A Deep Reinforcement Learning Decision-Making Approach for Adaptive Cruise Control in Autonomous Vehicles

Dany Ghraizi, Reine Talj, Clovis Francis

A Deep Reinforcement Learning Decision-Making Approach for Adaptive Cruise Control in Autonomous Vehicles*

Dany Ghraizi[1], Reine Talj[1] and Clovis Francis[2]

***Abstract*— In the evolving automobile industry, Adaptive Cruise Control (ACC) is key for aiding autonomous traffic navigation. Ideal ACC systems can decelerate to low speeds in stop-and-go traffic, maintain a safe following distance, minimize rear-end collision risks, and lessen the driver's need to continually adjust vehicle's speed to match traffic flow. In this paper, we offer a Deep Reinforcement Learning-based adaptive cruise control (DRL-ACC) system that creates safe, flexible, and responsive car-following policies agents. Instead of using discrete incremental and decremental values or a continuous action space, we suggest constructing a discrete high-level action space to accelerate, decelerate, and hold the current speed. We also provide a comprehensive, easy-to-interpret multi-objective reward function that reflects safe, responsive, and rational traffic behavior. This strategy, trained on a single steady-state flow car-following scenario, promotes steadiness, responsiveness, and shows better generalization to diverse car-following scenarios. Results are also compared to the conventional Intelligent Driver Model (IDM). We further explore the model's potential to avoid rear-end collisions and facilitate future integration of lane-change maneuvers, which will increase its effectiveness in emergency situations.**

I. INTRODUCTION

As the research towards fully autonomous vehicles is still in progress [1], users now have easy access to lesser degrees of vehicle automation with a variety of driver-assist technologies called Advanced Driver Assistance Systems (ADAS) [2], these technologies include (adaptive) cruise control [3], lane-keeping assistance [4], automated emergency braking [5], and lane departure warning, among others.

Adaptive Cruise Control (ACC) systems have become an essential component in the development of intelligent transportation systems, they contribute to safety as they help maintain a safe following distance to reduce the risk of rear-end collisions. They also reduce the need for the driver to constantly adjust the vehicle's speed in response to changing traffic conditions leading to almost full longitudinal and lateral accelerations thus enhancing passenger's comfort. Some more advanced ACC systems can even stop the vehicle completely and then resume moving, which can be very helpful in congested areas [6]. In contrast, a poor performing ACC system will lead to congestion and traffic oscillation, relying on Driver Intervention thus wasting commuter's time and increasing energy consumption and pollution while defeating the purpose of having an automated driving assistance feature and increasing the driver's workload and stress levels.

*The work was realized within an International Research Project: Approches de Diagnotic et de cONtrôle Intelligent des Systèmes (IRP ADONIS), funded by CNRS Université de technologie de Compiègne (UTC) and Université libanaise (UL).

[1]Sorbonne université, Université de technologie de Compiègne, CNRS, Heudiasyc UMR 7253, CS 60 319, 60 203 Compiègne, France name.surname@hds.utc.fr

[2]Arts et Métiers Paris Tech, 1, Rue Saint Dominique, 51000 Châlons en Champagne, France clovis.francis@ensam.eu

Elementary classical control theory models such as P, PI, or PID controllers can be employed [7], [8]. However, adjusting the controller gain to the optimal setting can pose challenges for more complex systems. More advanced approaches such as Model Predictive Control (MPC) have been used on a dynamic model of the system being controlled to predict future behavior and optimize a control sequence over a finite time horizon [9]. Alternatively, the Krauss model [10] or the Intelligent Driver Model (IDM) [11], both model-based strategies, consider the desired time headway, desired speed, and speed difference between the ego vehicle (the vehicle under control) and the leading vehicle to calculate the appropriate acceleration or deceleration. Constraints on these approaches arise from the ability to handle uncertainties, and the size of the prediction horizon.

Leveraging Deep Reinforcement Learning (DRL), we can now optimize autonomous vehicles' driving behaviors, significantly enhancing safety, efficiency, and performance in a way that was not feasible before [12]. The application of Deep Reinforcement Learning (DRL) to ACC has recently gained significant attention due to its ability to solve the issues mentioned while also achieving better performance by handling high-dimensional state spaces, discrete and continuous control actions, and intricate, dynamic environments [13]. When there are no modeling mistakes and the testing inputs fall within the training data range, the authors of [14] show that the DRL solution is equivalent to MPC with a long enough prediction horizon. Additionally, they draw attention to DRL's shortcomings with regard to machine learning generalization and its performance when there are modeling mistakes. In [15] they generate car-following policies that are safe, human-like, and comfortable. The methodology differs from current approaches by defining the action space of the DRL agent using discrete incremental/decremental actions instead of continuous ones, reflecting how human drivers adjust throttle and brake pedal levels and also include explicit actions for holding and coasting, which are typically excluded in ACC systems. The reward terms are also completely derived from the real-world dataset collected from a human driver. ACC 4S [16] is a DRL approach imposing state-specific safe sets as output constraints on the policy. The authors sought to prevent rear-end collisions with the vehicle in front where the safe sets were derived from the Responsibility-Sensitive Safety model [17] and regulatory standards, which provided an upper bound for the demanded acceleration. On the other hand, combining Deep Deterministic Policy Gradient (DDPG) [18] and Cooperative Adaptive Cruise Control (CACC) [19], the authors in [20] model the car-following process as a Markov decision process (MDP) to calculate CACC and DDPG concurrently at each frame. The highest reward determines which of the CACC and DDPG actions is better. A rule to guarantee that the acceleration change rate stays below a desired value is

also included in the approach in a similar way. However, integrating multiple control strategies introduces additional complexity, making the system more sensitive to modeling errors or inaccuracies.

In this study we describe the design and implementation of our DRL-based ACC system, providing an overview of the perception, decision-making, path planning and control modules. We also detail the observation-space, the high-level action-space, the architecture of the DRL agent and the reward function. The efficacy of the proposed system is evaluated through extensive simulations under varying traffic scenarios. The main contributions of this study are:

- Providing a modular and distinctive AI-based ACC system with good generalization capabilities which adaptively responds to dynamically changing traffic situations without the need for large datasets or the challenging process of fine-tuning. This proposed strategy alleviates the drawbacks of static model based techniques.
- Providing a high-level discrete-action model that aims to build the velocity profile (low-level) within a trajectory planner for the controller to follow through the accelerator and brake pedals paired with a holding action, which is a simpler action space for the neural network as opposed to the typical discretization of acceleration data into incremented/decremented values or the usage of a continuous action space.
- Placing the ACC behavior within the framework of an MDP structure with a complete and straightforward reward function, drawing on prior transportation research. By employing a deep-Q network, vehicles can respond safely and more effectively in complex and continually evolving traffic conditions.

The rest of the paper is organized as follows: Section II describes the proposed DRL-based ACC approach, including the DRL framework, the ACC system's design, and the reward function. Section III describes the methodology adopted in this study, including the training and its environment. Section IV presents the simulation results, demonstrating the effectiveness of the proposed system in optimizing collision avoidance, car following behavior, and speed modulation. Finally, Section V concludes the paper and discusses potential avenues for future research.

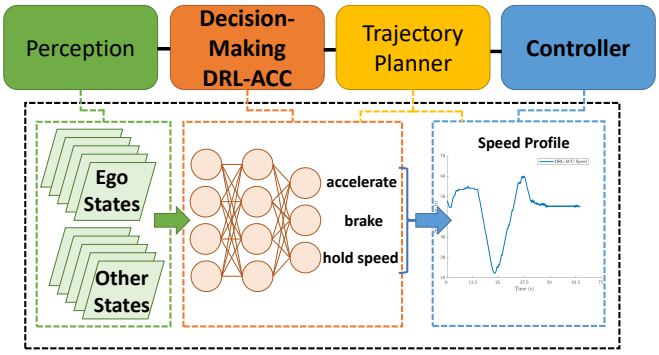

Fig. 1: Decision-Making DRL-ACC Architecture

II. Proposed DRL-Based ACC Architecture

Fig. 1 represents our DRL-based ACC Decision making architecture which is based on our previous work on Trajectory Planning [21] presented in Fig. 2.

A. Perception

The perception block is beyond the scope of this paper. In this study, the output of the perception module is considered as a representation of an occupancy grid, as explained in [21]. In summary, a global occupancy grid is generated from a global map. The local occupancy grid is then derived from the global occupancy grid based on the vehicle's position and orientation. The local occupancy grid consists of cells measuring 400 * 400, with each cell representing an area of 25 * 25 cm. These dimensions are chosen to match the perception system's horizon. To enhance collision checking in terms of accuracy, efficiency, and time consumption, the local occupancy grid is then converted into a clearance map [22].

B. Observation Space

The state of the MDP is defined by a tuple $\langle S_{ego}, S_i \rangle$ in the Frenet Frame [23] where the first vector S_{ego} is the ego vehicle states and S_i is the state vector of the 4 nearest surrounding vehicles where i indicates the i^{th} vehicle. $S_{ego} = (v_{ego}, t_{headway}, \psi_{ego}, d_{centerlane}, lane_{ego})$ consists of the ego vehicle's speed v_{ego}, the time headway to the preceding vehicle $t_{headway}$, the ego vehicle orientation ψ_{ego}, the distance to lane center $d_{centerlane}$, and the lane occupancy $lane_{ego}$. $S_{others} = (v_i, long_i, lat_i, \psi_i, lane_i)$ are the respective other vehicles' states which are the relative speed with respect to the ego vehicle v_i, relative longitudinal distance $long_i$, relative lateral distance lat_i, relative orientation ψ_i, and lane occupancy $lane_i$. If the ego vehicle is faster than the other vehicles then v_i would be negative, and vice versa if it is slower. Similarly, if the other vehicle is behind the ego vehicle then $long_i$ would be negative, and vice versa if it is in front of it. All the states that can be normalized are normalized before being passed into the network according to the equation below. For example the urban speed limits used are defined by the traffic laws in France [24]. Some of the observation states are shown in Fig. 3.

$$X_{Normalized} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

where X refers to the state to be normalized.

C. Action Space

The action space (a_1, a_2, a_3) consists of accelerate a_1, brake a_2, and a holding action that maintains the current speed a_3, where the vehicle follows a predefined velocity profile starting from a desired velocity profile of the base frame and is calculated for each point of the candidate path, detailed in [21]. This takes into consideration the speed limit ($V_{x_{\text{limit}}}$) [24], and is imposed by the velocity limits of the base frame and curvature of the road, and the lateral acceleration to improve vehicle stability and passenger comfort criteria by keeping the lateral acceleration under a maximum threshold $|a_{y_{\max}}| = 4m/s^2$ as stated in [25].

D. Vehicle Dynamic Model and Control

Our algorithm incorporates a comprehensive longitudinal and lateral vehicle model developed using the multi-body formalism described in [26]. The model takes wheel driving/braking torque (τ_w) and steering angle (δ) as inputs.

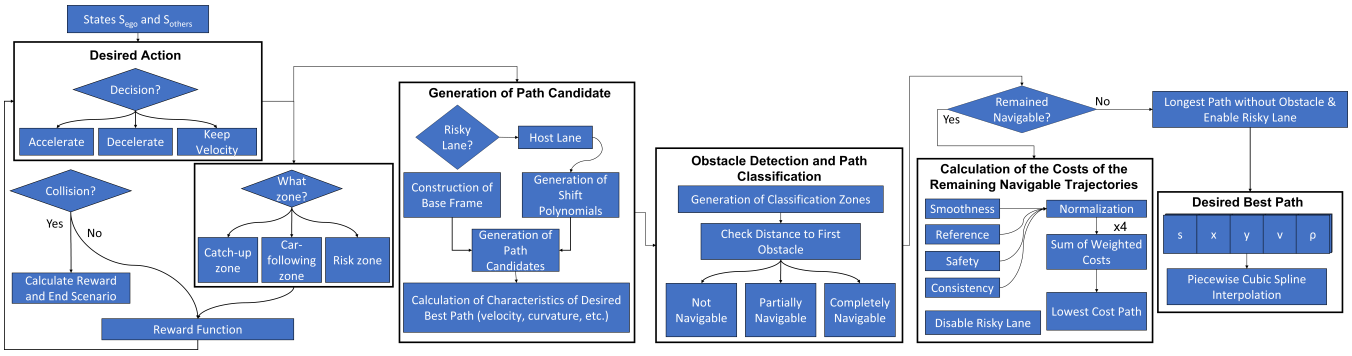

Fig. 2: Decision-Making Trajectory Planning Diagram inspired from [21]

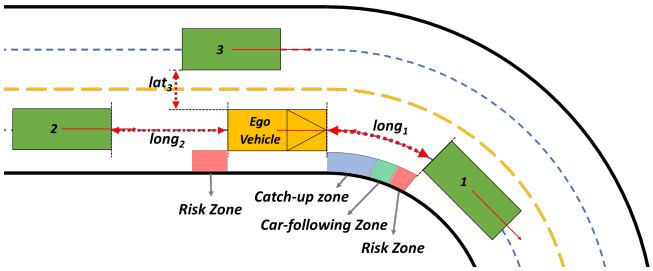

Fig. 3: Some observation states of the system and the zone occupancy of the ego vehicle. Ego vehicle in yellow and other vehicles in green.

Utilizing the Dugoff model for estimating tire forces and performing model matrix calculations, it generates the following outputs: longitudinal ($\ddot{x}$) and lateral ($\ddot{y}$) vehicle accelerations, yaw rate ($\dot{\psi}$), and wheel angular velocities (w_{ij}).

For control purposes, we have opted for a second-order sliding mode based on the super-twisting algorithm. This choice ensures robust stability while minimizing chattering, a common issue in sliding mode control. The complete model has been validated using the SCANeR studio simulator across various driving conditions. In summary, our algorithm incorporates a thorough vehicle model and employs advanced control techniques to optimize performance. Through extensive validation, we have ensured that the model behaves accurately and reliably in different scenarios.

E. Reward Function

If rewards are sparse, it's difficult for the agent to figure out which actions are beneficial and which are not. Therefore the reward function is a rich multi-objective function (consisting of safety, collision avoidance, and car-following behaviors) that utilizes the reward shaping technique, based on logical driving behavior, where additional rewards are provided to guide the agent towards the desired behavior. It consists of a safety reward R_{safety} and a speed reward R_{speed}. The safety reward consists of 3 sub-rewards where each of them is weighted w_i along with the speed reward in the main reward. The weights were chosen through testing in order to balance the bias of the agent. The Reward function is as follows:

$$\begin{cases} R_{safety} & = w_1 \cdot R_{front} + w_2 \cdot R_{back} + w_3 \cdot R_{collision} \\ R & = R_{safety} + w_4 \cdot R_{speed} \end{cases} \tag{2}$$

where w_i = [1, 0.5, 1, 0.35].

- **Safety** R_{safety}: The safety reward is split into 3 rewards that govern the collisions $R_{collision}$, and the front R_{front} and back R_{back} gap of the ego vehicle with other vehicles. According to Fig. 3 the vehicle could be in one of 3 zones at each time: 1) car-following zone where the ego vehicle is following the front vehicle at a safe distance with a fairly similar speed, 2) risk zone where the ego vehicle is too close to the front (or back) vehicle such that the relative distance and time headway are less than the desired values, and 3) catch-up zone where the ego vehicle is too far away from the front vehicle such that the relative distance and time headway are more than the desired values. The agent is rewarded in the car-following zone if it is within $+0.5s$ of the Desired Time Headway DTH which is set at 2 seconds or within +3 meters of the Distance Error DE and penalized otherwise, when it is in the other zones, according to the Time Headway TH and the Distance Error. Additionally, it is rewarded if it maintains the same speed as the other vehicle when in the car-following zone but penalized otherwise, in the risk zone it is rewarded if it slows down and penalized otherwise, and it is rewarded in the catch-up zone if it accelerates to catch up when it lags behind, but penalized if it is slower. The calculations are achieved according to the equations below:

$$\begin{cases} TH & = \frac{D}{V_{ego}} \ , \ DTH = 2 \\ DE & = D - D_{s0} + \frac{V_{ego}^2}{2a_{dec-max}} \end{cases} \tag{3}$$

Where $a_{dec-max}$ is the maximum deceleration, D_{s0} is a minimum safety gap and D is the distance between the two vehicles, V_{ego} is the ego vehicle velocity. Together, DE is the error with the safe-stop-distance.
In the risk zone, we apply the penalty P_1, and in the catch-up zone we apply the penalty P_2. The equations are shown below:

$$P_1 = -(k_1 \cdot (TH - DTH)^2 + k_2 \cdot DE^2) \tag{4}$$

$$P_2 = RB - (|D_{front}| \times k) \tag{5}$$

where $k_1 = 0.5$ and $k_2 = 0.5$ are weighting factors, $k = 1$ is the rate of the negative reward, $RB = -0.5$ is the Reward Baseline, D_{front} is the relative distance to the front vehicle.
Based on the relative speed of the ego vehicle, in the

car-following zone we apply the reward-penalty SR_1, in the risk zone we apply the reward-penalty SR_2, and in the catch-up zone we apply the reward-penalty SR_3.

$$SR_1 = \begin{cases} \frac{V_i-(-0.015)}{0.015-(-0.015)} & \text{if } -0.015 \leq V_i \leq 0.015 \\ -abs(V_i) \cdot k & \text{otherwise} \end{cases} \tag{6}$$

$$SR_2 = \begin{cases} \frac{V_i}{0.05} & \text{if } V_i \geq 0.05 \\ -abs(V_i) \cdot k & \text{otherwise} \end{cases} \tag{7}$$

$$SR_3 = \begin{cases} \frac{V_i-(-0.1)}{-0.05-(-0.1)} & \text{if } -0.1 \leq V_i \leq -0.05 \\ -abs(V_i) \cdot k & \text{otherwise} \end{cases} \tag{8}$$

where V_i is the normalized relative speed, and $k = 2$ is the rate of the negative reward.
For $\alpha = DTH + 0.5$, and the above equations motivating the vehicle to either speed up, slow down, or hold its speed, the front gap reward would be:

$$R_{front} = \begin{cases} SR_1 + 1 & \text{if } (DTH < TH \leq \alpha) \\ & \vee (0 \leq DE \leq 3) \\ SR_2 + P_1 & \text{if } TH \leq DTH \vee DE < 0 \\ SR_3 + P_2 & \text{otherwise} \end{cases} \tag{9}$$

For the vehicles behind the ego vehicle, there is only a penalty for when they are in the risk zone:

$$R_{back} = \begin{cases} SR_3 + P_1 & \text{if } (TH \leq DTH) \vee (DE < 0) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

In case of collisions the vehicle is also penalized:

$$R_{collision} = \begin{cases} P_{collision} & \text{if } collision \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

- **Speed** R_{speed}: The speed reward is based on the speed limits of 57.6 km/h and 36 km/h since we consider a common range in an urban environment [24]. This however does not limit the agent's decision to go beyond these limits as can be seen in the testing results in section V.

$$R_{speed} = \begin{cases} \frac{SP-0.5}{0.8-0.5} & \text{if } SP \leq 0.8 \\ -abs(SP) \cdot k & \text{otherwise} \end{cases} \tag{12}$$

where SP is the normalized speed of the ego vehicle.

F. Network Architecture

In Adaptive Cruise Control (ACC), the problem of learning optimal control policies is formulated as a Markov Decision Process (MDP) within Deep Reinforcement Learning (DRL) [27]. ACC involves the DRL agent dynamically adjusting the vehicle's acceleration or deceleration actions a based on the observed state s, which includes factors like the ego vehicle's speed, distance and relative speed with respect to the leading vehicle. The agent's objective is to maintain a safe and comfortable distance from the leading vehicle while adapting to traffic conditions. Through interactions with the environment, the agent receives rewards $R(s, a, s')$ during transition from state s to state s' by taking the action a. This reward function is evaluated based on the achieved performance in terms of safety, comfort, and efficiency (when energy consumption is considered) in the form of feedback s'. These units form the MDP tuple $\langle s, a, T, R \rangle$ where the transition function T(s, a, s') defines the probability of transitioning from state s to state s' upon taking action a.

In our approach, we employ a Double Deep Q-Network (DDQN) [28], [29] agent. A decision motivated by several compelling factors, a DQN can adeptly manage environments characterized by high-dimensional state spaces, effectively process and learn from complex inputs, and provide inherent generalization capabilities. Moreover, the stability and data efficiency offered by DQN were additional factors influencing our choice. Techniques such as Experience Replay [30] and Target Networks, integral to the DQN architecture, where adopted to significantly enhance the learning process.

The agent's architecture consists of a 2D convolutional layer, configured with a 3x3 kernel and 128 filters, a stride of 1 and padding set to 'same'. Subsequent to the convolution operation, a Rectified Linear Unit (ReLU) activation function is utilized to introduce non-linearity, supporting the extraction of complex features from the given inputs. Subsequent operations involve a sequence of three fully connected layers. The first layer houses 128 neurons, followed by a second layer with 64 neurons where both are supplemented by a ReLU activation function. The final layer comprises a number of neurons equivalent to the number of elements in the action space. The input of the architecture is the tuple $\langle S_{ego}, S_i \rangle$ and the output is one of the actions (a_1, a_2, a_3) which consists of accelerate a_1, brake a_2, and a holding action that maintains the current speed a_3. The complete network is designed to estimate Q-values for the agent's actions in response to its observations. The hyperparameters of the DRL-ACC model are summarized in Table I.

Hyperparameter	**Value**
Use DoubleDQN	true
Minibatch Size	32
Look-Ahead Period	64 Steps
Target Update Frequency	1 Step
Experience Buffer Length	50,000
Optimizer	ADAM
Learning Rate	0.01
Discount Factor	0.99

TABLE I: Hyperparameters of the DQN Agent

III. METHODOLOGIES

For training the DRL-Based ACC, the DRL inputs are the observation space and the output is one of the actions defined. The scenario used for training is car-following where only 1 vehicle is placed in front of the ego vehicle on the same lane where it accelerates and then maintains its velocity throughout the rest of the episode. The road chosen is a straight 2-lane road that extends up to $1000m$. The initial conditions for the position and speed are randomly generated within the intervals $[1, 100]m$, and $[36, 54]km/h$ respectively. Training is done on 5 seeds where the seed is randomly chosen. The training initial conditions are considered to represent common normal car-following scenarios

with the starting distance headway between the vehicles of each scenario varying between 15 and 40 meters. The training episode length is $1000steps$ where the episode could last between 30 and 90 seconds allowing the vehicle to reach the end of the road successfully, however the episode terminates if the leading vehicle goes out of the range of perception for more than $50steps$. This is because we allow the agent during training to correct its situation by catching up to the leading vehicle as defined within the reward function. We trained the policy for $800episodes$ of a total of 380 thousand steps, with a timestep of $t = 0.125s$, with an observed convergence of the discounted long-term reward, and the moving average reward $(windowsize = 50)$ in Fig. 5. Based on the ranges presented in Table II, the other vehicles are controlled using the Intelligent Driver Model (IDM) [31] with the following parameters: a maximum acceleration of $2m/s^2$, a maximum comfort deceleration of $-3m/s^2$, a randomly generated desired velocity in the range $[36, 54]km/h$, an acceleration exponent of 4, and desired distance and time headway of $3m$ and $1.8s$ respectively.

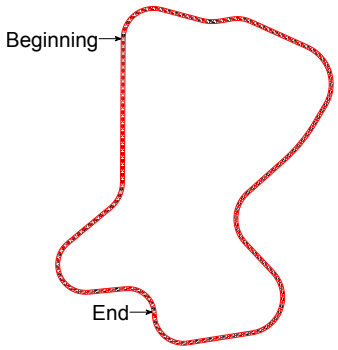

Fig. 4: Testing Map

TABLE II: IDM Parameters [31]

Parameter	Range	Normal	Aggressive
(δ) acceleration exponent	{2, 4}	4	4
(s_{min}) min. desired distance gap	4.0-1.0 m	2.0	1.0
(v^*) desired velocity	54-140 km/h	57.6	64.8
(t_{gap}) desired time gap	1.8-1.0 s	1.5	1.0
(a_{max}) max. acceleration	1.0-2.0 m/s^2	1.4	2.0
(b_{comf}) comfort deceleration	1.0-3.0 m/s^2	2.0	3.0

IV. Simulation Results

This section details the testing results of the DRL-Based ACC where we can see that the reward function designed was able to guide the agent towards the desired behavior in scenarios which it was not trained in. The scenarios used for testing are: 1) car-following a leading vehicle in front, 2) car-following with both a leading vehicle and a following vehicle in front and behind the ego vehicle respectively, 3) similar to the second scenario, but the road is no longer straight as it is a map of a rather realistic situation obtained from the SCANeR studio simulator, which has both curved and straight portions, as can be seen in Fig. 4, and finally 4) cut-in and cut-out (4.1 and 4.2 respectively) of the leading vehicle situated between the ego vehicle and another front vehicle. As an added perspective to the behavior of the agent, we test

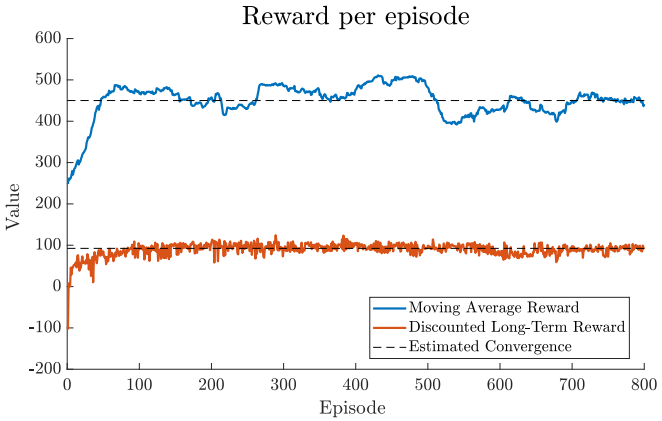

Fig. 5: Training Results

5) rear-collision avoidance with the vehicle behind the ego vehicle that does not slow down when it is close to the ego vehicle. We also added two additional scenarios, outside of the training data speed, to the ego vehicle testing: 6) similar to (2) where the leading vehicle accelerates aggressively, and 7) an emergency case with sudden braking to near-zero speed $(3.6km/h)$ then maintaining this speed. These scenarios follow the same generation process as in the training, however they were tested on 10 different seeds. The testing conditions mentioned enable us to test the performance, adaptability and response-time of our model trained only with one scenario of car following, in several scenarios for which it is not trained: car-following at different speed levels including near-zero speeds, collision avoidance in both front-end and rear-end cases, speed modulation (accelerate, decelerate and speed holding action), and cut-in and cut-out cases. Several performance indicators are considered and recalled in Table III: The average velocity of the ego vehicle and other vehicles, the average time headway and distance error (TH and DE) and number of collisions.

The agent successfully maintains a time headway of at least 2 seconds and a positive normalized distance error in a car-following scenario with a relatively similar speed to the vehicle it is following in scenarios (1 and 2), and even with changing road curvature in scenario (3). In the case of cut-in and cut-out (4.1 and 4.2), the agent is able to adjust to the changes in the environment while still maintaining acceptable time headway and distance error values. This is present with the lower average speed in case of cut-ins and higher average speed in case of cut-outs. However, the agent seems to struggle with cut-outs where the time headway dips under 2 seconds, but it was still able to maintain an acceptable distance error value. On the other hand, it is motivated to accelerate when there is no vehicle in front as can be seen in the average speed of scenario (5). We note that the time headway and distance error in this case are only recorded when the vehicle behind the ego vehicle is within the perception range which means the ego vehicle accelerated away from the previous vehicle for the majority of the episode. In (6) the agent showed the ability to adapt to a speed higher (Avg. V_{ego} = 61.5459km/h) than the training data speed. Meanwhile in (7) the agent showed its ability to slow down significantly to near-zero speeds, lower than the training data speed, and follow the leading vehicle accordingly presented in Fig. 9. In all 7 scenarios the agent

TABLE III: Testing Results

Model	Scenario	Avg. $\Delta V_{ego-front}$ (km/h)	Avg. TH_{front} (s)	Avg. DE_{front} (m)	Avg. $\Delta V_{ego-back}$ (km/h)	Avg. TH_{back} (s)	Avg. DE_{back} (m)	Collisions
DRL-ACC	(1)	1.5617	**2.3406**	**0.5805**	-	-	-	**0**
	(2)	1.066	**2.1874**	**0.5442**	1.0955	2.4295	0.5913	**0**
	(3)	2.0474	**1.9139**	**0.4874**	1.903	2.1570	0.3271	**0**
	(4.1)	**-2.4357**	**2.0836**	**0.4443**	-2.0962	2.6422	0.5381	**0**
	(4.2)	**3.7764**	**1.8268**	**0.4581**	2.8047	2.7694	0.7069	**0**
	(5)	-	-	-	**9.958**	2.5690	0.7549	**0**
	(6)	**1.4184**	2.1853	0.7505	-	-	-	**0**
	(7)	**2.1981**	1.9827	0.2480	-	-	-	**0**
IDM$_{Nor}$	(1)	1.741	**1.4602**	**0.3723**	-	-	-	**0**
	(2)	2.0466	**2.0053**	**0.4951**	4.1764	2.7669	0.6641	**0**
	*(3)	-4.036	**3.2410**	**0.6850**	4.8571	2.3698	0.5276	**0**
	*(4.1)	**0.2934**	**2.3758**	**0.5269**	4.8708	2.7999	0.5722	**0**
	*(4.2)	**-1.8994**	**3.1181**	**0.7015**	0.3369	3.1274	0.5984	**0**
	(5)	-	-	-	**9.126**	2.4731	0.7054	**1**
IDM$_{Agg}$	(1)	1.5398	**2.4396**	**0.5981**	-	-	-	**0**
	(2)	-2.8397	**2.5596**	**0.5803**	0.2005	2.8722	0.6321	**0**
	(3)	-1.5775	**2.4674**	**0.5943**	5.4626	2.6636	0.6134	**0**
	*(4.1)	**0.5508**	**2.3902**	**0.5796**	2.97	2.8298	0.6255	**0**
	*(4.2)	**4.446**	**2.3846**	**0.5890**	2.6759	3.0272	0.7245	**0**
	(5)	-	-	-	**18.9082**	2.6301	0.7764	**0**

The model does not keep the leading vehicle in the perception range at all time-steps.

has not experienced any front-end or rear-end collisions. We also present the results of the IDM in both Agressive IDM_{agg} and Normal IDM_{nor} driving style, where the IDM driving style parameters are according to the parameters in [31] summarized alongside our chosen values in Table II. IDM_{nor} appears to be less cautious in (1) and (2). It also loses sight of the leading vehicle in some cases of (3), (4.1), and (4.2) where it shows a slow response to changes in the environment and also manages to have 1 collision overall. In (5) it performs similarly to our agent. IDM_{agg} performs similarly to our agent in (1), (2), and (3). However, it also suffers from the same issues of IDM_{nor} in (4.1) and (4.2), albeit less significantly. In (5) it presents high acceleration, away from the vehicle behind it, as there is no leading vehicle. As for (6) and (7), the IDM in (6) shows similar performance, meanwhile in (7) it is noticed during testing that the IDM tends to reach a full-stop instead of maintaining the necessary very low speed which is not the desired outcome from this scenario.

For further discussion, we will analyze the behavior of the vehicle in the car-following, cut-out, and cut-in scenarios against the IDM in both Agressive IDM_{agg} and Normal IDM_{nor} driving styles.

- **Car-Following**. In the car-following scenario represented in Fig. 6, the leading vehicle brakes suddenly at $t = 12.5s$ and maintains its speed before it accelerates at $t = 25s$ and decelerates again at $t = 37.5s$ to reach a steady state. The DRL agent is able to follow the other vehicle and adjust its speed accordingly to a low $10km/h$, while maintaining acceptable time headway and distance errors. The actions of acceleration, braking and holding of the agent are also better at responding to the changes in the speed of the leading vehicle without a huge difference in time headway and distance error. Our DRL-Based ACC model outperforms the IDM models in car-following as the IDM is not even able to catch-up to the leading vehicle where the gap between them exceeds the perception range at $t = 6s$ and $t = 29.625s$ which triggers the termination condition for the episode if in training. A large steady state speed error is also observed with both IDM models with respect to the trained agent.
- **Cut-in**. In the cut-in scenario represented in Fig. 7, the cut-in occurs at $t = 6.25s$. Our model is able to follow the new leading vehicle's speed and to adjust the time headway and distance error faster and more appropriately than both the IDM models, even if the time headway dips into the low value of $0.5s$. However, it maintains a time headway of less than 2 seconds which is below the desired time headway. We notice that the speed of IDM_{nor} oscillates, this is because it struggles to keep the vehicle within the range of perception as can be seen in the time headway and distance error figures. We also notice that IDM_{agg} lags behind the leading vehicle but then overshoots in acceleration to catch up to it.
- **Cut-out**. In the cut-out scenario represented in Fig. 8, the cut-out occurs at $t = 6.25s$. Our model is able to quickly catch up to the leading vehicle and to maintain an acceptable time headway and distance error values. Both IDM models perform well in this scenario, however both models decelerate when cut-out occurs instead of accelerating as our agent does, and they both show long settling time to the reference speed, meanwhile the IDM_{nor} loses sight of the leading vehicle for $10s$ before catching up to it.

Overall the performance of the DRL-Based ACC is better than both IDM models, presenting fast response time, better speed following, comfortable driving, and the ability to handle uncertainties in the car-following scenarios. This is especially true since it was only trained on following 1 vehicle that accelerates to a constant speed which is

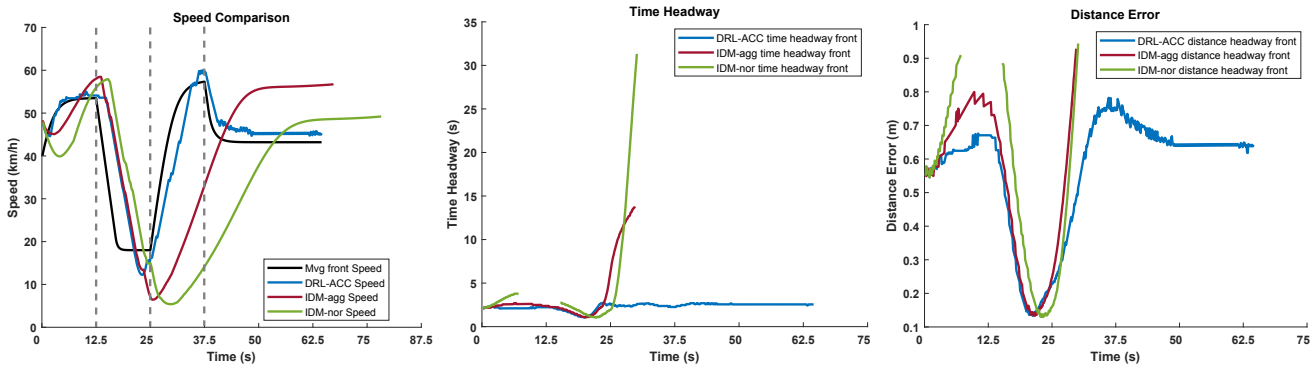

Fig. 6: Car Following Comparison between DRL-Based ACC and IDM

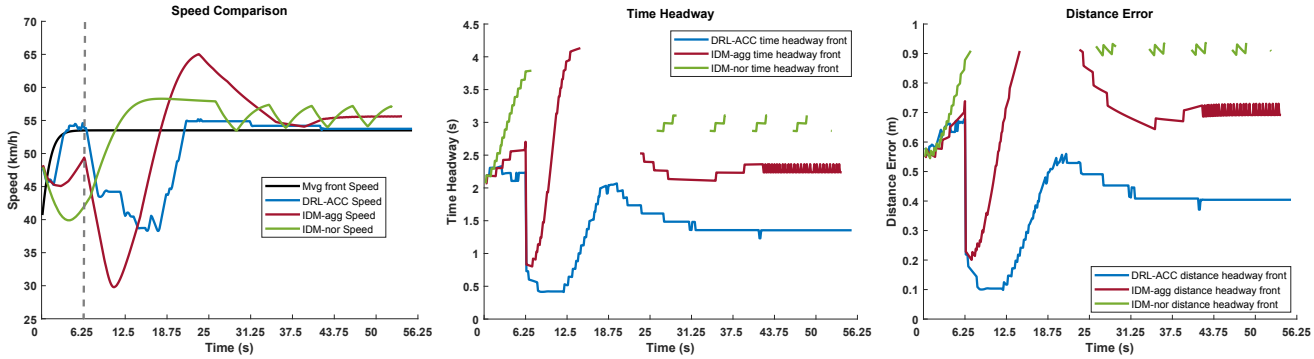

Fig. 7: Cut-in Comparison between DRL-Based ACC and IDM

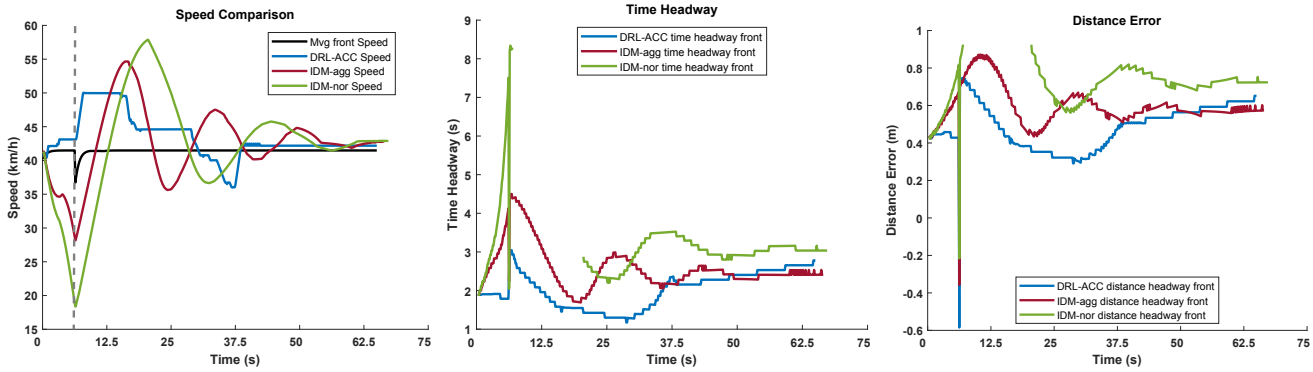

Fig. 8: Cut-out Comparison between DRL-Based ACC and IDM

maintained throughout the episode.

We believe that our model is therefore able to reduce traffic congestion through its response to these changes thus allowing a smoother traffic flow.

V. Conclusion

With respect to the training situation, we have built and validated a car-following policy based on a discrete action space of a DRL-ACC model with a logical and multi-objective reward function. In addition to outperforming the generally used comparable algorithm in terms of safety, comfort, and response time, our thorough evaluations, carried out in simulations encompassing numerous driving scenarios while using car-following ACC also indicate expected logical behavior in the evaluated circumstances. The suggested DRL-ACC policy generation and reward function, in our opinion, is a step in the right direction toward accomplishing tactical decision-making in the car-following environment and resulting in behaviors that are similar to those of a human being when accelerating, braking, and speed holding.

For future work, as it has demonstrated predictable reaction behavior to the rear-end vehicle's state in the environment, it may be scaled to incorporate lane-change behavior utilizing this observation space. One such example is the tested emergency lane change maneuver, shown in Fig. 10, where the vehicle showed the ability to avoid the sudden stop of a leading vehicle without having to decelerate significantly, or the speeding of the rear vehicle which lead to front-end and rear-end collisions respectively. Expanding the behavior to incorporate decision-making calls at crossroads and roundabouts would be another part. This would necessitate updating the reward function and doing

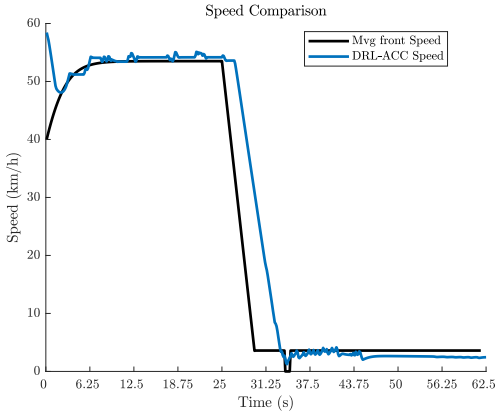

Fig. 9: Scenario (7) Near-zero speed car following

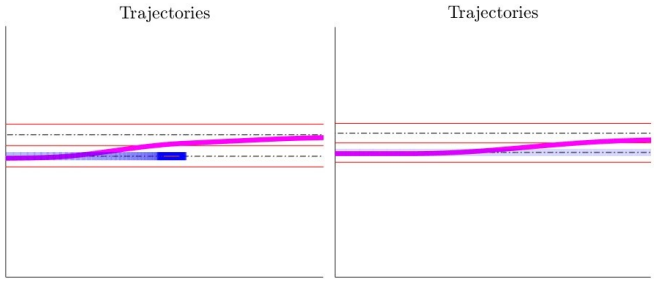

Fig. 10: Front (left) and Rear-End (right) Collision Avoidance by Emergency Lane Change of DRL-ACC

further testing as part of the autonomous vehicle's whole modular pipeline, which includes modules of perception, planning, decision-making, and control. Then, taking into account the fact that driving is safety-critical, we might move forward with real-world deployment. For this, we would need a safety monitoring algorithm that could act when necessary. By doing so, our DRL agent might be incorporated into human driving as a high level producer of longitudinal and lateral decisions.

REFERENCES

[1] O.-R. A. D. O. Committee, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, apr 2021.

[2] Y. He, B. Ciuffo, Q. Zhou, M. Makridis, K. Mattas, J. Li, Z. Li, F. Yan, and H. Xu, "Adaptive cruise control strategies implemented on experimental vehicles: A review," *IFAC-PapersOnLine*, vol. 52, no. 5, pp. 21–27, 2019, 9th IFAC Symposium on Advances in Automotive Control AAC 2019.

[3] L. Yu and R. Wang, "Researches on adaptive cruise control system: A state of the art review," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 2-3, pp. 211–240, 2022.

[4] S. Wei, P. E. Pfeffer, and J. Edelmann, "State of the art: Ongoing research in assessment methods for lane keeping assistance systems," *IEEE Transactions on Intelligent Vehicles*, pp. 1–28, 2023.

[5] S. Mosharafian and J. M. Velni, "Cooperative adaptive cruise control in a mixed-autonomy traffic system: A hybrid stochastic predictive approach incorporating lane change," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 136–148, 2023.

[6] A. Mishra, J. Purohit, M. Nizam, and S. K. Gawre, "Recent advancement in autonomous vehicle and driver assistance systems," in *2023 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2023, pp. 1–5.

[7] S. Chamraz and R. Balogh, "Two approaches to the adaptive cruise control (acc) design," in *2018 Cybernetics Informatics (KI)*, 2018, pp. 1–6.

[8] M. K. Rout, D. Sain, S. K. Swain, and S. K. Mishra, "Pid controller design for cruise control system using genetic algorithm," in *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016, pp. 4170–4174.

[9] B. Sakhdari and N. L. Azad, "Adaptive tube-based nonlinear mpc for economic autonomous cruise control of plug-in hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 11 390–11 401, 2018.

[10] A. Validi, T. Ludwig, A. Hussein, and C. Olaverri-Monreal, "Examining the impact on road safety of different penetration rates of vehicle-to-vehicle communication and adaptive cruise control," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, pp. 24–34, 2018.

[11] Y. Li, Z. Li, H. Wang, W. Wang, and L. Xing, "Evaluating the safety impact of adaptive cruise control in traffic oscillations on freeways," *Accident Analysis Prevention*, vol. 104, pp. 137–145, 2017.

[12] D. Ghraizi, R. Talj, and C. Francis, "An overview of decision-making in autonomous vehicles," *22nd IFAC World Congress*, 2023, (Accepted, to appear).

[13] A. Haydari and Y. Yılmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 1, pp. 11–32, 2022.

[14] 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, 2021.

[15] M. U. Yavas, T. Kumbasar, and N. K. Ure, "Toward learning human-like, safe and comfortable car-following policies with a novel deep reinforcement learning approach," *IEEE Access*, vol. 11, pp. 16 843–16 854, 2023.

[16] M. Brosowsky, F. Keck, J. Ketterer, S. Isele, D. Slieter, and M. Zöllner, "Safe deep reinforcement learning for adaptive cruise control by imposing state-specific safe sets," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 488–495.

[17] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *CoRR*, vol. abs/1708.06374, 2017.

[18] 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," 2019.

[19] Z. Wang, G. Wu, and M. J. Barth, "A review on cooperative adaptive cruise control (cacc) systems: Architectures, controls, and applications," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2884–2891.

[20] R. Yan, R. Jiang, B. Jia, J. Huang, and D. Yang, "Hybrid car-following strategy based on deep deterministic policy gradient and cooperative adaptive cruise control," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 2816–2824, 2022.

[21] A. Said, R. Talj, C. Francis, and H. Shraim, "Local trajectory planning for autonomous vehicle with static and dynamic obstacles avoidance," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 410–416.

[22] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory Comput.*, vol. 8, pp. 415–428, 2012.

[23] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," 06 2010, pp. 987 – 993.

[24] "Article r413-3 - code de la route - légifrance," accessed on: 19th July 2023, 17h43.

[25] R. Rajamani, *Vehicle Dynamics and Control*, ser. Mechanical Engineering Series. Springer US, 2014.

[26] A. Chebly, R. Talj, and A. Charara, "Coupled longitudinal/lateral controllers for autonomous vehicles navigation, with experimental validation," *Control Engineering Practice*, vol. 88, pp. 79–96, 07 2019.

[27] M. van Otterlo and M. Wiering, *Reinforcement Learning and Markov Decision Processes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 3–42.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 12 2013.

[29] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015.

[30] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016.

[31] A. Kesting, M. Treiber, and D. Helbing, "Agents for traffic simulation," *Multi-Agent Systems: Simulation and Applications*, 05 2009.