Decision-Making Strategy on Highway for Autonomous Vehicles

1st Siddahant Jain, 2nd Dhiram Ashok Buch, 3rd Arshnoor Singh Sachdeva, 4th Monark Bandishbhai Parekh
EEE 598: Reinforcement Learning in Robots
Arizona State University

***Abstract*—Autonomous driving is a promising technology that has the potential to improve driving efficiency and reduce accidents. In this project, we explore the overtaking maneuvers of self-driving vehicles in a highway environment using deep reinforcement learning techniques: Deep Q-Network (DQN), Dueling Deep Q-Network (Dueling DQN), and Double Deep Q-Network(DDQN). In the highway environment, the goal of the ego vehicle is to perform overtake maneuvers in a safe and efficient manner. The ego vehicle is controlled using two levels of controls: upper level and lower level controller. Upper-level control is responsible for making highway decisions, like decisions to overtake, decisions to slow down, etc., and the lower-level control is responsible for inputs like steering angle, throttle input, braking, etc. We implement reinforcement learning techniques for the upper-level controller for formulating a policy and evaluating the advantages and shortcomings of different implementations.**

***Index Terms*—DQN, Dueling DQN, Double DQN, ego-vehicle.**

I. Introduction

Autonomous vehicles are getting more attention because of their promise to reduce traffic and accidents that happen due to human errors. Autonomous vehicles have four important modules: perception, decision-making, planning, and control. Of the above-mentioned modules, this project explores the decision-making module and the controls module.

The Decision-making module performs tasks that would usually be performed by a "human brain" of the driver in a normal vehicle. It is responsible for making higher-level decisions for the vehicle including when to perform lane changes, lane keeps, overtake, brake, etc. Control Module on the other hand is responsible for lower-level controls of the vehicle. It receives signals from the decision-making module and is responsible for vehicle inputs like throttle, brake, and steering. In this project, we define a lower-level control module for the ego vehicle and then explore different Deep Reinforcement Learning (DRL) algorithms for the upper-level decision-making module. The proposed decision-making strategy is evaluated and estimated to be adaptive to other complicated scenarios.

The DRL methods tested on the decision-making module for upper-level control are DQN, Dueling DQN, and Double DQN. The work has compared all three DRL methods and also analyzed them theoretically. The performance of the proposed controllers is finally tested by executing simulation experiments in a highway environment.

II. Methods

A. Driving Environment and Control Module

The project utilizes an open-source highway driving scenario module. The construction of the Highway environment is done without losing generality. The lateral and longitudinal movements of the ego and surrounding vehicles are also managed by a hierarchical motion controller, which is also explained. Two models are present at the upper level: the intelligent driver model (IDM) and the minimizing overall braking induced by lane changes (MOBIL). The lower level is focused on controlling vehicle acceleration and speed.

1) Highway Driving Scenario: In autonomous driving, decision-making is choosing a series of plausible driving behaviors to carry out certain driving objectives. These actions on the highway include lane switching, lane keeping, accelerating, and braking. The major goals are avoiding crashes, operating effectively, and staying in the desired lane. Overtaking is a common driving maneuver that involves accelerating and surpassing other vehicles.

The driving scenario is shown in Fig1, which covers the decision-making challenge for autonomous vehicles on highways. The ego vehicle is the orange car, while the green cars are the surrounding vehicle.

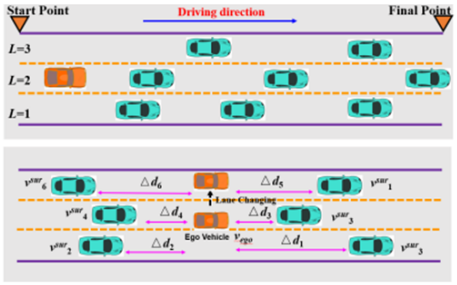


Fig. 1: Highway driving environment for decision making.

The ego vehicle's goal is to run as quickly as it can while avoiding collisions with other vehicles. Thus, efficiency and safety are interpreted as the two main objectives for running the vehicle on the highway. The initial speed and location of the surrounding vehicles are generated at random. It means that there are uncertainties in the actual driving situation. The ego vehicle can overtake other vehicles from the right or left side and likes to stay in lane 1 $(L = 1)$ in order to mimic real-world situations.

At the start of this driving task, all the surrounding vehicles are located in front of the ego vehicle to reduce the overall complexity. The ego vehicle may only continue under two circumstances: crashing other vehicles or running out of time. In this work, the process of running from the beginning point to the endpoint is referred to as one episode.

The following conditions for the driving scenario are established without losing generality: The ego vehicle's initial speed is selected between 23 to 25 m/s; its top speed is 40 m/s, and the length and breadth of all vehicles are 5 m and 2 m. One episode lasts 40 seconds, and the simulation frequency is 20 Hz. The beginning velocities of the nearby vehicles are randomly selected between 20 to 30 m/s, and IDM and MOBIL alter their behavior. These two models will be thoroughly discussed in the next section.

2) Vehicle Behaviour Controller: A hierarchical control system, as seen in Fig2, is responsible for controlling all vehicle movements in highway conditions. The lower level tries to make it possible for the ego vehicle to monitor a set target speed and follow a target lane while the top level uses IDM and MOBIL to control the vehicle behaviors. The ego vehicle in this work is controlled via the DRL approach. The bi-level structure in Fig2 is used as a benchmark to evaluate the DRL-based decision-making technique.

IDM is the microscopic model to implement vehicle-following and collision-free at the top level. In the adaptive cruise controller of automated vehicles, the longitudinal behavior is usually decided by IDM. In general, the longitudinal acceleration is IDM is determined as:

$$a = a_{max}[1 - (\frac{v}{v_{tar}})^{\delta} - (\frac{d_{tar}}{\Delta d})^2] \tag{1}$$

where v and a represent the current speed and acceleration of the vehicle, d is the distance to the front vehicle, a_{max} is the maximum acceleration, and is the constant acceleration parameter. The goal velocity and distance are v_{tar} and d_{tar}, respectively, and the target speed is achieved by a_{max} and d_{tar}. The front vehicle has an impact on the expected distance d_{tar} in IDM, which is calculated as follows:

$$d_{tar} = d_0 + Tv + \frac{v\Delta v}{2\sqrt{a_{max}b}} \tag{2}$$

where Δv is the relative speed between the two vehicles, b is the rate of deceleration in accordance with the comfortable purpose, and d_0 is the predetermined minimum relative distance.

IDM uses a priori definitions of relative speed and distance to infer the vehicle's velocity and acceleration at each time step. The default configuration was as follows: The ideal time gap T is 1.5 seconds, the maximum acceleration a_{max} is $6m/s^2$, the acceleration argument δ is 4, the comfortable deceleration rate b is -5 m/s2, and the minimum relative distance d_0 is 10 meters.

The MOBIL is used to decide on lateral lane changes since the IDM is used to assess longitudinal behavior. According to MOBIL, the safety criteria and incentive conditions are the two limits that should be applied to lane-changing behavior. These restrictions apply to the ego vehicle e, the ego vehicle's follower i in the current lane, and the follower j in the target lane change. Assuming that a_i^{old} and a_j^{old} represent

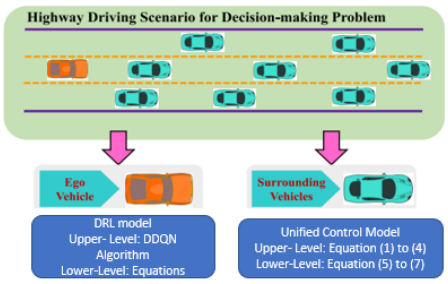


Fig. 2: Hierarchical control framework.

the followers' accelerations before changing, and a_i^{new} and a_j^{new} represent their accelerations after lane changing. The safety criterion requires the follower in the desired lane (after changing) to limit its acceleration to avoid a collision.

The mathematical expression is shown as

$$a_j^{new} \geq -b_{safe} \tag{3}$$

where b_{safe} is the maximum braking imposed on the follower in the lane-changing behavior. By following 3, collisions and accidents could be avoided effectively. The incentive condition is imposed on the ego vehicle and its followers by an acceleration threshold a_{th}:

$$a_e^{new} - a_e^{old} + z[(a_i^{new} - a_i^{old}) + (a_j^{new} - a_j^{old})] > a_{th} \tag{4}$$

where z is referred to as the politeness coefficient to determine the degree to which followers' lane-changing activities have an impact. Because of this incentive requirement, the preferred lane must be safer than the current lane. The settings in MOBIL are as follows: the politeness factor z is set to 0.001, the safe deceleration limit b_{safe} is set to 2 m/s^2, and the acceleration threshold is set to 0.2 m/s^2. After the upper level determines the longitudinal and lateral behaviors, the lower level is then used to apply the target speed and lane.

3) Vehicle Motion Controller: In the lower level, the motions of the vehicles in the longitudinal and lateral directions are controlled. The former regulates the acceleration by a proportional controller:

$$a = K_p(v_{tar} - v) \tag{5}$$

where K_p is the proportional gain.

In the lateral direction, the controller deals with the position and heading of the vehicle with a simple proportional-derivative action. The position indicates the lateral speed v_{lat} of the vehicle is computed as follows

$$v_{lat} = -K_{p,lat}\Delta_{lat} \tag{6}$$

where $K_{p,lat}$ is named as position gain and Δ_{lat} is the lateral position of the vehicle w.r.t center line. Then, the heading control is related to the yaw rate is given as

$$\dot{\varphi} = K_{p,\varphi}(\varphi_{tar} - \varphi) \tag{7}$$

where φ is the target heading angle to follow the desired lane and $K_{p,\varphi}$ is the heading gain.

Hence, the movements of the surrounding vehicles are achieved by the bi-level control framework in Fig2. The position, speed, and acceleration of these vehicles are assumed to be known as the ego vehicle. This limitation propels the ego vehicle to learn how to drive in the scenario via the trial-and-error procedure. In the next section, the DRL approach is

introduced and established to realize this learning process and derive the highway decision-making policy.

4) Variable Specification: To derive a decision-making strategy, initialization and calculative procedure are easily transformed into an analogous driving environment. The control actions are the longitudinal and lateral acceleration ($a1$ and $a2$) with the units m/s^2.

$$a_1 \in [-5, 5]m/s^2 \tag{8}$$

$$a_2 \in [-1, 1]m/s^2 \tag{9}$$

When these two accelerations are zeros, the ego vehicle adopts an idling control. After obtaining the acceleration action, the speed, and position of the vehicle can be computed as:

$$v_1^{t+1} = v_1^t + a_1 \cdot \Delta t \tag{10}$$

$$v_2^{t+1} = v_2^t + a_2 \cdot \Delta t \tag{11}$$

$$d_1^t = v_1^t \cdot \Delta t + \frac{1}{2} \cdot a_1 \cdot \Delta t^2 \tag{12}$$

$$d_2^t = v_2^t \cdot \Delta t + \frac{1}{2} \cdot a_2 \cdot \Delta t^2 \tag{13}$$

Where v_1, and v_2 are longitudinal and lateral speeds of vehicles similar to d_1 & d_2 with the frequency of 1Hz, which also indicate the time interval Δ_t as 1sec. The above equations are applicable for ego vehicles and surrounding vehicles simultaneously, and these expressions are considered as the transition model P in RL. The state variables defined as the relative speed and distance between the ego vehicle and the nearby vehicle are:

$$\Delta d_t = |d_t^{ego} - d_t^{sur}| \tag{14}$$

$$\Delta v_t = |v_t^{ego} - v_t^{sur}| \tag{15}$$

Where the superscript ego and sur represent the ego vehicle and surrounding vehicle. The reward model R is constituted by the optimal control objectives, Which are avoiding collision, running as fast as possible, and trying to drive on lane 1 (L=1)

$$r_t = -1 \cdot collision - 0.1 * (v_{ego}^t - v_{ego}^{max})^2 - 0.4 * (L-1)^2 \tag{16}$$

where collision 0, 1 and the goal of the highway decision-making strategy is maximizing the cumulative rewards, and the proposed decision-making policy is trained and evaluated in the simulation environment based on the open AI gym python toolkit.

With the number of lanes and surrounding vehicles is 4 and 30. The discount factor γ and learning rate α are 0.8 & 0.2. The layers of the value network and advantage network are both 128. The value of epsilon decreases from 1 to 0.05 with the time step of 6000. The training episode in different DRL approaches is 2000. The next section discusses the effectiveness of the presented decision-making strategy for autonomous vehicles.

III. Deep Reinforcement Learning Methods

In this section, all the deep reinforcement learning methods used and compared in this project are described. In summary, the project utilized 3 DRL methods: Deep Q-Network (DQN), Double DQN, and Dueling DQN.

A. Deep Q Network

A deep Q network (DQN) synthesizes the strengths of deep learning (neural network) and Q-learning to obtain the new state-value function. In the common Q-learning, the updating rule of this function is narrated as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max Q(s', a') - Q(s,a)] \tag{17}$$

where α ∈[0, 1] is named as a learning rate to trade-off the old and new learned experiences from the environment. s' and a' are the state and action at the next time step.

The conventional Q-learning method cannot handle problems with a huge space of state variables because it takes a long time to obtain the mutable Q table. In order to approximate the Q table as $Q(s; a; \theta)$, a neural network is used. The state-value function is the neural network's output, and its inputs are arrays of state variables and control actions. To measure the discrepancy between the approximated and

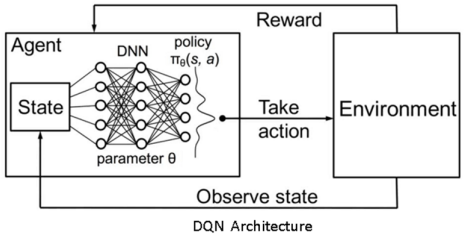


Fig. 3: Deep Q Network Algorithm(DQN).

actual Q table in DQN, the loss function is introduced like the following expression:

$$L(\theta) = E[\sum_{t=1}^{N}(y_t - Q(s,a;\theta))^2] \tag{18}$$

where

$$y_t = r_t + \gamma \underset{a'}{max} Q(s', a'; \theta') \tag{19}$$

As can be seen, the neural network has two parameters θ and θ' that assign two networks in DQN. These networks are target and prediction networks. While the latter seeks to provide the desired value, the former is used to estimate the present control action. Every predetermined number of time steps. The target network would typically replicate the parameters from the prediction network. By doing this, the network instability will be somewhat mitigated as the target Q table converges to predict one. In DQN, the online neural network is updated by gradient descent as follows:

$$\Delta_\theta L(\theta) = E[(y_i - Q(s,a;\theta))\Delta_\theta Q(s,a;\theta)] \tag{20}$$

The states and rewards are obtained using a unique criterion, and this action turns the DQN into an off-policy algorithm. Epsilon greedy is the name of the rule, which denotes that the agent uses probability "to explore (choose a random action) and probability 1-e to exploit (taking the best action available at the time)."

B. Double Deep Q Network

Double DQN uses two identical neural network models. One learns during the experience replay, just like DQN does, and

the other one is a copy of the last episode of the first model. The Q-value is actually computed with this second model. In DQN, Q-value is calculated with the reward added to the next state's maximum Q-value. Obviously, if every time the Q-value calculates a high number for a certain state, the value that is obtained from the output of the neural network for that specific state will become higher every time.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma Q'(s', a") - Q(s,a)] \quad (21)$$

where

$$a" = \underset{a}{max} Q(s', a) \quad (22)$$

Each output neuron value will get higher and higher until the difference between each output value is high. Now if let's say for state s action a is a higher value than action b, then action a will get chosen every time for state a. Now consider if for some memory experience action b becomes the better action for the state s. Then since the neural network is trained in a way to give a much higher value for action a when given state s, it is difficult to train the network to learn that action b is the better action in some conditions.

Hence a secondary model is a copy of the main model from the last episode and obviously since the difference between the values of the second model is lower than the main model, we use this second model to attain the Q-value. That is the way

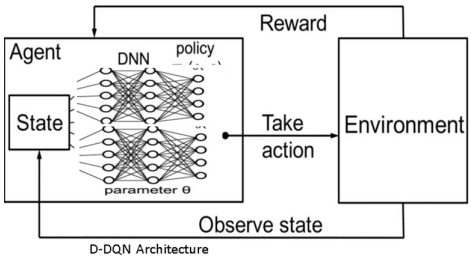


Fig. 4: Double Deep Q Network Algorithm(Double DQN).

Q-value gets calculated in Double DQN. We find the index of the highest Q-value from the main model and use that index to obtain the action from the second model.

C. Dueling Deep Q Network

DQN Reinforcement learning method tends to overestimate the rewards because it only uses the maximum value. In this project, it means that the DQN algorithm may not produce negative rewards for cases that would cause accidents. Dueling Deep Q-Network eliminates this problem by splitting the neural network into two branches, one estimates the states, and the other estimates the advantage function and combines the results later.

It consists of two streams of fully connected layers that are used to estimate the state-value function $V(s)$ and the advantage function $A(s,a)$ of each action. Since it is evident that the output of this new dueling network is likewise a Q table, it is also possible to approximate this Q table using the neural network used in DQN. The network with two parameters is computed as:

$$Q^{\pi}(s,a;\theta) = V^{\pi}(s;\theta_1) + A^{\pi}(s,a;\theta_2) \quad (23)$$

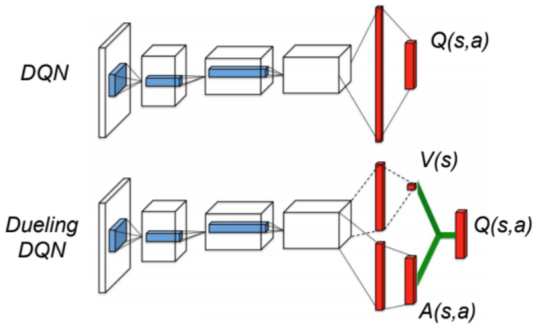


Fig. 5: Dueling DQN Algorithm

where θ_1 and θ_2 are the parameters of state-value function and advantage function, respectively.

To update the Q table in Dueling DQN and achieve the optimal advantage function respectively. To update Q table in Dueling-DQN and achieve the optimal control action, the above equation is reformulated as:

$$Q^{\pi}(s,a;\theta) = V^{\pi}(s;\theta_1) + (A^{\pi}(s,a;\theta_2) - \underset{a'}{max} A^{\pi}(s,a';\theta_2)) \quad (24)$$

$$a^* = \underset{a'}{argmax} Q(s,a';\theta) = \underset{a'}{argmax} A(s,a';\theta_2) \quad (25)$$

It can be discerned that the input-output interfaces in Dueling DQN and DQN are the same. Hence, the gradient descent in Eq 20 is capable of being recycled to train the Q table.

IV. Implementation & Simulation

The proposed decision-making control policies are trained and evaluated in the highway-env simulation environment based on the OpenAI gym and Python toolkit. The highway-env configuration is set up as follows: The number of lanes and surrounding vehicles is 4 and 50 respectively and the distance between the ego vehicle and surrounding vehicle is set to 200m for initial spawn as shown in Fig1. The observation parameter is set as Kinematics where the vehicle is represented by a dynamical system: a modified bicycle model. Its state is propagated depending on its steering and acceleration action. In highway-env other vehicles follow intelligent driver model (IDM) behavior with discrete meta actions. For training, the duration of the simulation is the 40s and the training episode in different DRL approaches is 2000, and the test simulation time is doubled to 80s to check the robustness of control policies. To implement and deploy the control policy in highway-env we used a 2.30 GHz Intel i7 vPro CPU with 4 GB RAM and Nvidia GTX 3050-Ti GPU.

TABLE I: Training & testing times

Techniques	Training Time(hr)	Testing Time(s)*
DQN	18	64
Double DQN	15	78
Dueling DQN	17	68

*Testing scenario is different from training scenario.

Table I provides the training and testing time of the DQN, Double-DQN, and Dueling-DQN approaches. The learned

parameters and policies can be applied online even though the training time can only be experienced offline. This motivates us to implement our decision-making policy in the visualization simulation environment. Testing simulation video links are provided below:

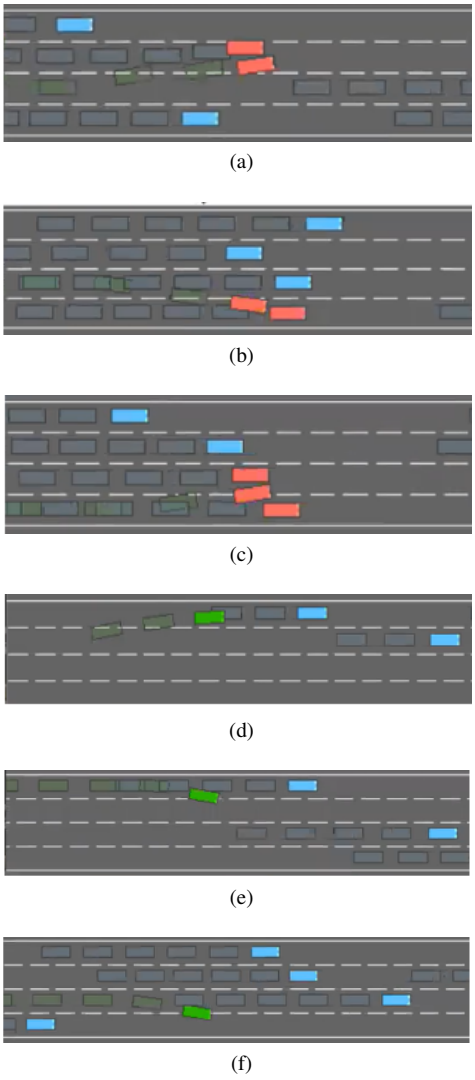
1) DQN link
2) Double DQN link
3) Dueling DQN link

V. Results

In this section, the optimality is analyzed by comparing DQN, Dueling-DQN, and Double DQN methods. We used a benchmark method for comparing the proposed decision-making policy while keeping the same parameters and initial conditions. Furthermore, to check the adaptability we tested our model in the same highway-env but in different driving scenarios.

A. Driving Conditions

The training process comprises some recurring cases that have been explored in this section. The figures: Fig 6a, Fig 6b and Fig 6c depict failure conditions that were widely occurring for a good part of the training process. Fig 6a shows that the agent changes the lane when it doesn't need to, which causes it to collide with another vehicle. Fig 6b shows that the agent needs to change the lane but chooses the wrong lane change. Finally, in Fig 6c the agent has no correct lane change option available, in which case the agent should have been patient to wait for an opportunity, which it does not do. During the testing process of the agent, it was discovered that the agent was performing well but it still had some problems when the environment was sparse as shown in Fig 6d. After training the agent again with 10 times more episodes it was discovered to have better results as shown in Fig 6e and Fig 6f, where the ego vehicle is performing exceptionally well while going through both sparse and dense environments.



(a) (b) (c) (d) (e) (f)

Fig. 6: Driving Conditions

Fig. 7: Average reward variation of DQN, Double DQN, and Dueling DQN

B. Optimality Evaluation

The DQN, Double-DQN, and Dueling-DQN adopted a hierarchical control framework. The model used two types of control level 1) Lower-level control which utilized the Eq5 - Eq7 to regulate the position, acceleration, and heading. 2) Upper-level control which utilized the DQN, Double-DQN, and Dueling-DQN algorithm as discussed in the section.

Fig7 Represent the total_reward of the three algorithms for the 40s time frame. According to the definition of a reward function (eq), a higher reward denotes driving in the chosen lane while performing more effective actions. As we can see in Fig7 The Dueling DQN achieved higher rewards faster than the other two algorithms because of the advantage of Eq23 parameters.

It is also noticeable that the Double DQN and Deuling DQN have better training stability than DQN methods. This is because the Double DQN and Deuling DQN networks could assess the worth of the chosen action at each step, which helps the ego vehicle to find a better decision-making policy.

Our results of DQN and dueling DQN match with the reference paper results in Fig8. The only difference is that in the reference paper the author normalized the reward and we didn't because we want to see the actual rewards for better

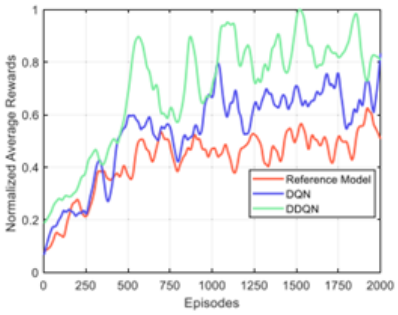


Fig. 8: Paper Reference: Average reward variation of algorithms

visualization and understanding.

All the models were trained for 2000 episodes to observe the trajectories of state variables in this work. Fig9 shows the average vehicle traveling time in these three compared techniques. A higher average time length indicates that the ego vehicle completes the driving scenario faster and achieves greater cumulative rewards. Vehicle speed and collision conditions both have an impact on travel time. The higher travel time means the ego vehicle could cover the long distance without collision for the given constraints and policy.

Fig. 9: Vehicle travel length of the ego vehicle at each episode

The travel time (of the ego vehicle) will fluctuate until the training is over due to the random set of the initial speeds and positions of nearby vehicles. These findings accurately reflect the demand for efficiency and safety. The noticeable differences can attest to the proposed algorithm's efficiency.

C. Adaptability Estimation

A short number of test episodes (10 episodes) is used to test the ability of vehicles in highway environments after training and learning. In the testing, we increased the time length and kept the default setting the same as training i.e the number of lanes and surrounding vehicles, etc. The neural network's learned parameters are preserved and can be used directly under different circumstances. The average reward of the testing process is the most concerning element.

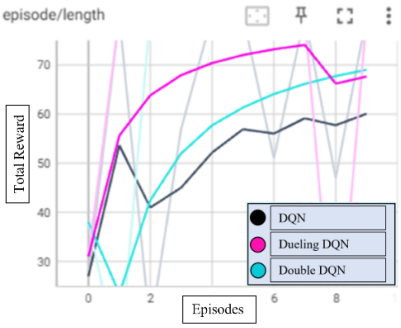


Fig. 10: Rewards of Test set simulation.

VI. Discussions & Conclusions

This project focused on the highway decision-making problem using the DRL technique. By applying the DQN, Dueling DQN, and Double DQN algorithms in the designed driving environments, an efficient and safe control framework is constructed and comparisons are made between the algorithms. Depending on a series of simulation experiments, the optimality, convergence rate, and adaptability are demonstrated. In addition, the testing results are analyzed, and the potential of the presented methods to be applied in real-world environments is reviewed. Finally, based on the benchmark factors and the simulation results, it is concluded that statistically, the Dueling DQN performs the best when compared to the regular DQN and Double DQN. Future work includes the online applications of highway decision-making by executing hardware-in-loop (HIL) experiments and the application of practical constraints in our model that may further refine the algorithm. Moreover, the real-world collected highway database can be used to estimate the related overtaking strategy.

From this project, we experienced firsthand how Reinforcement Learning algorithms solve an extremely complicated task such as highway navigation. While coding we faced several challenges ranging from extremely simple yet unnoticeable, like a mismatch of package versions, to complicated problems such as tensor multiplication errors. By solving these problems we got a better understanding of how to handle a number of tensors and other matrices simultaneously, and carry out operations on them. Finally, we also learned about several benchmark parameters and how to decide which parameters provide a better understanding of our results and help us make better comparisons.

References

[1] Liao, J., Liu, T., Tang, X., Mu, X., Huang, B., & Cao, D. (2020). Decision-making strategy on highway for autonomous vehicles using deep reinforcement learning. IEEE Access, 8, 177804-177814.

[2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

[3] Leurent, E.. (2018). rl-agents: Implementations of Reinforcement Learning algorithms.

[4] Leurent, E.. (2018). An Environment for Autonomous Driving Decision-Making.