US DOT National
University Transportation Center for Safety

Carnegie Mellon University

Penn University of Pennsylvania | The Ohio State University | CCAC | Community College of Philadelphia | The University of Texas Rio Grande Valley | Morgan State University

---

Hierarchical Decision Making and Control in RL-based Autonomous Driving for Improved Safety in Complex Traffic Scenarios

Extensive Exploration in Complex Traffic Scenarios using Hierarchical Reinforcement Learning

Keith Redmill (PI) (https://orcid.org/0000-0003-1332-1332)
Zhihao Zhang (https://orcid.org/0009-0009-7932-2128)
Ekim Yurtsever (https://orcid.org/0000-0002-3103-6052)

FINAL RESEARCH REPORT - July 31, 2024

Contract # 69A3552344811/69A3552348316

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Hierarchical Decision Making and Control in RL-based Autonomous Driving for Improved Safety in Complex Traffic Scenarios Extensive Exploration in Complex Traffic Scenarios using Hierarchical Reinforcement Learning		**5. Report Date** July 31, 2024 **6. Performing Organization Code**
7. Author(s) Keith A. Redmill, Ph.D. https://orcid.org/0000-0003-1332-1332 Zhihao Zhang https://orcid.org/0009-0009-7932-2128 Ekim Yurtsever, Ph.D. https://orcid.org/0000-0002-3103-6052		**8. Performing Organization Report No.**
9. Performing Organization Name and Address The Ohio State University Center for Automotive Research 930 Kinnear Road Columbus, OH 43212		**10. Work Unit No.** **11. Contract or Grant No.** Federal Grant No. 69A3552344811
12. Sponsoring Agency Name and Address Safety21 University Transportation Center Carnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213		**13. Type of Report and Period Covered** Final Report (July 1, 2023 - June 30, 2024) **14. Sponsoring Agency Code** USDOT

15. Supplementary Notes

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

16. Abstract

Developing an automated driving system capable of navigating complex traffic environments remains a formidable challenge. Unlike rule-based or supervised learning-based methods, Deep Reinforcement Learning (DRL) based controllers eliminate the need for domain-specific knowledge and datasets, thus providing adaptability to various scenarios. Nonetheless, a common limitation of existing studies on DRL-based controllers is their focus on driving scenarios with simple traffic patterns, which hinders their capability to effectively handle complex driving environments with delayed, long-term rewards, thus compromising the generalizability of their findings. In response to these limitations, our research introduces a pioneering hierarchical framework that efficiently decomposes intricate decision-making problems into manageable and interpretable subtasks. We adopt a two step training process that trains the high-level controller and low-level controller separately. The high-level controller exhibits an enhanced exploration potential with long-term delayed rewards, and the low-level controller provides longitudinal and lateral control ability using short-term instantaneous rewards. Through simulation experiments, we demonstrate the superiority of our hierarchical controller in managing complex highway driving situations.

17. Key Words	18. Distribution Statement
hierarchical deep reinforcement learning, automated driving, complex traffic scenarios,	No restrictions.

19. Security Classif. (of this report)	20. Security Classif. (of this page)		22. Price
Unclassified	Unclassified		

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Abstract

Developing an automated driving system capable of navigating complex traffic environments remains a formidable challenge. Unlike rule-based or supervised learning-based methods, Deep Reinforcement Learning (DRL) based controllers eliminate the need for domain-specific knowledge and datasets, thus providing adaptability to various scenarios. Nonetheless, a common limitation of existing studies on DRL-based controllers is their focus on driving scenarios with simple traffic patterns, which hinders their capability to effectively handle complex driving environments with delayed, long-term rewards, thus compromising the generalizability of their findings. In response to these limitations, our research introduces a pioneering hierarchical framework that efficiently decomposes intricate decision-making problems into manageable and interpretable subtasks. We adopt a two step training process that trains the high-level controller and low-level controller separately. The high-level controller exhibits an enhanced exploration potential with long-term delayed rewards, and the low-level controller provides longitudinal and lateral control ability using short-term instantaneous rewards. Through simulation experiments, we demonstrate the superiority of our hierarchical controller in managing complex highway driving situations. Our code is open-source and publicly available: `https://github.com/Zhang1998-06/HRL_highway` .

Keywords: Hierarchical deep reinforcement learning, automated driving, complex traffic scenarios

The results presented in this report have also been submitted for potential publication in a referreed academic journal

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The emergence of autonomous vehicles has introduced an innovative trajectory in the narrative of highway driving. Autonomous highway driving, as studied in this paper, pertains to the decision-making and vehicle control processes through which autonomous vehicles navigate and interact within highway environments A typical high-level pipeline for the vehicle sensing and control architecture is shown in Figure 1.1. It represents the convergence of cutting edge technologies and involves the integration of advanced artificial intelligent and control algorithms, sensors and sensor fusion, and real-time data processing to enable vehicles to make informed choices, such as lane changes, merging, overtaking, and responding to intricate and dynamic traffic conditions [1, 2], promising enhancements in traffic management, safety, and overall efficiency [3]. The goal is to ensure safe, efficient, and reliable autonomous travel on highways by enabling vehicles to analyze complex scenarios, anticipate potential hazards, and execute actions that align with traffic rules and user preferences [3–6]. Yet, despite remarkable strides, the task of crafting autonomous vehicles with the capability to make discerning decisions remains a formidable obstacle. This complexity emanates from the complicated amalgamation of diverse disciplines and the intricacies of real-world scenarios, necessitating profound innovation and interdisciplinary collaboration [3, 6–8].

In the rapidly evolving field of automated driving systems, three fundamental methodologies have emerged as essential strategies for addressing the complex challenge of decision-making [6,8].The first approach, characterized as conventional, embraces a modular framework that governs the decision-making process within highway scenarios [2, 9–14]. Within this paradigm, distinct modules are meticulously crafted, each responsible for addressing specific aspects of decision-making. These modules encompass specific behaviors such as lane changes, merging, overtaking, and other maneuvers crucial for highway navigation.

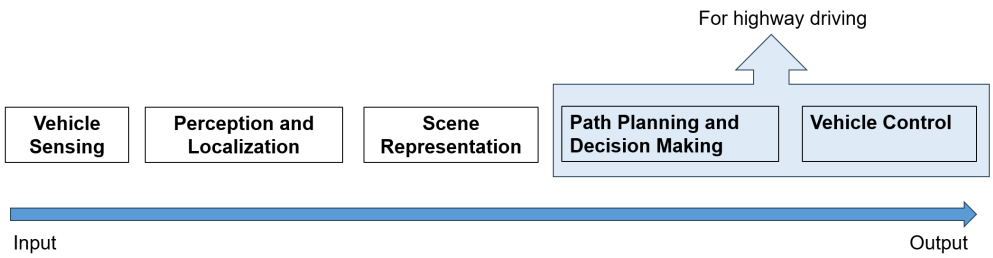


Figure 1.1: A pipeline of vehicle control process.

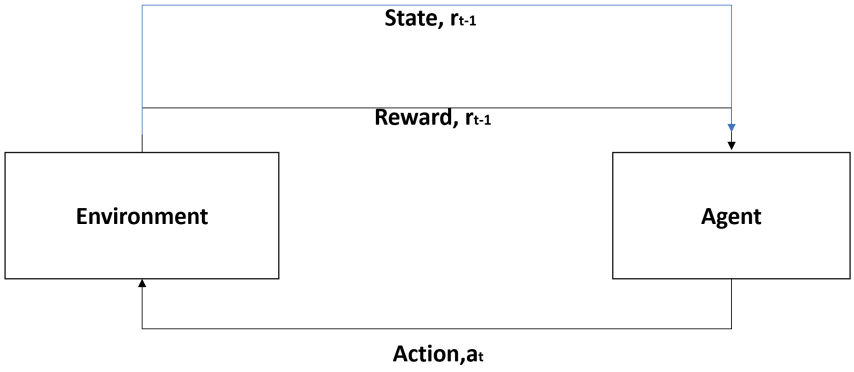


Figure 1.2: RL agent interacts with the environment and learn from the environment.

Ultimately, the amalgamation of these modules along with a behavior planning and selection algorithm culminates in a cohesive decision-making system, which then interfaces with vehicle control mechanisms. This approach, while promoting transparency and interpretability in decision processes, demands the meticulous calibration of diverse driving factors. It necessitates a comprehensive understanding of adapting to an array of driving scenarios while accounting for diverse safety considerations.

Conversely, data-driven methodologies, exemplified by supervised learning, take a different path towards decision-making refinement [15–19]. This approach simplifies the process by using machine learning methods to replicate expert driving behavior. Developing such a controller entails concurrently gathering and correlating sensor data with the respective steering, brake, and throttle actuator actions performed by human drivers. However, a notable caveat of this method is its demand for extensive data collection. Finally, reinforcement learning emerges as another compelling paradigm, aiming to equip autonomous systems with decision-making capabilities [1, 20, 21]. Through simulated interactions with the environment, the self-driving agents accumulate data and experience, facilitating the acquisition of driving strategies as illustrated in Figure 1.2. However, DRL controllers often rely on exploring the environment in the early stages of their training. Insufficient exploration might hinder the DRL controller from discovering better decision-making behaviors, which are essential for coping with more complex traffic environments [9]. Moreover, existing studies gravitate towards simpler traffic scenarios, restricting the generalizability of their findings when facing more intricate and unpredictable real-world conditions [22, 23].

1.1 Related Work

Automated driving systems typically encompass two primary components: environment modeling and vehicle control. These systems deploy an array of sensors, such as lidars and cameras, to observe the environment. Through data interpretation algorithms, these systems can detect, classify, and track relevant features and objects, while simultaneously pinpointing their own location within a dynamic driving scene [1, 2].

Once the driving environment is adequately represented, the vehicle's autonomous agent engages in the planning and decision-making process. Hierarchical controllers introduce a layered architecture, decomposing complex decision-making into manageable

layers in order to enhance system efficiency, manageability, and interpretability relative to single-level controllers. Often, the automated driving tasks are artificially segmented to simplify the overall driving challenge [8, 22–29]. For instance, separate high-level controllers might be designed for distinct traffic scenarios or for each high-level decision. This hierarchical decomposition allows for targeted solutions to specific driving tasks, with the potential for model-based control methods at the lower levels to ensure lightweight and robust responses.

The common limitations associated with current hierarchical DRL controllers for autonomous vehicles can be encapsulated within three primary aspects:

1. Oversimplified Testing Environments: The complexity of testing environments is significantly limited by factors such as homogeneous traffic types [23], sparse traffic volume [22, 27], reduced vehicle speeds [27], and fewer lanes [6]. Another common limitation is a constrained vehicle action space such that it can engage in either a lateral or longitudinal control action, but not both simultaneously [8]. These constraints do not accurately reflect the intricate and variable conditions typical of real-world automated driving.

2. Controller Design Constraints: For some hierarchical DRL controllers, the optimal driving solutions are primarily derived from the high-level controller. The low-level controller relies on rule-based and model-based [28, 30] approaches. This design approach hampers the possibilities for optimizing low-level motion planning strategies that are adaptive to the current high-level commands. Consequently, there remains a necessity to design or train an optimal low-level controller for each specific high-level controller, which does not fully exploit the potential of reinforcement learning for vehicular control.

3. Environmental Exploration Capabilities: Current implementations do not fully enable the hierarchical DRL controller's ability to improve the exploration of environments. Integrating the high-level controller for long-term strategy and the low-level controller for immediate actions can improve the system's performance and adaptability in complex scenarios. Huilkarni et al. [31] proposed a hierarchical framework that facilitates efficient exploration in complex environments, demonstrating its efficacy in an ATARI game with sparse feedback. This hierarchical DRL improves environment exploration by simultaneously training a high-level controller for goal setting and a low-level controller for action execution. However, transferring this framework to the context of highway driving presents unique challenges. The dynamic nature of highway traffic, with its continuously evolving scenarios, demands a more adaptable approach. Moreover, the distinct reward systems for high and low-level controllers necessitate careful consideration to ensure that the low-level controller's actions align with the overarching driving objectives.

In addressing these challenges, our work builds upon the hierarchical DRL paradigm, aiming to refine controller robustness and adaptability.

1.2 Exploration in Complex Traffic Scenarios using Hierarchical DRL

To address these limitations, we introduce a novel hierarchical framework that can decompose complex highway decision-making problems and empower the driving agent with the ability to effectively navigate intricate traffic environments. The design of the two-level controller also facilitates exploration at various scales. The high-level controller employs high-level actions for broader exploration, while the lower-level controller uses low-level actions for precise vehicle control and localized exploration. To ensure the high-level con-

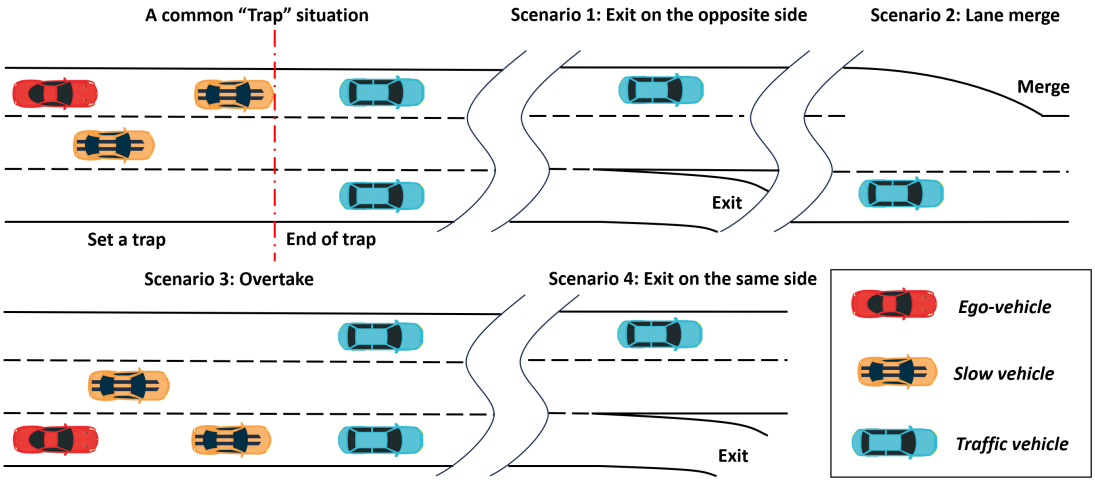


Figure 1.3: Experiment Setting to Evaluate Exploration Ability: In this setup, the DRL agent initially encounters a group of two slow-moving traffic vehicles. This scenario tests the agent's ability to navigate through this 'trap'. Upon successfully maneuvering out of this situation, the agent is then introduced to normal traffic conditions, further assessing its adaptability and exploration capabilities.

troller's extensive exploration of the environment, we train the high-level and low-level controllers separately. The high-level controller is trained using a critic function specifically designed for highway driving environments.

To test the feasibility of this hierarchical DRL controller, we have devised a highway driving scenario that includes a 'trap' situation, as shown in Figure 1.3. This 'trap' situation is represented by two vehicles moving slower than the typical traffic, forcing the ego vehicle to reduce speed to avoid a collision. The agent's objective is to maneuver out of this "Trap" formed by slow-moving traffics and subsequently accelerate to the ideal cruising speed.

Vehicles may encounter the necessity to navigate around slow-moving traffic under the following four scenarios:

1. The ego vehicle is initially positioned in a lane far from the exit it intends to take, requiring the vehicle to cross multiple lanes toward the opposite side of the highway to reach the exit.

2. The ego vehicle is in a left lane that is merging into the main traffic flow on the right. Escape from the "trap" allows the ego vehicle to integrate into the denser traffic without disrupting the flow or causing safety concerns.

3. The ego vehicle, positioned in the right lane, desires to use the far-left overtaking lane to pass and move beyond the slower vehicle.

4. The ego vehicle is in a lane close to an exit while slow-moving vehicles intend to slow down and merge to the exit, but the ego vehicle needs to continue on the main road. Escape from the "trap" will prevent the ego vehicle from following the slow-moving vehicles to the exit.

Our experiments and analyses are centered on the common 'trap' situation depicted in Figure 1.3. The subsequent four specific scenarios that necessitate the ego vehicle to navigate around slow-moving traffic fall outside the scope of this paper's investigation.

Specifically, our simulations of the "Trap" situation involves the ego vehicle, the nearby trap vehicles, and further upstream traffic vehicles which are governed by a rule-based controller. The traffic vehicles create a more complex and varied traffic environment for the agent to navigate after it moves past slow-moving traffic vehicles. This diversity of traffic patterns is important in demonstrating the stronger adaptability of automated driving strategies, while also preventing the model from overfitting to just one type of traffic pattern.

Our main contributions are as follows:

- A novel hierarchical DRL-based controller for highway driving scenario is proposed. The high-level controller enhances the vehicle's exploration potential with long-term planning and allows it to escape from slow traffic traps. The lower-level controller undertakes the intricate task of managing the vehicle's basic control maneuvers, fostering fine-grained control over driving dynamics.

- A two-step training process for the high-level controller and the low-level controller is designed and implemented.

- A speed-biased highway driving reward function is created. A trap scenario is created to test the agent's ability to explore long-term benefits in highway driving.

Chapter 2

Problem Formulation

In our methodology, we formulate the task of automated driving on highways as a Markov Decision Process (MDP) problem [32]. MDP can be described by a tuple $\langle S, A, P_{ss'}, R_s \rangle$ where the states S encapsulate the vehicle's environment and its internal status, while the actions A reflect the vehicle's potential maneuvers. The transition between state-actions is denoted as $P_{ss'} = P\left[S_{t+1} = s' | S_t = s\right]$. The rewards $R_s = E\left[R_{t+1} \mid S_t = s\right]$ are designed to guide the vehicle towards optimal driving behavior. A DRL-based controller aims to optimize the cumulative reward within a single episode, which is derived from the external reward function.

2.1 State

The observation model in our study comprehensively captures the state information of the ego vehicle and its interaction with four surrounding vehicles, incorporating a total of 26 distinct factors. The state of the ego vehicle is characterized by several parameters. The presence of the ego vehicle is constantly indicated by a binary flag, I_{ego}, which in our scenario is always set to true, reflecting the ego vehicle's persistent presence. The

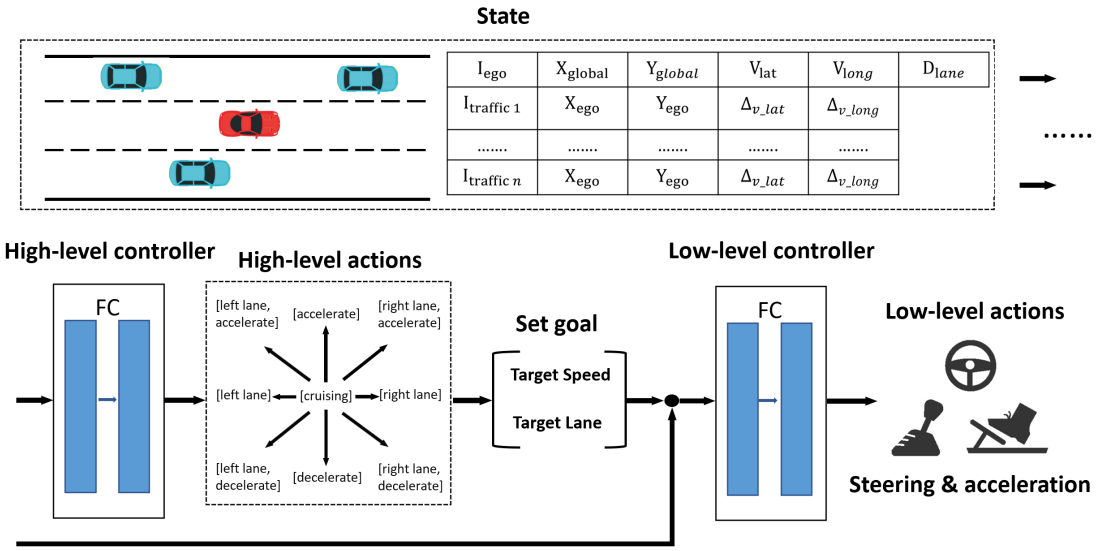


Figure 2.1: Hierarchical DRL framework for highway driving.

Table 2.1: The observation features for highway driving scenarios.

Feature	**Description**
I_{present}	Present indicator (binary indicator that vehicle in in region of interest)
X_{global}	The ego-vehicle's location in global coordinate.
Y_{global}	The ego-vehicle's location in global coordinate.
X_{ego}	The lateral distance to the ego-vehicle.
Y_{ego}	The longitudinal distance to the ego-vehicle.
V_{lat}	Lateral velocity of ego-vehicle
V_{long}	Longitudinal velocity of ego-vehicle
$\Delta v_{\text{\_lat}}$	The relative lateral speed to the ego-vehicle.
$\Delta v_{\text{\_long}}$	The relative longitudinal speed to the ego-vehicle.
D_{lane}	Lateral distance to the closest lane center.

ego vehicle's position in the global coordinate system is represented by X_{global} and Y_{global}, denoting its longitudinal and lateral locations, respectively. Additionally, the ego vehicle's velocity is captured in terms of V_{lat} and V_{long}, representing its lateral and longitudinal speeds in the global coordinate system. d_{lane} represents the lateral distance of the ego vehicle from the nearest lane center. For each of the surrounding traffic vehicles, the state information includes a binary presence indicator, $I_{traffic}$, specifying whether the vehicle exists and is within the Region of Interest (ROI) of the ego vehicle. X_{ego} and Y_{ego} represent the longitudinal and lateral distances, respectively, of the surrounding vehicles within the ROI from the ego vehicle. The lateral and longitudinal velocities of these vehicles relative to the ego vehicle's velocity are denoted by $\Delta v_{\text{\_lat}}$ and $\Delta v_{\text{\_long}}$. Detailed description of each feature is shown in Table 2.1.

2.2 Action

In this framework, at each timestep, the high-level controller processes the state information, S, and selects a high-level action, A_{high}. The available high-level actions for the high-level controller are categorized into lateral and longitudinal decision-making processes. Lateral decisions include behaviors such as left lane change, maintaining the current lane, and right lane change. Longitudinal decisions involve adjustments to the vehicle's speed, either by increasing or decreasing it by a specified increment, δ m/s. Lateral decisions do not interfere with longitudinal decisions. The selected high-level action is then interpreted as instructions for setting the target goal for the low-level controller, represented as $(l_{\text{target}}, V_{\text{target}})$. Here, l_{target} specifies a target lane index, and V_{target} designates a target speed.

For the low-level DRL-based controller, we employ a discrete action space. The low-

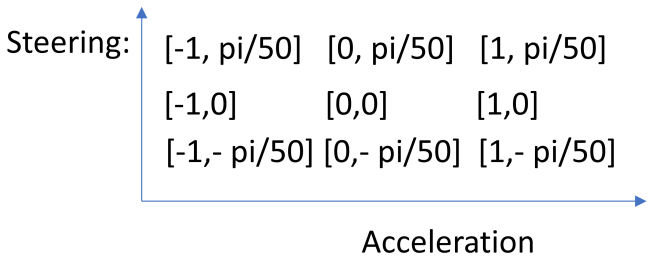


Figure 2.2: Action space for Single-level DRL controller.

level action, A_{low}, consists of a steering and acceleration pair (a, θ). The acceleration a and the steering angle θ action spaces are defined as follows:

$$a \in \{-1, 0, 1\} m/s^2 \tag{2.1}$$

$$\theta \in \{-\pi/50, 0, \pi/50\} rad \tag{2.2}$$

Thus the automated driving agent operates with a total of 9 discrete low-level actions(Figure 2.2).

2.3 Reward

In our experiment settings, we define four primary reward terms: speed, lane centering, and steering rewards, along with a penalty for accidents. In the highway driving scenario, we mainly consider three types of accidents: the vehicle comes to a stop on the highway, the vehicle oversteers and leaves the highway, and the vehicle collides with another vehicle. An episode concludes either when an accident occurs or when the maximum timesteps for an episode is reached.

Speed reward

$$r_v = \begin{cases} e^{-(v-15)^2} & , v > 15 \\ \frac{8}{25}v - \frac{19}{5} & , 12.5 < v \leq 15 \\ \frac{2}{75}v - \frac{2}{15} & , 5 < v \leq 12.5 \\ 0 & , v \leq 5 \end{cases} \tag{2.3}$$

As shown in Figure 2.3, the speed reward is structured around three distinct zones: the Dangerous Speed Zone ($v \in [0, 5]$ m/s), the Low Speed Zone ($v \in (5, 12.5]$ m/s), the Ideal Speed Zone ($v \in (12.5, 15]$ m/s), and the Over Speed Zone ($v \in (15, \infty)$ m/s). The Dangerous Speed Zone is characterized by speeds too low for highway safety, posing a risk of traffic accidents. The Low Speed Zone, while not hazardous, represents suboptimal vehicle speeds. Our target lies within the Ideal Speed Zone, with a specified speed of 15 m/s. If the ego vehicle's speed drops below this threshold, the controller encourages an acceleration back to 15 m/s. Conversely, the Over Speed Zone denotes speeds exceeding the limit, which are discouraged.

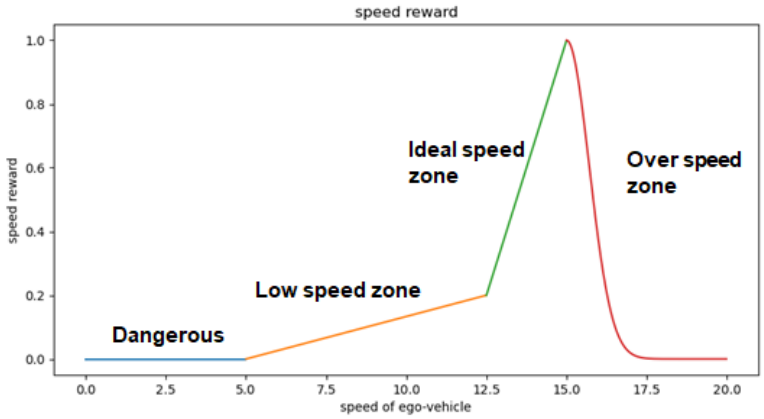


Figure 2.3: Speed Reward Function in a Highway Environment: The vehicle's speed is categorized into four distinct zones for effective management. These are: 1) Dangerous Low Speed Zone, for speeds significantly below the optimal range; 2) Low Speed Zone, for speeds moderately below the ideal; 3) Ideal Speed Zone, representing the optimal speed range for safety and efficiency; and 4) Over Speed Zone, for speeds exceeding the safe upper limit.

Lane-centering reward

$$r_y = e^{-1.5\Delta d^2} \tag{2.4}$$

As shown in Figure 2.4,the lane-centering reward is designed to encourage the vehicle to maintain its position in the center of the lane, a critical factor for safe and efficient driving, especially on curves [33]. Δd represents the distance to the center of the current lane.

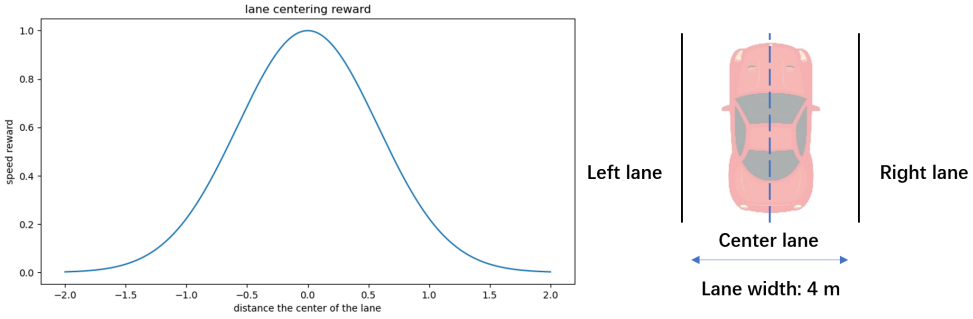


Figure 2.4: Lane Centering Reward: This reward mechanism is designed to ensure that the vehicle maintains proximity to the lane's center while cruising.

Steering reward

$$r_\theta = -|\sin(\theta)| \tag{2.5}$$

The steering reward aims to prevent excessive steering, promote smoother driving, and reduce the risk of overcorrection.

Penalty for accident

$$r_{\text{total}} = \begin{cases} \frac{w_v r_v + w_\theta r_\theta + w_y r_y}{w_v + w_\theta + w_y} & \text{if no accident} \\ -10 & \text{if accident} \end{cases} \tag{2.6}$$

Each of these reward terms is assigned a specific weight factor, which helps to balance their influence on the vehicle's learning process. Additionally, the model imposes a significant penalty of -10 for critical failures such as crashes, driving out of the lane, or stopping in the lane. Specific weight values can be found in Table 3.1.

2.4 Traffic Vehicle Control

To control the traffic vehicles, we implement the Intelligent-Driver Model (IDM) [13] for longitudinal control and the Minimizing Overall Braking Induced by Lane Change (MOBIL) model [14] for lateral decision-making.

IDM [13] is a time-continuous vehicle-following model used to simulate the behavior of individual vehicles in a traffic flow. In the IDM model, each vehicle aims to keep a safe distance from the preceding vehicle while maintaining its desired speed. The desired speed is a function of the driver's preference and the posted speed limit. The model also includes the driver's reaction time and the vehicle's acceleration and deceleration capabilities. The IDM model ensures that vehicles maintain a safe following distance, avoid collisions, and adjust their speed according to the traffic flow. It is a car-following model that considers various factors such as the distance between vehicles, their relative velocity, and the desired speed of the vehicle. The IDM model uses these factors to control the acceleration and deceleration of a vehicle and maintain a safe distance from other vehicles. This represents an autonomous vehicle model that is collision-free and has self-adaptive capabilities on highways.

The IDM model [13] calculates the desired acceleration of the vehicle using:

$$acc = a \left[1 - \left(\frac{v}{v_{desired}} \right)^{\delta} - \left(\frac{s_{desired}}{s} \right)^2 \right] \tag{2.7}$$

where s is the gap to the front vehicle, $s_{desired}$ is the desired distance to the preceding vehicle, v is the vehicle's current velocity, a is a parameter for acceleration, and $v_{desired}$ is the target velocity for the controlled vehicle. The desired distance to the preceding vehicle is calculated using

$$s_{desired} = S_0 + Tv + \frac{v\Delta v}{2\sqrt{ab}} \tag{2.8}$$

where S_0 is the desired distance gap, T the desired time gap to the preceding vehicle, and a and b are acceleration and deceleration parameters. The desired distance is calculated based on the vehicle's speed and the relative speed between the current vehicle and the preceding vehicle.

MOBIL [14] is a model that governs the lane-changing behavior of autonomous vehicles. MOBIL aims to minimize the total amount of braking induced by lane changes, while also maximizing the overall traffic flow. It considers factors such as the relative speed and position of surrounding vehicles, as well as the ego-vehicle's acceleration and deceleration capabilities, to determine whether changing lanes would be beneficial.

The MOBIL model incorporates several decision rules to determine the benefit of a lane change. First, the MOBIL model checks if there is a gap in the target lane that the

vehicle can occupy without causing any conflict with other vehicles. If there is a gap, the model checks if the vehicle can accelerate to its desired speed before reaching the gap. If the vehicle can safely accelerate to the desired speed, it performs the lane change. If there is no gap in the target lane, MOBIL checks if the lane change would allow the vehicle to travel at a higher speed than the current lane. If so, the vehicle can perform the lane change if it can safely accelerate to the desired speed. If neither of these conditions is met, the vehicle will remain in its current lane. By using MOBIL, autonomous vehicles can make safe and efficient lane-changing decisions.The MOBIL model decides when to perform a lane change based on the impact on other drivers:

$$(a'_e - a_e) + p[(a'_b - a_b) + (a'_a - a_a)] > a_{\text{th}} \tag{2.9}$$

where $(a'_e - a_e)$, $(a'_b - a_b)$, $(a'_a - a_a)$ represent the acceleration difference of the driver's vehicle, the following vehicle before the lane change and the following vehicle after the lane change, respectively, p represents the politeness factor which weighs the disadvantages imposed on other drivers due to the lane change, and a_{th} is the acceleration threshold. The control parameters for the traffic vehicles are detailed in Table 2.2.

Table 2.2: Traffic vehicle parameters

Parameters	value
Vehicle length	$5m$
Vehicle Width	$2m$
Road Width	$4m$
Steering range a_t	$[-1, 1]m/s^2$
Acceleration range θ_t	$[-\pi/36, \pi/36]rad$
Politeness factor p	0.5
Acceleration threshold a_{th}	$0.2m/s^2$
Acceleration a	$0.5m/s^2$
Deceleration b	$0.5m/s^2$
Exponent value δ	4
Desired distance $s_{desired}$	$10m$
Desired velocity $v_{desired}$	$12.5m$
Desired distance gap S_0	$10m$
Desired time gap T	$1.5s$

2.5 Low-level Motion Planner

Our hierarchical controller employs a two-step training process. To train the high-level controller, as shown in Figure 2.5, we implement a lateral low-level controller to execute lane changes and a longitudinal low-level controller to track the target speed (details can be found in Section 4.2). The same lateral low-level controller is also used to execute lane changes for traffic vehicles using the MOBIL controller. The lateral low-level controller is based on the lateral distance to the target lane center:

$$v_{lateral} = -K_{lat}\Delta d_{\text{\_target}} \tag{2.10}$$

$\Delta d_{\text{\_target}}$ is the lateral distance to target lane center line. K_{lat} is the proportional gain for this speed planner. In the lateral direction, the heading controller deals with the heading

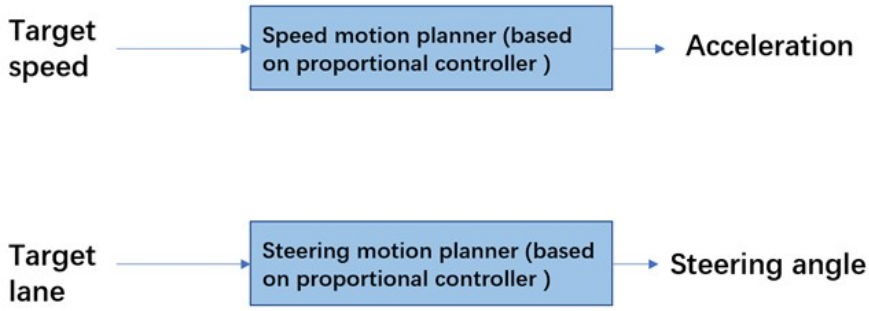


Figure 2.5: Speed and steering motion planner

of the vehicle with a similar proportional-derivative action:

$$\theta_{lateral} = K_\theta(\theta_{target} - \theta) \tag{2.11}$$

θ is the vehicle's current heading angle. θ_{target} is the vehicle's heading angle after lane change behavior. The heading angle after lane change will remain aligned with the direction of the target lane. K_θ is a proportional gain for this heading controller. This lane change motion planner will plan the trajectory of a vehicle from an initial lane to a target lane. Once the lane change decision is made, the motion planner will provide steering angle for each time step until the ego-vehicle reached the target lane center line.

In longitudinal direction, we interpret the act of accelerating or decelerating at a high-level as setting a target speed. Speed up means to increase the target speed by δ m/s (a numeral factor) and slow down means to decrease the target speed by δ m/s. δ is a parameter than we can set based on the ideal speed of autonomous driving problem. Since we are studying a low-speed lane change problem with an ideal speed of 15m/s, we set δ to 2.5 m/s. After we set the target speed, a speed motion planner will be implemented to control the acceleration and track the target speed:

$$a_{acc} = K_{speed}\Delta v_{\text{-target}} \tag{2.12}$$

where K_{speed} is the proportional gain for speed motion planner.

2.6 Trap Initialization

As previously mentioned, we define a typical trap scenario where the ego vehicle is in the far-left lane, with the trap vehicles positioned in front and in the lane to the right of the ego vehicle as shown in Figure 2.6. "Trap Vehicle 1" is initialized with a longitudinal distance D_1 from the ego vehicle. "Trap Vehicle 2" maintains a close longitudinal distance D_2 from the ego vehicle, ensuring that it will block the ego vehicle if it attempts to change lanes or accelerate while maintaining its current speed. Both trap vehicles are set to travel at a speed slower than the normal traffic flow. To maintain the presence of this trap, we set both trap vehicles to the same speed for the entire episode. The initial distances between the ego vehicle and the trap vehicles D_1 and D_2 vary during training and testing. During training, we initial distances are drawn from a uniform distribution to introduce variability. For testing, we use more relaxed conditions. Detailed information can be found in Table 3.1.

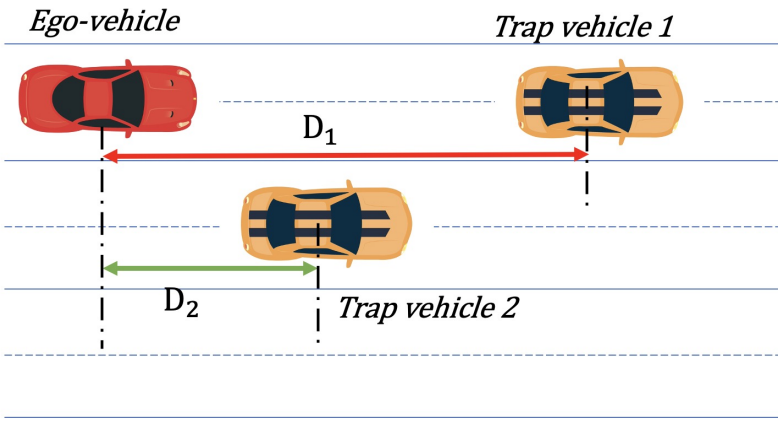


Figure 2.6: The trap vehicles are initialized with a longitudinal distance with respect to the ego vehicle.

2.7 Deep Reinforcement Learning

A single-Level DRL controller directly map the ego-vehicle's observation to low-level action, as shown in Figure 2.7. In this project, we use Q-learning [34] reinforcement learning algorithm with a discrete action space. Its objective is to learn an action-value function Q(s,a), which estimates the expected cumulative reward for taking action a in state s. The Q-value update function can be expressed as follow:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \Bigg(\tag{2.13}$$

The Q-learning algorithm updates the Q-value function repeatedly, using the Bellman equation. This equation decomposes the Q-value of a state-action pair into two parts: the immediate reward r that the agent obtains from taking the action a in the state s, and the discounted maximum expected future reward $\gamma \max_{a\prime} Q(s\prime, a\prime)$ that the agent can obtains form taking the action $a\prime$ in state $s\prime$. The parameter γ is a discount factor that balancing the significance of future rewards with respect to immediate rewards. the learning rate α is the step size for each update during the training process. The temporal difference error is calculated as $(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$, which is the difference between the expected Q-value and the actual Q-value observed in the current state-action pair. The update rule adjusts the Q-value towards the updated estimate.

Deep Q-Network (DQN) [35] is an RL algorithm that use deep neural networks as the approximate function to learn a Q-function that can handle large and complex state and action spaces. In DQN, the agent learns to maximize its expected cumulative reward by

Figure 2.7: A single-level DRL controller for highway decision making

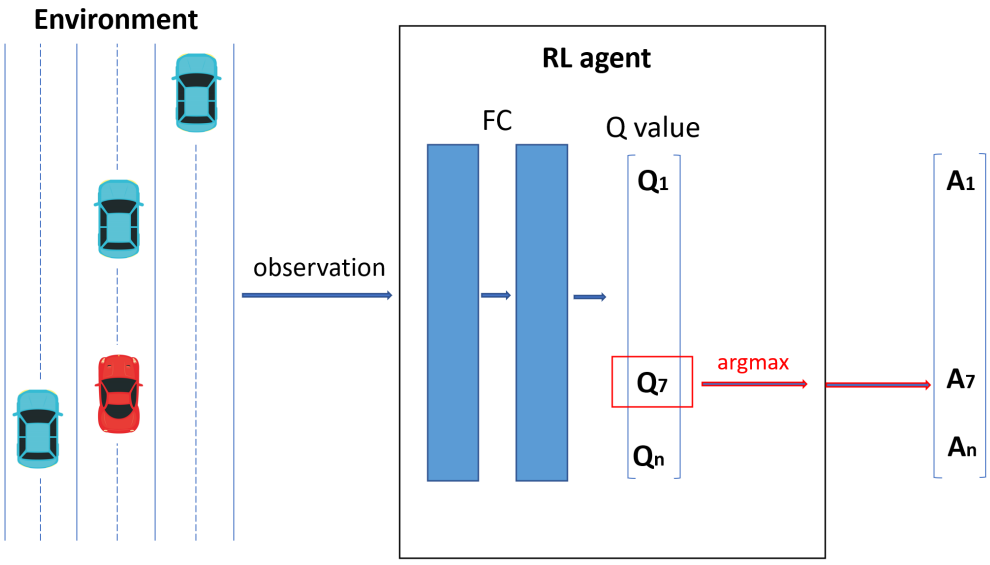


Figure 2.8: A Q-learning based DRL controller for highway driving

updating the network parameters using stochastic gradient descent (SGD) to minimize the difference between the predicted Q-values and the target Q-values. DQN also employs two techniques to improve its stability and convergence: experience replay and target networks. Experience replay involves storing past experiences in a buffer and randomly sampling a subset of them to train the neural network, which reduces the correlation between the updates and improves learning efficiency. Target networks involve creating a separate copy of the Q-network to generate target Q-values during training, which reduces the impact of temporal difference errors and improves stability. A Q-learning based Single-level DRL controller is shown in Figure 2.8. The pseudocode of the DQN Algorithm is shown in Algorithm 1.

Algorithm 1 DQN Algorithm

Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
Initialize experience replay memory D with capacity N
Initialize state s_1
for $i = 1$ to T **do**
    Choose a random action a_t with probability ϵ
    otherwise select $a_t = \arg\max_{a \in A} Q(s_t, a; \theta)$ (greedy policy)
    Execute action a_t in the emulator and observe next state s_{t+1} and reward r_t
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    for a minibatch: random sample of transitions (s_j, a_j, r_j, s_{j+1}) from D **do**
        if s_{j+1} is terminal **then**
            $y_j \leftarrow r_j$
        else if s_{j+1} is non-terminal **then**
            $y_j \leftarrow r_j + \gamma \max_{a' \in A} \hat{Q}(s_{j+1}, a^{'}; \theta^{-})$
        end if
        Update Q by minimizing the loss $(y_j - Q(s_j, a_j; \theta))^2$
                with respect to network parameters θ using gradient descent
        Every C steps, reset $\hat{Q} \leftarrow Q$
    end for
end for

Despite its success, DQN can suffer from overestimation of the Q-values, which can lead to suboptimal policies. Double DQN algorithm [36] is an extension of DQN that addresses this issue by decoupling the action selection and Q-value estimation steps, using one network to select the best action and another network to evaluate its value.

This decoupling of the action selection and Q-value estimation reduces the overestimation of the Q-values and improves the accuracy of the learned policies [36]. DDQN has been shown to outperform DQN on a variety of benchmark tasks, and it is now a widely used and popular algorithm in RL. The core of the DDQN's [35, 37] efficacy lies in its Q-value function. The Q-value function, denoted as $Q(s, a)$, estimates the expected utility of taking action a in state s, and then following the optimal policy thereafter. Formally, the Q-value function in the context of the DDQN is defined as follows:

$$L(\theta) = \mathbb{E}\left[\left(r_t + \gamma \max_{a'} Q(s'_t, a'; \theta^-) - Q(s_t, a_t; \theta)\right)^2\right]\Bigg(\tag{2.14}$$

where r_t is the reward for taking action a_t in state s_t. γ is the discount factor. The loss is computed as the mean squared error (MSE) of the difference between the target Q-value (computed with θ^-) and the predicted Q-value (computed with θ) for the current state-action pair. The pseudocode of Double-DQN Algorithm is shown in Algorithm 2.

Algorithm 2 Double-DQN Algorithm

Initialize replay memory D with capacity N
Initialize main Q-network parameters θ
Initialize target Q-network parameters $\theta^- = \theta$
Initialize exploration rate ϵ
Set minibatch size M
for *episode* $= 1$ to M **do**
    Reset environment state S_0
    Set cumulative reward $R \leftarrow 0$
    for *timestep* $= 1$ to T **do**
        With probability ϵ select a random action a_t
        otherwise select $a_t = \arg\max_{a \in A} Q(s_t, a; \theta)$
        Execute action a_t in the emulator and observe next state s_{t+1} and reward r_t
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
        Sample random minibatch of transitions (s_i, a_i, r_i, s_{i+1}) from D
        Set target $y_i = r_i + \gamma Q(s_{i+1}, \arg\max_{a' \in A} Q(s_{i+1}, a'; \theta); \theta^-)$
        Update main Q-network parameters using gradient descent:

$$\nabla_\theta \frac{1}{M} \sum_{i=1}^{M} \Big(y_i - Q(s_i, a_i; \theta)\Big)^2$$

        Every C timesteps, update target Q-network parameters: $\theta^- \leftarrow \theta$
    end for
    Update exploration rate: $\epsilon \leftarrow max(\epsilon_{min}, \epsilon_{decay} * \epsilon)$
    Update cumulative reward: $R \leftarrow R + r_t$
end for

Chapter 3

Proposed Method

3.1 Hierarchical DRL Framework

Inspired by h-DQN [31], we introduce our hierarchical DRL framework, as depicted in Figure 3.1. The high-level controller is defined with an input layer corresponding to the number of observation features and an output layer corresponding to the number of goal features. It consists of two fully connected layers, each with 512 neurons, utilizing ReLU activations. The low-level controller takes the goal and new observations as inputs and outputs the discrete action pair. In practice, the goals will be interpreted as the difference in distance of the current state to the target lane center, $\Delta d_{\text{\_target}}$, and the difference between the current longitudinal speed and the target longitudinal speed,$\Delta v_{\text{\_target}}$. These can be represented by the following equations:

$$\Delta d_{\text{\_target}} = X_{\text{target\_lane}} - X_{\text{global}} \tag{3.1}$$

$$\Delta v_{\text{\_target}} = V_{\text{target}} - V_{\text{long}} \tag{3.2}$$

The low-level controller has the same structure as the high-level controller. The low-level action is the discrete steering and acceleration pair (a, θ) mentioned previously. Dur-

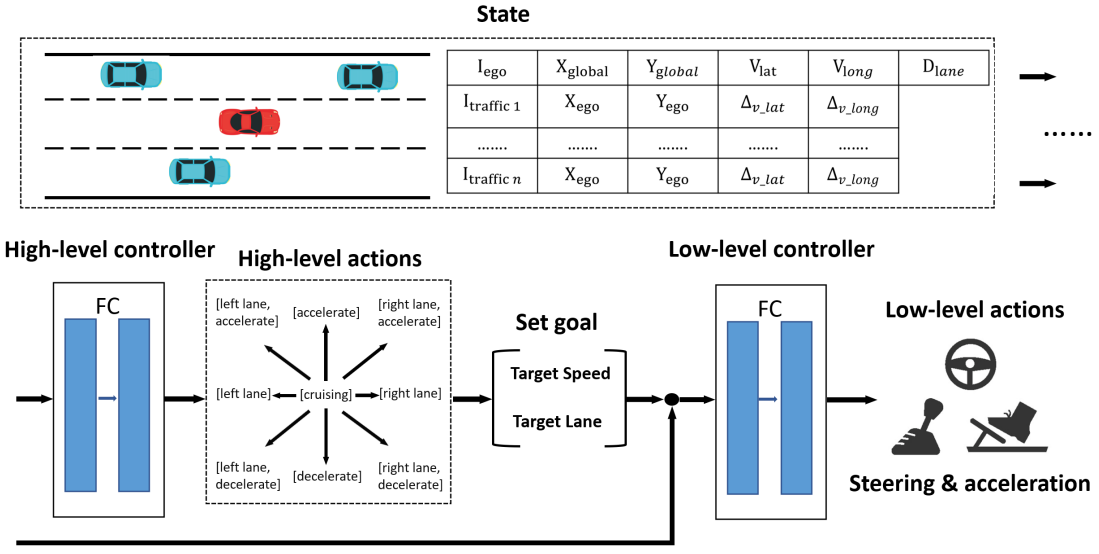


Figure 3.1: Hierarchical DRL framework for highway driving.

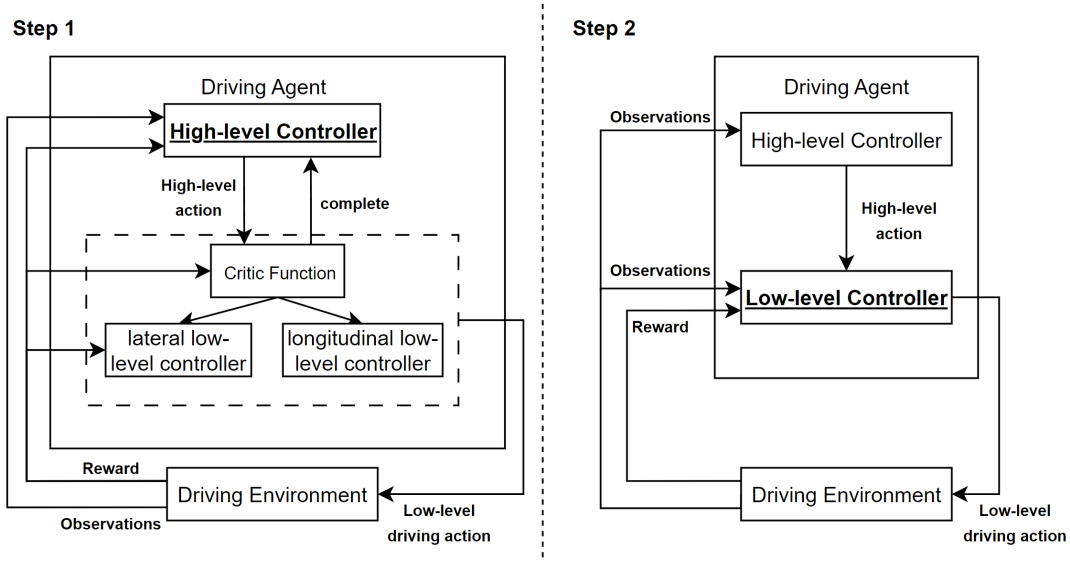


Figure 3.2: Two-Step Training Process for High-Level and Low-Level Frameworks: Initially, the high-level controller is trained using a model-based motion planner and a critic function. Subsequently, this trained high-level controller is employed to facilitate the training of the low-level controller.

ing testing, at each timestep, the high-level controller first updates its goal based on the new observations. The low-level controller then combines this new goal with the new observations to generate the low-level actions within the same timestep. It is important to note that, unlike the h-DQN [31] design, where the performance of the low-level controller is critiqued by the intrinsic reward [38], the vehicle's performance in our framework is exclusively dependent on the external environment reward, which aligns with the high-level controller's training reward.

3.2 Training the Hierarchical DRL Agent

To separate the exploration ability of the high-level and low-level controller, we implement a two-step training method for the architecture shown in Figure 3.2. The initial step only involves training the high-level controller. To facilitate using the high-level controller to control the vehicle exclusively, in this first step we use a rule-based motion planner to control the vehicle's steering and acceleration based on the high-level goal. This also allows us to concentrate on the training and optimization of the high-level controller without the complexities introduced by an untrained low-level controller. To be specific, once the high-level controller sets the goal, the rule-based motion planner uses low-level action within the defined action space to reach the goal. To evaluate whether the goals set by the high-level controller are being effectively reached, we introduce a critic function. If the lateral distance between the ego vehicle and the target lane center is less than a threshold D_δ, the vehicle has successfully reached the target lane. If the speed difference between the current speed and target speed is less than a threshold V_δ, the vehicle has reached the target speed. The lateral and longitudinal critic functions are thus:

$$|\Delta d_{\_\text{target}}| < D_\delta \tag{3.3}$$

$$|\Delta v_{\_\text{target}}| < V_\delta \tag{3.4}$$

If both the lateral and the longitudinal critic functions are satisfied, this means the rule-based motion planner has fully executed the high-level goal. The high-level controller is permitted to set a subsequent goal only upon the achievement of the current goal. We train the high-level controller for 1000 episodes and store the controller that has the best mean rewards for 10 episodes.

Once the high-level controller is adequately trained, we proceed to the second step: training the low-level controller. In this phase, we use the previously trained high-level controller and focus on the low-level controller, which operates at a frequency of 2 Hz. All controllers are using Epsilon-Greedy exploration strategy. The exploration probability is reduced from an initial value 0.5 to a final value of 0.02 over 1000 steps. Other training parameters are detailed in Table 3.1. This targeted training allows the low-level controller to learn how to accurately control the vehicle under the guidance of the already optimized high-level controller.

Table 3.1: Experiments parameters

Parameters	value
speed reward weight w_v	1.5
steering reward weight w_θ	0.05
lane centering reward weight w_y	0.05
threshold D_δ	0.3 m
threshold V_δ	0.3 m/s
training episode	2000
Duration	250
batch size	64
experience reply memories	5E4
learning rate	1E-3
discount	0.8
D_1 during training	[14.80,16.44] m
D_2 during training	[4.06,7.43] m
D_1 during testing	15.62 m
D_2 during testing	6.61 m

Chapter 4

Experimental Results

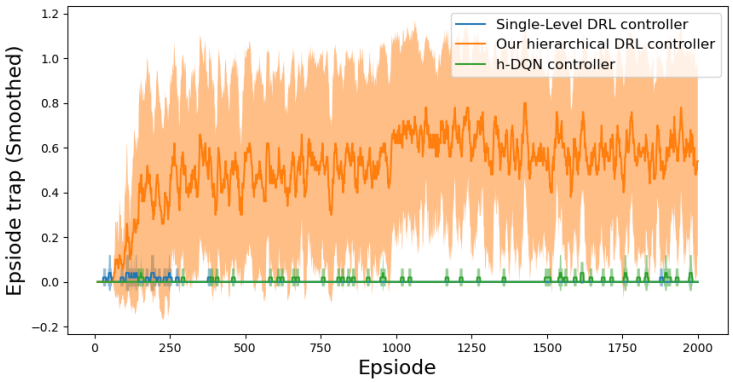
To test the algorithm's capability in environmental exploration while prioritizing long-term rewards, our approach involved integrating slow-moving vehicles as a constraint on the ego vehicle's maneuvers. Our experiments were conducted in the highway-env [39]. The criterion for successfully escaping a traffic trap is defined as the ego vehicle's ability to find an overtaking path within the episode duration. Specifically, the ego vehicle is deemed to have successfully escaped the trap if its rear bumper surpasses the front of all trap vehicles before the episode terminates.

4.1 Training Process Results

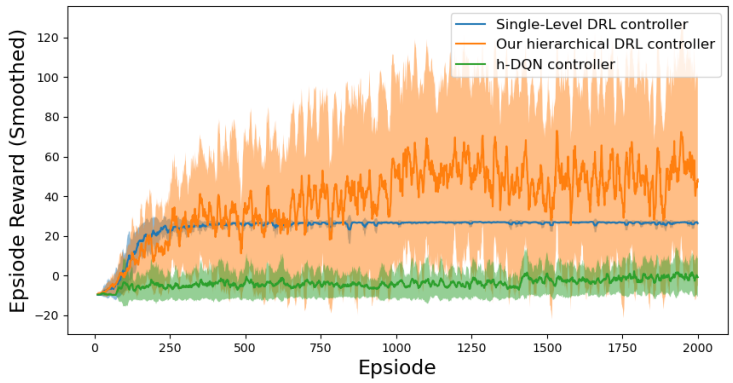
In our study, the single-level DRL controller employs a direct mapping from observation to optimal low-level action, utilizing two fully connected layers, each with 512 parameters, and is trained in a single step, in contrast to the hierarchical DRL controller which is trained using the two-step process described above. We compared the training processes of the hierarchical DRL controller and the single-level DRL controller in three aspects: the average success rate of escaping traps (Figure 4.1a), the average reward (Figure 4.1b), and the average speed (Figure 4.1c) within an episode. Figure 4.1a shows that during the initial exploration phase, both the hierarchical DRL controller and the single-level DRL controller exhibited tendencies to escape from traps. However, as the exploration probability decreased, the single-level DRL controller showed a minimal probability of escaping from difficult situations.

Figure 4.1c and Figure 4.1b compare the differences in average speed and average reward per episode between the hierarchical DRL controller and the single-level DRL controller. Throughout the learning process, the single-level DRL controller's strategy revolved around maintaining speed and safety distance from the trap vehicle ahead. It failed to adopt overtaking strategies, limiting the ego vehicle's potential for long-term speed gains. Consequently, due to the adoption of a cruising strategy that matched the speed of the vehicle ahead, the ego vehicle achieved nearly constant average speed and reward in each episode. On the other hand, our hierarchical DRL controller enabled the agent to accelerate and closely approach the target speed of 15m/s after overtaking the trap vehicle, thereby obtaining a higher long-term speed reward.

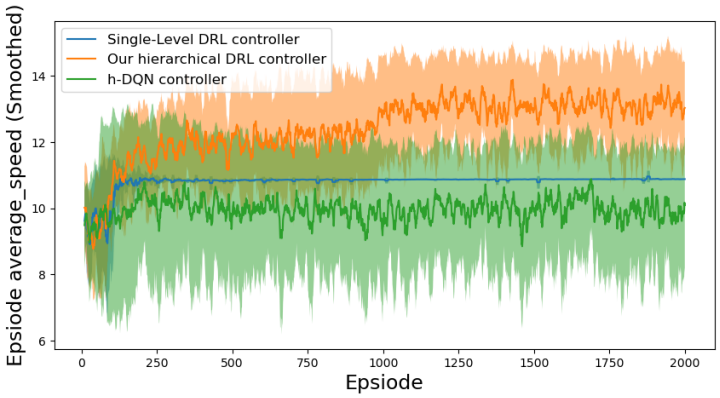
It is noteworthy that the hierarchical DRL controller exhibited greater variance in all of the evaluation criteria compared to the single-level DRL controller. The single-level DRL controller has learned a relatively simple following strategy from a straightforward trap traffic pattern which has less uncertainty, whereas the hierarchical DRL controller had the additional challenge of learning overtaking maneuvers while subsequently adapting to the surrounding traffic.



(a)



(b)

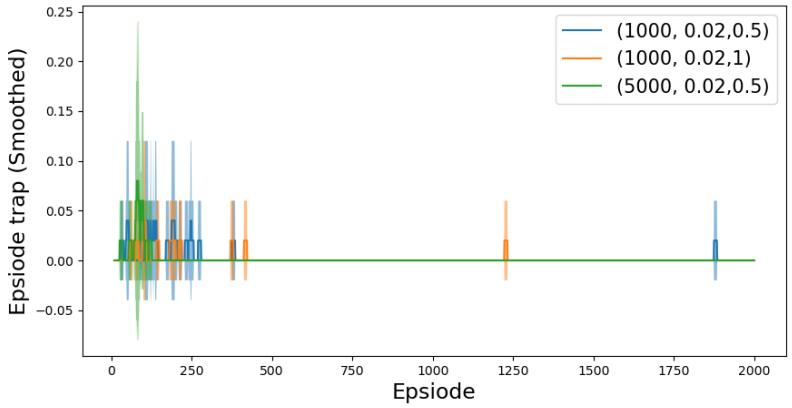


(c)

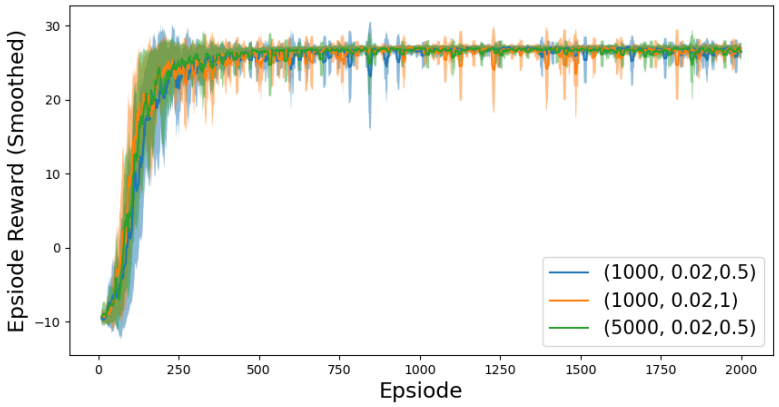
Figure 4.1: Single-level, hierarchical DRL, and h-DQN controller performance during training episodes. (a) Average success rate: the hierarchical DRL controller exhibits a higher likelihood of successfully navigating out of situations encumbered by low-speed traffic. (b) Average reward: The hierarchical DRL demonstrates superior performance, yielding higher average rewards per episode. (c) Average speed: The hierarchical DRL controller consistently achieves a higher average speed.

4.2 Exploration Capability

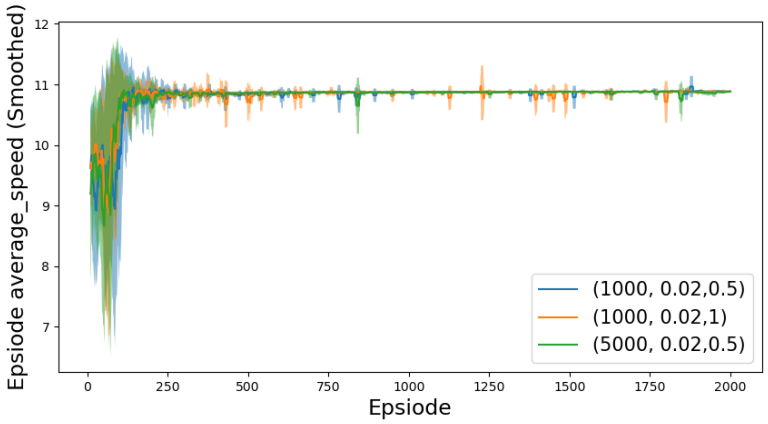
To compare the exploration capability of the hierarchical controller in the environment, we enhanced the exploration ability of the single-level DRL controller using Epsilon-Greedy strategy to a certain extent. We refined the initial exploration strategy by tuning the epsilon parameter in the Epsilon-Greedy algorithm of a single-level DRL controller. Within the same setting, we executed five training experiments, adjusting the epsilon value to anneal from an initial value of 1 to a final value of 0.02 across 1000 steps, and from 0.5 to 0.02 over 5000 steps. Despite these modifications, Figures 4.2a, 4.2b, and 4.2c reveal that the success rate for evading traps showed no significant enhancement compared to the baseline, essentially remaining negligible after 2000 episodes. This result implies that simply increasing the initial exploration probability within a certain range does not guarantee an improved ability to detect viable escape paths. In contrast, the hierarchical DRL controller steadily improved its ability to escape traps during the early stages and continued to show enhancements even when the exploration value stabilized.



(a)



(b)



(c)

Figure 4.2: Comparing three single-level DRL controllers with different initial exploration. (a) The success rate in evading low-speed Traffic within a training episode. (b) Average reward within a training episode. (c) Average speed during training.

4.3 Exhibited Driving Behaviors

To elucidate the driving policies for managing slow-moving vehicles, as implemented by both single-level and hierarchical DRL controllers, we present in Figure 4.3 the vehicle's position changes in an ego vehicle centered point of view at selected video rendering moments. At moments t=1s, 2s, 3s and 4s, the single-level DRL controller demonstrated its ability to match the speed of the vehicle ahead while maintaining a safe distance. In contrast, the hierarchical DRL controller, guided by high-level objectives, initially reduced speed and then performed two lane changes to overtake the slower vehicle. This strategy led to a noticeable decrease in immediate rewards due to deceleration. By t=9s, after successfully navigating past the slow-moving vehicle, the hierarchical DRL controller achieved significantly higher immediate and accumulated rewards compared to the single-level DRL controller.

Figures 4.4a and 4.4b further elucidate the driving policies over the initial 40 timesteps. The hierarchical DRL controller decreased its speed in the first 5 timesteps, then accelerated to approach the optimal speed. Although the single-level DRL controller also initially reduced speed, its objective was to maintain a safe following distance. Once this distance was secured, the single-level controller increased its speed to minimize the gap. The initial speed reduction by the hierarchical controller was more substantial than that of the single-level controller, necessitating the overcoming of the negative impact associated with prolonged periods of low reward. The hierarchical DRL controller decelerated and maintained a safe distance to execute a secure overtaking maneuver.

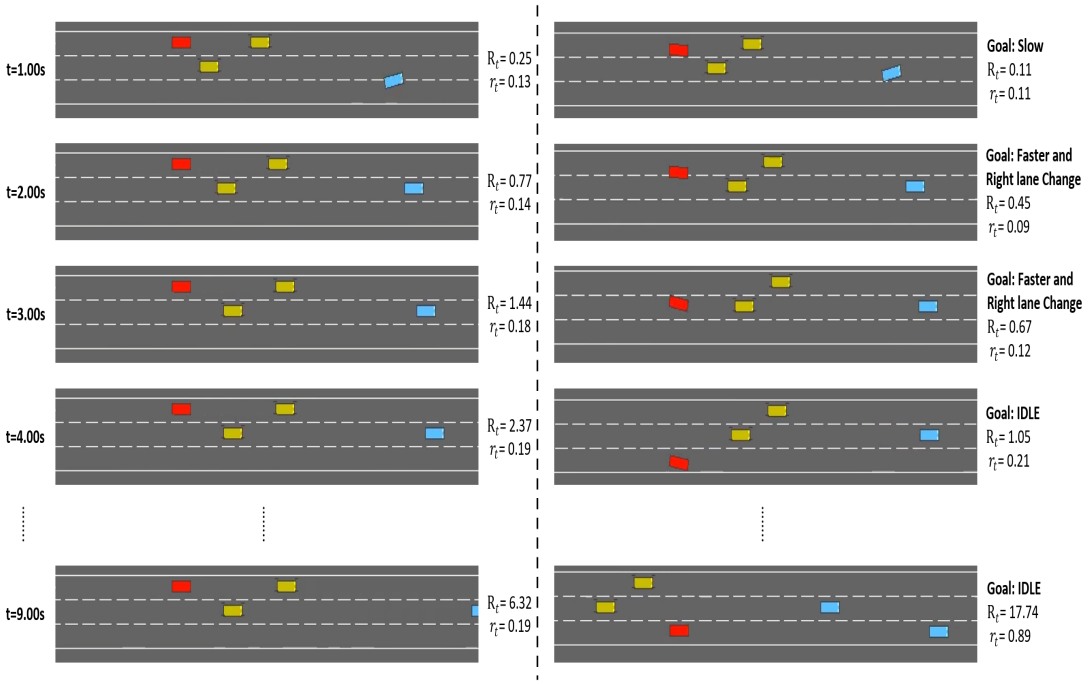
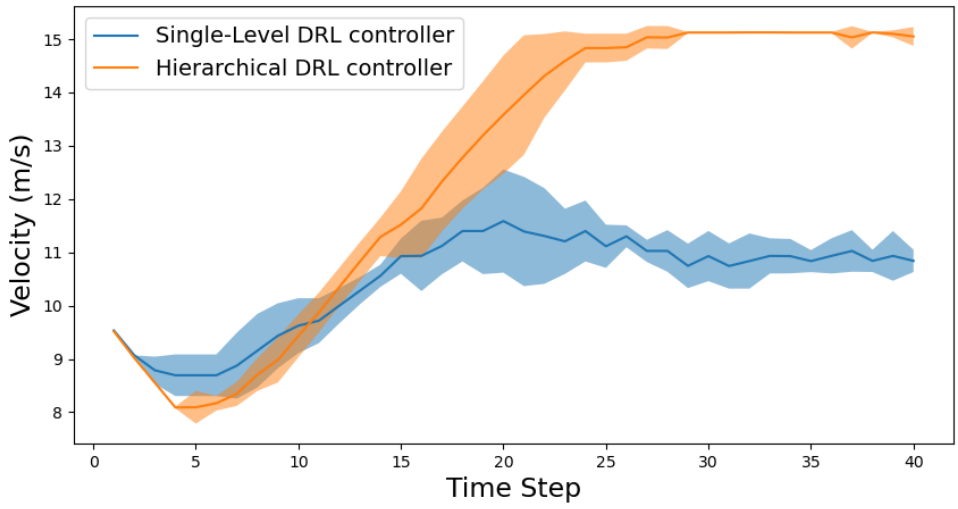
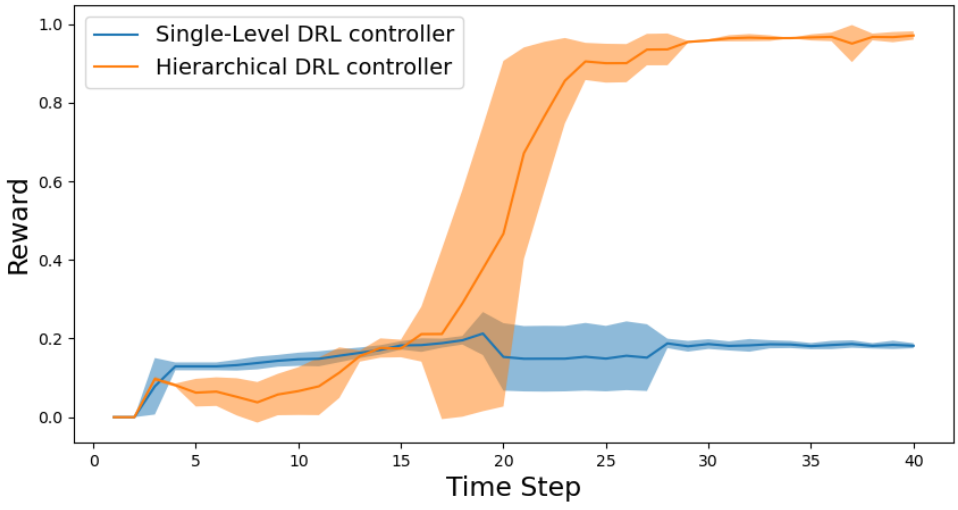


Figure 4.3: The driving policies for slow-moving vehicles as implemented by both single-level and hierarchical DRL controllers. The red, yellow, and blue vehicles represent the ego, trap, and traffic vehicles, respectively. r_t represents the instant reward received at video rendering time t. The accumulated reward $R_t = \sum_{k=0}^{t} r_t$ represents the sum of instance rewards over time.



(a)



(b)

Figure 4.4: (a) Velocity profile of the initial 40 timesteps of the single-level and hierarchical DRL controllers. (b) Reward profile of the initial 40 timesteps of the single-level and hierarchical DRL controllers.

We further compare the average results for reward, speed, and the success rate of escaping traps over the final 50 episodes of the training period in 5 runs, as summarized in Table 4.1. The hierarchical framework enhances the ego vehicle's ability to learn and execute escape maneuvers from traps through low-level actions.

Table 4.1: Training Phase Comparison

Training phase	Single-level DRL Controller	Hierarchical DRL Controller
Average reward	25.82	47.94
Average speed	10.89	13.089
Average success rate to get out of trap	0%	60%

After training the controllers, we conducted 300 episodes of tests in a relaxed trap scenario, each consisting of 25 timesteps. The results of these tests are compiled in Table 4.2. In this consistent testing environment, the hierarchical DRL controller outperformed the single-level DRL controller in terms of average reward, average speed, and the average success rate of escaping traps, also covering a greater distance within the same time frame.

Table 4.2: Performance Comparison of Controllers in Overtaking Maneuver

Testing phase	Single-level DRL Controller	Hierarchical DRL controller
Average accumulated reward	3.32	13.20
Average speed	10.29	13.42
Average success rate to get out of trap	0%	97.67%
Average travel distance	257.29	331.60
Accident rate	0%	2.33%

As shown in Figure 4.5, our hierarchical controller achieves higher reward compare to the single-level DRL controller. The major motivation for escaping from the trap is the accumulated speed reward after escaping. As shown in Figure 4.6, the acculumated speed reward within an episode achieved within an episode for the hierararchcal controller is higher then the single-level DRL controller.

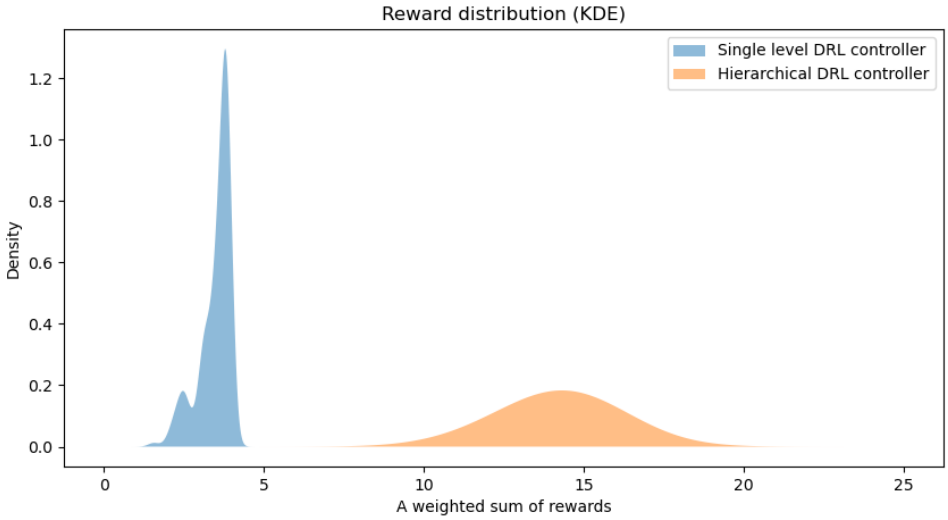


Figure 4.5: The average reward distribution within an episode

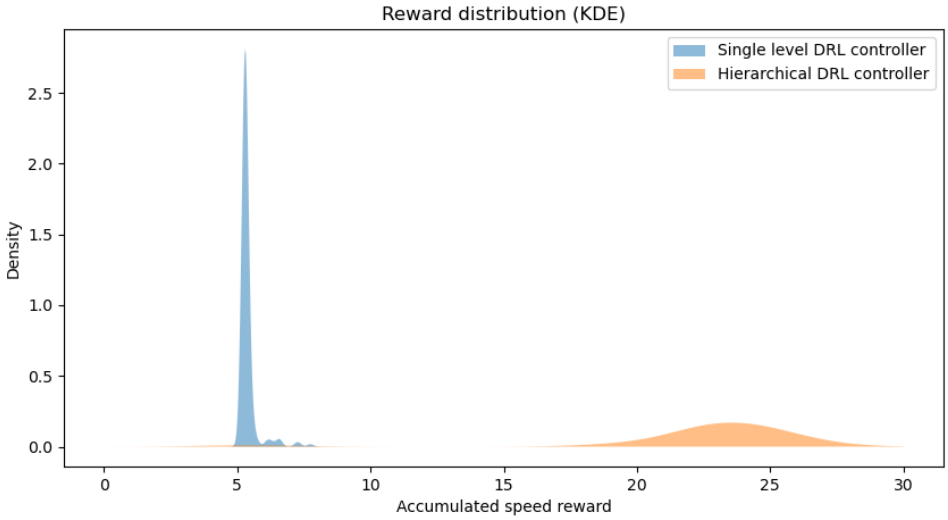


Figure 4.6: Distribution of accumulated speed reward achieved within an episode

For this environment, we encourage the vehicle to stay in an ideal speed within 12.5m/s to 15m/s and we don't encourage the ego vehicle to continue driving behind the trap vehicle. As shown in Figures 4.7 and 4.8 the single-level DRL controller spends most of the time under 12.5m/s and stays behind the trap vehicle, while the hierarchical DRL controller, as shown in Figure 4.9 can not only escape from the trap but also staying close the to the ideal speed of the highway.

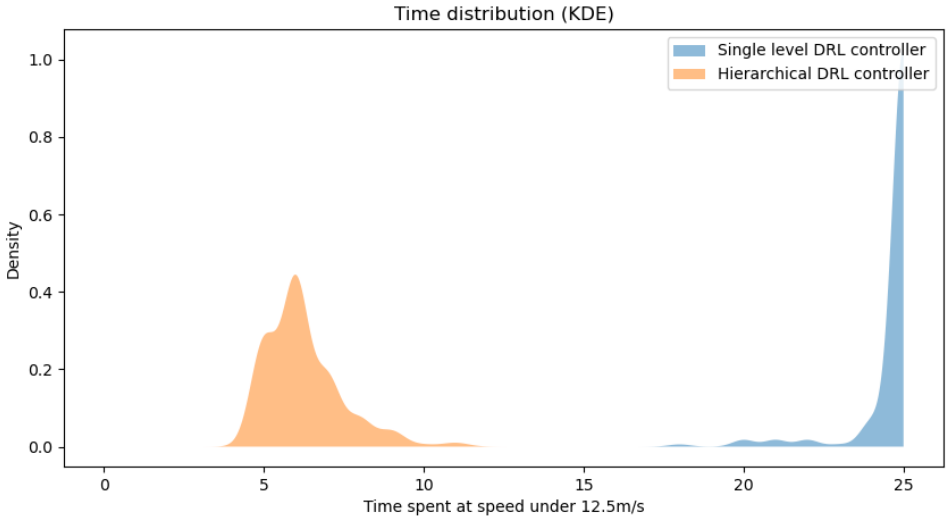


Figure 4.7: Distribution of time spent at speed under 12.5m/s

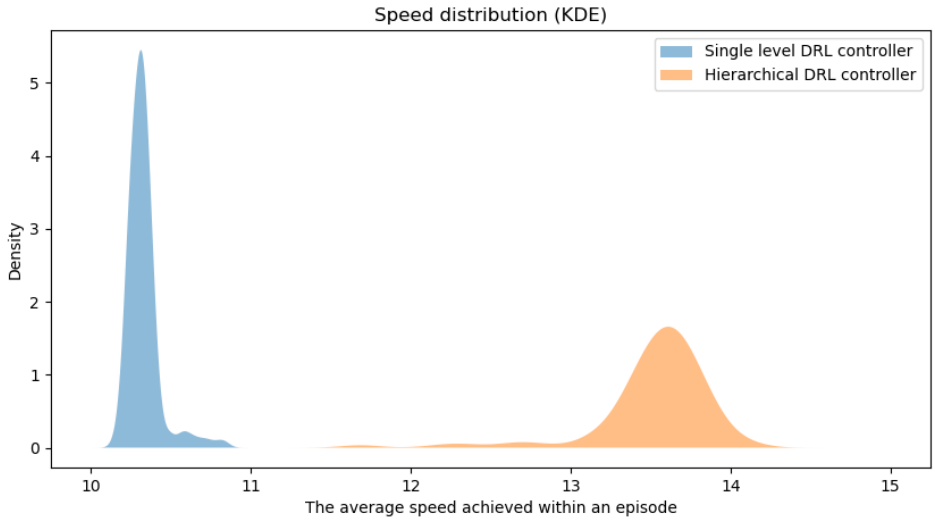


Figure 4.8: Distribution of average speed achieved within an episode

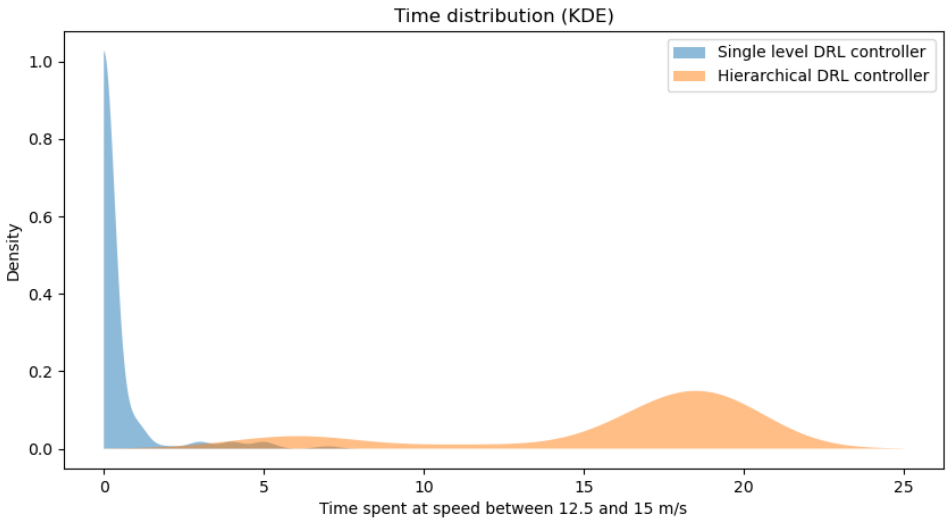


Figure 4.9: Distribution of time spent at ideal speed zone

However, due to the more conservative and simplistic driving strategy learned by the single-level DRL controller, this controller maximizes the speed reward by staying close to the proceeding vehicle, in this case is the trap vehicle, while keeping a safety distance to the front vehicle as shown in Figure 4.10. Since the controller is not able to change lanes because of it's incapability of discovering the speed reward of staying in the ideal speed zone, the single-level DRL controller hardly steers and stays close to it's inital lane center as shown in Figure 4.11. This also results in an accumulated lane centering reward of single-level DRL controller that is higher than the hierarchical DRL controller with in an episode, as shown in Figure 4.12.

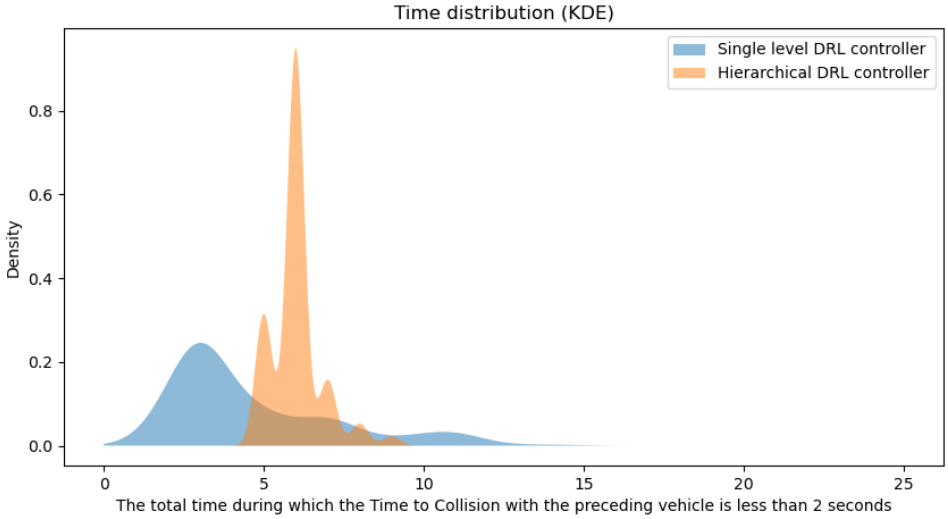


Figure 4.10: Distribution of time spent with TTC under 2 seconds

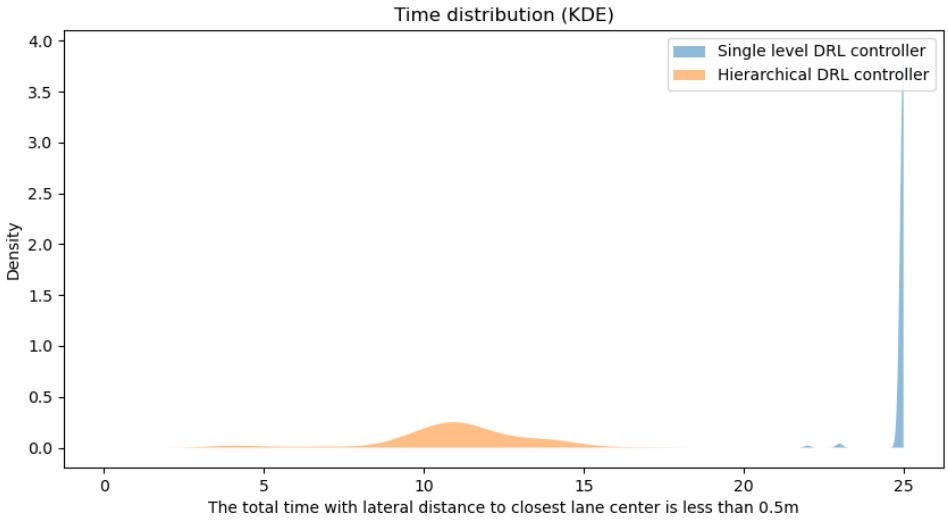


Figure 4.11: Distribution of time spent with a lateral distance to the closest lane center less than 0.5m within an episode

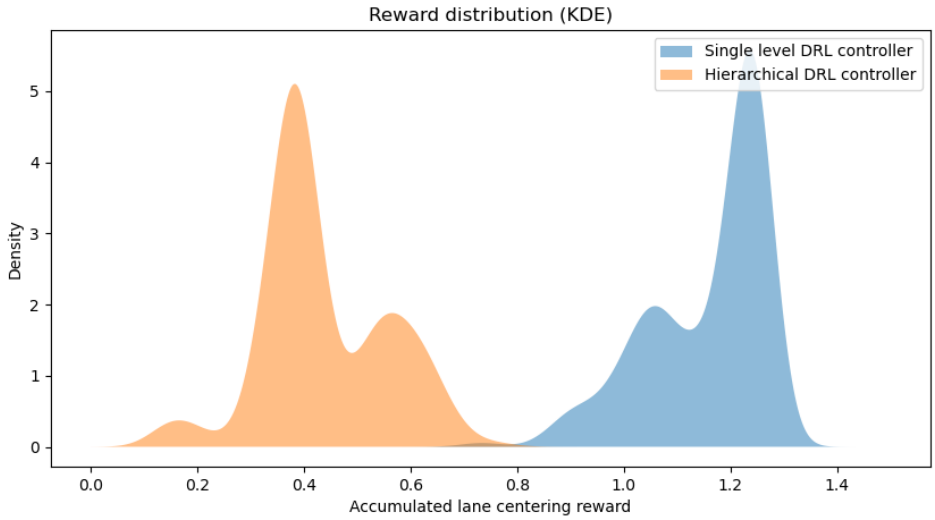


Figure 4.12: Distribution of lane centering reward

Chapter 5

Conclusion

In this study, we have developed a hierarchical controller framework for autonomous highway driving to improve the robustness of handling complex traffic scenarios using DRL. We proposed using a high-level controller for long-term planning and broad exploration, while the low-level controller focuses on detailed vehicle maneuvering. To enhance the controller's effectiveness, we implemented a two-step training process, where each step focuses on training one controller.

We demonstrated the effectiveness of the hierarchical controller using a "Trap" situation, which is commonly seen in many highway scenarios. Based on our experiment, we proved that the hierarchical framework has a superior understanding of the "Trap" and therefore has a better chance to gain long-term rewards during training and testing episodes.

As discussed in Chapter 1, our study primarily focuses on DRL-based autonomous driving strategies for highway scenarios. Real-world traffic and road conditions may be much more complex than those encountered in our experiment [40], including driving in residential areas. At the same time, DRL-based ADS are difficult to achieve a level of driving beyond that of a human [20]. Such driving environments may involve timely avoidance of pedestrians [41], navigating single lane passing, and dealing with winding roads, among other challenges. For example, in urban driving, our autonomous vehicles need to be proficient in complex, crowded traffic environments and familiar with traffic regulations [41]. In comparison, the highway autonomous driving scenario studied in our research is relatively simple in terms of road traffic environment. Designing autonomous driving strategies that can flexibly adapt to various complex traffic environments remains a significant challenge.

The reinforcement learning algorithms used in this experiment are Deep Q-network and Double Deep Q-network, which store learned driving strategies in neural networks. Now many researchers implement other sequential models that focus more on time in ADS. For highway autonomous driving strategies, when a vehicle falls into a non-ideal state, it is often due to a series of actions. We can compare the performance differences between sequential models and neural networks. Future work could focus on utilizing more complex neural networks [42, 43] or incorporating safety constraints to ensure safer driving strategies while still being capable of handling complex traffic scenarios.

Reinforcement learning requires sufficient exploration of the environment to search for the optimal strategy. In our experiment, we employed the annealing algorithm to decrease the probability of selecting random actions over the course of training. However, we can still attempt to use other exploration strategies and conduct an analysis of their effectiveness.

Bibliography

[1] 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.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58443–58469, 2020.

[3] R. Schubert, "Evaluating the utility of driving: Toward automated decision making under uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 354–364, 2011.

[4] P. Varaiya and S. E. Shladover, "Sketch of an ivhs systems architecture," in *Vehicle Navigation and Information Systems Conference, 1991*, vol. 2, pp. 909–922, IEEE, 1991.

[5] D. C. K. Ngai and N. H. C. Yung, "A multiple-goal reinforcement learning method for complex vehicle overtaking maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 509–522, 2011.

[6] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 10, pp. 3884–3897, 2018.

[7] J. Rosenzweig and M. Bartl, "A review and analysis of literature on autonomous driving," *E-Journal Making-of Innovation*, pp. 1–57, 2015.

[8] J. Peng, S. Zhang, Y. Zhou, and Z. Li, "An integrated model for autonomous speed and lane change decision-making based on deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21848–21860, 2022.

[9] 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.

[10] C. Hatipoglu, U. Ozguner, and K. A. Redmill, "Automated lane change controller design," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 1, pp. 13–22, 2003.

[11] R. E. Chandler, R. Herman, and E. W. Montroll, "Traffic dynamics: studies in car following," *Operations research*, vol. 6, no. 2, pp. 165–184, 1958.

[12] P. G. Gipps, "A behavioural car-following model for computer simulation," *Transportation research part B: methodological*, vol. 15, no. 2, pp. 105–111, 1981.

[13] 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.

[14] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[15] T. Han, J. Jing, and Ü. Özgüner, "Driving intention recognition and lane change prediction on the highway," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, pp. 957–962, IEEE, 2019.

[16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[17] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[18] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 4693–4700, IEEE, 2018.

[19] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklethwaite, N. Griffiths, A. Shah, *et al.*, "Urban driving with conditional imitation learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 251–257, IEEE, 2020.

[20] E. Yurtsever, L. Capito, K. Redmill, and U. Ozgune, "Integrating deep reinforcement learning with model-based path planners for automated driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1311–1316, IEEE, 2020.

[21] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254, IEEE, 2019.

[22] J. Chen, Z. Wang, and M. Tomizuka, "Deep hierarchical reinforcement learning for autonomous driving with distinct behaviors," in *2018 IEEE intelligent vehicles symposium (IV)*, pp. 1239–1244, IEEE, 2018.

[23] E. Sonu, Z. Sunberg, and M. J. Kochenderfer, "Exploiting hierarchy for scalable decision making in autonomous driving," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2203–2208, IEEE, 2018.

[24] A. Kurt and Ü. Özgüner, "Hierarchical finite state machines for autonomous mobile systems," *Control Engineering Practice*, vol. 21, no. 2, pp. 184–194, 2013.

[25] K. Rezaee, P. Yadmellat, M. S. Nosrati, E. A. Abolfathi, M. Elmahgiubi, and J. Luo, "Multi-lane cruising using hierarchical planning and reinforcement learning," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 1800–1806, IEEE, 2019.

[26] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1379–1384, IEEE, 2018.

[27] J. Wang, Y. Wang, D. Zhang, Y. Yang, and R. Xiong, "Learning hierarchical behavior and motion planning for autonomous driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2235–2242, IEEE, 2020.

[28] M. Moghadam and G. H. Elkaim, "A hierarchical architecture for sequential decision-making in autonomous driving using deep reinforcement learning," *arXiv preprint arXiv:1906.08464*, 2019.

[29] S. V. Albrecht, C. Brewitt, J. Wilhelm, B. Gyevnar, F. Eiras, M. Dobre, and S. Ramamoorthy, "Interpretable goal-based prediction and planning for autonomous driving," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1043–1049, IEEE, 2021.

[30] T. Shi, P. Wang, X. Cheng, C.-Y. Chan, and D. Huang, "Driving decision and control for automated lane change behavior based on deep reinforcement learning," in *2019 IEEE intelligent transportation systems conference (ITSC)*, pp. 2895–2900, IEEE, 2019.

[31] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, 2016.

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

[33] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 2326–2331, IEEE, 2019.

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

[35] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] A. G. Hado and D. Silver, "Deep reinforcement learning with double q-learning," *arXiv preprint arXiv:1511.06581*, 2015.

[37] 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.

[38] A. G. Barto, "Intrinsic motivation and reinforcement learning," *Intrinsically motivated learning in natural and artificial systems*, pp. 17–47, 2013.

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

[40] B. Weng *et al.*, "Towards guaranteed safety assurance of automated driving systems with scenario sampling: An invariant set perspective," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 638–651, 2021.

[41] M. Koç *et al.*, "Pedestrian emergence estimation and occlusion-aware risk assessment for urban autonomous driving," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021.

[42] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," *Advances in neural information processing systems*, vol. 34, pp. 15084–15097, 2021.

[43] M. Janner, Q. Li, and S. Levine, "Offline reinforcement learning as one big sequence modeling problem," *Advances in neural information processing systems*, vol. 34, pp. 1273–1286, 2021.