time steps. In [80], 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 [110]. To address the complexity of temporal data, extraction modules have been proposed to distill historical information into useful state representations. For instance, raw trajectory data was used to summarize interactive driving dynamics in [63], where an LSTM encoder was trained through supervised learning. Similarly, in [56], 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 [111].

However, challenges remain in constrained training of extractors and ensuring the distilled states sufficiently capture all critical environmental details [112]. 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.

3.5 Reward Function

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 3.7.

3.5.1 Safety

Safety is a paramount concern for AV, and it involves avoiding collisions or near-collisions with other vehicles or 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 [75, 76, 78, 81], while others treat any collision as a terminating state with a large negative reward [68–70]. Similarly, [49] uses a negative terminating reward for encountering terminating events such

Table 3.7: Studies organized by reward functions

Rewards		References
Safety	Speed-based penalty	[51, 70, 77]
	Distance penalty	[52, 56, 60, 61, 63, 67, 84]
	Action-based penalty	[48, 50, 52, 55, 56, 58, 72]
Efficiency	Speed-based	[48, 49, 51, 52, 58, 60, 63, 65, 68, 75, 77, 84]
	Social utility	[50, 55, 56, 59, 72, 74, 81]
	Position-based or maneuver-based	[73–76, 78, 80, 80]
Comfort	Jerk minimization	[43, 48, 51, 60, 70, 73, 76, 77]
	Control inputs smoothing	[55, 64, 77]
	Speed/distance tracking	[49, 51, 67, 84]
	Supplementary techniques	[73, 77]
Adaptability		[59]

as collisions or leaving the road. The safety reward can be represented as follows:

$$r_t(a, s_t) = -100 \times (1 - \mathbb{I}(E)), \tag{3.3}$$

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.

A more sophisticated way of calculating the safety penalty is used in Ref. [51]. 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:

$$r = \begin{cases} -\left(\frac{a}{a_{\max}}\right)^2 - 1, & \text{if gap} > L_{\text{threshold}} \text{ or } a > a_{\text{conflict}} \\ -\left(\frac{a}{a_{\max}}\right)^2, & \text{else} \end{cases}, \tag{3.4}$$

where gap is the headway distance of the ego vehicle, and a_{max} is the acceleration bound. On the other hand, [77] uses time to collision with a four seconds threshold to determine what is considered unsafe. Similarly, in [61], 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, [70] 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. [50] penalizes collisions with other vehicles or pedestrians, while [52, 58] penalize collisions with other vehicles or lane boundaries. Frequent lane changes is penalized in [55, 72], as they are associated with higher risks. Ref. [56] penalizes getting too close to other vehicles, while [60] incorporates collision avoidance and different expected lane-changing distances in its reward function. Ref. [63] 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, [52, 56, 58] 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. [67] 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 [73], safety is handled implicitly by the lower-level motion planning layer.

3.5.2 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, [55, 56, 60, 63] include components in their reward function that reward higher speeds of the ego vehicle. [51] penalizes the agent for having low speeds to avoid inefficient low-speed driving states. Similarly, [49, 68] 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, [73, 74, 76, 78, 80] provide positive rewards for successfully completing the merging maneuver. Ref. [75] rewards forward progress along the road. In [80], 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 [74]. For instance, [50] aims to maximize the average speed of all vehicles at the intersection, while [56] rewards maximizing the average speed of all CAVs. A social reward is included in [72] that accumulates the progress of all vehicles, while [81] uses a time penalty factor to incentivize reaching the goal position quickly. Similarly, [55] rewards maximizing the velocity of both CAVs and HDVs. Ref. [59] rewards optimizing social utility, which involves cooperation among AVs to achieve

socially desirable outcomes. The reward of vehicle, i, can be defined as

$$R_i(s,a) = \cos(\phi) \times r_i^{\text{ego}} + \sin(\phi) \times r_i^{\text{social}}, \tag{3.5}$$

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 [65] rewards driving as fast as possible down the highway. Finally, [77] rewards reducing the time taken to approach the target lane center, while [52,58] 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.

3.5.3 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.

3.5.3.0.1 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, [48, 51, 70, 73, 76, 77] impose absolute or squared penalties on the magnitude of jerk. In particular, [73] filters out jerks above a threshold so only large uncomfortable jerks are penalized, while [51] penalizes exceeding acceleration limits. [48] uses a comfort objective evaluated by the jerk in lateral and longitudinal directions,

while, Ref. [60] penalizes minimizing the angular velocity of the steering wheel and jerk, represented as:

$$R_{\text{comfort}} = k_w \times \dot{\theta}_w + k_a \times j, \tag{3.6}$$

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.

3.5.3.0.2 Smooth control inputs Maintaining smooth steering and throttle/braking control is also important. In this regard, [64] punishes large yaw rate and yaw acceleration for smooth lane changes,

$$r_s = -w_1 \times |\omega_{\text{yaw}}| - w_2 \times |a_{\text{lat}}|, \tag{3.7}$$

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 [77] penalizes large angular speeds of the steering wheel. Finally, [55] also penalizes acceleration, deceleration, and jerk that exceed a threshold.

3.5.3.0.3 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 [51] penalizes going slower than acceptable speeds. Similarly, [67] 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:

$$\text{Reward}_{\text{headway}} = \begin{cases} -100, & \text{if x} \leq 0, \\ -100(1 - \sqrt{(1-(x-1)^2)}), & 0 < \text{x} \leq 1, \\ 0, & \text{x} > 1. \end{cases} \tag{3.8}$$

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.

3.5.3.0.4 Supplementary techniques Some papers use supplementary techniques in addition to reward design to improve comfort. For instance, [73] handles collisions at a lower level so comfort can be prioritized in the reward, while [77] adapts weights based on proximity to the mandated lane change point.

3.5.4 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 [59] 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. Authors define an adaptation error (A_{error}) that explicitly rewards adaptability of the trained AVs to new scenarios. This is calculated as:

$$A_{\text{error}} = w_s \times (C) + w_e \times \left(1 - \frac{DT}{DT_{\text{max}}}\right), \tag{3.9}$$

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:

$$R_i(s,a) = R_{\text{ego}} + R_{\text{social}}, \tag{3.10}$$

where the ego reward is defined as:

$$R_{\text{ego}}(s,a) = \cos(\phi_i)\, r_i(s,a), \tag{3.11}$$

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:

$$R_{\text{social}} = \sin(\phi_i)\left[\sum_j r_{i,j}^{\text{AV}}(s,a) + \sum_k r_{i,k}^{\text{HV}}(s,a) + \sum_k r_{i,k}^{M}(s,a)\right] \tag{3.12}$$

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.

3.6 Summary

This chapter reviewed the state-of-the-art RL 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.

CHAPTER FOUR

AV MERGING CONTROL USING CENTRALIZED RL

This chapter presents a case study on cooperative platoon merging in a road reduction scenario using a Centralized Training Centralized Execution (CTCE) approach. We investigate the merging behavior of two platoons of connected and automated vehicles (CAVs) as they navigate a lane reduction. Vehicle platooning is a promising road management system to reduce congestion, fuel consumption, and accidents [113]. In a platoon, multiple partially or fully automated vehicles are arranged in a train-like formation, coordinated to move at the same speed while maintaining a desired inter-vehicle distance [114]. In some road situations, the platoon has to perform lateral transitional maneuvers essential for safety and driving efficiency, such as joining, merging, and leaving the platoon [115]. However, platoon merging for CAVs remains challenging due to various factors including multi-vehicle interactions, real-time control requirements, and potential interference from unintentional vehicles [116] [117] [118].

Several approaches have been proposed to address the challenges of platoon merging. Table 4.1 summarizes key articles that addresses platoon maneuver problems, including merging scenarios. For instance, the merging of heterogeneous vehicular platoons was studied in [119], where the authors concluded that the proposed controllers' performance is satisfactory, but a more complicated scenario is needed for testing. A distributed MPC was proposed to generate the merging trajectories, while a linear quadratic regulator (LQR) controller was designed to create a gap for the merging platoon. Reference [120] proposed a novel PID controller for heavy-duty vehicle platoon maneuvers, while the authors of [121] showed that using cooperative adaptive cruise control (CACC) for highway-merging scenarios improves traffic-flow stability and efficiency. However, the

proposed approach of [121] only considered longitudinal vehicle control, which means only vehicle speed would be automated using vehicle-to-vehicle communication, while the active steering of the ego vehicle and the behavior of the surrounding vehicles were not considered.

The most relevant study to this work is [122], where the authors proposed a distributed controller utilizing a state feedback law to guarantee a collision-free vehicle merging when facing a road reduction. However, the lateral movement of the merging vehicles was not included in the analysis. Furthermore, the optimal merging location (as measured by the distance between the end of the lane and the merging vehicle) at which the merging vehicle should initiate a merge request was not investigated but assumed. Note the merging location can significantly affect the system level efficiency and safety, and the optimal merging location is not obvious given a particular scenario. Therefore, the assumption that the optimal merging location is known, as made by [122], is not realistic. This work fills this gap by utilizing RL to interactively learn the optimal merging location to improve fuel efficiency and ride comfort. Our study aims to develop an optimal merging strategy that maximizes road capacity utilization while ensuring safety and passenger comfort. We employ an actor-critic style maskable proximal policy optimization (MPPO) algorithm to learn effective merging policies. The RL agent determines the optimal distance at which each merging vehicle should initiate its lane change maneuver. To handle the low-level vehicle control, we utilize Bézier curves for trajectory generation and PID controllers for longitudinal and lateral control. This setup allows the RL agent to focus on high-level decision-making while ensuring smooth vehicle movements. This chapter explores various reward function designs, including minimizing time, energy consumption, and jerk, as well as maximizing average speed. We analyze how these different objectives impact the learned merging strategies and overall traffic flow efficiency. By studying these metrics, we aim to improve road capacity utilization, reduce traffic oscillations, and

Table 4.1: State of the art articles on platoon control maneuvers

References	Vehicle Dynamics	Environment	Evaluation	Application	Control Technique
[123]	Longitudinal	MATLAB	Fuel consumption	On-ramp merging	Optimal control
[119]	Longitudinal and lateral	Not mentioned	Controller stability	Platoon Merging	Distributed MPC
[120]	Not mentioned	VISSIM [124]	String stability	Merging and splitting of platoons	PID
[122]	Longitudinal	Not mentioned	Collision avoidance	Platoon merging facing road reduction	Distributed state feedback controller
[82]	Longitudinal	PLEXE [125] and SUMO	Fuel consumption, connectivity strength, platoon stability, platoon size, and time	Platoon formulating	Hybrid DRL
[68]	Longitudinal	SUMO	Traffic oscillation and platoon stability	Platoon longitudinal control	Soft actor critic (SAC)
[126]	Longitudinal and lateral	AUDRIC/ Dynacar	Safety	Platoon Merging	Feedforward and feedback controller
[127]	Longitudinal and lateral	MATLAB and ROS	Safety	Platoon maneuver protocols	PID, adaptive MPC, and Lyapunov controller
[128]	Longitudinal	PLEXE	String stability	Joining and leaving platoon	Consensus-based controller
[50]	Longitudinal and lateral	SUMO	Traffic flow, average speed, and delay time	Platoons at non signalized intersection	PPO
[70]	Longitudinal	SUMO	String and controller stability	Platoons gap closing/opening	Deep deterministic policy gradient (DDPG)
[129]	Longitudinal and lateral	MATLAB	Controller robustness	Multi-vehicle merging into platoon	Nonlinear MPC
This work	Longitudinal and lateral	Python	Fuel consumption, time, jerk, maximum jerk, and speed	Platoon merging facing road reduction	Maskable PPO

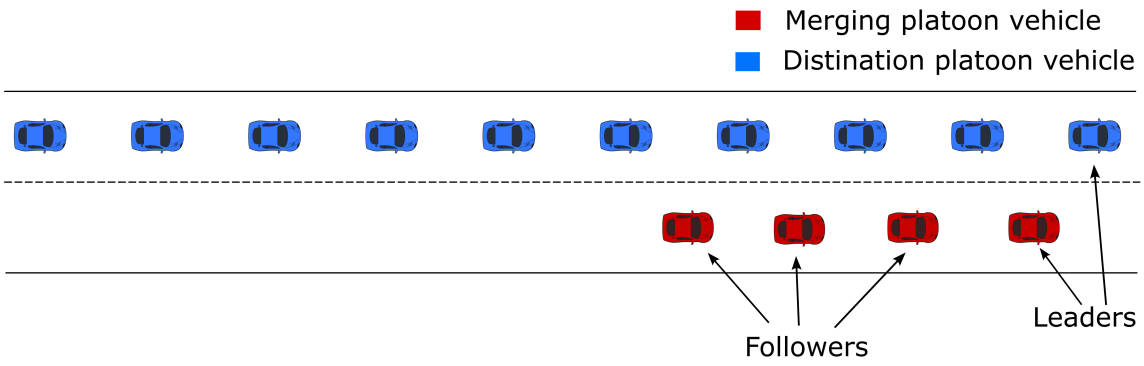

Figure 4.1: Platoons initial configuration.

enhance the driving range of electric vehicles. Through this case study, we investigate the potential of CTCE RL approaches in addressing complex traffic scenarios and optimizing cooperative behaviors among connected vehicles. Our work contributes to the development of efficient platoon merging strategies, which can significantly impact road safety, traffic conditions, and environmental sustainability in urban areas.

4.1 Simulation Environment

4.1.1 Vehicle Platoon

The platoon configuration we consider in this case study is shown in Fig. 4.1, where the destination platoon consists of ten vehicles to measure the impact of the merging technique (nine followers and one leader). These vehicles are initialized to be 11 meters away from each other. On the other side, the merging platoon consists of four vehicles (three followers and one leader). Suppose that the lane for the merging platoon is about to end, and the goal here is to find the optimal merging location for the destination platoon. Therefore, by the end of the simulation, all the merging vehicles should be merged to the destination platoon on the other lane to form one single platoon of fourteen vehicles (thirteen followers and a single leader).

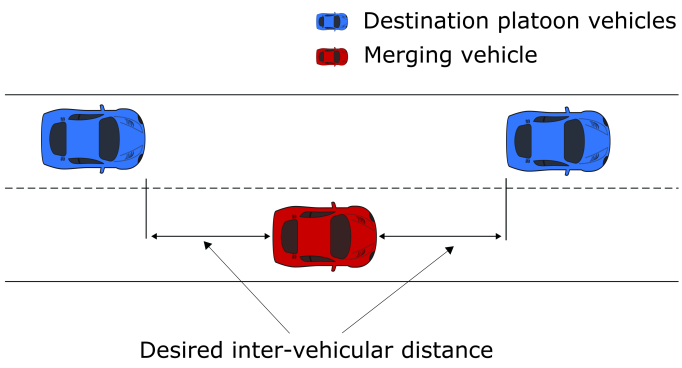

Figure 4.2: Illustration of gap generation.

The platoon travels as one unit without the need to physically couple the vehicles of the platoon, which can be achieved by maintaining a fixed spacing distance between the platoon's members. Two typologies are used in the literature to achieve that, i.e., *constant spacing policy* and *time headway policy*. In the constant spacing policy, the platoon ensures the desired spacing between each vehicle in the platoon regardless of the velocity of the platoon. In the headway time policy, the desired spacing changes with respect to the vehicle's velocity. So that the spacing distance is more extensive for higher velocities to ensure safety by providing more time for the follower vehicle to react to breaks, the platoon in this chapter uses the constant distance spacing policy.

At the start of the scenario, the initial speed of the vehicles and the inter-vehicular gaps are equal to their respective desired values. Desired speed, v_d, equals 10 m/s. All the merging platoon vehicles can ask to merge at any time during the simulation. Furthermore, since the platoons consist of CAVs that are capable to communicate with each other for cooperating merging when any vehicle asks to merge, a gap generation operation will be cooperatively performed by nearby vehicles in both platoons to ensure sufficient space for merging vehicle to perform lane change. Particularly, the controller selects which vehicles

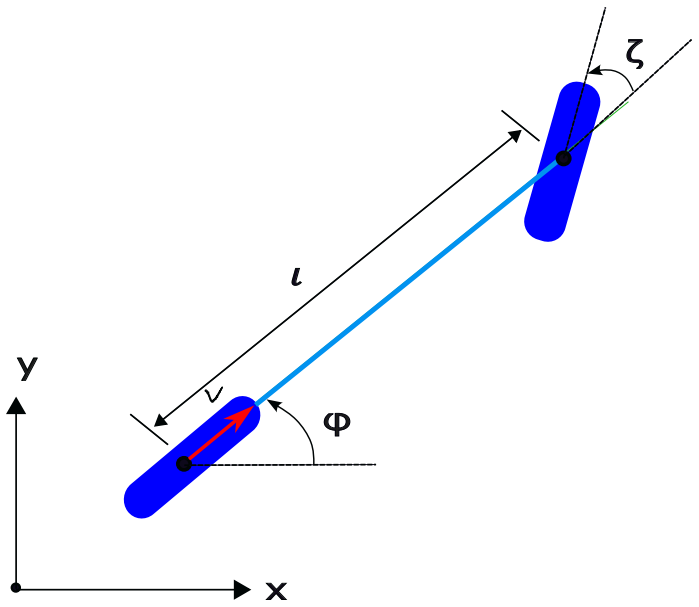

Figure 4.3: Schematics of the vehicle dynamics model.

to increase their spacing distance so that there is a safe distance for the merging vehicle to merge into (Fig. 4.2). The selection will be based on the position of the merging vehicle. When the gap generation operation is done, the lateral controller of the merging vehicle will perform a lane change to merge to the destination platoon. After the merging vehicle arrives at the target lane, a platoon reformulation occurs. The reformulation reassigns the leader of the platoon to the front vehicle and the target vehicle for each vehicle.

4.1.2 Vehicle Model

Both the longitudinal and lateral dynamics of vehicles are taken into consideration. The vehicle dynamic model is briefly described in this section, and interested readers are refer to relevant reference, e.g., [5, 130, 131]. The model of the vehicle used is depicted in

Fig. 4.3 and can be formulated as follows [130]:

$$\dot{v} = a \tag{4.1a}$$

$$\dot{p}_x = v\cos(\phi) \tag{4.1b}$$

$$\dot{p}_y = v\sin(\phi) \tag{4.1c}$$

$$\dot{\phi} = \frac{v}{l}\tan(\zeta), \tag{4.1d}$$

where (p_x, p_y) denotes the position of the vehicle, l is the wheelbase, and ϕ is the yaw angle. The control variables are the acceleration a and the steering angle ζ.

4.1.3 Longitudinal Control

In longitudinal control, the controller tracks the difference between the longitudinal position of the follower vehicle and the longitudinal position of its target vehicle (the vehicle in the front) to the desired value for each follower vehicle by controlling the acceleration of the vehicle. A conventional PID controller is used to control the longitudinal inter-vehicular distance between each vehicle and the vehicle in front of the same platoon. The PID is formulated as

$$u_k(t) = k_p\, e_k(t) + k_d \frac{d}{dt} e_k(t) + k_i \int_0^t e_k(t)\, dt, \tag{4.2}$$

where k_p, k_d, and k_i represent the proportional, derivative, and integral gain of the controller, respectively. $u_k(t)$ and $e_k(t)$ are control variable and the error signal of the kth vehicle, respectively. The error signal can be calculated as

$$e_k(t) = x_{k+1} - x_k - d_{ref}, \tag{4.3}$$

where x_{k+1} and x_k are the longitudinal coordinates of the k_{th} vehicle and its target vehicle, respectively. d_{ref} represents the desired inter-vehicular distance.

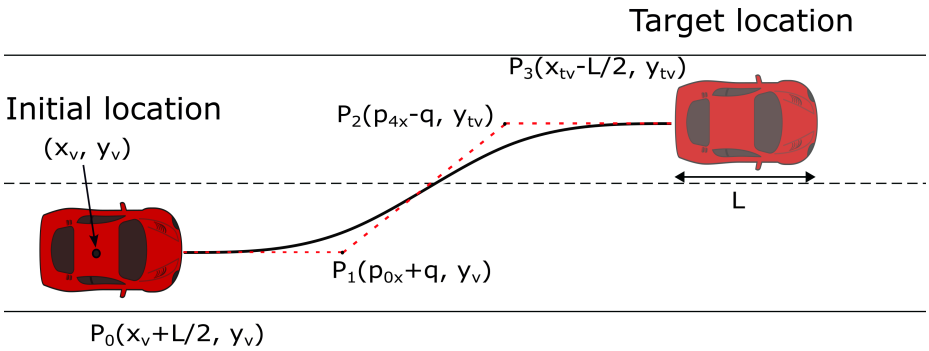

Figure 4.4: Lane changing cubic Bézier curve.

4.1.4 Lateral Control

When the gap generation operation is done, the merging vehicle generates a lane-changing path and follows it to the other lane. Using Bézier curves to generate the reference trajectory results in smoother routes that are easy to track by the merging vehicles [132]. With $n+1$ control points, a Bézier curve of order n is formulated as described by [132]

$$P_{[t_0,t_1]}(t) = \sum_{i=0}^{n} B_i^n(t)\, P_i, \tag{4.4}$$

where P_i are control points, $B_i^n(t)$ is the Bernstein polynomial given by

$$B_i^n(t) = \binom{n}{i} \left(\frac{t_1 - t}{t_1 - t_0}\right)^{n-i} \left(\frac{t - t_0}{t_1 - t_0}\right)^{i} i \in \{0, 1, ..., n\} \tag{4.5}$$

Bézier curve has several unique properties, but the most satisfactory for lane-changing maneuvers is that the curve's starting and ending segments are tangent to the first and last points. So, the line between the first two control points and the line between the last two control points can be selected to be parallel to the lanes (Fig. 4.4). By doing this, at the end of the lane change, the vehicle will have the same heading angle as the lanes.

A third order Bézier curve with four control points (p_0, p_1, p_2, and p_3) is used in this work. Therefore, (4.4) reduces to

$$P(t) = (1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3, \tag{4.6}$$

with $t \in [0,1]$. As shown in Fig. 4.4, the first and last control points (P_0, P_3) are positioned at the front of the merging vehicle and the back of its target vehicle, respectively. The orientation of lines P_0P_1 and P_2P_3 are parallel to the lane lines to reduce the vehicle's post-curve adjustment time. Furthermore, setting

$$q = P_{0,x} - P_{1,x} = P_{2,x} - P_{3,x}, \tag{4.7}$$

yields a symmetric Bézier curve around the path center, making q the only hyperparameter to be tuned to get a smooth curve.

Similar to longitudinal control, a PID controller is used to track the lateral offset between the vehicle and the Bézier trajectory. When the merging vehicle reaches the center of the target lane, the PID lateral controller is then used to track the center line so that the vehicle will be performing lane-keeping.

4.2 Proximal Policy Optimization Algorithm

In this work, the proximal policy optimization (PPO) algorithm is used [133, 134], which is a policy-based on-policy policy gradient RL algorithm. In general, policy gradient methods attempt to optimize the policy directly [135]. The policy, π, is a function approximator, usually a neural network, parameterized with respect to a set of parameters θ. Essentially, gradient ascent is used to change θ towards the increase of the cumulative rewards. Policy gradient methods are significantly faster in practice [136], but they suffer from some fundamental problems. For example, the agent's training data is based on the current policy when the data was collected, which makes the rewards and observations

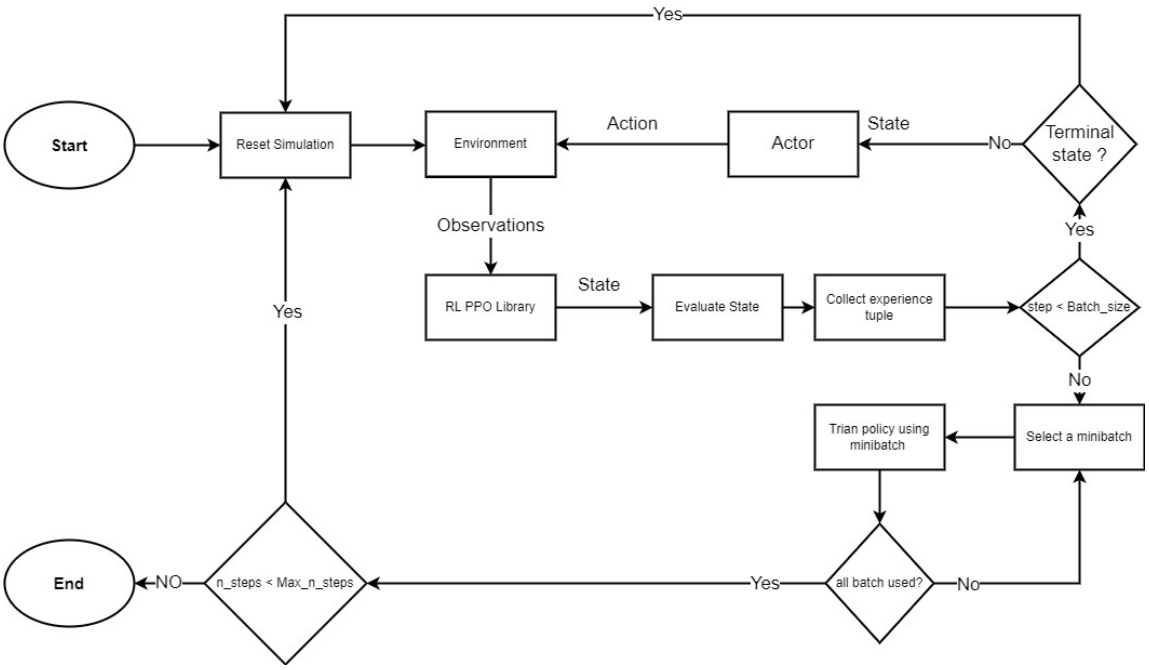

Figure 4.5: Data flow diagram of the PPO algorithm.

distribution constantly changing based on the current policy. This change leads to instability in the whole training process. Also, policy gradient methods are susceptible to hyper-parameters like entropy coefficient, learning rate, and weights initialization, to name a few. To address these issues, PPO has been proposed in the literature as a scalable, robust, and sample-efficient policy gradient algorithm that is also relatively easy to code.

For policy gradient methods, the loss is defined as follows:

$$L^{PG}(\theta) = E[\log \pi_\theta(a_t|s_t)A_t], \tag{4.8}$$

where E is the expected return over a batch of data, A_t is the estimation of the advantage function at time step t, and π_θ is a stochastic policy. $\pi_\theta(a_t|s_t)$ is likelihood of choosing the action a given the state s. The advantage function can be calculated as

$$A_t = G_t - V_t(s), \tag{4.9}$$

where G_t is the total discounted rewards, v_t function or value estimation of the state s. Making multiple optimization steps on this loss using the same data collected from the environment is not advised because that might change the policy too much towards that specific trajectory. TRPO, [135], has already tried to solve this issue, but their solution (trust region optimization method) includes a second-order derivative and its inverse, which is very computationally expensive. PPO solved the same problem by introducing a soft constraint that makes the objective function solvable using a first-order optimizer. The new objective function will prevent the policy from changing too much by clipping the objective value, making it possible to run multiple optimization steps on the cost function without moving the policy too far in the parameter space. The loss function proposed by PPO is as follows [134]:

$$L^{CLIP}(\theta) = E\left[min(\Gamma_t(\theta)A, clip(\Gamma_t(\theta), 1-\varepsilon, 1+\varepsilon))A\right], \tag{4.10}$$

where Γ_t is the probability ratio of the policy before the new policy and the policy before the update $\pi_{\theta_{old}}(a_t|s_t)$. Epsilon is a hyperparameter that defines how much an update can change the policy. In the PPO algorithm, the agent collects data by interacting with the environment. Next, the advantage estimate of each state is calculated. Finally, for k epochs, stochastic gradient descent is applied with N mini-batches of the collected data to update the policy. A pseudocode of the PPO algorithm is shown in Algorithm 4.1. Finally, Fig. 4.5 shows flow chart of the PPO algorithm.

4.3 RL-based Merging Strategy

4.3.1 States Observation and Action Space

In this simulation, there are fourteen autonomous vehicles. Four are in the merging platoon, and the rest belong to the destination platoon. The states should describe all the essential information about every vehicle so that the agent can have enough information to

Algorithm 4.1: Centralized-RL AVs Merging Control

1. Initialize policy parameters θ, value function parameters ϕ
2. **for** *each iteration* **do**
3. Set $\theta_{old} \leftarrow \theta$
4. **for** *actor = 1, 2, ..., N* **do**
5. Run policy $\pi_{\theta_{old}}$ in environment for T timesteps
6. Compute advantage estimates A_t
7. **end**
8. **for** *epoch = 1, 2, ..., K* **do**
9. **for** *each minibatch* **do**
10. Compute value function loss: $L^{VF}(\phi) = (V_\phi(s_t) - G_t)^2$
11. Update ϕ to minimize $L^{VF}(\phi)$
12. Compute policy loss: $L^{CLIP}(\theta)$
13. Update θ to maximize $L^{CLIP}(\theta)$
14. **end**
15. **end**
16. **end**

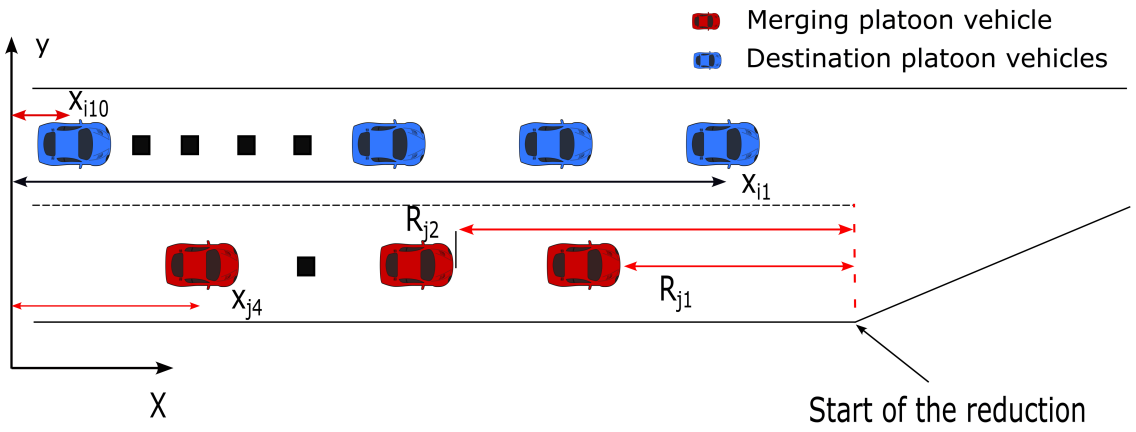

Figure 4.6: Observation space measurements.

take reasonable actions. The global x position of every vehicle is provided, and the relative distance of each merging vehicle to the start of the road reduction is provided as shown in Fig 4.6. The state of whether every merging vehicle is merged or not is also fed to the network. It can be observed that there are continuous and discrete attributes, and each has its own maximum and minimum values, meaning that in order to get a fast convergence, normalization is inevitable. The state vector can be formed as follows:

$$S = \begin{bmatrix} x_{i1} & x_{i2} & \cdots & x_{i10} & R_{j1} & \cdots & R_{j4} & s_{j1} & \cdots & s_{j4} \end{bmatrix} \tag{4.11}$$

where i and j denote the destination and merging platoon vehicles, respectively, as shown in Fig 4.6. x is the global x coordinate, R is the relative distance between the corresponding vehicle and the starting point of the road reduction, and s is the status of the vehicle as follows:

$$s_v = \begin{cases} 1, & => \text{if vehicle } v \text{ is merged} \\ 0, & => \text{if vehicle } v \text{ is not merged} \end{cases} \tag{4.12}$$

Since we have four vehicles in the merging lane, the action space size is four, one for each vehicle. The action should stimulate the corresponding merging platoon vehicle to ask to merge to the other lane. There are two options for the action space: discrete or continuous. The continuous option means that the agent will select a relative distance at the start of the simulation to ask to merge and ultimately try to find the optimal distance. That approach works only if perfect prediction of all vehicles future behavior is available. On the other hand, with the discrete action space, the agent will make real-time actions based on the observations it is receiving. The action for the vehicle v is as follows:

$$a_v = \begin{cases} 1, & => \text{Request to merge} \\ 0, & => \text{Stay in the same lane} \end{cases} \tag{4.13}$$

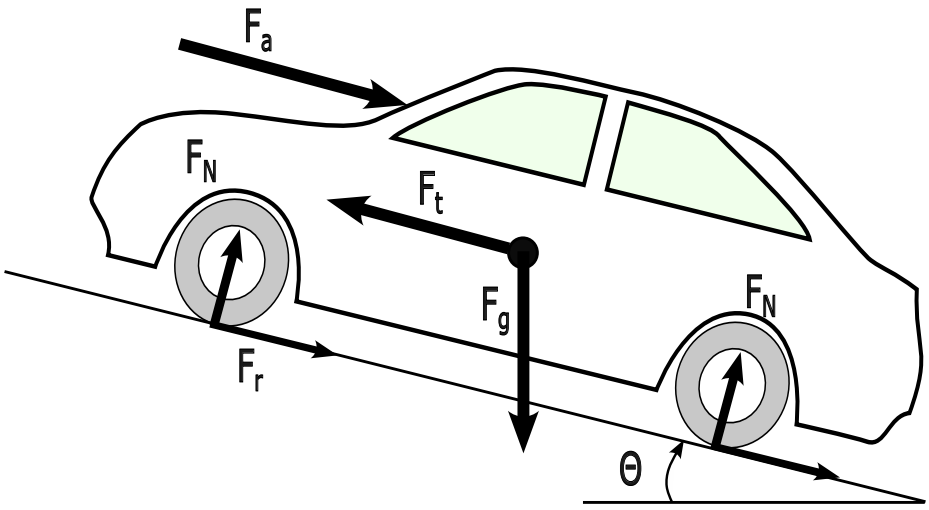

Figure 4.7: Free Body Diagram of the vehicle.

4.3.2 Reward Functions

In this chapter, different reward functions will be used to train the RL model to investigate their impact on the merging strategy. The vehicles' time consumed, energy consumption, mean jerk, maximum jerk, and relative position are characteristics used to incentivize or discourage the agents' decisions. The first important index is that all the vehicles merge to the not-ending lane and do not crash. A penalty of negative rewards is returned to the RL algorithm for every non-merged vehicle that gets close to the start of the road reduction.

4.3.2.1 The Energy Consumption The amount of energy consumed during the maneuver is essential in evaluating the model behavior. In this work, an electric vehicle energy model is used to calculate the energy consumed by all vehicles to finish the merge. Using newton's second law, the forces on the wheel can be formed as follows in equation (4.14).

$$\sum F_x = ma, \tag{4.14}$$

where a is the vehicle acceleration, m is the mass of the vehicle, and F_x is the summation of forces applied on the vehicle in the x direction. Substituting the forces shown in Fig. 4.7 expressed in equation (4.15).

$$F_t - F_a + F_g + F_r = ma, \tag{4.15}$$

where $F_a = 0.5 C_d(D) \rho A v^2$ is the aerodynamic resistance, $F_r = m C_r g \cos(\theta(t))$ is the friction force, $F_g = m g \sin(\theta(t))$ is the gravity force, and $F_t = m a_w$ is the traction force. Note that here a_w is the wheel acceleration, A is the frontal area of the vehicle, ρ is the air density, g is the acceleration of gravity, $\theta(t)$ is the gradient of the road, C_d is the air drag coefficient, C_r is the rolling resistance coefficient, g is the gravity acceleration, a is vehicle acceleration, and D is the relative distance between the vehicle and the vehicle in front of it [137].

Reorganizing and substituting the forces formulas in equation (4.15) yields:

$$a_w = a + \frac{0.5 C_d(D) \rho A v^2}{m} + C_r g \cos(\theta(t)) + g \sin(\theta(t)) \tag{4.16}$$

This work adopts a 2019 Nissan LeafSV EV from [123]. The energy consumption of the vehicle during the simulation time ts is as follows:

$$R_e = \int_0^{ts} (m a_w v + \frac{b (m r_t)^2}{\xi^2} a_w^2) \, dt, \tag{4.17}$$

where ξ is the gear ratio, r_t is the radius of the tire, and b is the motor loss coefficient, measured experimentally.

4.3.2.2 The Vehicle Jerk Passengers' comfort has been studied thoroughly, especially for automated vehicles, as it can affect the adoption of autonomous vehicles. Repetitive exposure to low-frequency motions can develop motion sickness [138], and regular exposure to high-frequency motions can lead to lower back pain [139, 140]. The jerk can be used to sense these discomfort and sudden acceleration changes and ultimately optimize

the autonomous vehicle's behavior to ensure comfortable driving. This work uses the mean and the maximum jerk as reward functions to train the RL agent. For the mean jerk, the absolute value of the jerk of every vehicle is calculated, and the mean is sent as the reward. The reward function is expressed as follows:

$$R_j = \frac{-1}{N} \sum_{n=1}^{N} \left(\frac{a_{n,k} - a_{n,k-1}}{dt} \right)^2, \tag{4.18}$$

where R_j is the step rewards, k is the time step, and N is the number of vehicles.

For the maximum jerk as a reward function, only the maximum jerk of all vehicles is returned as the step reward. In this case, the reward function is expressed as:

$$R_{mj} = -||J_k||_\infty \tag{4.19}$$

where J_k is a vector of the absolute values of all the vehicles' jerk at time step k.

4.3.2.3 Time Another metric used to train the RL is time. Reducing the time it takes all the vehicles to finish the merge reduces traffic congestion. For every time step, a negative reward will be sent to the RL agent until all of the merging platoon vehicles have already merged to the other lane. This will incentivize the agent to merge all the vehicles as soon as possible.

$$R_t = \begin{cases} -r, & => \text{if merging vehicles did not merge yet} \\ 0, & => \text{if all merging vehicles have merged} \end{cases} \tag{4.20}$$

4.3.2.4 Speed Another reward function is proposed to encourage the model to get all the vehicles to go through the merge faster. A relative position (longitudinal velocity) of the last vehicle in the destination platoon is returned to the agent at each time step. The reward function can be obtained as follows:

$$R_s = x_{v_l,k} - x_{v_l,k-1}, \tag{4.21}$$

where $x_{v_l,k}$ is the global x position of last vehicle in the destination platoon at time step k.

4.3.3 Maskable PPO

Based on the nature of our simulation, the valid actions change based on the state of the environment. So, for example, a gap generation operation will start when one of the vehicles asks to merge. The vehicle's state will be changed accordingly to "merged", which means the agent should not be able to ask a vehicle to merge again after it is already merged into the target lane. That means for the rest of the simulation, the only proper action for a merged vehicle is "stay in the same lane".

There are three methods to solve this problem.

- The first one is to build the simulation environment to ignore invalid actions. However, this method is not sampling-efficient since sampling ignored actions that do not affect the environment will waste a significant amount of time.
- In the second approach, a negative reward is set to penalize choosing an invalid action so that the agent will eventually learn only to select valid actions. This method will add an unnecessary complication for the policy to learn, increasing the required convergence time.
- In the third approach, a mask is used to block invalid action allowing the policy to only choose within the available valid actions at that state. In [141], the theoretical justification for using masking in policy gradient methods is proved.

All three approaches have been implemented in this work, and it was determined that the third approach, namely, Maskable PPO, performs the best. Fig. 4.8 shows the difference in convergence time between a regular PPO, where the environment ignores invalid actions, and Maskable PPO (MPPO). All numerical results presented in the rest of this chapter are collected using MPPO.

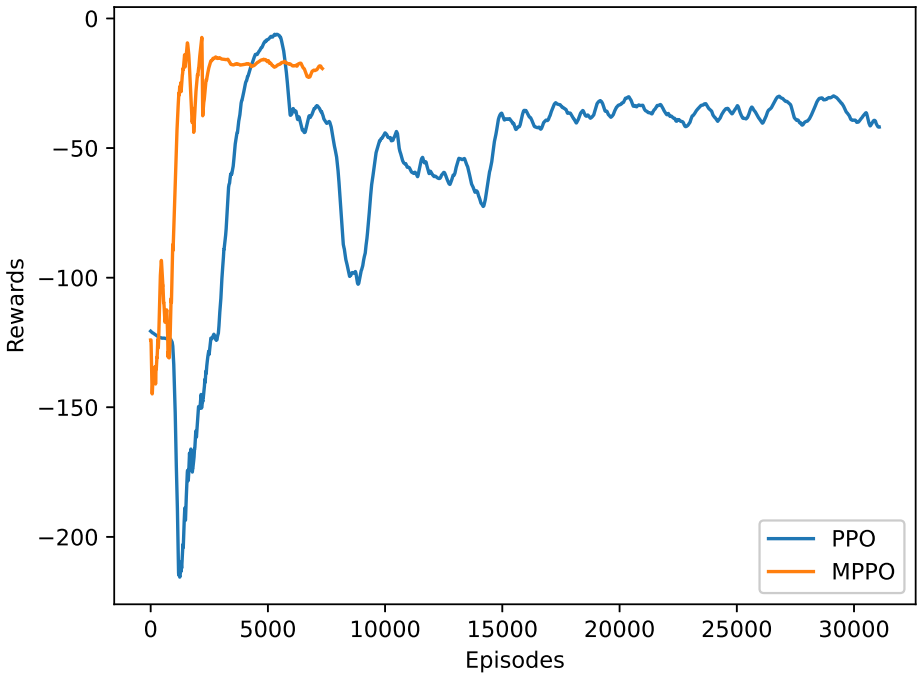

Figure 4.8: Comparison of training progress of MPPO and PPO.

4.4 Numerical Results and Discussion

A series of tests were performed to evaluate the performance of the proposed framework using our recently developed object-oriented toolbox for Python. The system includes a collection of tools and interfaces for simulating and displaying the movement of the vehicles within an intelligent transportation system environment. It also calculates the performance indices to evaluate the merging technique of the trained model. The toolbox consists of two main components, the DRL model and the environment. The PPO algorithm represents the DRL model. However, there are multiple implementations for the PPO algorithm (DRL model) [142]. Therefore, the maskable PPO from [133] is adopted in this work. The environment is created to be an OpenAI Gym class [143], with built-in functions to perform the low-level controllers to manage the simulation of non-RL related actions. The toolbox operation consists of two stages, training and evaluation. After training, the model will be used to predict actions, and a test scenario will be simulated to evaluate the performance indices of the model behavior. This section presents numerical results. We start with single objective RL, followed by the multi-objective RL that combines all important metrics. Table 4.2 lists all the parameters used in this simulation. Simulations are performed with high-performance computing (HPC) nodes running Red Hat Enterprise Linux release 8.6 (Ootpa) with 192 GB of RAM and 40 CPU Cores at 2.50 GHz. The training time of a maskable PPO model with 2048 steps for each rollout and 64 batch size is 6000 episodes, around 10 hours. On the other hand, the time required to simulate an entire scenario and evaluate the model actions is 400 milliseconds.

4.4.1 Single Objective RL

Evaluation results of each single objective RL model are summarized in Table 4.3, where the "Simple Early Merge" model is a manually designed merging strategy for benchmarking. Specifically, this simple merging behavior would make all vehicles ask to

Table 4.2: Simulation Parameters

Parameters	Value	Description
N	14	Total number of vehicles
r	1	Time model negative reward
q $[m]$	6	Distance between control points to obtain a smooth Bézier curve
θ_a	22 x 64 x 64 x 8	Actor Network Architecture
θ_c	22 x 64 x 64 x 1	Critic Network Architecture
v_d $[m/s]$	10	Leaders desired speed
d_{ref} $[m]$	11	Desired inter-vehicular distance
Batch Size	64	Number of tuples propagate the network
ε	0.2	Clipping hyperparameter
Number of Epochs	10	The times experiences are used to train the network
RL discount factor γ	0.99	Defines the priority of immediate rewards
l $[m]$	4	Wheelbase length
m $[kg]$	1618.87	Mass of the vehicle
ρ $[kg/m^3]$	1.28	Air density
A $[m^2]$	2.5334	Frontal area of the vehicle
$\theta(t)$ $[rad]$	0	Gradient of the road
g $[m/s^2]$	9.81	Acceleration of gravity
C_r	0.015	Rolling resistance coefficient
ξ	8.193	Gear ratio
r $[m]$	0.4318	Radius of the tire
b	1.0355	Motor loss coefficient
k_p	0.2	Proportional gain
k_d	0.7	Derivative gain
k_i	0.00034	Integral gain

Table 4.3: Evaluation results of different single objective RL models

RL Model	R_e	R_j	R_{mj}	R_s	R_t	Simple Early Merge
Energy Consumed [MJ]	**21.24**	44.76	91.5	92.77	80.2	91.5
Average Jerk [m/s^3]	0.4907	**0.4478**	0.6841	0.692	0.634	0.6841
Maximum Jerk [m/s^3]	2.95	2.436	**2.3019**	3.09	2.565	**2.3019**
Last Vehicle's Average Speed [m/s]	7.67	6.17	8.869	8.865	**8.91**	8.869
Time [s]	27.1	28.8	18.2	18.1	**17.4**	18.2

merge at the start of the simulation, yielding a zipper-like configuration of the resultant single platoon. Detailed discussions on the results are given as follows.

4.4.1.0.1 Results for minimizing energy consumption only For the first case, the RL agent is trying to reduce the energy consumed by the vehicles. The average of all the vehicles' energy consumed is returned every time step. As shown in Table 4.3, the RL agent reduced the energy to around 21.24 MJ, which is more than 76% better than doing an early merge of all the vehicles at the start of the simulation. Furthermore, the average jerk is also significantly reduced. The RL model performed a zipper merge, starting with the first vehicle of the merging platoon asking to merge, and the last vehicle of the merging platoon being the last one to ask to merge. Fig. 4.9 shows the training progress of ten different seeds.

4.4.1.0.2 Results for minimizing the time required to finish the merge The RL agent is trying to reduce the time required to merge all the vehicles into one platoon. The apparent attempt to minimize the time required to finish the merge is to start merging as soon as possible to minimize the time required. However, the RL agent found a better cooperative

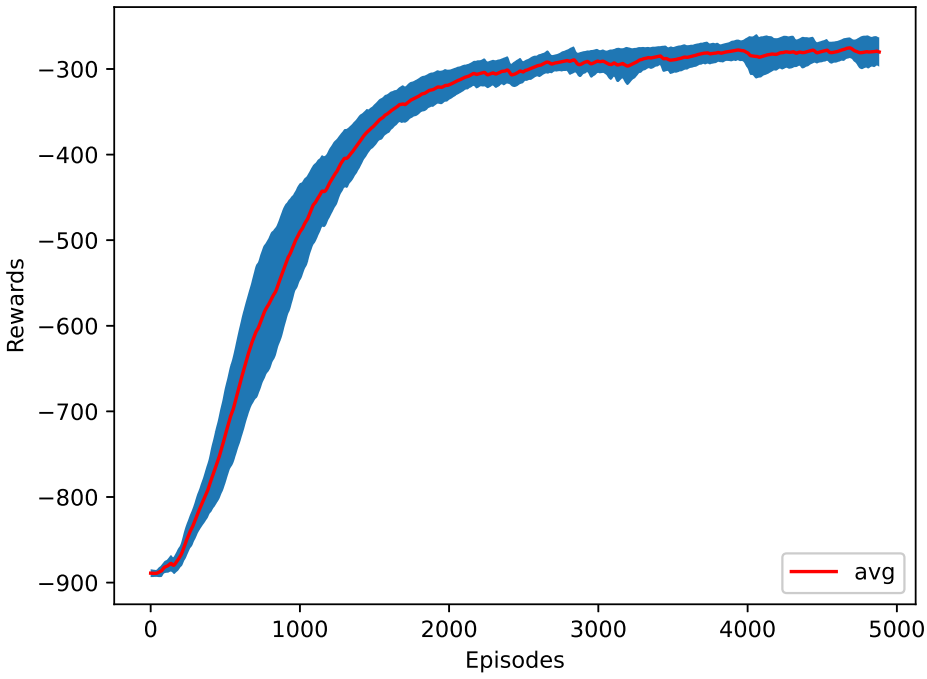

Figure 4.9: The training plot of the energy as a reward function.

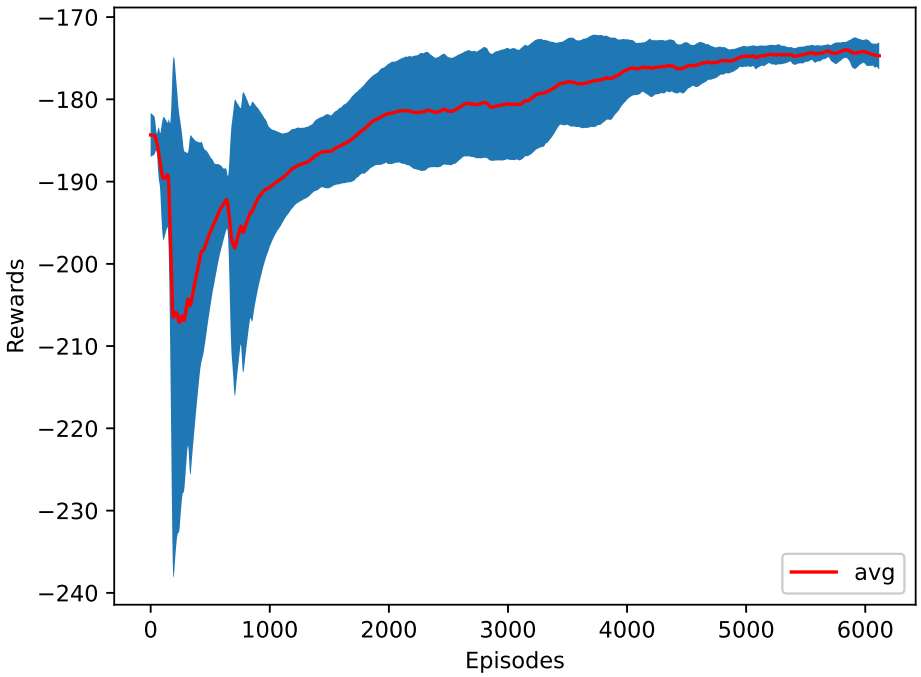

Figure 4.10: The training plot of the time as a reward function.

behavior that does not merge all vehicles at the start. Instead, some vehicles surprisingly wait some time before asking to merge, which proves that the pattern or behavior of merging significantly affects the merging performance. As a result, the agent learns to perform an early zipper merge to finish in only 17.4 seconds, as shown in Table 4.3. The training progress is shown in Fig. 4.10. It can be observed that even with the untrained model (random actions), the time rewards achieved are relatively good. The reason is that at each time step, the agent has two options for each vehicle, merge or stay in the same lane, which makes starting probability of each action to be chosen by the agent is 50%. Given that the agent will be asked to choose an action ten times every second, it is very likely that the agent will ask all the vehicles to merge in the first second.

4.4.1.0.3 Results for maximizing the speed of the last vehicle in the destination platoon In the third case, the change in the x position of the destination platoon's last vehicle is returned to the agent. The change in the x position represents the longitudinal velocity. Increasing the longitudinal speed of the last vehicle increases the traffic flow. The agent's average speed of the last vehicle is 8.8 m/s, where the desired speed of the last vehicle is set to 10 m/s. Fig. 4.11 shows the training progress of ten different seeds. It is worth noting that in this case, the last vehicle's speed is actual lower that the case of R_t. This is likely due to the fact that rewarding based on one single vehicle can take a longer time for RL to converge and there can be multiple local optimal for RL training algorithm. However, as can be seen from Table 4.3 that R_s did perform better than the cases of benchmarking Simple Early Merge model and R_{mj}.

4.4.1.0.4 Results for minimizing the mean jerk of all the vehicles In the fourth case, the RL agent reduces the changes of acceleration and/or deceleration of the vehicles. The average jerk of all of the vehicles is returned to the agent. The RL learned to merge in 28.8 seconds with only 0.4478 m/s^3 average jerk. Reducing the average jerk reduces the vehicle's changes of acceleration and deceleration, and therefore decreasing energy consumption by more than 51% less than a regular early merge. Fig. 4.12 shows the training progress of ten different seeds.

4.4.1.0.5 Results for minimizing the maximum jerk Reducing the average jerk does not necessarily mean that the jerk is satisfied for every vehicle at each time step. In the fifth case, the RL is encouraged to reduce the maximum jerk of all vehicles. The maximum value of all the vehicles' jerks is returned to the agent at every time step. The RL agent successfully reduced the maximum jerk to 2.3019 m/s^3, 5.5%, and decreased the time spent by 36.8%, better than the average-jerk RL. This comes with the cost of increasing energy

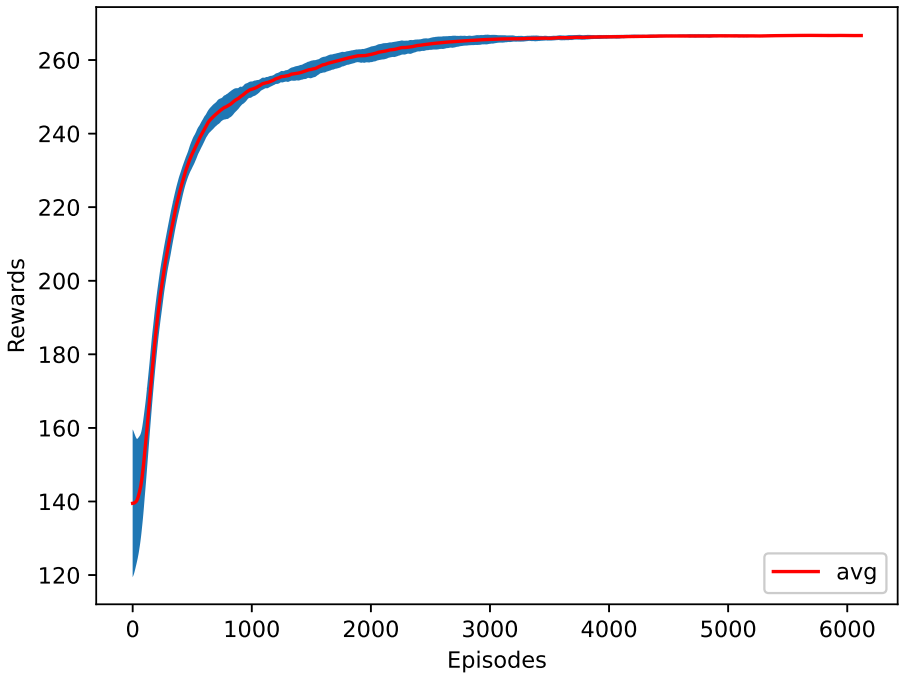

Figure 4.11: The training plot of the speed as a reward function.

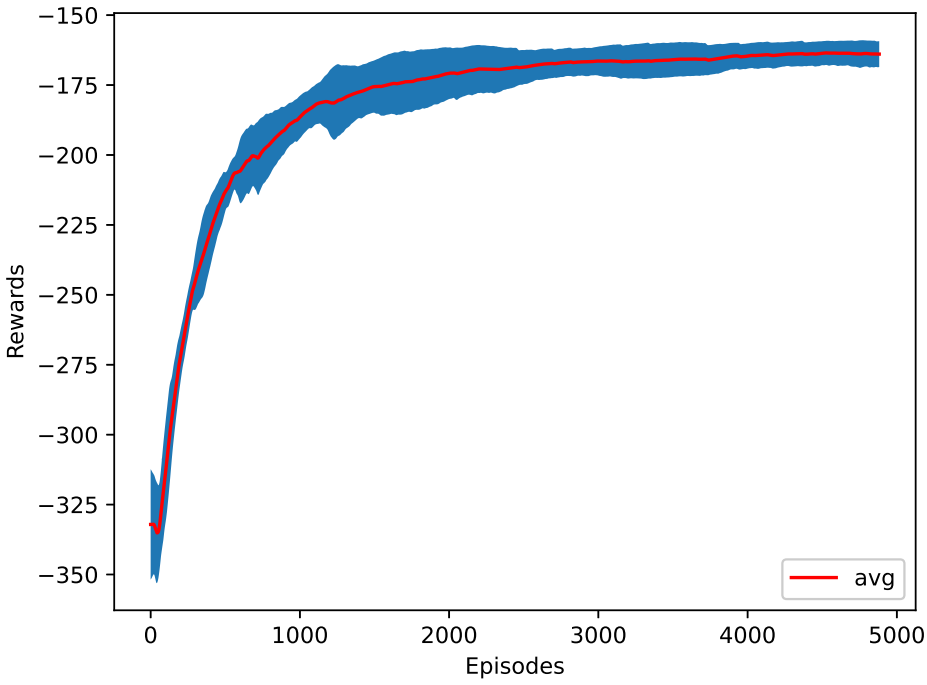

Figure 4.12: The training plot of the average jerk as a reward function.

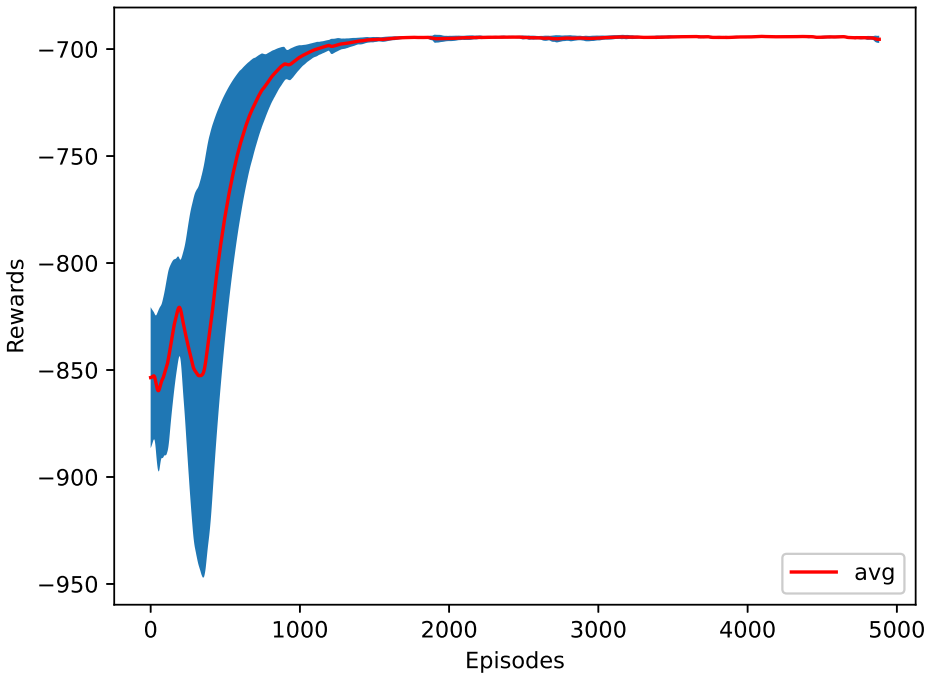

Figure 4.13: The training plot of the maximum jerk as a reward function.

consumption by 51% and the average jerk by 52.7%. The training progress of ten different seeds is shown in Figure (4.13).

4.4.2 Multi-Objective RL

It can be observed that when the average jerk is minimized, the agent takes too long to finish the merge, but when time is the main objective of the RL the energy consumed and the jerk rise high. This means there should be a balance based on the type of drive required. A weighted sum of all the individual rewards is returned to the agent every time step, which can be formulated as follows:

$$R_{\text{multi-objective}} = -\delta_1 R_t + \delta_2 R_s - \delta_3 R_j - \delta_4 R_e, \tag{4.22}$$

Table 4.4: Evaluation results of the multi-objective RL with different weights

	Case 1	Case 2	Case 3
Max Jerk $[m/s^3]$	3.01	**2.6**	3.35
Avg Jerk $[m/s^3]$	**0.52**	0.54	0.57
Last Vehicle's Average Speed $[m/s]$	7.75	**8.9**	7.97
Energy Consumed $[MJ]$	56.66	**46.48**	83.68
Time $[s]$	25.5	**21.6**	22.4

where $\delta_{1,2,3,4} \in [0,1]$ are the weights. R_e, R_j, R_t, and R_s, are formulated in equations (4.17), (4.18),(4.20), (4.21), respectively. Figs. 4.14 and 4.15 show the training progress of ten different seeds with weights $\delta_1 = 0.2, \delta_2 = 0.1, \delta_3 = 0.3, \delta_4 = 0.3$ (Case 1) and $\delta_1 = 0.1, \delta_2 = 0.1, \delta_3 = 0.3, \delta_4 = 1$ (Case 2), respectively. The agent will accommodate time and energy for the first set of weights while maintaining a low amount of mean jerk. While the agent for Case 2 will care more about energy which increases the time by a few seconds, as shown in Table 4.4, which lists an additional result for Case 3 with $\delta_1 = 0.2, \delta_2 = 0.1, \delta_3 = 0.4, \delta_4 = 0.4$. As can be seen from Table 4.4, the merging strategy is greatly influenced by the weights for (4.22). Usually, the balance of each metric is up to the policymaker, and the proposed framework is flexible to accommodate a variety of merging strategies by simply changing the weights of (4.22), hence avoiding manual control design for each scenario.

4.5 Summary

This chapter explored the cooperative merging of two platoons of electrified Connected and Automated Vehicles (CAVs) during a lane reduction scenario. We proposed a CTCE DRL framework using an actor-critic style maskable Proximal Policy

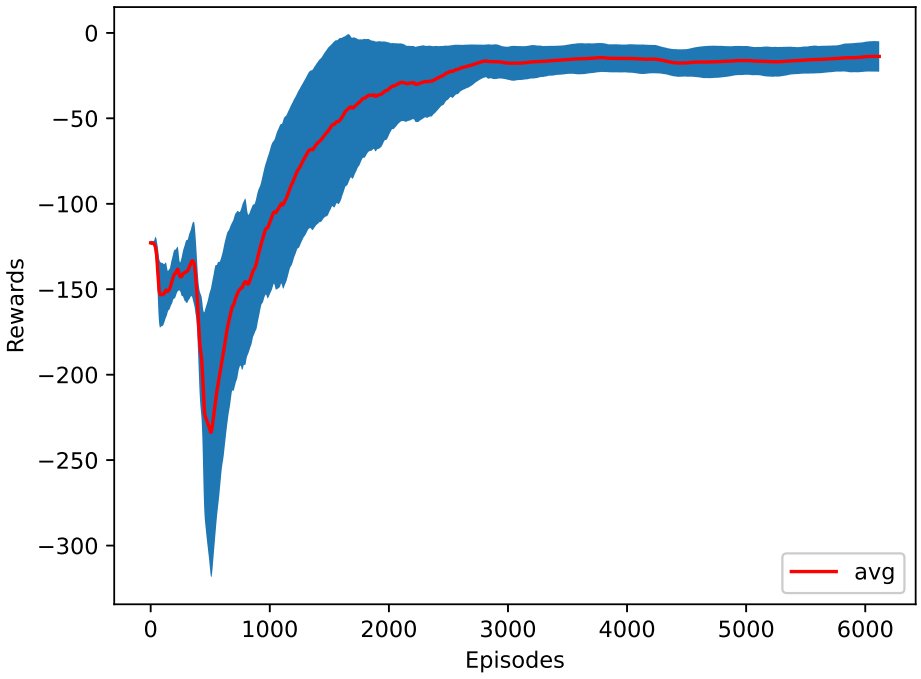

Figure 4.14: The training plot of the multi-objective reward function (Case 1).

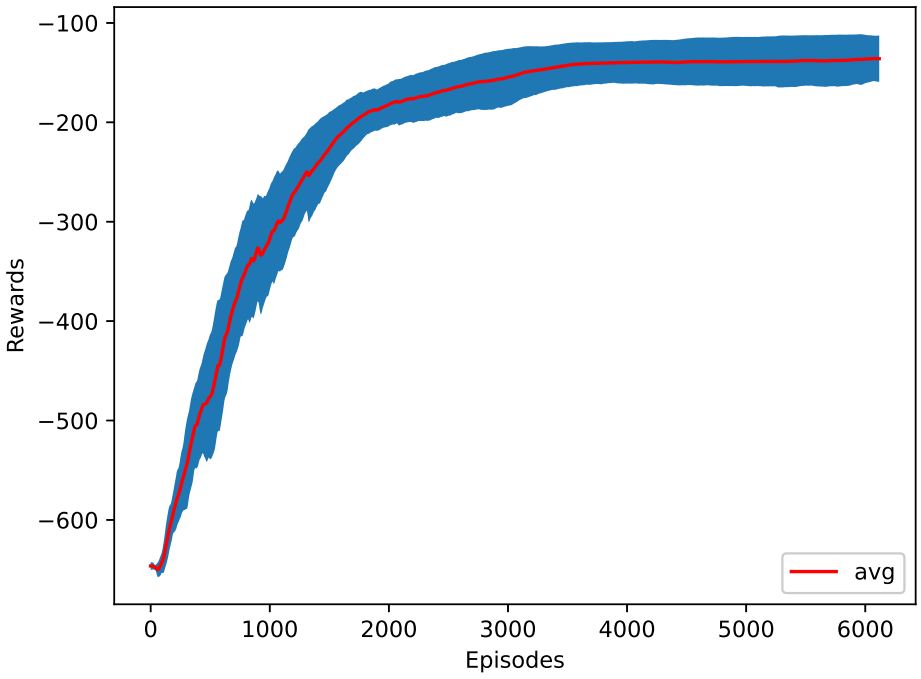

Figure 4.15: The training plot of the multi-objective reward function (Case 2).

Optimization (MPPO) algorithm to learn optimal merging policies. The framework aimed to maximize road capacity utilization while ensuring safety and passenger comfort. Our approach successfully determined the optimal distance for merging vehicles to request lane changes, with low-level control handled by Bézier curves and PID controllers. The DRL method demonstrated significant improvements, achieving a 76.7% reduction in energy consumption and a 50% reduction in average jerk compared to baseline strategies. These results highlight the substantial impact of merge timing on traffic flow, energy efficiency, and passenger comfort. The study revealed the DRL framework's ability to balance multiple objectives, including minimizing time, energy consumption, and jerk, while maximizing average speed. However, we acknowledge limitations in the current approach, particularly its focus on a two-lane reduction scenario with fully cooperative vehicles. While the centralized controller demonstrated promising results in our study, it faces inherent limitations that could hinder its application in larger, more complex traffic scenarios. The reliance on a single control point makes the system vulnerable to communication failures or latency issues, which could significantly impact the performance and safety of the entire platoon. Moreover, as the number of vehicles increases, the computational complexity grows exponentially, potentially leading to slower decision-making and reduced responsiveness. This scalability issue is particularly concerning in real-world traffic situations where the number of vehicles can be much larger and more dynamic than in our controlled experiment.

CHAPTER FIVE

MARL-BASED AV MERGING CONTROL

This chapter presents a case study on applying MARL to highway merging control in road reduction scenarios. Building on the centralized approach explored in Chapter 4, this study investigates a Centralized Training Decentralized Execution (CTDE) framework to address scalability limitations. Specifically, MARL is employed to control AVs attempting to merge onto a lane already occupied by human-driven vehicles (HDVs), with the goal of maximizing traffic flow, passenger comfort, and fuel efficiency. The framework utilizes an actor-critic architecture, where a self-attention network in the critic handles the varying number of AVs. The proposed method is validated using the SUMO traffic simulator [144], which provides a realistic highway simulation environment.

Several researchers have applied CTDE to highway merging scenarios for AVs, as shown in Table 5.1. For example, authors in [145] utilized CTDE for connected and automated vehicles at merging roadways, where the critic receives the concatenated local observations of all agents while actors map local observations to local actions. The system was trained with a total of three AVs, and it was demonstrated that the learned policy can be applied to scenarios with up to 8 AVs. Similarly, reference [74] proposed an Interaction-aware Decision Making with Adaptive Strategies (IDAS) approach using CTDE, employing a pair of critics, one centralized and one decentralized, along with decentralized actors. The centralized critic encouraged cooperative behavior and smooth traffic flow, while the decentralized critic helped agents learn rule-following and individual behaviors.

The CTDE approach has shown promise in achieving smooth and safe merging while optimizing traffic flow. By leveraging global information during training, these

methods can learn cooperative behaviors that consider the overall traffic situation, while still allowing for decentralized, scalable execution based on local observations. Yet, despite these advancements, CTDE approaches in current research still face certain limitations, particularly when it comes to handling dynamic traffic environments. Additionally, as shown in Table 4.1, the majority of current research on MARL for highway maneuvers (e.g., lane change, on-ramp merging, and road reduction) considers a fixed, limited number of AVs that don't change during the training episode [43, 44, 54, 58, 59, 71, 72, 145, 146]. Although this makes the learning problem simpler, it does not accurately represent real-world traffic situations, as vehicles constantly join and exit the road network and cause the number of vehicles to change dynamically over time.

Compared to existing work reviewed above, the present study is different and novel from the following aspects. (1) The number of vehicles considered in this paper is significantly higher than existing work. In particular, a total number of 50 vehicles are considered during one single episode. (2) The proposed MARL CTDE framework is capable of handling a variable number of agents, and therefore significantly reduce the network size and decrease the training time. (3) As most of existing work only evaluate one or two metric to assess the performance of the learned policy, we consider five different metrics ranging from passenger comfort to fairness.

5.1 Problem Definition and Proposed Approach

In this study, we focus on the problem of multiple AVs merging in a road reduction scenario using MARL. The environment, as shown in Fig. 5.1, is modeled as a two-lane highway where the right lane terminates after 300 meters, necessitating the merging of vehicles into a single lane. This traffic scenario is simulated using SUMO (Simulation of Urban Mobility), an open-source microscopic traffic simulator [144]. Vehicles in the main lane are expected to maintain safe following distances and create gaps to accommodate

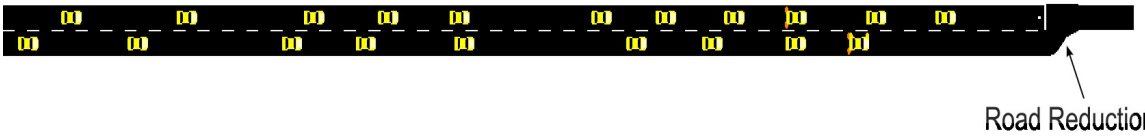

Figure 5.1: The AVs merging scenario in SUMO simulation environment, where each vehicle on the bottom lane is an RL agent. The goal is to learn an optimal merging strategy for each vehicle on the bottom lane so that the overall long term return is maximized. Note that the number of vehicles in the merging zone varies within a single episode, making the RL problem challenging as the number of agents is not fixed.

Table 5.1: MARL Approaches for AV Control in Various Highway Merging Scenarios

References	MARL Approach	# of AVs	Evaluation Metrics	Merging Scenario
[43]	Multi-agent Q-learning	2	Collision rates	Taper-type on ramp
[44]	DTDE-MADDPG	2	Collision rates	Taper-type on ramp
[54]	DTDE-MARL	≤ 4	Collision rates	Parallel-type on ramp
[71, 72]	DTDE-MA-A2C	Not mentioned	altruistic behavior in AVs	Parallel-type on ramp
[59]	DTDE-DQN	≤ 5	altruistic behavior in AVs	Parallel-type on ramp
[146]	DTDE-MA2C	≤ 8	Safety and efficiency	Parallel-type on ramp
[58]	DTDE-MA2C	≤ 6	Safety and efficiency	Highway lane change
[147]	DTDE-MARL	≤ 3	Collisions rate	Taper-type on ramp
[145]	CTDE-MADDPG	≤ 8	Eliminate stop and go	Taper-type on ramp
[41]	CTCE	≤ 30	Safety and efficiency	Taper-type on ramp
[74]	CTDE-MADDPG	≤ 8	Outflow, inflow, and speed	Taper-type on ramp
Our approach	CTDE	50	Comfort, fuel, traffic flow, lane distribution fairness, position shift fairness, and speed	Road reduction (Lane-drop)

merging vehicles when a merging lane AV asks to merge. To mimic real-world conditions, vehicles enter the highway randomly from either lane, creating a dynamic environment with a varying number of learning agents. This approach tests the scalability of MARL algorithms to changing traffic conditions and evaluates the generalization capabilities of the learned policies. The simulation uses SUMO's Traffic Control Interface (TraCI) [148] to facilitate interactions between the environment and RL agents. Vehicle longitudinal dynamics are governed by the Krauss car-following model [149], lane change decisions are made by RL agents, and the lateral dynamics are controlled using SUMO's default model [150].

The complexity of this problem arises from several factors:

- Multiple agents: Each AV is an independent agent that must make decisions based on local information while considering the actions of other vehicles.
- Efficiency goals: The merging process should minimize traffic slowdowns and maintain a smooth flow of vehicles through the bottleneck.
- Partial observability: Each vehicle has limited information about the state of other vehicles and must make decisions based on this partial view of the environment.
- Dynamic environment: The number of vehicles involved in the merging process can vary over time, requiring a flexible and scalable solution.

5.1.1 Environment Overview

The problem is modeled as a decentralized partially observable MDP (Dec-POMDP) [151] defined by the tuple: $(\mathbf{S}, \mathbf{O}, \mathbf{A}, r, p, \gamma)$. Here, $\mathbf{S}$ is the state space, $\mathbf{O} : \mathbf{O_1} \times \ldots \times \mathbf{O_n}$ is the joint local observation space of all agents n, $\mathbf{A} : \mathbf{A_1} \times \ldots \times \mathbf{A_n}$ is the joint action space for all agents n, $p(S' \mid S, A)$ is the transition probability to state S' given the current S and action $A = (a_1, \ldots, a_n)$, γ is the discount factor, and r is the immediate

shared reward received when taking action A in state S. This work uses an MARL CTDE framework, adopted from [152], for multi-agent automated traffic control with a dynamic number of agents. During the training phase, the critic has access to global information through a central coordinator. However, in the execution phase, each agent can only observe a limited part of the environment, which aligns with realistic connected vehicle scenarios. Specifically, each agent's local observation $o_i \in \mathbf{O_i}$ consists of information from nearby vehicles within a predefined communication range, representing vehicle-to-vehicle (V2V) connectivity limitations. This partial observability during execution ensures that the learned policies are applicable in real-world settings where full environmental information is not available to individual vehicles.

5.1.1.1 State Space In this work, the state of the environment at time t, S_t, is represented by the joint local observations of all active agents K_t:

$$S = [o_1, o_2, ..., o_{K_t}] \tag{5.1}$$

Note that since the number of agents is dynamic, K_t represent the participating agents at time t. The local observation of each agent i, denoted as o_i, is defined as a vector of dimension $4 + 3N^i$, where N^i is the number of observed surrounding vehicles within an 8-meter range in front of and behind the AV. Fig. 5.2 illustrates this local observation space for each AV. The local observation vector of AV i is represented as:

$$o_i = [e_i, x_1, x_2, ..., x_{N^i}], \tag{5.2}$$

where:

- $e_i = [v_i, d_i, m_i]$ represents the ego vehicle's state:
  - v_i: speed of the ego vehicle
  - d_i: distance to the merge point

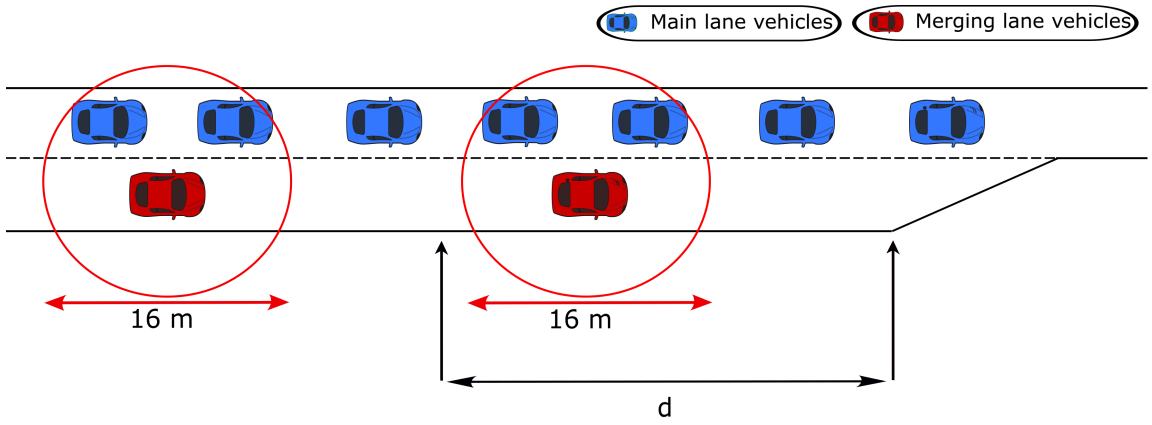

Figure 5.2: Illustration of the local observation space for an AV. Each AV can observe neighboring vehicles within a total 16-meter range.

  - m_i: merge state (1 if merged, 0 if not yet merged)

- $x_j = [\Delta x_j, v_j, \Delta v_j]$ for $j = 1, ..., N^i$ represents the state of each observed vehicle:

  - Δx_j: relative longitudinal position to the ego vehicle
  - v_j: speed of the observed surrounding vehicle
  - Δv_j: relative speed to the ego vehicle

This local observation o_i is normalized and then passed to the actor network.

5.1.1.2 Action Space The action space A_i of agent i is defined as a binary set representing high-level lane change decision:

$$A_i = \begin{cases} 1 & \Rightarrow \text{ Request to merge} \\ 0 & \Rightarrow \text{ Stay in current lane.} \end{cases} \tag{5.3}$$

With a selected high-level decision, lower-level controllers in the SUMO simulation environment produce the corresponding steering and throttle control signals to maneuver

the AVs. The overall action space of the system is the joint actions from all active k AVs at time step t:

$$A = A_1 \times A_2 \times \cdots \times A_{k_t} \tag{5.4}$$

This formulation allows for a simplified yet effective representation of the key decisions required in the merging scenario, while delegating the detailed vehicle control to the simulation environment. This adopted formulation is widely used in the literature, such as [153], [146], [71, 72], and [59].

5.1.2 Reward Functions

In the proposed CTDE framework, a shared reward structure is employed, where a normalized global reward is received at each time step to indicate the quality of the joint actions taken at that time step. The general form of the reward function is:

$$r_t = f(s_t, \mathbf{a_t}) \tag{5.5}$$

where s_t is the current state, $\mathbf{a_t}$ is the joint action of all agents. In this study, various reward function designs are explored. Detailed description of these reward functions are presented in Section 5.2.

5.1.3 MARL Algorithm

We propose an MARL algorithm designed to address the challenges of scalability, non-stationarity, and dynamic agent numbers in a road reduction environment. The proposed algorithm is shown in Algorithm 5.1. Our approach utilizes a CTDE framework, as illustrated in Fig. 5.3, consisting of decentralized actors for each agent, a centralized critic, and a centralized counterfactual baseline network [92]. This structure allows us to

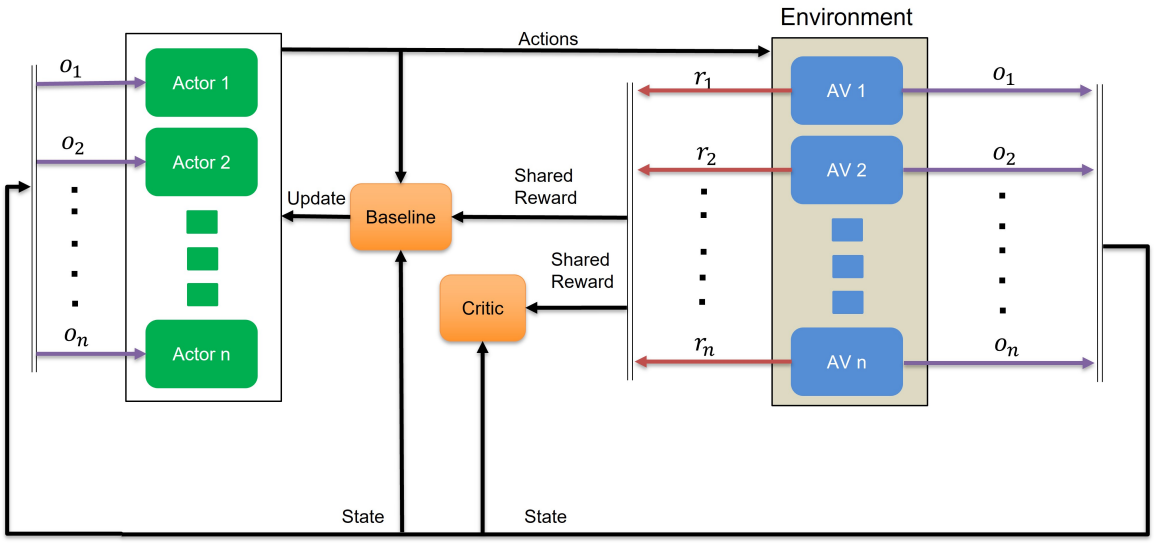

Figure 5.3: CTDE Actor-Critic MARL Architecture adopted in the proposed merging control framework.

maintain the benefits of decentralized policies while leveraging global information during training to mitigate non-stationarity issues.

5.1.3.1 Centralized Critic and Counterfactual Baseline The centralized critic estimates the value function $V_\pi(S_t)$, representing the expected cumulative discounted reward from the global state S_t under the joint policy π. The critic takes the global state S_t as input and outputs a scalar value estimate.

However, when using a centralized critic network with a global reward to guide the agents' behavior, a new challenge arises: the credit assignment problem. The global reward is given to a joint set of actions that moved all the agents from state S_t to state S_{t+1} in terms of the global environment state space. As a result, calculating each agent's individual contribution to the global reward becomes a complex task.

To address this challenge, a counterfactual baseline is introduced:

$$b^i_{\psi,t}(S,\mathbf{a}) = \mathbb{E}_{a_i\sim\pi(\cdot|o_i)}\left[Q_\pi\left(S_t,(\mathbf{a}^{-\mathbf{i}},a_i)\right)\right], \tag{5.6}$$

where $\mathbf{a}^{-\mathbf{i}}$ represents the actions of all agents except agent i, and o_i is the observation of agent i. The advantage A^i_t quantifies how much better (or worse) the chosen action of agent i is compared to the average action, given the actions of other agents.

$$A^i = Q_\pi(S,\mathbf{a}) - b_\psi(S,\mathbf{a}), \tag{5.7}$$

where the Q-value function $Q_\pi(S_t,\mathbf{a_t})$ is defined as:

$$Q_\pi(S_t,\mathbf{a_t}) = E_\pi[r(S_t,\mathbf{a_t}) + \gamma V_\pi(S_{t+1})], \tag{5.8}$$

with $r(S_t,\mathbf{a_t})$ being the global reward function.

The centralized critic is updated to minimize the following mean squared error loss:

$$L_c(\phi) = \sum_t \left[r(S_t,\mathbf{a}_t) + \gamma V_\phi(S_{t+1}) - V_\phi(S_t)\right]^2 \tag{5.9}$$

Similarly, the baseline network b_ψ is updated using a similar mean-squared error loss:

$$L_b(\psi) = \sum_t \left[\sum_{i=1}^{K_t}\left[b_\psi(S_t,\mathbf{a}_t^{-i}) - \left(r_t + \gamma V_\phi(S_{t+1})\right)\right]^2\right]. \tag{5.10}$$

5.1.3.2 Actor Network Each agent i is represented by an actor network parameterized by θ. The actor's objective function is defined as:

$$\begin{aligned} J(\theta) = E_{\pi_\theta}\Big[\sum_{i=1}^{N}\sum_t \min\big(&\Gamma^i_t(\theta)A^i_t, \\ &\text{clip}(\Gamma^i_t(\theta),1-\varepsilon,1+\varepsilon)A^i_t\big) \\ &+\beta H(\pi_i(o_i))\Big], \end{aligned} \tag{5.11}$$

Algorithm 5.1: CTDE-MARL AVs Merging Control

Initialize weights: Critic ϕ, Baseline ψ, Policy θ
for *each iteration* **do**
  while *nsteps* $<$ *buffer size* **do**
    Observe state S, local observation o, active agents k_t
    for *each agent i in* K_t **do**
      Sample action $a_i \sim \pi_\theta(a_i|o_i)$
    end
    Take joint action $\mathbf{a}$, observe r, S'
    Store $(S, O, \mathbf{a}, R, S', k_t)$ in buffer $\mathscr{B}$
    $V_t = \text{Critic}_\phi(S)$
  end
  Compute discounted returns G_t for every state S_t
  Compute Critic loss $L_c(\phi)$
  Update ϕ to minimize $L_c(\phi)$
  for *time step t* **do**
    for *each agent j in* K_t **do**
      Calculate Baseline $b^j_\psi(o_j, (o_i, a_i) 1 \leq i \leq k_t, i \neq j)$
      Compute advantage $A_j = G_t - Q^j_\psi$
    end
  end
  Compute loss $L_b(\psi)$
  Update ψ to minimize $L_b(\psi)$
  Compute $J(\theta)$ using A_j
  Update θ to maximize $J(\theta)$
end

where $\Gamma_t^i(\theta)$ is the probability ratio defined as $\frac{\pi_\theta(a_t^i|o_t^i)}{\pi_{\theta_{old}}(a_t^i|o_t^i)}$, β is the entropy coefficient, ε is a clipping parameter, H is the policy entropy given by $H(\pi_i(o_i)) = E_{\pi_i}[-\log(\pi_i(o_i))]$, and A_t^i is the advantage function for agent i at time t.

5.1.3.3 Neural Networks In this study, we adopt a framework that utilizes both centralized and decentralized neural networks. The centralized networks are the critic and baseline networks, while each actor has its own decentralized neural network. Given that our agents (AVs) are homogeneous with identical local state space and action space, we employ parameter-sharing. This technique, where all agents share the same policy network parameters, has been shown to enhance learning efficiency [154].

5.1.3.3.1 Actor Network Architecture The decentralized actor network is designed to efficiently process the local observations of each agent. The observation vector o_i, which includes information about the ego vehicle and nearby vehicles within an 8-meter range, is fed directly into the network as illustrated in Fig. 5.4. This observation vector is processed through a series of fully connected layers to extract relevant features and capture complex relationships within the state information and producing a set of logits z_i. These logits are then passed through a Softmax layer, yielding the action probability distribution: $\pi_\theta(o_i) = \text{softmax}([z_1, z_2])$. The final action a_i is sampled from this probability distribution: $a_i \sim \pi_\theta(o_i)$.

5.1.3.3.2 Critic Network Architecture For the centralized critic network, the input is a concatenation of all agents' local observations at each time step. However, our environment presents a challenge: the number of participating AVs fluctuates as vehicles complete merges or enter the highway, resulting in a variable global state size. To address this issue, we propose a self-attention based architecture. Self-attention layers can handle inputs of varying lengths, allowing our model to process observations from a changing

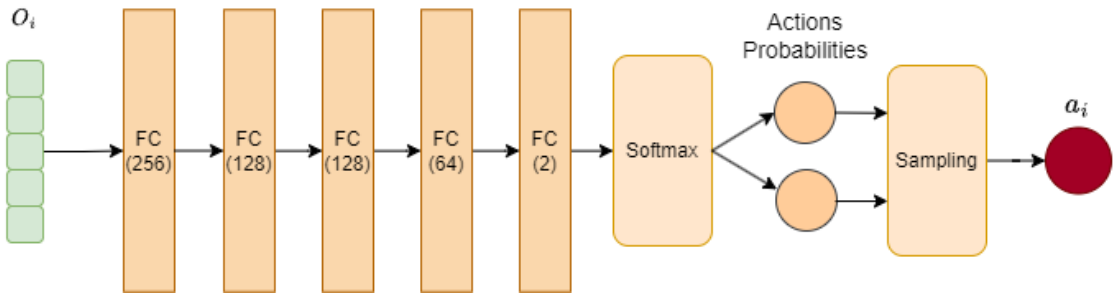

Figure 5.4: Actor network architecture.

number of agents. This approach is also applied to the centralized baseline, which takes as input the concatenation of local observations and actions from active agents. The self-attention mechanism, introduced by Google Research and Google Brain, is a powerful technique used in deep learning and natural language processing (NLP) tasks [155]. It has demonstrated effectiveness in various applications, including computer vision [156], semantic segmentation [157], and object detection [158]. In the self-attention mechanism, each element in the input becomes a query, key, and value. These are derived from the input embeddings by multiplying them with a weight matrix. The attention function can be described mathematically as:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{5.12}$$

where Q, K, and V represent the query, key, and value matrices respectively, and d_k is a scaling factor.

The critic network begins with a feature extractor that processes each observation individually using a fully connected layer followed by layer normalization (see Fig. 5.5). This initial processing allows the network to extract relevant features from each agent's observation independently. The output of this stage is then fed into the self-attention mechanism, comprising query, key, and value vectors, which feed into a Multi-head

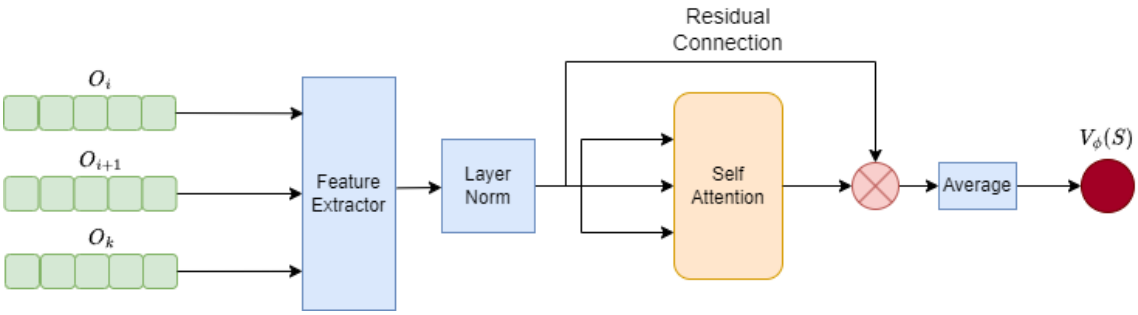

Figure 5.5: Critic network architecture.

Attention layer. The self-attention layer enables the network to dynamically weigh the importance of different agents' information, adapting to the changing number of agents in the environment. Following the attention mechanism, a residual connection combines the output of the attention layer with the encoded observations, allowing the network to preserve important individual agent information alongside the relational data. The final stages of the critic network include an average layer, which average the processed information across all agents, and output a scalar. This structure allows the critic to effectively evaluate the global state while accommodating a variable number of agents.

5.1.3.3.3 Baseline Network Architecture The baseline network shares a similar structure with the critic but incorporates some distinct features to process both observations and actions. The baseline is used to estimate $Q_\pi(S_t, \mathbf{a_t^{-i}})$, which can be defined as:

$$Q_\pi(S_t, \mathbf{a_t^{-i}}) = V_\pi(o_i) + Q_\pi(S_t^{-i}, \mathbf{a_t^{-i}}). \tag{5.13}$$

Hence, baseline network has two inputs: the local observation of agent i, and the observation-action pair of all of the other active agents. See Fig. 5.6. Each input will go through a different feed forward layer. Following that, the baseline network employs a self-attention mechanism similar to the critic network, allowing the baseline to

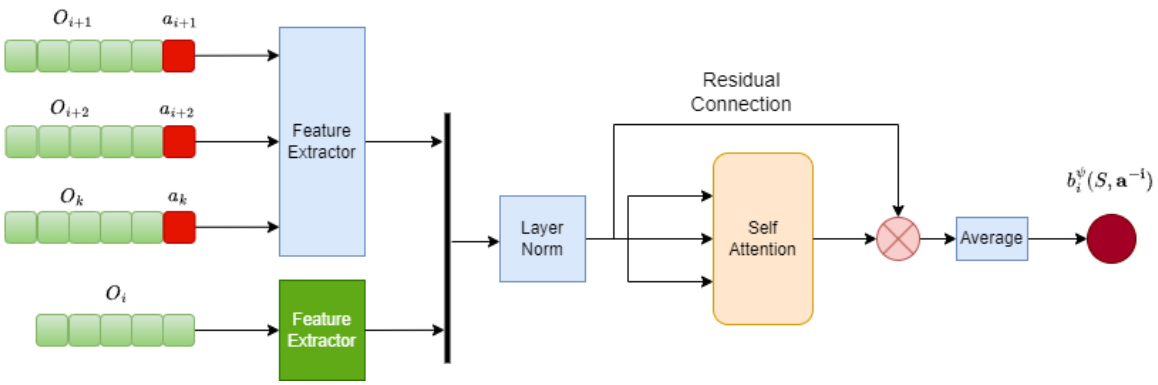

Figure 5.6: Baseline network architecture.

consider relationships between different agents' observations and actions when estimating the baseline value. The output of the attention mechanism is then processed through additional layers, including a residual connection and average layer, before producing the final baseline estimate. This architecture enables the baseline network to generate accurate estimates while adapting to the varying number of agents in the environment.

5.2 Reward Functions

Extensive experiments are conducted to assess the performance of the proposed approach under various reward functions formulations. In particular, the following reward functions are explored.

- M1: Local speed with safety optimization
- M2: Global speed with safety optimization
- M3: Local speed with jerk minimization
- M4: Local speed with fuel consumption minimization

5.2.1 M1: Local Speed with Safety Optimization

This reward function aims to optimize the speed and safety based on local observations of AVs. In order to ensure safety, agents will be penalized for getting too close to the road reduction point. The safety penalty will start at a specific threshold and increase as the agent gets closer to the road reduction point. The speed term is based on the average speed of the AV and its surrounding vehicles. For an active agent j, the local speed reward and safety reward are defined as follows.

$$r^j_{\text{local speed},t} = \frac{1}{|N^j_t|+1} \sum_{i \in j \cup \mathcal{N}^j_t} s^i_t \tag{5.14}$$

$$r^j_{\text{safety},t} = \begin{cases} -\left(\frac{x^j_t - d}{d}\right)^2, & \text{if } x^j_t \leq d \\ 0, & \text{otherwise,} \end{cases} \tag{5.15}$$

where N^j_t is the set of observed surrounding vehicles of agent j at time t with $|N^j_t|$ being its cardinality of the set, s^i_t is the speed of vehicle i at time t, x^j_t is the longitudinal distance from the agent j to the road reduction point, and d is a predefined threshold distance. For example, for the leftmost merging vehicle in Fig. 5.2, $r_{\text{local speed},t}$ corresponds to the average speed of the three vehicles in the leftmost circle. In addition, for the left merging vehicle, $r_{\text{safety},t} = 0$ as it is outside of the predefined threshold d, while the right merging vehicle, $r_{\text{safety},t}$ is non-zero.

Therefore, for M1 model, the reward can be defined as follows.

$$r_t = \frac{1}{K_t} \sum_{j=1}^{K_t} \left(w_1 r^j_{\text{local speed},t} + w_2 r^j_{\text{safety},t} \right), \tag{5.16}$$

where w_1 and w_2 are weighting factors and K_t is number of active agents at time t.

5.2.2 M2: Global Speed with Safety Optimization

Different from M1, this reward function is designed to maximize the global speed. The reward function incorporates two key components: the average speed of all vehicles in both lanes that are before the road reduction point, and a safety reward that discourages the agents from approaching the merge point too closely. Resultant reward can be defined as:

$$r_t = \frac{1}{C_t} \sum_{n=1}^{C_t} w_1 s_t^n + \frac{1}{K_t} \sum_{j=1}^{K_t} w_2 r_{\text{safety},t}^j, \tag{5.17}$$

where C_t is the total number of vehicles in the environment at time t. It's important to note that the safety term is zero for vehicles in the main lane, so the second term only sums over all active agents at time t.

Remark 1 *The key difference between the reward function (5.16) of M1 and the reward function (5.17) of M2 lies in the treatment of vehicles in the main lane. Take Fig. 5.2 for example, the vehicles in the main lane that do not belong to any circle will not contribute to the calculation of (5.16), but they do impact the reward of (5.17). The idea here is that, (5.17) aims to capture the impacts of RL decisions on the entire traffic by including all vehicle speed, hence referred to as "global speed". On the other hand, (5.16) captures the vehicle speed that are directly impacted by RL decision. For example, if a vehicle in the main lane is far away from any merging vehicles, its speed may not be directly impacted (or only minimally impacted) by the merging decision, and therefore are excluded in the reward function. Therefore, (5.16) is referred to as "local speed".*

5.2.3 M3: Local Speed with Jerk Minimization

M3 is a dual-objective reward function designed to maximize local speed while simultaneously minimizing vehicle jerk. The reward function balances two key components: the average speed of all agents and its immediate neighbors, and a term for

abrupt changes in acceleration (jerk). Research has shown that motion sickness can be induced by repetitive exposure to low-frequency motions [138], while lower back pain can be induced by regular exposure to high-frequency motions [139, 140]. The AVs behavior can be optimized to ensure comfortable driving by minimizing the jerk, which correlates to these distress and sudden acceleration changes [32]. Total reward can be defined as:

$$r_t = \frac{1}{K_t} \sum_{j=1}^{K_t} \left(w_1 r^j_{\text{local speed},t} + w_2 r^j_{\text{local jerk},t} \right) \tag{5.18}$$

with the local jerk defined as

$$r^j_{\text{local jerk},t} = -\frac{1}{|N^j_t| + 1} \sum_{l \in j \cup \mathcal{N}_t{}^j} \left(\frac{acc^l_t - acc^l_{t-1}}{dt} \right)^2, \tag{5.19}$$

where acc^l_t is the acceleration of vehicle l at time t. This combination encourages smooth, efficient movement while maintaining a comfortable ride for passengers.

5.2.4 M4: Local Speed with Fuel Consumption Minimization

The goal of M4 is to minimize fuel consumption and optimize local speed at the same time. The reward function combines the average speed of the ego vehicle and its immediate neighbors, with a term for excessive fuel usage expressed by promoting both efficient movement and environmentally friendly driving behaviors. The fuel consumption evaluation is based on SUMO's PHEMlight model [159], which calculates fuel consumption as a function of the vehicle's current power demand. The power demand P is computed as:

$$P = (P_{Roll} + P_{Air} + P_{Accel} + P_{Grad})/\eta_{gb}, \tag{5.20}$$

where P_{Roll} is the power needed to overcome rolling resistance, P_{Air} is the power needed to overcome air resistance, P_{Accel} is the power needed for acceleration, P_{Grad} is the

Table 5.2: Overview of the Four RL Models Evaluated in This Chapter.

Model Label	Reward Function	Weights	Results
M1	Local speed with safety optimization	$w_1 = 1$ & $w_2 = 3$	Tables 5.3-5.5; Figs. 5.7, 5.11 & 5.12
M2	Global speed with safety optimization	$w_1 = 1$ & $w_2 = 3$	Tables 5.3-5.5; Figs. 5.8, 5.11 & 5.13
M3	Local speed with jerk minimization	$w_1 = 1$ & $w_2 = 0.4$	Tables 5.3-5.5; Figs. 5.9, 5.11 & 5.14
M4	Local speed with fuel consumption minimization	$w_1 = 1$ & $w_2 = 1$	Tables 5.3-5.5; Figs. 5.10, 5.11 & 5.15

power needed to overcome road gradient, and η_{gb} is the drivetrain efficiency. The fuel consumption rate is then determined using characteristic curves specific to each vehicle type, which is a function of power demand, denoted as $f(P)$.

The total reward can be expressed as:

$$r_t = \frac{1}{K_t} \sum_{j=1}^{K_t} \left(w_1 r^j_{\text{local speed},t} + w_2 r^j_{\text{local fuel},t} \right), \tag{5.21}$$

where the local fuel consumption is defined as

$$r^j_{\text{local fuel},t} = -\frac{1}{|N^j_t| + 1} \sum_{i \in j \cup \mathcal{N}^j_t} f(P^i_t). \tag{5.22}$$

5.3 Results and Discussion

This section presents a comprehensive evaluation of the proposed MARL framework for AVs merging control in highway scenarios (described in Section 5.1). To benchmark our method, its performance is compared against (1) the Multi-Agent Proximal Policy Optimization (MAPPO) algorithm [154], a well-established approach in MARL, and (2) a rule-based zipper merge strategy [160, 161]. Four RL models are analyzed, each with one of the reward functions discussed in Section 5.2. The label of each model, the corresponding reward function and results, are summarized in Table 5.2.

5.3.1 Simulation Setup

We employed an on-policy training algorithm, as detailed in Algorithm 5.1. Each training rollout consists of 6 episodes (around 6000 steps), allowing the agents to experience a wide range of traffic scenarios and learn effective merging strategies. Each time step (sampling time) is 0.2 second. To ensure the robustness of our approach, we conducted training using 8 different random seeds, to assess the consistency and reliability of the results. That is, 8 separate trainings each with different initialization of weights and randomness for both the MARL algorithm and SUMO sides.

The simulations were conducted using high-performance computing (HPC) nodes running Red Hat Enterprise Linux release 8.6 (Ootpa). Each node was equipped with 192 GB of RAM and 40 CPU cores operating at 2.50 GHz, providing the necessary computational power to handle the complex MARL training process. The SUMO simulator was interfaced with our MARL implementation using TraCI (Traffic Control Interface) [148], allowing seamless interaction between the RL agents and the simulated environment.

5.3.2 Training Convergence Analysis

The learning performance of our proposed MARL framework across four different reward functions is evaluated in comparison to the benchmark MAPPO algorithm [154]. In Figs. 5.7-5.10, the x-axis represents the number of training rollouts, while the y-axis shows the average reward achieved. Our proposed model is represented by the red line, while MAPPO is shown in blue. The shaded areas around each line indicate the variance in performance across multiple random seeds.

For M1, the proposed approach demonstrates faster convergence and higher final rewards compared to MAPPO, shown in Fig 5.7. The agents are able to learn effective merging policies that balance safety and efficiency, achieving an average episodic reward of approximately 87.2 after around 2300 training updates. In contrast, MAPPO converges

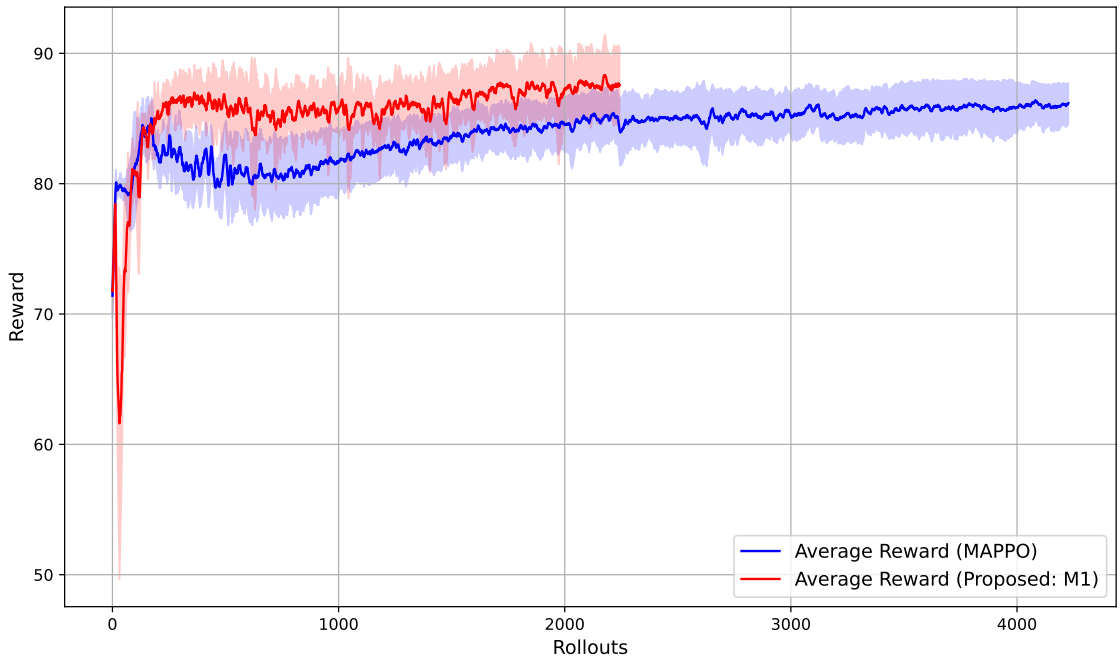

Figure 5.7: Convergence comparison of the proposed method (M1) and the benchmark MAPPO method.

to a lower episodic reward of approximately 85.3 and requires nearly twice as many training updates. However, despite our model requiring fewer updates to converge, it utilizes an additional centralized baseline network. As a result, the overall training time to convergence is comparable to MAPPO.

For M2, Fig. 5.8 shows a clear upward trend in average reward in the proposed method, converging to approximately 217. This performance significantly surpasses that of the MAPPO approach, which converges to a lower value of around 195. Such a substantial improvement in convergence value indicates that our model can more effectively optimize global speed while maintaining safety constraints.

Fig. 5.9 illustrates the progression of the average reward for over the course of training rollouts for both the proposed M3 model and the MAPPO benchmark. The

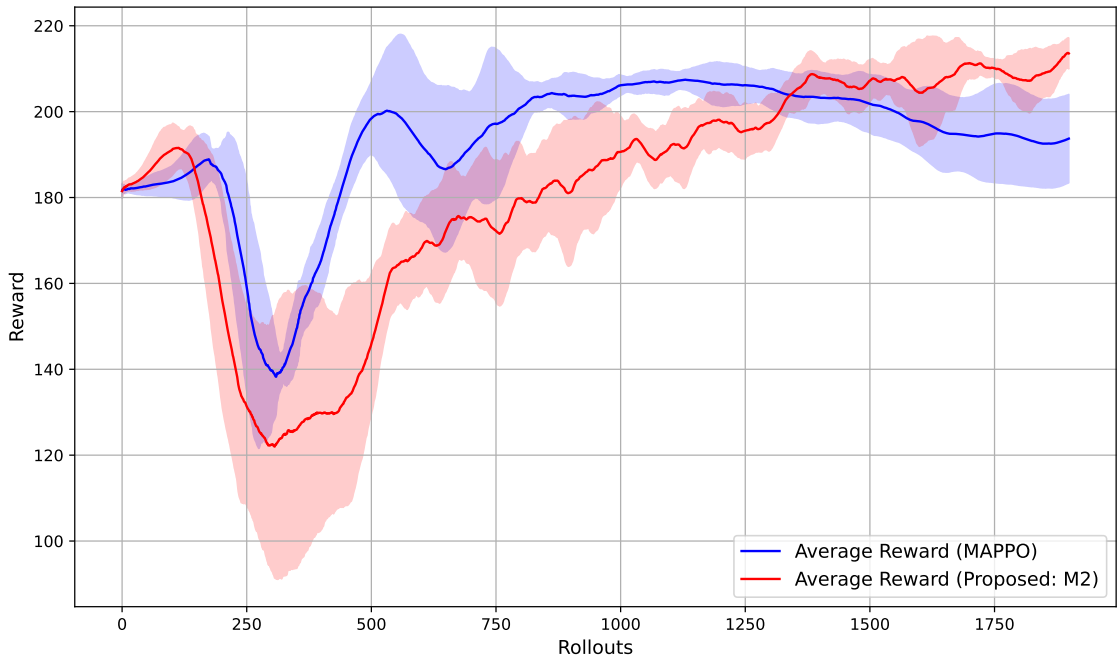

Figure 5.8: Convergence comparison of the proposed method (M2) and the benchmark MAPPO method.

results demonstrate a clear advantage of our proposed model over the MAPPO benchmark. Both approaches show rapid initial improvement, indicating quick learning in the early stages of training. However, our model (represented by the red line) consistently outperforms MAPPO (blue line) throughout the training process. By the end of the training period (around 700 rollouts), M3 model achieves and maintains an average reward of approximately 320, while MAPPO plateaus at a lower value of about 310. Notably, in addition to the faster convergence, the proposed M3 model also exhibits greater robustness, as evidenced by the narrower shaded area which suggests that the proposed approach has less training variance. Finally, it is also worth mentioning that MAPPO method seems to be unstable in this case study, as the reward starts to continusouly decline after 600 rollouts.

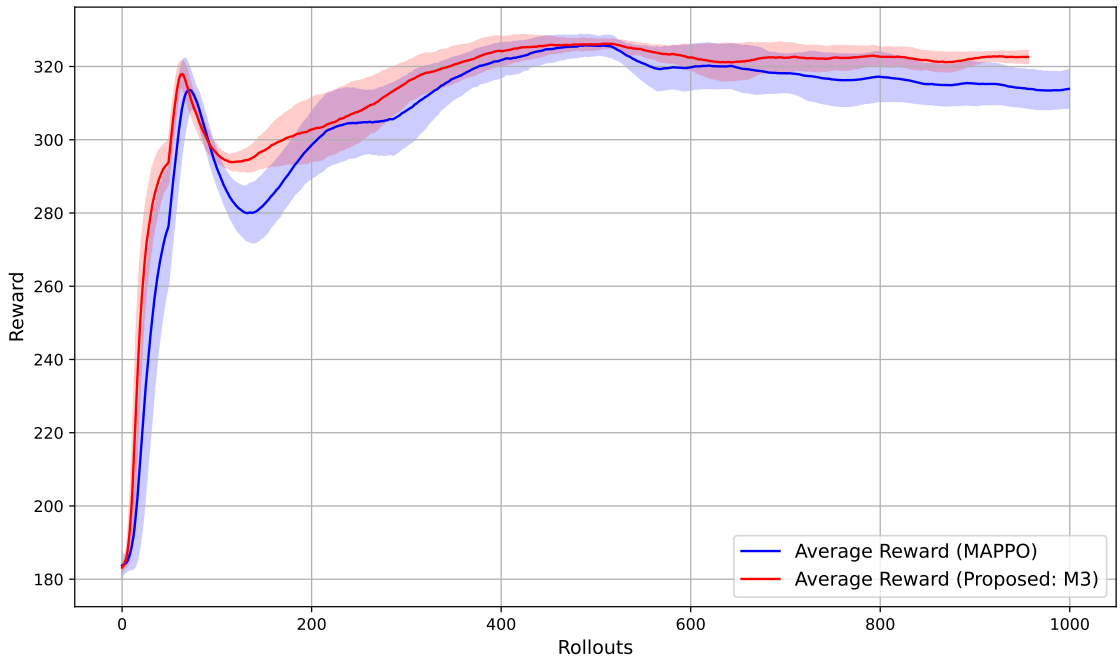

Figure 5.9: Convergence comparison of the proposed method (M3) and the benchmark MAPPO method.

Lastly, the training performance of the proposed M4 model in comparison to the MAPPO benchmark is shown in Fig. 5.10. By the end of the training period, which spans approximately 800 rollouts, our model attains an average reward of about 76 surpassing that of MAPPO, which reaches a slightly lower value of around 72. This shows that the proposed M4 model is able to effectively balance the objectives of speed optimization and fuel efficiency, as indicated by the higher reward.

Overall, the proposed framework exhibits higher training efficiency, as evidenced by the shorter convergence time and the higher reward. In all four models, the proposed method achieve smaller variance, and hence is more robust. Moreover, as can be seen from Figs. 5.8 and 5.9, MAPPO suffers stability issue where the reward starts to decline, while the proposed method demonstrates greater stability in all four case studies.

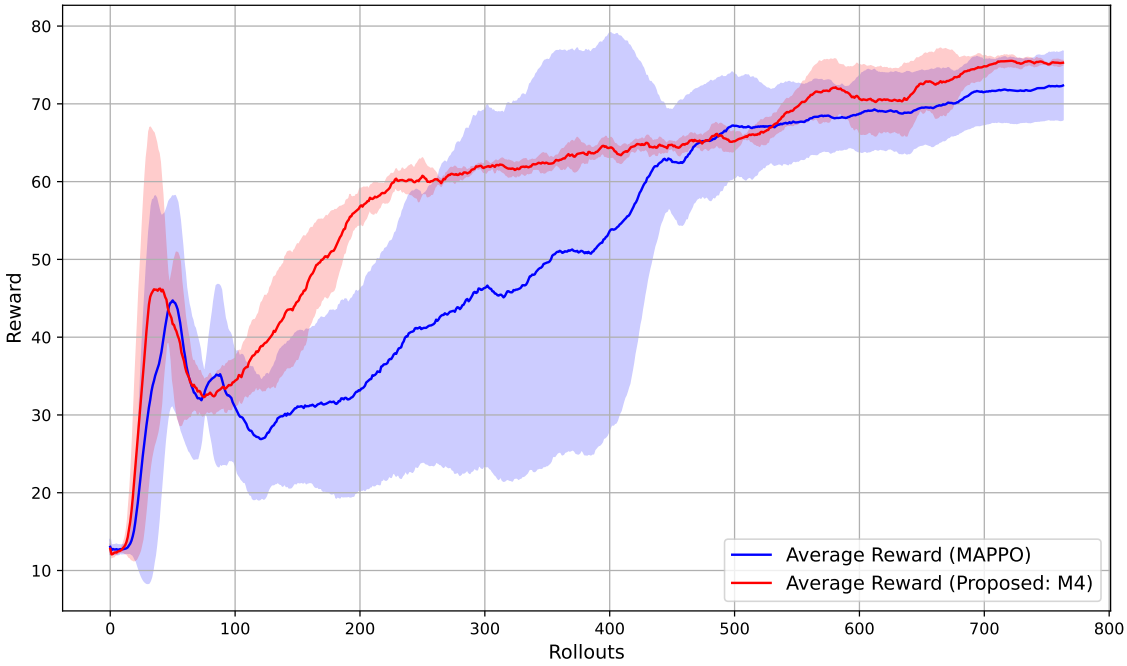

Figure 5.10: Convergence comparison of the proposed method (M4) and the benchmark MAPPO method.

Table 5.3: Evaluation Results of M1-M4 Models with 10 m/s Maximum Speed.

Model	Flow [$\#/h$]	Fuel [mg/s]	Avg Jerk [m/s^3]	Avg Speed [m/s]
M1	1488.83	122701.95	4.01	**5.3**
M2	**1498.75**	124711.08	3.44	5.23
M3	1430.84	159143.18	**3.09**	4.18
M4	1482.7	**114995.16**	3.93	5.17
Zipper Merge	1304.35	169540.15	3.45	4.02

5.3.3 Merging Performance Comparison

To assess the effectiveness of the proposed MARL approach, we deployed the best-performing trained models for evaluation. Specifically, for each reward model (M1-M4), the saved model checkpoint that achieved the highest rewards during training

Table 5.4: Evaluation Results of M1-M4 Models with 20 m/s Maximum Speed.

Model	Flow [$\#/h$]	Fuel [mg/s]	Avg Jerk [m/s^3]	Avg Speed [m/s]
M1	1986.75	169531.47	3.66	9.31
M2	**2135.23**	183102.56	3.816	**11.46**
M3	1595.74	186751.38	**3.01**	5.72
M4	1803.61	**155711.8**	3.65	8.01
Zipper Merge	1333.33	21265.34	3.1	4.54

is then tested in various conditions to quantitatively compare the merging performance. To validate the robustness and generalizability of the proposed approach, the model training was conducted with a maximum speed of 10 m/s and the evaluation was conducted at both 10 m/s and 20 m/s. Notably, the performance trends remain consistent across both evaluation conditions, demonstrating the robustness of the proposed approach. In addition, a rule-based zipper merge strategy [160, 161] is also evaluated as benchmark.

The performance evaluation is conducted using a suite of metrics designed to capture various aspects of merging efficiency and safety, including:

- Traffic flow (unit: $vehicles/hour$).
- Fuel consumption (unit: mg/s): The average fuel consumption of all vehicles before the road reduction at each time step and then averaged by the total time steps.
- Average jerk (as a measure of ride comfort; unit: m/s^3): The average jerk of all vehicles before the road reduction at each time step and then averaged by the total time steps.
- Average speed (unit: m/s): The average speed of all vehicles before the road reduction at each time step and then averaged by the total time steps.

These metrics provide a holistic view of the effectiveness and efficiency of our approach in managing highway merging scenarios. Note that when calculating these metrics, all

vehicles in both lanes will be included, and hence different from some of the rewards function used for training. This is because when analyzing the performance, we want to use the metrics that correspond to the overall system efficiency, even if some vehicles are not observed by any local agents.

Tables 5.3 and 5.4 present the evaluation results for both evaluation conditions, i.e., 10 m/s and 20 m/s maximum speeds, respectively. Across both scenarios, all MARL models consistently outperform the rule-based zipper merge strategy in most metrics, underscoring the advantages of our learning-based approach. Note that performance of the benchmark, i.e., the zipper merge strategy, is presented in the last row of each table. With 10 m/s evaluation condition, (Table 5.3), M2 model achieves the highest flow rate of 1498.75 vehicles per hour, a 14.9% improvement over the zipper merge baseline. M1 model, while slightly behind in flow rate, achieves the highest average speed of 5.3 m/s. Additionally, M4 model demonstrates the best fuel efficiency, consuming only 114,995.16 mg/s, a 32.17% reduction compared to the zipper merge baseline.

The performance improvements further increase at higher speeds, as evident in Table 5.4. With $20m/s$ evaluation condition, M2 model, whose reward function focuses on global speed, achieves a remarkable flow of 2135.23 vehicles per hour, a 60.14% improvement over the zipper merge baseline. It also maintains the highest average speed of 11.46 m/s, significantly outpacing other approaches. These results suggest that optimizing the global speed leads to more efficient overall traffic management, especially in high-speed scenarios. However, the superior speed and flow of the global speed model come with a trade-off in fuel consumption. The speed with M4 models continues to demonstrate the best fuel efficiency among MARL models, achieving a 14.96% reduction in fuel consumption compared to M2 model. This highlights the importance of considering multiple objectives in AV control.

Consistently across both evaluation conditions, the M3 model, which combines local speed and jerk in the reward function, achieves the lowest average jerk, i.e., 3.09 m/s^3 and 3.01 m/s^3, respectively). While this potentially offers the smoothest ride for passengers, it comes at the cost of reduced traffic flow and average speed. Such trade-off between ride comfort and system efficiency is an important consideration in real-world applications. Finally, it's noteworthy that despite being trained at 10 m/s, the proposed MARL models generalize very well to the 20 m/s condition, maintaining their relative performance and consistently outperforming the zipper merge benchmark.

5.3.4 Merging Strategy Visualization

In this and the following section, we evaluate the behavior of various trained models in terms of their merging strategies, focusing on aspects such as merging positions, vehicle order post-merge, and the fairness of these strategies. By quantifying these metrics, we aim to understand the strengths and weaknesses of each model's approach to traffic management. To visualize the merging position, Fig. 5.11 presents a heat map of the vehicle merging zone. This spatial representation indicates where merging decisions most frequently occur, with the intensity of color in different areas reflecting the concentration of merging actions. As can be seen, M1 model implements a merging strategy that is both earlier and distributed, demonstrated by the heat map where the merging activities are distributed across a broad region with greater concentrations observed during the early region of the merging zone, i.e., between approximately 100m and 160m along the merging zone. For M2 model, the optimal merging strategy demonstrates a clear preference for late merging behavior. As evidenced in Fig. 5.11, the heat map shows a high concentration of merging activities towards the end of the merging zone, i.e., between 250-300m along the merging zone. Such a late merging behavior suggests that M2 model encourages vehicles to utilize both lanes for as long as possible before executing merge maneuvers, which can

help maintain higher flow rate. For M3 model, the heat map shows intense merging activity concentrated between 250m and 300m in the merging zone, indicating a late, clustered merging strategy. This approach allows for the formation of vehicle groups before merging, potentially contributing to smoother overall traffic flow despite the significant reordering of vehicles. See the next section for more details on this regard. Finally, M4 model presents a balanced approach among the extremes observed in the other models. Fig. 5.11 reveals a more even distribution of merging activities along the merge zone, but with a noticeable concentration towards the 285-300m range.

Another attempt to visualize the merging strategies is done in Figs. 5.12-5.15, which depict the relationship between the initial order of vehicles and their final position in the platoon after the merge. In the top portions of these figures, the x-axis represents the order in which vehicles finished the merge, where zero (far right) represents the first vehicle to finish the merge, and 99 represents the last. The y-axis (and also the numerical labels placed on each data point) denotes the vehicle entering sequence, with lower values indicating earlier entry times. Additionally, the color of the points denotes the lane from which the vehicle first entered the traffic, with red corresponding to the Main lane and blue representing the merging lane. To provide a reference for comparison, a dotted diagonal line is included in these plots to represent the ideal zipper merge strategy. This line illustrates the expected final positions of vehicles if they were to merge in a perfect alternating pattern from the main and merging lanes, i.e., using zipper merge strategy. The deviation of the actual data points from this diagonal line visualizes how the learned merging strategies differ from the zipper merge approach. For example, if a vehicle is above the dotted line (e.g., vehicle 14 in Fig. 5.12), it means this vehicle has sacrificed its position to either let other vehicles pass. Similarly, if a vehicle is below the dotted line (e.g., vehicle 26 in Fig. 5.12), it means this vehicle has gained advantage by surpassing

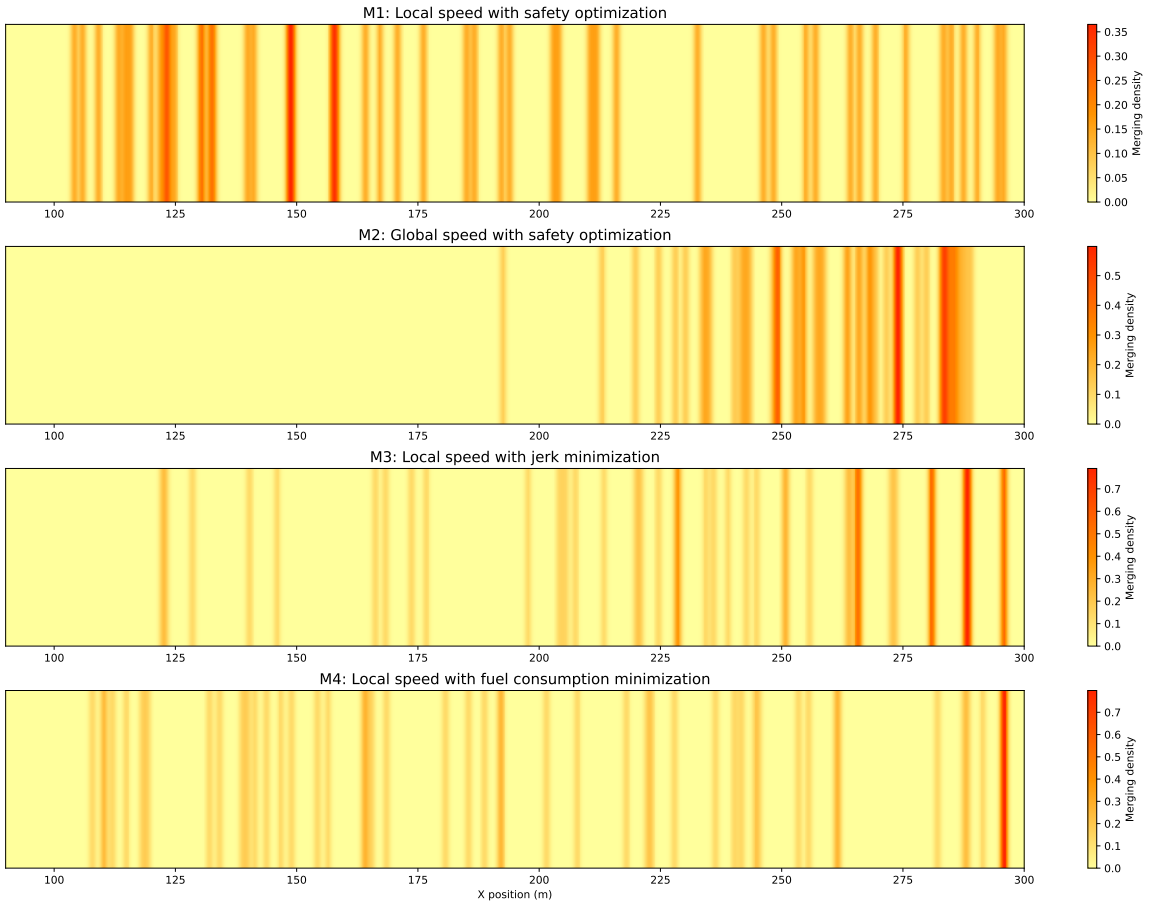

Figure 5.11: Heat-map of vehicles merging point for each MARL model. The intensity of the red color indicate high frequency that each MARL model decides to merge.

other vehicles. Moreover, the bottom portions of Figs. 5.12-5.15 plot the final formation after passing the merging point, with the number for each point being their initial order.

Note that Figs. 5.12-5.15 provide an intuitive way to visualize different merging strategy. For example, Fig. 5.14 clearly demonstrates that M3 model uses a clustering merging strategy, where groups of main lane vehicles and groups of merging lane vehicles alternate in the final formation. This motivates us to quantify the fairness of merging strategies. Two metrics are proposed, Lane Fairness and Individual Fairness. The following section presents more details on the fairness comparison of the four merging strategies.

5.3.5 Fairness Comparison

The Lane Fairness, inspired by [162], measures how evenly vehicles from the two lanes are distributed in the final formation. An ideal merging strategy would result in a balanced distribution of vehicles across all lanes, ensuring no single lane is disproportionately congested. Intuitively speaking, the bottom portions of Figs. 5.12-5.15 should present a pattern with alternating colors and the larger deviation from such a pattern, the less fair it is. Mathematically, Lane Fairness can be calculated as follows.

$$\text{Lane Fairness} = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}, \tag{5.23}$$

where x_i is the ratio of actual to expected number of vehicles for a lane in a segment and n is the number of segments ($n = 2$ is chosen for the present study, resulting the expected number of vehicles for each lane in a single segment equals 1).

While the Lane Fairness effectively quantifies the distribution of vehicles across lanes post-merge, it does not account for the individual delays experienced by vehicles during the merging process. To address this limitation and provide a more comprehensive assessment of fairness, we introduce an additional metric, Individual Fairness. Individual Fairness measures the average position shift for individual vehicle. An ideal merging

Table 5.5: Summary of Merging Styles and Fairness.

Model Label	Merging Styles	Lane Fairness	Individual Fairness
M1	Distributed towards early merge	0.7692	0.9664
M2	Late merge	0.7812	0.966
M3	Distributed towards late merge	0.5319	0.7528
M4	Distributed towards late merge	0.7692	0.928
Zipper Merge	Late merge	1	1

strategy would preserve the order for all vehicles. Intuitively speaking, the top portions of Figs. 5.12-5.15 should perfect align with the dotted diagonal line, and the large deviation from it, the less fair. Mathematically, Individual Fairness can be calculated as follows.

$$\text{Individual Fairness} = 1 - \frac{\sum_{i=1}^{n} |p_i - q_i|}{\text{Maximum Displacement}}, \tag{5.24}$$

where p_i is the initial spawning order of vehicle i before the merge, q_i is the final order of vehicle i after the merge, and n is the total number of vehicles. Maximum Displacement represents the theoretical upper limit of cumulative position shifts across all vehicles, given by:

$$\text{Maximum Displacement} = \begin{cases} \frac{n^2-1}{2} & \text{for odd } n, \\ \frac{n^2}{2} & \text{for even } n. \end{cases} \tag{5.25}$$

Note that (5.25) corresponds to an extreme case where the first vehicle ends up being the last one after merge, the second vehicle ends up being the second last one after merge, and so on.

The merging strategies and corresponding fairness indexes for each model, together with those of the benchmark zipper merge strategy, are summarized in Table 5.5. Note that for both Lane Fairness and Individual Fairness, the higher score, the more fair it is.

For M1 model, Fig. 5.12 suggests that although reordering occur during the merging process, they are relatively minor, resulting in the Individual Fairness being

0.9664. The Lane Fairness for this model is 0.7692, which is slightly lower than the M2 model. This suggests that there is a moderate level of lane alternation and the vehicles are evenly distributed among the lanes. Note that the longest streak of vehicles from the same lane is only 4.

For M2 model, the close alignment of points along the diagonal line in Fig. 5.13 indicates minimal reordering of vehicles, which is quantitatively supported by the high Individual Fairness score of 0.966. This score suggests that the model prioritizes maintaining the initial order of vehicles throughout the merging process, likely contributing to a smooth and predictable traffic flow. The Lane Fairness of 0.7812 indicates a relatively balanced utilization of both lanes. This is further supported by the longest same-lane streak of only 5 vehicles, suggesting frequent alternation between vehicles from the main and merging lanes. These metrics, combined with the visual evidence from Fig. 5.13, demonstrate that M1 model, which focus on global speed optimization, achieves a high degree of both lane-wise and individual fairness while optimizing for overall traffic speed and safety.

On the other hand, M3 model exhibits the most distinctive behavior among all four models. Fig. 5.14 reveals a striking pattern of clustered points, clearly illustrating a traffic light-like behavior where groups of vehicles from each lane alternate in merging. This clustering is quantitatively supported by the model's low Lane Fairness of 0.5319 and the remarkably long same-lane streak of 24 vehicles. These metrics indicate that the model strongly favors merging vehicles in alternating groups from each lane, likely as a strategy to minimize overall jerk by reducing the frequency of lane changes. Furthermore, the individual Fairness for this model is 0.7528, significantly lower than the other models. This low score reflects the extreme disruption to the initial vehicle ordering, with some points in Fig. 5.14 showing shifts of 60-70 positions.

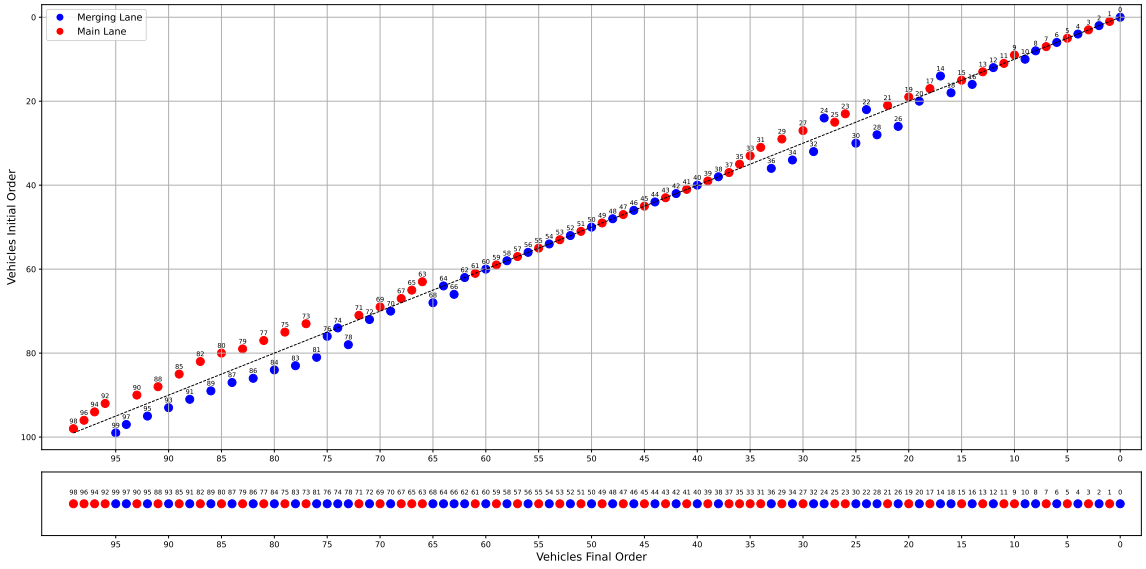

Figure 5.12: Comparison of starting and final positions for M1 model. *Top*: the *x*-axis represents the order in which vehicles pass the reduction point, and the *y*-axis denotes the vehicle entering sequence. *Bottom*: the final formation after passing the lane reduction point.

Lastly, M4 model presents a balanced approach between the extremes observed in the other models. Fig. 5.15 shows a merging pattern that falls between the tightly ordered M2 model and the highly reordered M3 model. The Individual Fairness of 0.928 indicates that while reordering occurs, it is more equitably distributed among vehicles compared to the M3 model. The Lane Fairness of 0.7692, identical to the M1 model, suggests a similar balance in lane utilization. However, the longest same-lane streak of 9 vehicles indicates a tendency towards longer groups from the same lane. This grouping behavior is visible in Fig. 5.11, particularly towards the end of the graph, where clusters of 5-10 vehicles from the same lane maintain their relative positions. This strategy combined with the tendency for late merging shown in the heat map is likely to reduce stop-and-go traffic, potentially decreasing overall fuel consumption by minimizing full stops and subsequent accelerations.

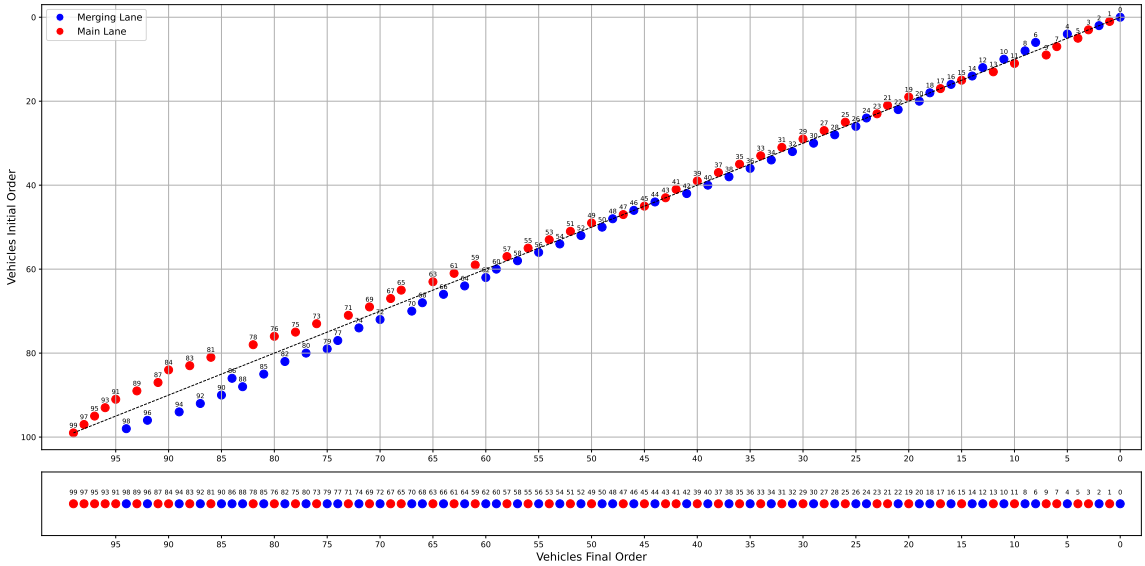

Figure 5.13: Comparison of starting and final positions for M2 model. *Top*: the x-axis represents the order in which vehicles pass the reduction point, and the y-axis denotes the vehicle entering sequence. *Bottom*: the final formation after passing the lane reduction point.

5.4 Summary

This chapter presents a multi-agent reinforcement learning (MARL) framework for decentralized coordination of autonomous vehicles (AVs) in highway merging scenarios. The approach is founded on a decentralized partially observable Markov decision process (Dec-POMDP) formulation, allowing each vehicle to make independent decisions based on local observations. Utilizing a centralized training with decentralized execution paradigm, the framework incorporates self-attention mechanisms in the critic and baseline networks to effectively manage varying numbers of agents. The framework's implementation and validation were conducted using the SUMO traffic simulator, employing four distinct reward functions: global speed, local speed, fuel efficiency, and ride comfort. Simulation results demonstrate the framework's efficacy in managing complex trade-offs in AV

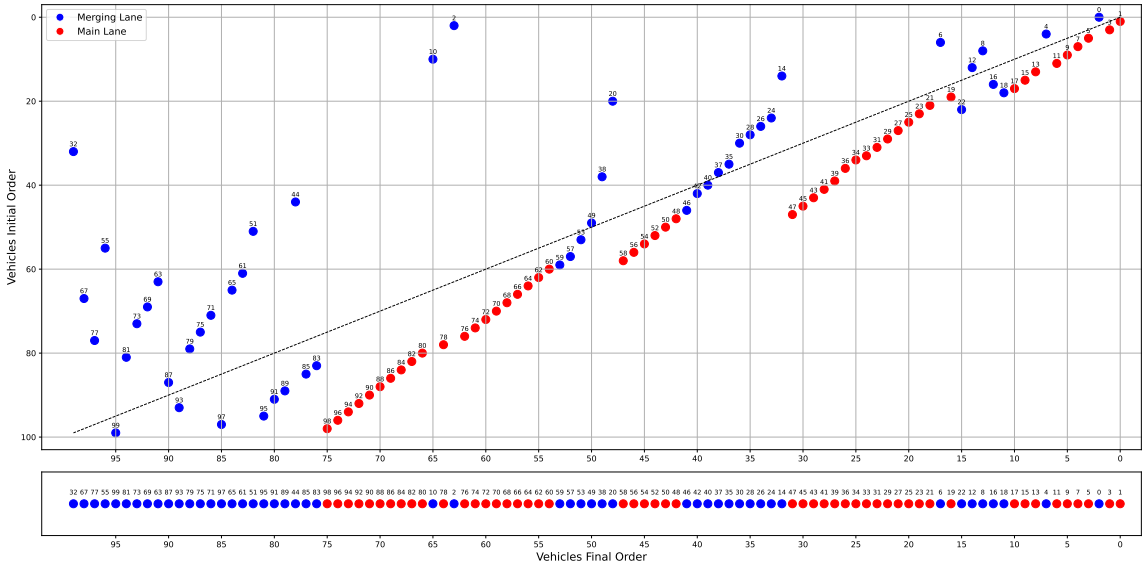

Figure 5.14: Comparison of starting and final positions for M3 model. *Top*: the x-axis represents the order in which vehicles pass the reduction point, and the y-axis denotes the vehicle entering sequence. *Bottom*: the final formation after passing the lane reduction point.

control. The framework's generalizability is evidenced by its successful evaluation in higher speed conditions, despite being trained at lower speeds. Additionally, two fairness indexes are introduced to compare how different merging strategies preserve lane-wise order and individual vehicle order.

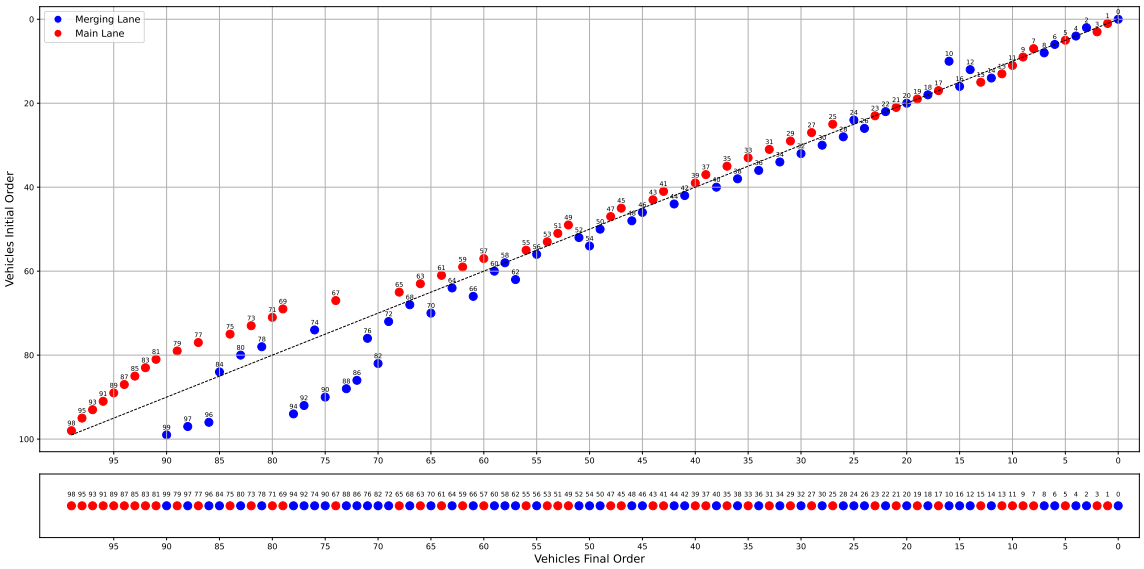

Figure 5.15: Comparison of starting and final positions for M4 model. *Top*: the x-axis represents the order in which vehicles pass the reduction point, and the y-axis denotes the vehicle entering sequence. *Bottom*: the final formation after passing the lane reduction point.

CHAPTER SIX

MARL-BASED AV MERGING CONTROL WITH DECENTRALIZED TRAINING

This chapter addresses the multiple AVs merging problem introduced in Chapter 5, using a Decentralized Training and Decentralized Execution (DTDE) scheme to further evaluate the proposed Centralized Training and Decentralized Execution (CTDE) algorithm.

6.1 DTDE-based AVs Merging Control

The problem is modeled as a decentralized partially observable MDP (Dec-POMDP) [151] defined by the tuple: $(\mathbf{S}, \mathbf{O}, \mathbf{A}, R, p, \gamma)$. Here, $\mathbf{S}$ is the state space, $\mathbf{O} : \mathbf{O_1} \times \ldots \times \mathbf{O_n}$ is the joint local observation space of all agents n, $\mathbf{A} : \mathbf{A_1} \times \ldots \times \mathbf{A_n}$ is the joint action space for all agents n, $p(S' \mid S, A)$ is the transition probability to state S' given the current S and action $A = (a_1, \ldots, a_n)$, γ is the discount factor, and $R = \{r_1, \ldots, r_n\}$ is the set of individual reward functions for each agent. This work uses a MARL DTDE framework for multi-agent automated traffic control with a dynamic number of agents. In this approach, both training and execution phases are fully decentralized. Each agent learns and acts independently based on its local observations, without access to global information or a central coordinator. Specifically, each agent's local observation $o_i \in \mathbf{O_i}$ consists of information from nearby vehicles within a predefined range. This decentralized approach during both training and execution ensures that the learned policies are directly applicable in real-world settings where full environmental information is not available to individual vehicles and centralized coordination is not feasible.

The DTDE MARL AVs Merging Control algorithm (Algorithm 6.1) implements the following key computations. The local advantage for agent i is computed as:

$$A_i = r_i + \gamma V_{\phi_i}(o'i) - V\phi_i(o_i), \tag{6.1}$$

where r_i is the local reward for agent i, o'_i is the next local observation, and γ is the discount factor.

The actor network π_{θ_i} for each agent i is updated to maximize the expected advantage:

$$L_\pi(\theta_i) = \mathbb{E}[\log \pi_{\theta_i}(a_i|o_i)A_i], \tag{6.2}$$

where a_i is the action taken by agent i.

The critic network V_{w_i} for each agent i is updated to minimize the mean squared TD error:

$$L_V(w_i) = \mathbb{E}[(r_i + \gamma V_{w_i}(o'i) - Vw_i(o_i))^2]. \tag{6.3}$$

To facilitate learning in a multi-agent environment, experiences from all agents are collected in a shared replay buffer $\mathscr{D}$. Each experience tuple (o_i, a_i, r_i, o'_i) is stored in this buffer. During the learning phase, batches of experiences are sampled from this shared buffer to update the actor and critic networks. We adopted parameter sharing in this framework, where only a single actor network π_θ and a single critic network V_ϕ are used for all AVs. This approach significantly reduces the number of parameters to be learned and promotes consistency in behavior across agents. The shared actor network takes an agent's local observation as input and outputs an action, while the shared critic network estimates the value function based on the local observation. This shared buffer and parameter sharing approach allows for efficient knowledge transfer between agents, as each agent can learn from the experiences of others, and all agents contribute to updating

Algorithm 6.1: DTDE-MARL AVs Merging Control

Initialize shared weights: Value function ϕ, Policy θ
for *each iteration* **do**
  while *nsteps* $<$ *buffer size* **do**
    for *each agent i in* K_t **do**
      Observe local observation o_i
      Sample action $a_i \sim \pi_\theta(a_i|o_i)$
      Take action a_i, observe r_i, o_i'
      Store (o_i, a_i, r_i, o_i') in shared buffer $\mathscr{D}$
    end
  end
  Sample batch of experiences (o, a, r, o') from $\mathscr{D}$
  Compute TD errors: $r + \gamma V_\phi(o') - V_\phi(o)$
  Compute value function loss: $L_V(\phi)$
  Update ϕ to minimize $L_V(\phi)$
  Compute advantages: A
  Compute policy loss: $L_\pi(\theta)$
  Update θ to maximize $L_\pi(\theta)$
end

the same set of parameters. However, it's important to note that during both training and execution phases, each agent still acts independently based on its local observations.

6.2 State and Action Spaces

In this decentralized approach, each agent i operates solely based on its local observation o_i at time t. The global state S_t of the environment, while not directly accessible to any individual agent, can be conceptualized as the collection of all agents' local observations:

$$S_t = [o_1, o_2, ..., o_{K_t}] \tag{6.4}$$

where K_t represents the number of active agents at time t, which may vary dynamically.

The local observation o_i for each agent i is a vector of dimension $3+3N^i$, where N^i is the number of observed surrounding vehicles within an 8-meter range in front of and behind the AV. This local observation space is illustrated in Fig. 5.2. Specifically, o_i is defined as:

$$o_i = [e_i, x_1, x_2, ..., x_{N^i}], \tag{6.5}$$

where $e_i = [v_i, d_i, m_i]$ represents the ego vehicle's state:

- v_i: speed of the ego vehicle
- d_i: distance to the merge point
- m_i: merge state (1 if merged, 0 if not yet merged)

$x_j = [\Delta x_j, v_j, \Delta v_j]$ for $j = 1, ..., N^i$ represents the state of each observed vehicle:

- Δx_j: relative longitudinal position to the ego vehicle
- v_j: speed of the observed surrounding vehicle
- Δv_j: relative speed to the ego vehicle

Each element in the local observation o_i is normalized to [0, 1] based on its maximum and minimum values. This normalized local observation is the sole input to the agent's actor network for making predictions and decisions. It's important to note that each agent has access only to its own local observation and does not have information about the global state or the observations of other agents.

The action space of each AV is a high-level lane change decisions and can be expressed as:

$$a_i = \begin{cases} 1 & \Rightarrow \text{ Request to merge} \\ 0 & \Rightarrow \text{ Stay in current lane.} \end{cases} \tag{6.6}$$

6.3 Reward Functions

In our DTDE framework, each agent receives an individual reward based on its local observations and actions. We explore three different reward functions, each designed to optimize different aspects of the merging behavior.

6.3.1 R1: Local Speed with Safety Optimization

This reward function aims to balance speed optimization with safety considerations. For an agent j, the reward is composed of two terms:

$$r^j_{\text{speed},t} = \frac{1}{|N^j_t| + 1} \sum_{i \in j \cup N^j_t} s^i_t \tag{6.7}$$

$$r^j_{\text{safety},t} = \begin{cases} -\left(\frac{x^j_t - d}{d}\right)^2, & \text{if } x^j_t \leq d \\ 0, & \text{otherwise} \end{cases} \tag{6.8}$$

Here, N^j_t is the set of observed neighboring vehicles, s^i_t is the speed of vehicle i, x^j_t is the distance to the merge point, and d is a safety threshold. The total reward for agent j is:

$$r^j_t = w_1 r^j_{\text{speed},t} + w_2 r^j_{\text{safety},t} \tag{6.9}$$

6.3.2 R2: Local Speed with Jerk Minimization

This reward function aims to maximize speed while minimizing jerk for a smoother ride:

$$r_t^j = w_1 r_{\text{speed},t}^j + w_2 r_{\text{jerk},t}^j \tag{6.10}$$

where $r_{\text{speed},t}^j$ is as defined in R1, and

$$r_{\text{jerk},t}^j = -\frac{1}{|N_t^j|+1} \sum_{l \in j \cup N_t^j} \left(\frac{acc_t^l - acc_{t-1}^l}{dt} \right)^2 \tag{6.11}$$

Here, acc_t^l is the acceleration of vehicle l at time t.

6.3.3 R3: Local Speed with Fuel Consumption Minimization

This reward function balances speed optimization with fuel efficiency:

$$r_t^j = w_1 r_{\text{speed},t}^j + w_2 r_{\text{fuel},t}^j \tag{6.12}$$

where $r_{\text{speed},t}^j$ is as defined in R1, and

$$r_{\text{fuel},t}^j = -\frac{1}{|N_t^j|+1} \sum_{i \in j \cup N_t^j} f(P_t^i) \tag{6.13}$$

Here, $f(P_t^i)$ is the fuel consumption rate based on the power demand P_t^i of vehicle i, calculated using SUMO's PHEMlight model. In all reward functions, w_1 and w_2 are weighting factors to balance different objectives.

6.4 Results

We evaluated the three reward functions (R1, R2, R3) in our DTDE MARL framework for AVs merging control. Figures (6.1 - 6.3) illustrate the learning curves for our DTDE MARL approach using reward functions R1, R2, and R3 respectively. The learning curve in Fig. 6.1 for R1 shows an overall upward trend, indicating that the DTDE approach successfully learned to optimize speed while considering safety constraints. Fig. 6.2, representing R2, presents a learning curve with a similar upward trajectory, suggesting effective learning in balancing speed optimization and ride smoothness. The learning curve in Fig. 6.3 for R3 also demonstrates an increasing trend, indicating that the DTDE approach learned to optimize speed while minimizing fuel consumption.

When compared to our previous CTDE results (Chapter 5), the DTDE approach consistently achieved lower reward values across all three reward functions compared to the CTDE method. Additionally, the CTDE approach exhibited more stable learning curves for all reward functions, while the DTDE learning curves showed greater variability. One the other hand, despite the lack of access to global information or a central coordinator, the DTDE approach successfully learned effective policies for AVs merging control. This demonstrates the viability of fully decentralized learning and execution in multi-agent traffic scenarios, even when individual agents have limited environmental awareness.

6.5 Summary

This chapter presented a Decentralized Training and Decentralized Execution (DTDE) approach for multi-agent reinforcement learning in automated vehicle merging control. We modeled the problem as a Dec-POMDP, enabling decentralized decision-making based on local observations. Our fully decentralized algorithm operates without access to global information or a central coordinator in both training and execution phases. We implemented parameter sharing with a single actor network and a single critic

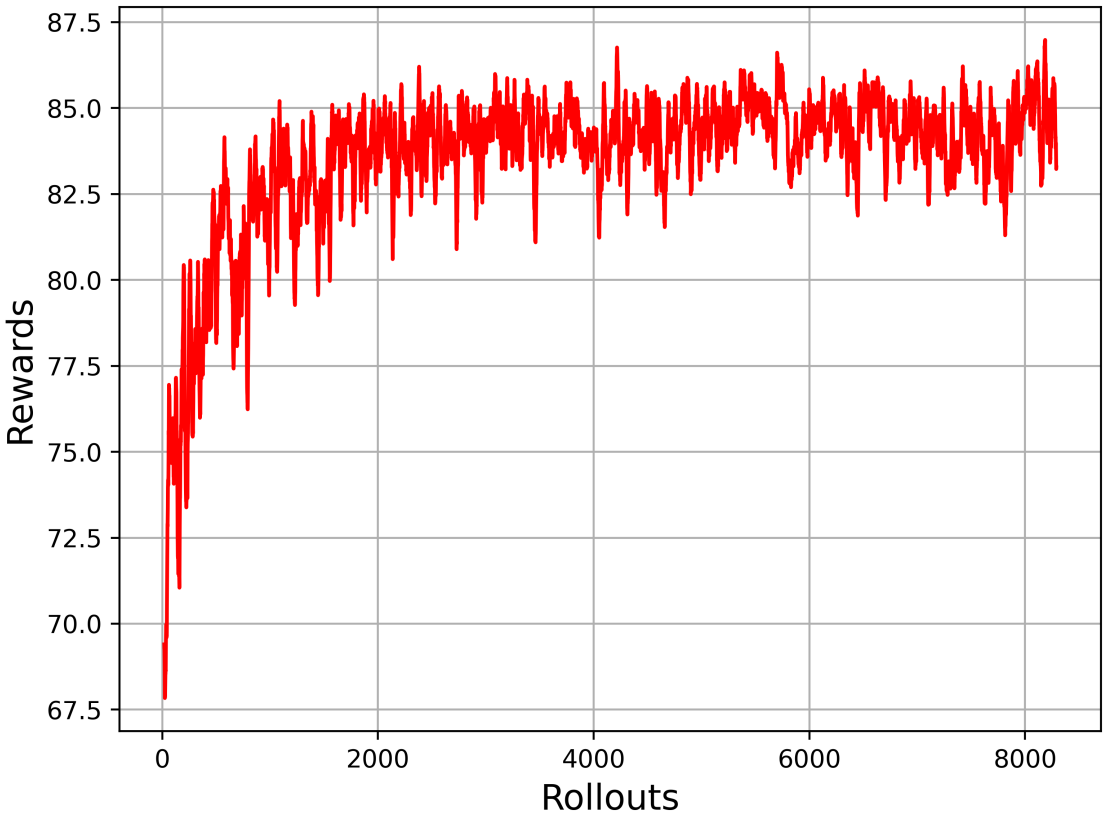

Figure 6.1: Learning curve for R1 reward function.

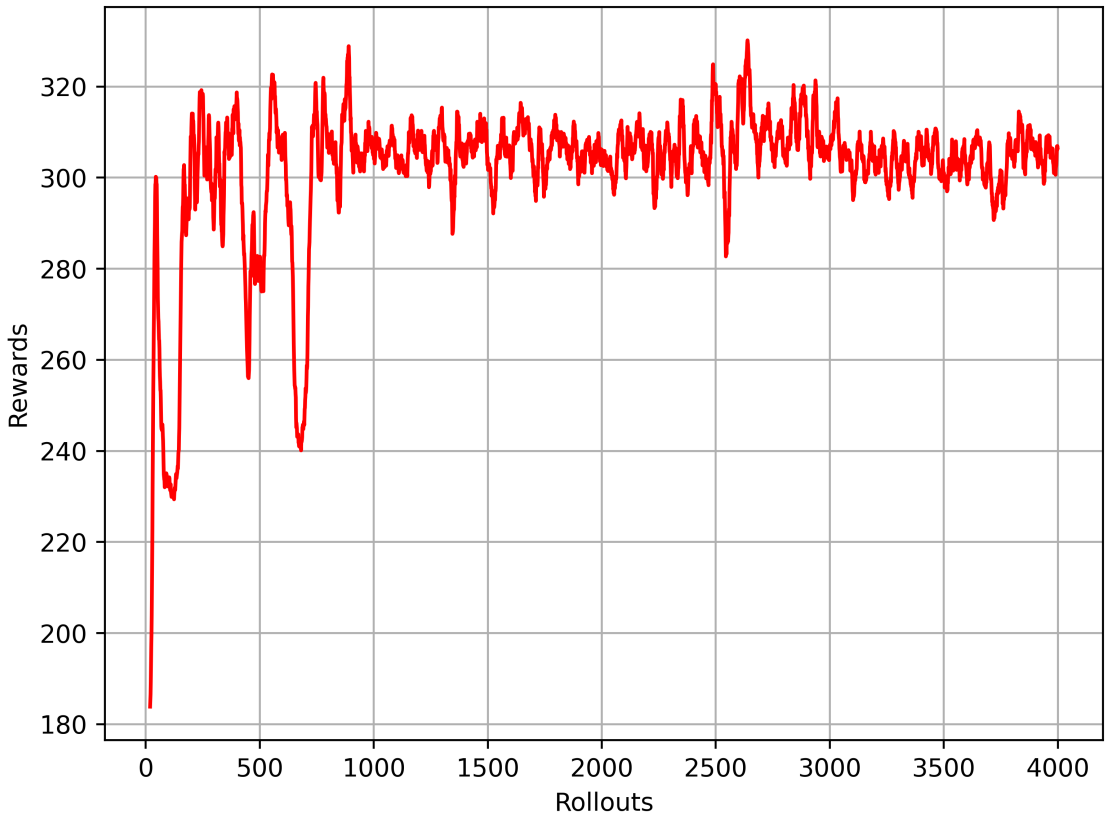

Figure 6.2: Learning curve for R2 reward function.

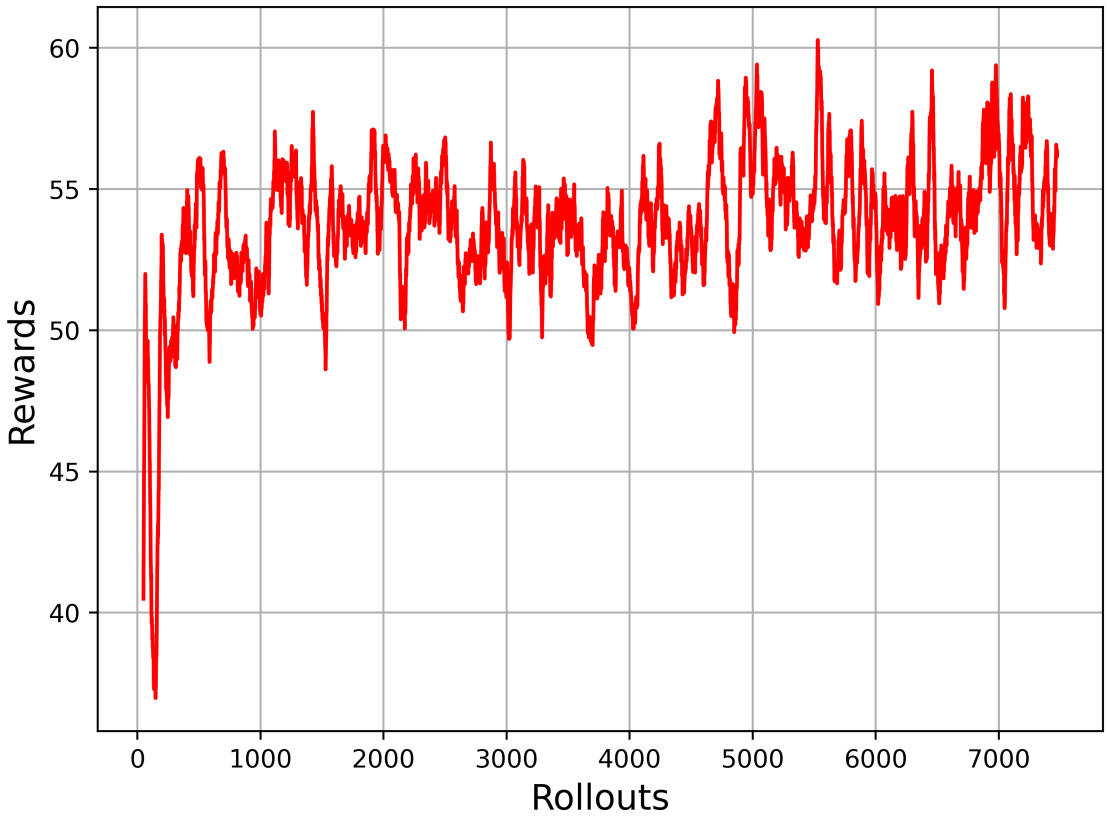

Figure 6.3: Learning curve for R3 reward function.

network for all AVs, promoting efficient learning and behavioral consistency. A shared replay buffer facilitates indirect knowledge transfer between agents while maintaining decentralized execution. We explored three different reward functions to optimize various aspects of merging behavior, including speed, safety, ride smoothness, and fuel efficiency. The DTDE approach addresses key challenges in multi-agent systems, such as scalability to dynamic numbers of agents and applicability to real-world scenarios where centralized control is infeasible. Learning curves demonstrated the effectiveness of our approach across different reward functions.

CHAPTER SEVEN

CONCLUSION AND FUTURE WORK

This dissertation explored the application of multi-agent reinforcement learning (MARL) to autonomous vehicle (AV) control, with a focus on highway merging scenarios. Three case studies were presented, a centralized approach for platoon merging and another two decentralized multi-agent frameworks where AVs act independently based on local observations. The first case study demonstrated the potential of centralized RL to optimize cooperative platoon merging strategies. The proposed maskable proximal policy optimization (MPPO) algorithm successfully learned policies to balance multiple objectives including safety, efficiency, energy consumption, and passenger comfort. Results showed significant improvements over baseline strategies, with up to 76.7% reduction in energy consumption and 50% reduction in average jerk.

The second case study addressed key limitations of centralized approaches by developing a scalable multi-agent reinforcement learning (MARL) framework. The proposed approach is built on a decentralized partially observable Markov decision process (Dec-POMDP) formulation, enabling each vehicle to make independent decisions based on local observations. The framework employs a centralized training with decentralized execution (CTDE) paradigm, utilizing self-attention mechanisms in the critic and baseline networks to effectively handle varying numbers of agents. The fairness and efficiency of the learned merging strategies were evaluated using novel quantitative metrics. Both the centralized and decentralized MARL approaches consistently outperformed benchmark RL methods and a rule-based zipper merge strategy across various metrics. Notably, the proposed methods achieved up to 60.14% improvement in traffic flow at higher speeds, demonstrating their potential to significantly enhance highway merging efficiency. The

generalizability of the framework was demonstrated by successfully applying policies trained in low-speed scenarios to high-speed situations. This capability is crucial for real-world deployment, where traffic conditions can vary widely.

The third case study presents the decentralized training and decentralized execution (DTDE) MARL approach where agents act and learn independently based solely on local information. This study showed that even with the lacking global information or central coordination, the DTDE approach effectively learned merging policies. This demonstrates that fully decentralized learning and execution can succeed in multi-agent traffic scenarios, even with agents' limited environmental awareness.

The present research opens up several promising avenues for future investigation and expansions. A crucial area for future work is investigating the effects of uncertainties in the system, including studying noise's impact on vehicle-level control and errors or delays in inter-vehicle communication. Furthermore, the use of multiple past-time steps in the state space, and/or the use of recurrent network capable of handling time series data, can potentially provide a richer context for decision-making. Lastly, another potential direction for future research can be exploring safety-based reinforcement learning approaches that incorporates safety constraints directly into the learning process.

REFERENCES

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

[2] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning," 2020.

[3] 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, pp. 3547–3555, June 2023.

[4] Z. Zhou and J. Chen, "Modeling driver lane change behavior using inverse reinforcement learning," in *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, pp. 1–5, IEEE, 2024.

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

[6] 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.

[7] A. Irshayyid, J. Chen, and G. Xiong, "A review on reinforcement learning-based highway autonomous vehicle control," *Green Energy and Intelligent Transportation*, p. 100156, 2024.

[8] V. W. Inman, S. Jackson, B. H. Philips, *et al.*, "Cooperative adaptive cruise control human factors study: Experiment 1-workload, distraction, arousal, and trust," tech. rep., United States. Federal Highway Administration, 2016.

[9] N. C. Spiller, K. Blizzard, R. Margiotta, *et al.*, "Recurring traffic bottlenecks: A primer focus on low-cost operational improvements," tech. rep., United States. Federal Highway Administration. Office of Operations, 2017.

[10] 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.

[11] Y. Lyu, W. Luo, and J. M. Dolan, "Probabilistic safety-assured adaptive merging control for autonomous vehicles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10764–10770, IEEE, 2021.

[12] K. A. Mustafa, D. J. Ornia, J. Kober, and J. Alonso-Mora, "Racp: Risk-aware contingency planning with multi-modal predictions," *IEEE Transactions on Intelligent Vehicles*, 2024.

[13] X. Tang, K. Yang, H. Wang, J. Wu, Y. Qin, W. Yu, and D. Cao, "Prediction-uncertainty-aware decision-making for autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 4, pp. 849–862, 2022.

[14] F. Lucidi, A. Bosco, L. Mallia, and A. Setti, "Factors underpinning and influencing drivers' aberrant behaviors across the life course," *Frontiers in Psychology*, vol. 10, p. 504670, 2020.

[15] H. Yang, Y. He, Y. Xu, and H. Zhao, "Collision avoidance for autonomous vehicles based on mpc with adaptive apf," *IEEE Transactions on Intelligent Vehicles*, 2023.

[16] M. Capallera, L. Angelini, Q. Meteier, O. Abou Khaled, and E. Mugellini, "Human-vehicle interaction to support driver's situation awareness in automated vehicles: A systematic review," *IEEE Transactions on intelligent vehicles*, vol. 8, no. 3, pp. 2551–2567, 2022.

[17] J. Liu, D. Zhou, P. Hang, Y. Ni, and J. Sun, "Towards socially responsive autonomous vehicles: A reinforcement learning framework with driving priors and coordination awareness," *IEEE Transactions on Intelligent Vehicles*, 2023.

[18] D. Xu, P. Liu, H. Li, H. Guo, Z. Xie, and Q. Xuan, "Multi-view graph convolution network reinforcement learning for cavs cooperative control in highway mixed traffic," *IEEE Transactions on Intelligent Vehicles*, 2023.

[19] D. Zhou, Z. Ma, and J. Sun, "Autonomous vehicles' turning motion planning for conflict areas at mixed-flow intersections," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 204–216, 2019.

[20] 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.

[21] W. Li, F. Qiu, L. Li, Y. Zhang, and K. Wang, "Simulation of vehicle interaction behavior in merging scenarios: A deep maximum entropy-inverse reinforcement learning method combined with game theory," *IEEE Transactions on Intelligent Vehicles*, 2023.

[22] T. Nishi, P. Doshi, and D. Prokhorov, "Merging in congested freeway traffic using multipolicy decision making and passive actor-critic learning," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 2, pp. 287–297, 2019.

[23] L. Davis, "Effect of adaptive cruise control systems on mixed traffic flow near an on-ramp," *Physica A: Statistical Mechanics and its Applications*, vol. 379, no. 1, pp. 274–290, 2007.

[24] 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.

[25] D. Marinescu, J. Čurn, M. Bouroche, and V. Cahill, "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach," in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 900–906, IEEE, 2012.

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

[27] C. Dong, J. M. Dolan, and B. Litkouhi, "Smooth behavioral estimation for ramp merging control in autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1692–1697, IEEE, 2018.

[28] J. Rios-Torres and A. A. Malikopoulos, "Automated and cooperative vehicle merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 780–789, 2016.

[29] W. Cao, M. Mukai, T. Kawabe, H. Nishira, and N. Fujiki, "Cooperative vehicle path generation during merging using model predictive control with real-time optimization," *Control Engineering Practice*, vol. 34, pp. 98–105, 2015.

[30] J. Chen, L. Zhang, and W. Gao, "Reconfigurable model predictive control for large scale distributed systems," *IEEE Systems Journal*, 2024.

[31] S. Li, W. Zou, J. Gao, Y. Yin, D. Kim, S. Yang, and S. E. Li, "Fast online computation of mpc-based integrated decision control for autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, 2024.

[32] A. Irshayyid and J. Chen, "Comparative study of cooperative platoon merging control based on reinforcement learning," *Sensors*, vol. 23, no. 2, pp. 1–23, 2023.

[33] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017.

[34] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre,

G. Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 10 2017.

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

[36] Y. Lin, J. McPhee, and N. L. Azad, "Comparison of deep reinforcement learning and model predictive control for adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 221–231, 2020.

[37] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[38] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[39] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[40] A. R. Kreidieh, C. Wu, and A. M. Bayen, "Dissipating stop-and-go waves in closed and open networks via deep reinforcement learning," in *2018 21st international conference on intelligent transportation systems (itsc)*, pp. 1475–1480, IEEE, 2018.

[41] J. Cui, W. Macke, H. Yedidsion, A. Goyal, D. Urieli, and P. Stone, "Scalable multiagent driving policies for reducing traffic congestion," 2021.

[42] C. Wang and Y. Wang, "Safe autonomous driving with latent dynamics and state-wise constraints," *Sensors*, vol. 24, no. 10, p. 3139, 2024.

[43] 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)*, pp. 3461–3468, IEEE, 2019.

[44] L. Schester and L. E. Ortiz, "Automated driving highway traffic merging using deep multi-agent reinforcement learning in continuous state-action spaces," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 280–287, IEEE, 2021.

[45] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pp. 66–83, Springer, 2017.

[46] H. Nekoei, A. Badrinaaraayanan, A. Sinha, M. Amini, J. Rajendran, A. Mahajan, and S. Chandar, "Dealing with non-stationarity in decentralized cooperative multi-agent deep reinforcement learning via multi-timescale learning," in *Conference on Lifelong Learning Agents*, pp. 376–398, PMLR, 2023.

[47] L. M. Schmidt, J. Brosig, A. Plinge, B. M. Eskofier, and C. Mutschler, "An introduction to multi-agent reinforcement learning and review of its application to autonomous mobility," in *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1342–1349, IEEE, 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)*, pp. 1746–1752, IEEE, 2020.

[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)*, pp. 21–26, IEEE, 2020.

[50] 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.

[51] 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.

[52] 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.

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

[54] 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)*, pp. 1050–1053, IEEE, 2022.

[55] 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.

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

[61] 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.

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

[63] 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)*, pp. 1–6, IEEE, 2017.

[64] 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)*, pp. 1454–1460, IEEE, 2019.

[65] 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)*, pp. 1–6, IEEE, 2019.

[66] 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.

[67] 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.

[68] 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.

[69] 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)*, pp. 4079–4084, IEEE, 2019.

[70] 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.

[71] 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)*, pp. 4517–4524, IEEE, 2021.

[72] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Altruistic maneuver planning for cooperative autonomous vehicles using multi-agent advantage actor-critic," *arXiv preprint arXiv:2107.05664*, 2021.

[73] 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)*, pp. 804–811, IEEE, 2021.

[74] 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)*, pp. 151–158, IEEE, 2019.

[75] 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)*, pp. 1595–1600, IEEE, 2020.

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

[77] 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. 17594–17606, 2022.

[78] 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)*, pp. 1–6, IEEE, 2020.

[79] 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.

[80] 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)*, pp. 216–221, IEEE, 2020.

[81] 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)*, pp. 3441–3447, IEEE, 2019.

[82] 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. 13340–13350, 2021.

[83] 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)*, pp. 1–9, IEEE, 2019.

[84] 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)*, pp. 4355–4361, IEEE, 2020.

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

[86] 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.

[87] 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.

[88] 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, 2016.

[89] 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*, pp. 1995–2003, PMLR, 2016.

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

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

[92] 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, 2018.

[93] 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.

[94] 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.

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

[96] 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.

[97] 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*, pp. 411–423, Springer, 2018.

[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] 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)*, pp. 3420–3431, IEEE, 2019.

[100] 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.

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

[102] 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.

[103] 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.

[104] 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.

[105] 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*, pp. 2722–2730, 2015.

[106] 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.

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

[108] 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.

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

[110] 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.

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

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

[113] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola, *Autonomous Vehicle Technology: A Guide for Policymakers*. Santa Monica, CA: RAND Corporation, 2016.

[114] *Vehicle Platooning: A Brief Survey and Categorization*, vol. Volume 3: 2011 ASME/IEEE International Conference on Mechatronic and Embedded Systems

and Applications, Parts A and B of *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 08 2011.

[115] F. Fakhfakh, M. Tounsi, and M. Mosbah, "Vehicle platooning systems: Review, classification and validation strategies," *International Journal of Networked and Distributed Computing*, vol. 8, 06 2020.

[116] C. Englund, L. Chen, J. Ploeg, E. Semsar-Kazerooni, A. Voronov, H. H. Bengtsson, and J. Didoff, "The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 146–152, 2016.

[117] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, *et al.*, "Lane change and merge maneuvers for connected and automated vehicles: A survey," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 105–120, 2016.

[118] A. Paranjothi, M. Atiquzzaman, and M. S. Khan, "Pmcd: Platoon-merging approach for cooperative driving," *Internet Technology Letters*, vol. 3, no. 1, p. e139, 2020.

[119] H. Liu, W. Zhuang, G. Yin, Z. Tang, and L. Xu, "Strategy for heterogeneous vehicular platoons merging in automated highway system," in *2018 Chinese Control And Decision Conference (CCDC)*, pp. 2736–2740, 2018.

[120] S. Dasgupta, V. Raghuraman, A. Choudhury, T. N. Teja, and J. Dauwels, "Merging and splitting maneuver of platoons by means of a novel pid controller," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2017.

[121] B. van Arem, C. J. G. van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 429–436, 2006.

[122] A. Schwab and J. Lunze, "Vehicle platooning and cooperative merging," *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 353–358, 2019. 9th IFAC Symposium on Advances in Automotive Control AAC 2019.

[123] Z. Su and P. Chen, "Optimal platoon merging and catch-up approach for connected electric vehicles," in *2022 American Control Conference (ACC)*, pp. 1964–1969, 2022.

[124] N. E. Lownes and R. B. Machemehl, "Vissim: a multi-parameter sensitivity analysis," in *Proceedings of the 2006 Winter Simulation Conference*, pp. 1406–1413, IEEE, 2006.

[125] M. Segata, R. Lo Cigno, T. Hardes, J. Heinovski, M. Schettler, B. Bloessl, C. Sommer, and F. Dressler, "Multi-Technology Cooperative Driving: An Analysis

Based on PLEXE," *IEEE Transactions on Mobile Computing (TMC)*, 2 2022. to appear.

[126] C. Hidalgo, R. Lattarulo, C. Flores, and J. Pérez Rastelli, "Platoon merging approach based on hybrid trajectory planning and cacc strategies," *Sensors*, vol. 21, no. 8, p. 2626, 2021.

[127] A. Farag, A. Hussein, O. M. Shehata, F. García, H. H. Tadjine, and E. Matthes, "Dynamics platooning model and protocols for self-driving vehicles," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1974–1980, IEEE, 2019.

[128] S. Santini, A. Salvi, A. S. Valente, A. Pescapè, M. Segata, and R. L. Cigno, "Platooning maneuvers in vehicular networks: A distributed and consensus-based approach," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 59–72, 2018.

[129] M. Goli and A. Eskandarian, "Mpc-based lateral controller with look-ahead design for autonomous multi-vehicle merging into platoon," in *2019 American Control Conference (ACC)*, pp. 5284–5291, IEEE, 2019.

[130] A. De Luca, G. Oriolo, and C. Samson, *Feedback control of a nonholonomic car-like robot*, pp. 171–253. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998.

[131] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[132] J.-w. Choi and G. H. Elkaim, "Bézier curves for trajectory guidance," in *World Congress on Engineering and Computer Science, WCECS*, pp. 22–24, Citeseer, 2008.

[133] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[134] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[135] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1889–1897, PMLR, 07–09 Jul 2015.

[136] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[137] 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.

[138] T. Irmak, K. N. de Winkel, D. M. Pool, H. H. Bülthoff, and R. Happee, "Individual motion perception parameters and motion sickness frequency sensitivity in fore-aft motion," *Experimental brain research*, vol. 239, no. 6, pp. 1727–1745, 2021.

[139] M. J. Griffin and J. Erdreich, "Handbook of human vibration," 1991.

[140] K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau, "Standards for passenger comfort in automated vehicles: Acceleration and jerk," *Applied Ergonomics*, vol. 106, p. 103881, 2023.

[141] S. Huang and S. Ontañón, "A closer look at invalid action masking in policy gradient algorithms," in *Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2022, Hutchinson Island, Jensen Beach, Florida, USA, May 15-18, 2022* (R. Barták, F. Keshtkar, and M. Franklin, eds.), 2022.

[142] S. Huang, R. F. J. Dossa, A. Raffin, A. Kanervisto, and W. Wang, "The 37 implementation details of proximal policy optimization," *The ICLR Blog Track 2023*, 2022.

[143] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[144] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO–simulation of urban mobility: an overview," in *Proceedings of SIMUL 2011, The Third Int. Conf. on Advances in System Simulation*, 2011.

[145] S. K. S. Nakka, B. Chalaki, and A. A. Malikopoulos, "A multi-agent deep reinforcement learning coordination framework for connected and automated vehicles at merging roadways," in *2022 American Control Conference (ACC)*, pp. 3297–3302, IEEE, 2022.

[146] D. Chen, M. R. Hajidavalloo, Z. Li, K. Chen, Y. Wang, L. Jiang, and Y. Wang, "Deep multi-agent reinforcement learning for highway on-ramp merging in mixed traffic," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[147] D. Wang, "Multi-agent reinforcement learning for safe driving in on-ramp merging of autonomous vehicles," in *2024 14th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 644–651, IEEE, 2024.

[148] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: an interface for coupling road traffic and network simulators," in *Proc. Comm. Net. Simulation Symp.*, pp. 155–163, 2008.

[149] S. Krauß, P. Wagner, and C. Gawron, "Metastable states in a microscopic model of traffic flow," *Physical Review E*, vol. 55, no. 5, p. 5597, 1997.

[150] J. Erdmann, "Sumo's lane-changing model," in *Modeling Mobility with Open Data: 2nd SUMO Conference 2014 Berlin, Germany, May 15-16, 2014*, pp. 105–123, Springer, 2015.

[151] F. A. Oliehoek, C. Amato, *et al.*, *A concise introduction to decentralized POMDPs*, vol. 1. Springer, 2016.

[152] A. Cohen, E. Teng, V.-P. Berges, R.-P. Dong, H. Henry, M. Mattar, A. Zook, and S. Ganguly, "On the use and misuse of absorbing states in multi-agent reinforcement learning," *arXiv preprint arXiv:2111.05992*, 2021.

[153] A. Irshayyid and J. Chen, "Highway merging control using multi - agent reinforcement learning," in *2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI)*, pp. 1–2, 2024.

[154] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," in *Neural Information Processing Systems*, 2021.

[155] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[156] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[157] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, "Coca: Contrastive captioners are image-text foundation models," *arXiv preprint arXiv:2205.01917*, 2022.

[158] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu, "Video swin transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3202–3211, 2022.

[159] S. Hausberger and D. Krajzewicz, "Colombo deliverable 4.2: Extended simulation tool phem coupled to sumo with user guide," 2014.

[160] "Zipper merge." `https://www.dot.state.mn.us/zippermerge/`. Accessed: 2022-11-16.

[161] E. Lammers, J. G. Pigman, B. Howell, A. Kirk, *et al.*, "Applicability of zipper merge versus early merge in kentucky work zones," tech. rep., University of Kentucky Transportation Center, 2017.

[162] R. K. Jain, D.-M. W. Chiu, W. R. Hawe, *et al.*, "A quantitative measure of fairness and discrimination," *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA*, vol. 21, p. 1, 1984.