Journal of Future Artificial Intelligence and Technologies

E-ISSN: 3048-3719

Research Article

Exploring Deep Q-Network for Autonomous Driving Simulation Across Different Driving Modes

Marcell Adi Setiawan [1], De Rosal Ignatius Moses Setiadi [1,2,*], Erna Zuni Astuti [1], T. Sutojo [1,2], and Noor Ageng Setiyanto [1,2]

[1] Informatics Engineering Department, Faculty of Computer Science, Dian Nuswatoro University, Semarang 50131, Indonesia; e-mail: tendavid2@gmail.com; moses@dsn.dinus.ac.id; erna.zuni.astuti@dsn.dinus.ac.id; sutojo@dsn.dinus.ac.id; nasetiyanto@dsn.dinus.ac.id

[2] Research Center for Quantum Computing and Materials Informatics, Faculty of Computer Science, Dian Nuswantoro University, Semarang 50131, Indonesia

* Corresponding Author : De Rosal Ignatius Moses Setiadi

Abstract: The rapid growth in vehicle ownership has led to increased traffic congestion, making the need for autonomous driving solutions more urgent. Autonomous Vehicles (AVs) offer a promising solution to improve road safety and reduce traffic accidents by adapting to various driving conditions without human intervention. This research focuses on implementing Deep Q-Network (DQN) to enhance AV performance in different driving modes: safe, normal, and aggressive. DQN was selected for its ability to handle complex, dynamic environments through experience replay, asynchronous training, and epsilon-greedy exploration. We designed a simulation environment using the Highway-env platform and evaluated the DQN model under varying traffic densities. The performance of the AV was assessed based on two key metrics: success rate and total reward. Our findings show that the DQN model achieved a success rate of 90.75%, 94.625%, and 95.875% in safe, normal, and aggressive modes, respectively. Although the success rate increased with traffic intensity, the total reward remained lower in aggressive driving scenarios, indicating room for optimization in decision-making processes under highly dynamic conditions. This study demonstrates that DQN can adapt effectively to different driving needs, but further optimization is needed to enhance performance in more challenging environments. Future work will focus on improving the DQN algorithm to maximize both success rate and reward in high-traffic scenarios and testing the model in more diverse and complex environments.

Keywords: Autonomous Vehicles; Deep Q-Network; Driving Modes; Highway-env; MLP; Reinforcement Learning; Traffic Simulation.

Received: August, 31st 2024
Revised: October, 10th 2024
Accepted: October, 17th 2024
Published: October, 18th 2024
Curr. Ver.: October, 18th 2024

1. Introduction

The rapid growth of vehicles has led to increased road congestion, thus requiring higher driving skills, especially in complex traffic conditions. Recent studies have shown a significant increase in vehicle ownership, contributing to traffic congestion on highways and urban areas[1], [2]. Therefore, the need for safer and more efficient driving solutions is increasingly urgent. One promising approach to address this problem is the development of Autonomous Vehicles (AVs)[3]–[8]. AVs have the potential to move autonomously without human intervention, making them a viable solution to improve road safety and reduce traffic accidents. One important aspect of AV operation is adjusting the driving mode, especially on highways with various traffic conditions. For example, when traffic is smooth, AVs can operate at high speeds, but when traffic is heavy, AVs must reduce speed and maintain a safe distance from other vehicles. This adjustment of the driving mode is generally regulated through reinforcement learning (RL) algorithms[6]–[8]. Reinforcement learning helps AVs make decisions based on Action Space (A), State Space (S), Reward (R), and Discount Factor (γ)[9], allowing AVs to function effectively in dynamic and complex environments. Reinforcement learning (RL) methods are generally divided into two types: model-free reinforcement learning

(MFRL) and model-based reinforcement learning (MBRL)[10]. In MFRL, agents learn directly from the experience of interacting with the environment without requiring a predictive model of the environment itself. Examples of MFRL models that are often used are Soft Actor-Critic (SAC)[11], [12], Deep Q-Learning (DQN)[5], [13]–[16], Advantage Actor-Critic (A2C)[17], [18], and Proximal Policy Optimization (PPO)[13], [19], [20]. In contrast, MBRL uses a more structured approach by modeling the environment to predict the outcomes of various actions taken. One of the popular MBRL methods is Monte Carlo (MC), which requires specific sample data to ensure prediction accuracy[21]–[23].

The main advantage of MFRL is its flexibility in environmental exploration and decision-making, as it is not bound by a rigid environmental model[21]. However, MFRL usually requires more interactions with the environment to achieve optimal results. On the other hand, MBRL is often more efficient regarding sample data usage, as the agent is trained based on a more structured environmental model. However, the success of MBRL depends heavily on how accurately the environmental model can represent real conditions[23]. Of the various methods available in RL, several approaches, such as PPO, A2C, and SAC have been widely used in autonomous vehicle simulation. PPO, for example, is known for its ability to handle dynamic environments but often requires intensive hyperparameter tuning and faces challenges in training stability. Although A2C and SAC are effective in some aspects, they have limitations in terms of sample and memory usage efficiency.

One of the prominent methods in the MFRL category is DQN. DQN has several advantages, such as prioritizing experience replay, asynchronous training, and calculating the loss matrix, reducing sample selection complexity[24]. In addition, the recursive nature of DQN allows for improved performance in partially observed environments and can be easily scaled as the complexity of observations increases. DQN is also known to be stable in training and can handle exploration-exploitation problems through the epsilon-greedy technique [25]. Compared with methods such as PPO, DQN is more efficient in sample and memory usage. Meanwhile, PPO often faces stability challenges in more complex environments and requires deeper hyperparameter tuning to achieve optimal performance[26]. After considering these methods' various advantages and disadvantages, this study focuses on exploring DQN as the main model. This study aims to:

- Develop the best setting of the DQN model with different driving modes (safe, normal, and aggressive), so that the Autonomous Vehicle (AV) can be more flexible in adjusting to user needs.
- Evaluate the model's performance by measuring and analyzing the success rate and total reward obtained by the DQN agent in driving scenarios with varying traffic conditions.

This paper is structured as follows: Section 2 contains a literature review on reinforcement learning methods in autonomous vehicles. Section 3 describes the methodology, including the design of the simulation environment and DQN architecture. Section 4 presents the results and discussion of the experiments, and Section 5 offers conclusions and directions for further research.

2. Related Works

In recent years, there has been a large number of studies studying autonomous vehicles, given their increasingly massive production rates. Along with that, advances in the field of RL have driven its application in various fields, especially in autonomous vehicles. Several studies have proposed different methods according to the problems' complexity. Previous research [19] used PPO to accelerate the training process by testing three different approaches, namely Curriculum Proximal Policy Optimization (CPPO) and two PPOs with clipping parameters settings ($\varepsilon = 0.15, 0.25$), which were applied to the highway-env environment with the intersection type. The three methods were trained using a fully connected network with one hidden layer consisting of 128 units for the action network, and 64 units for the critical network. Optimization was performed using the Adam optimizer and consistent parameters, such as a discount factor of 0.9 and 20 epochs. The results showed that CPPO was able to accelerate the training process by 47.2% faster than PPO1, and 40.2% faster than PPO2. In terms of success rate and collision rate, CPPO showed better results with a success rate of 78.5%, compared to PPO1 (76.5%) and PPO2 (72%). Meanwhile, the failure rate (collision rate) of CPPO was lower, at 21.5%, compared to PPO1 (23.5%) and PPO2 (28%). Another study [12] used SAC as an RL method, with SVL SUMO and Carla-based environmental simulation

spaces. Testing was carried out in a roundabout scenario, with five different scenarios focused on variations in roundabout diameter (16m, 20m, 32m, 40m, and 50m). The model was randomly trained using 52 routes involving various roundabout sizes for 5700 episodes, with a discount factor of 1.0. This model uses three hidden layers for the q-model and three for the policy model. The test results show that the SAC agent can achieve a success rate of 73% in some scenarios but still shows limitations in scenarios with larger diameters.

Another study used A2C) in a highway exit-entry scenario, which distinguished two types of agents: egoistic agents (egoistic AVs) and altruistic agents (altruistic AVs)[18]. Egoistic agents tended to perform worse, only able to cover a distance of 227.6m with a high collision failure rate (77.6% and 78.2%), while altruistic agents were able to cover a distance of up to 498.2m with a much lower collision failure rate (2.1% and 16.3%).

Another study examined DQN applied to autonomous vehicles in a highway-env simulation environment consisting of a straight highway with varying levels of traffic density[27]. The results of the four scenarios show that at low to medium traffic density (20 to 40 vehicles/frame), the DQN agent can achieve a success rate of up to 94.875%, with an average reward of 34.45. However, at higher traffic density (60 to 80 vehicles/frame), the DQN performance drops significantly, with the success rate dropping to 61% and the reward reaching only 19 points. This indicates that the DQN agent must still be optimized to adapt to denser environments. Although many studies have been conducted to apply various RL methods, such as PPO, SAC, A2C, and DQN in autonomous vehicle simulations, most studies focus on training efficiency and vehicle performance under stable traffic conditions. However, studies evaluating the performance of DQN under different driving modes—such as safe, normal, and aggressive—are still very limited. In addition, there is a lack of understanding of how DQN performs in high-density and dynamic traffic scenarios. This study aims to fill this gap by exploring the performance of DQN in three different driving modes and more dynamic traffic conditions, making new contributions to understanding and optimizing DQN in autonomous vehicle simulation.

3. Methodology

This section will provide a comprehensive system overview, starting with an overview to provide context. Next, we will discuss the development of the toll road environment and scenarios, along with some explanations. Finally, we will describe the in-depth design of each DQN component, detailing the architecture and rationale behind each design decision.

3.1. System Overview

This study uses modules such as DQN agent, action, and environment, which are then integrated into one system architecture. Figure 1 illustrates a summary of the system on the autonomous car agent. Highway-env as an environment is used to take the value of the observation results when the agent is in the current state using the Python API. With the previous state's reward, it is entered into the DQN agent framework to be trained as input using the MLP policy. The DQN agent will issue output in the form of actions taken based on the Q-value [5], [28] of the MLP policy, and provide a response to the environment that continues to the next system series.

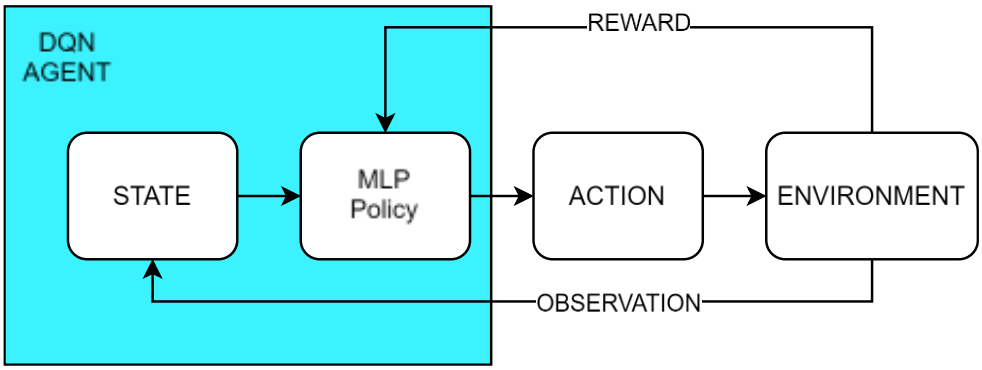

Figure 1. An overview of the system architecture

3.2. Simulation Environment

Several open-source libraries for the environment are widely developed for training and testing on RL algorithms. Many simulators are used in the field of autonomous vehicle research, such as Highway-env[19] and [29]. Highway-env is used to train the autonomous vehicle control function or agent (in green), as depicted in Figure 2.

In training autonomous vehicles in the environment, the Stable-Baseline 3 assistance tool is used to observe and develop DQN agents against the environment provided. Stable-Baseline 3 also has a replay buffer feature which plays a crucial role in storing training result data, namely [state, action, reward] so that agents can adapt to the environment quickly.

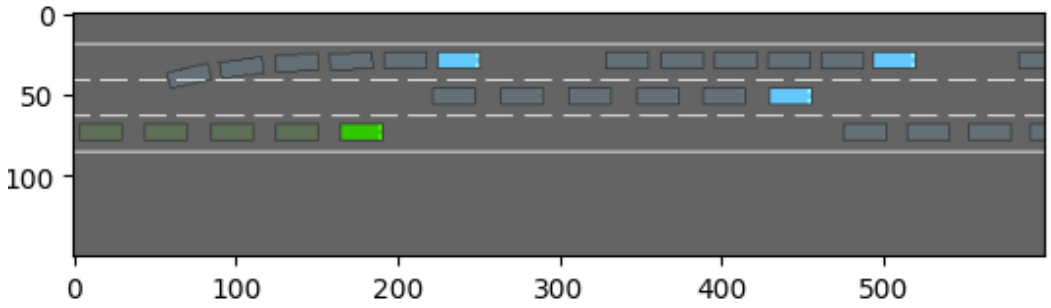

Figure 2. A snapshot of the autonomous vehicle environment during training simulation

3.3. Custom Scenarios Development

Many types of environments can be used when using the Highway-env simulator. The research used the type of environment in the form of a toll road. The environment can be configured as in Table 1 according to the scenario. The configuration carried out is changing the number of lanes in the toll road environment to 3 lanes, the configuration of the value of each reward, the length of the agent training period in the environment, the number of obstacles which in the case of autonomous cars is the number of surrounding cars, and the level of traffic density.

Table 1. Environment configurations

Configuration	Values
action_type 1	DiscreteMetaAction
lanes_count	3
duration	40
vehicles_count	20
vehicles_density	1
collision_reward	-1
right_lane_reward	0.1
high_speed_reward	0.3
lane_change_reward	0.1

In each scenario, the setting used is a toll road simulation. The toll road is filled with agents and several vehicles surrounding the agent. The initial position, speed, and destination of vehicles other than the agent are initialized randomly. The agent will run at a speed that has been adjusted in Table 2. The goal of the agent itself is to survive as long as possible on the toll road simulation and avoid collisions with other vehicles.

Table 2. Driving speed scenario

Scenario	Speed range
Safe	[17,20, 22]
Normal	[17,20, 25]
Aggressive	[17,25, 28]

Table 3 shows an example of kinematic observations taken by the ego vehicle during the simulation. In this simulation, several vehicles are monitored by the agent, including the ego vehicle itself and two other vehicles (vehicle 1 and vehicle 2). Each vehicle is monitored based on its position (x dan y) and its velocity on the horizontal (v_x) and vertical (v_y) axes.

Table 3. Observasi environment oleh Agent

Vehicle	x	y	v_x	v_y
ego-vehicle	0.05	0.04	0.75	0
vehicle 1	-0.15	0.00	-0.15	0
vehicle 2	0.08	0.04	-0.075	0

In Table 3, the ego vehicle has an x position of 0.05 and a y position of 0.04, indicating its location relative to its surroundings. Its horizontal velocity (v_x) is 0.75, indicating that the vehicle is moving to the right at a reasonably high-speed while v_y is zero, indicating no movement in the vertical axis. Vehicle 1 has an x position of -0.15, meaning it is to the left of the ego vehicle. Its velocities (v_x) and v_y) are zero, indicating that the vehicle is stationary at that location. The vehicle is to the right of the ego vehicle with an x position of 0.08 and a y position of 0.04 but is moving slowly to the left with a horizontal velocity v_x) of -0.075. Just like the ego vehicle, there is no vertical movement in this vehicle (v_y= 0). The actions that can be taken by the agent in the scenario are a set of five actions designed in set A, shown in Equation (1).

$$A = \begin{bmatrix} \text{lane\_left} \\ \text{idle} \\ \text{lane\_right} \\ \text{faster} \\ \text{slower} \end{bmatrix} \quad (1)$$

In action set A, the agent can take actions to turn left, maintain a fixed position, turn right, increase speed, and decrease speed according to the observations made. The rewards applied to the agent in the environment are designed as follows:

- Agent receives a reward of 0.1 if it takes the right lane
- Agent receives a reward of 0.3 if it increases its speed
- Agent receives a reward of 0.1 if it takes the left lane
- Agent receives a reward if it is only at the speed limit [20, max(agent speed)] in m/s
- Agent receives a reward of -1 if it experiences a collision

This reward scheme encourages agents to perform maneuvers and arrive at their destination as quickly as possible while avoiding collisions.

3.4. Evaluation Metrics

This study uses two evaluation metrics to assess safety issues (success rate and total reward) concerning the DQN agent model. The success rate calculates the percentage of episodes where the ego vehicle (agent) successfully accomplishes its goal (without collision) within a specified duration. This also considers the functionality of the autonomous vehicle agent, which evaluates the DQN model's ability to learn behavior and complete tasks. In this test, the autonomous driving scenario in the environment is set with a time limit representing the maximum training duration per episode. The success rate can be calculated by taking the duration the agent survives in each episode and the duration of the environment in each episode used for training, as shown in Equation (2).

$$\text{Success\_rate}(\%) = \frac{\text{survived\_duration}}{\text{environment\_duration}} \times 100 \quad (2)$$

The total reward is a standard metric used to evaluate the performance of the ego vehicle over several episodes, which is calculated using Equation (3).

$$\text{Total Reward} = \sum_{i=1}^{n} r_i \quad (3)$$

Where r_i is the reward received at the i-th step, and n n is the total number of steps in the testing episode.

In this testing, the total reward received by the ego vehicle is collected after taking several actions within the environment in each episode. As the ego vehicle improves its learning, the reward value earned by the ego vehicle will increase in subsequent episodes.

3.5 Experimental procedure

This study was conducted by testing the performance of the DRL DQN model using three different test scenarios. The first scenario trains the ego vehicle with safety mode, the second scenario trains the ego vehicle with normal mode, the last scenario trains the ego vehicle with aggressive mode. All scenarios will be trained by the DQN model with the same architectural settings as in Table 4 below.

Table 4. Training configuration

Hyperparameter	Values
policy	'MlpPolicy'
learning_rate	5e-4
buffer_size	15000
learning_starts	100
batch_size	32
discount_factor	0.8

The training will be run for 12000 step training episodes, then the ego vehicles testing section will be tested with 20 episodes. After testing is carried out, the results of each reward and the duration of the agent's survival in the environment in each episode will be stored as a value to calculate the success rate and total reward.

4. Results and Discussion

In this section, the results of the experiment will be discussed and evaluated. Testing is done by placing the agent in a simulation environment. This section consists of three main parts, namely the training phase, the testing phase, and the discussion.

4.1. Training Phase Evaluation Result

The training results of the DQN model for 12000 step episodes with various driving modes. Figure 3 shows the model performance in three scenarios: Safe, Normal, and Aggressive. The model performance is seen from the reward value (ep_rew_mean) and the agent survival duration (ep_len_mean) during training. The ep_rew_mean value shows how effectively the agent makes decisions to reach the goal safely and efficiently. A higher reward indicates that the agent has learned a better policy and is more efficient in completing its tasks. The ep_len_mean value reflects the agent's ability to survive longer in the simulated environment. If the episode duration is longer, the agent successfully avoids obstacles and survives longer in the simulated driving situation.

In the safe scenario, ep_len_mean (Figure 3a) consistently increases after a brief dip at around 2,000 steps. After this exploration phase, the agent shows stability and consistency and persists longer in each episode. The persisting duration increases significantly after 4,000 steps, indicating that the agent's policy is starting to perform optimally. This increase continues until the end of training at 12,000 steps, with the episode duration almost reaching its maximum. In contrast, ep_rew_mean (Figure 3b) also follows a similar pattern, with a steady increase after the initial exploration phase. The agent's reward increases significantly to nearly 32 points at 12,000 steps. This indicates that the agent successfully maximizes its reward through better decisions in a safe scenario. Thus, both ep_len_mean and ep_rew_mean are consistent in showing a gradual increase in agent performance after the initial exploration period.

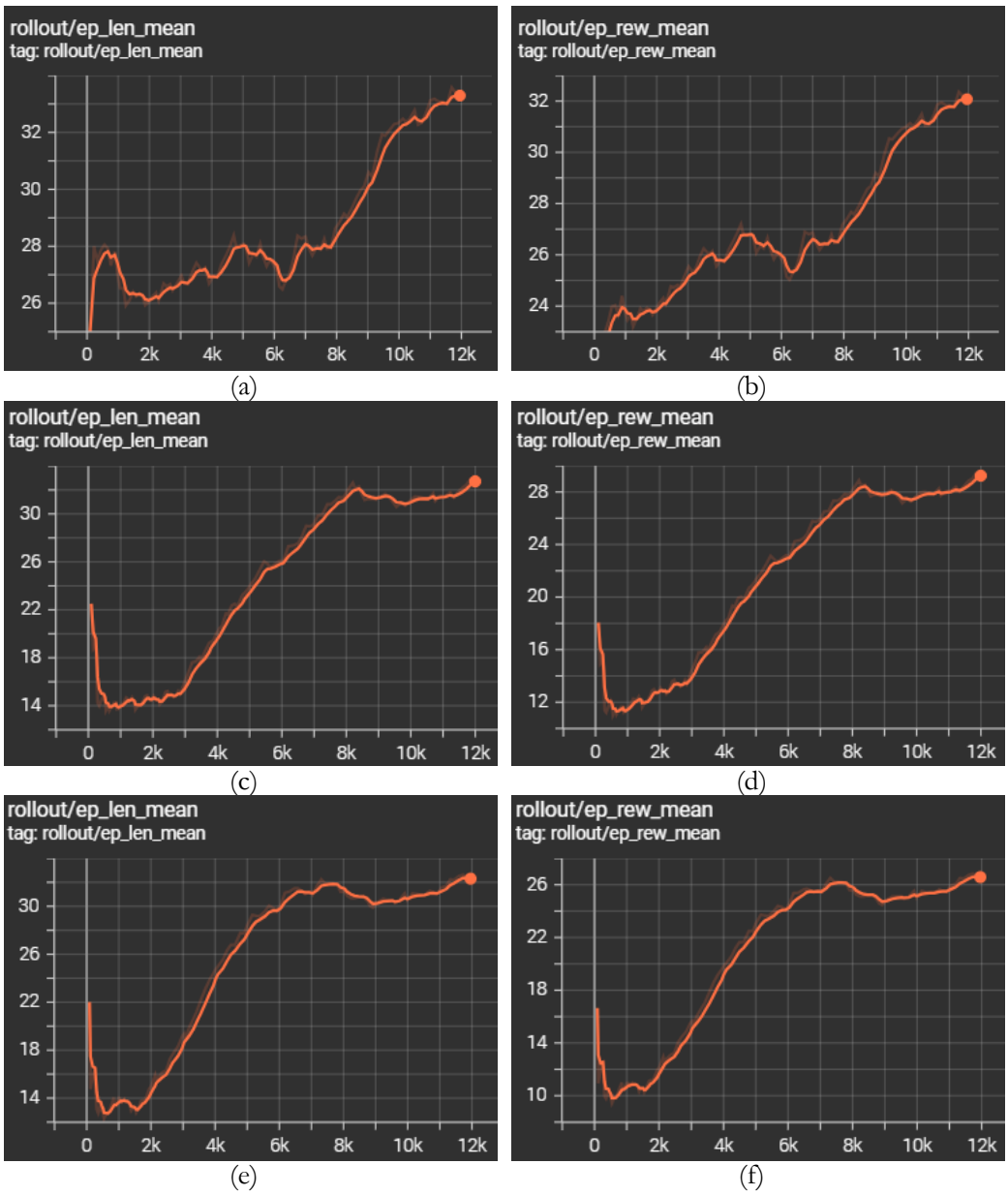

Figure 3. Showcase of training episodes using DQN in (**a**) ep_len_mean of safe scenario; (**b**) ep_rew_mean of safe scenario; (**c**) ep_len_mean of normal scenario; (**d**) ep_rew_mean of normal scenario; (**e**) ep_len_mean of aggressive scenario; and (**f**) ep_rew_mean of aggressive scenario.

In the normal scenario, ep_len_mean (Figure 3c) shows a more stable improvement pattern than the safe scenario. Although the agent also experiences a similar initial performance drop at around 2,000 steps, the survival time quickly increases consistently after 4,000 steps. The episode duration approaches 30 seconds towards the end of training, indicating that the agent is starting to be able to survive longer in a more dynamic environment. Meanwhile, ep_rew_mean (Figure 3d) shows a more volatile pattern. The reward increases after a fairly sharp initial drop at around 2,000 steps, but the increase is not as rapid as ep_len_mean. Although the reward increases with more episodes, it peaks under the safe scenario, with the reward approaching 28 points at the end of training. This suggests that although the agent can survive longer, it has not yet fully maximized its reward in this scenario, indicating potential for further optimization.

In the aggressive scenario, ep_len_mean (Figure 3e) shows the greatest challenge among the three scenarios. The agent's survival time drops significantly in the early stages, around 1,000 training steps, indicating that the agent struggles to find an effective policy in this scenario. However, after around 3,000 steps, the agent shows gradual improvement, although the improvement is slower than in the other scenarios. Up to 12,000 steps, the survival time

remains around 30 seconds, indicating that despite the improvement, the agent still struggles to maintain its performance consistently. In Figure 3f, ep_rew_mean also reflects the same difficulty, with the reward decreasing sharply in the early stages of training. The reward only starts to increase significantly after 3,000 steps, but the increase is more moderate than the safe and normal scenarios. The agent's reward in the aggressive scenario reaches around 26 points at the end of training, which is the lowest value compared to the other scenarios. This indicates that the agent still struggles to maximize its reward in more dynamic and aggressive driving conditions.

Overall, there is consistency between the improvements in ep_len_mean and ep_rew_mean in the safe and normal scenarios, where both metrics support each other and increase as training progresses. However, despite improvements in both metrics in the aggressive scenario, the reward improvement seems to lag behind the survival duration, indicating that the agent is not fully optimized in exploiting the conditions of the more aggressive scenario. This indicates the need for further improvements to the policy algorithm to address the challenges in scenarios with higher difficulty levels.

4.2 Testing Phase Evaluation Result

In this section, the test results of the DQN model for 20 episodes with three driving modes are presented in Table 5.

Table 5. The testing performance of DQN DRL models in drive mode scenarios.

Scenarios	Metrics	Results
Safe	Average Success rate	90.750%
	Average reward	35.70
	Average speed	21.8 m/s or 78.4 km/h
Normal	Average Success rate	94.625 %
	Average reward	33.42
	Average speed	24.9 m/s or 89.6 km/h
Aggressive	Average Success rate	95.875 %
	Average reward	31.85
	Average speed	25.2 m/s or 90.7 km/h

In the safe scenario, it can be observed that the model achieved a success rate of 90.75%. This figure shows that the model can run well in a safe scenario, completing the tasks or challenges with a fairly high success rate. The average reward obtained by the model in this scenario is 35.70 points. This result illustrates that the model has successfully formed an optimal policy during testing.

In the normal scenario, the model is able to show better performance with a success rate of 94.625%. This result indicates that the model can adapt well to environmental conditions in normal driving mode. The average reward obtained in this scenario is 33.42 points, indicating that the model is not only successful in dealing with tasks but also efficient in making decisions.

In the last scenario, namely aggressive mode, the model can show its best performance with a success rate of 95.875%. However, despite the high success rate, the average reward obtained in this scenario is 31.85 points, which is lower than other scenarios. This shows that the model successfully completes the task, but there is room for improvement in terms of maximizing rewards in more dynamic and challenging driving scenarios.

4.3 Discussion

Based on the training evaluation (Figure 3), it can be seen that the DQN model in the safe scenario experienced a decline in performance at around 6000-7000 step episodes. Still, the reward graph slowly began to increase after 8000 step episodes until it reached the training limit at 12000 step episodes. If the training episode limit is increased, the performance in the safe scenario can likely continue to increase and obtain higher rewards. For the normal and aggressive scenarios, the reward graphs began to stabilize at around 7000-12000 step episodes, indicating that the agent has achieved maximum performance. However, if the episode limit

is increased, the reward in the normal scenario is estimated to remain at around 33.42 points, while in the aggressive scenario, the reward will be at around 31.85 points, indicating that the model is not yet fully optimal in a more dynamic driving scenario.

Based on the testing evaluation in Table 5, the DQN model can effectively improve safety performance in autonomous vehicle simulations. In the Safe scenario, the agent obtained a high average reward of 35.70 points with a success rate of 90.75%, slightly lower than previously reported. In Normal mode, the agent performed better, with a reward of 33.42 points and a success rate of 94.625%. Meanwhile, in the aggressive scenario, the agent obtained a reward of 31.85 points, with a success rate of 95.875%. This evaluation shows that although the success rate increases, the rewards obtained in the aggressive scenario tend to be lower. This indicates that the model still faces challenges in maximizing performance in more challenging driving scenarios. With the high success rate, it can be concluded that the policy is running well, but further optimization is needed to maximize rewards, especially in more dynamic driving conditions.

5. Comparison

In comparing test results, we have adjusted the environment by using the average reward metric and setting the vehicle density level to 20 vehicles/frame, according to the first test scenario in previous research [27]. After adjusting the environment, we compared the test results in research [27] with an average reward of 35.61 points and a success rate of 90.075%. Compared to the test results in the Safe scenario, the average reward we obtained was slightly higher, which was 35.70 points, with almost the same success rate of 90.75%. Although this increase in reward is relatively small, it shows that our model can perform slightly better than previous research. In the Normal and Aggressive scenarios, the average reward obtained tends to be lower than Safe mode, with a value of 33.42 points for Normal and 31.85 points for Aggressive. However, the success rate for both scenarios is much higher, 94.625% for Normal and 95.875% for Aggressive, compared to the success rate in the study[27]. This indicates that our model is more successful in facing more dynamic environmental challenges, although the rewards obtained are slightly lower.

Overall, the test results show that our DQN model is able to outperform previous studies in terms of success rate, especially in the Normal and Aggressive scenarios. Although the average reward is lower, the model's overall performance in avoiding failures and completing driving tasks effectively increases. Adjustments made to the environment, such as speed range, reward settings, and several other configurations, positively affect DQN performance, especially in more dynamic scenarios. These results indicate that there is still potential for further improvement, especially in maximizing rewards in more complex traffic conditions, while maintaining a high success rate stability.

6. Conclusions

This study has successfully developed the best settings for the DQN model) in three different driving modes: safe, normal, and aggressive. With these variations in driving modes, A is able to adjust its performance based on user needs in different situations. The model performance is evaluated by measuring the success rate and total reward obtained by the DQN agent.

The experimental results show that the proposed DQN model is able to achieve very good performance in the safe scenario, with a success rate reaching 91.375%. Although the average reward obtained is slightly lower than the safe scenario in the normal and aggressive scenarios, the success rate has increased significantly to 94.625% and 95.875%, respectively. This indicates that DQN can adapt well to meet more dynamic and challenging driving needs, although there is room for further improvement in maximizing rewards in both scenarios.

However, this study has several limitations. First, although the DQN model has been tested on various driving modes, the test was only conducted in one type of environment (highway-env), which may limit the generalization of the results to other driving environments. Second, although the success rate is quite high, the reward obtained in the aggressive scenario is still relatively low, indicating that the DQN agent policy is not fully optimal in more complex scenarios. For further research, development can focus on improving the DQN algorithm in dealing with denser and more diverse traffic conditions and testing the model in other more complex simulation environments. This research also opens up

opportunities to explore the integration of DQN with other reinforcement learning models to improve the adaptability and efficiency of AVs in various driving needs.

Author Contributions: Conceptualization: M.A.S and D.R.I.M.S; Methodology: M.A.S and D.R.I.M.S.; Software: M.A.S; Validation: X M.A.S and D.R.I.M.S; Formal analysis: E.Z., T, T.S. and N.A.S; Investigation: X.X.; Resources: M.A.S.; Data curation: M.A.S.; Writing—original draft preparation: M.A.S; Writing—review and editing: D.R.I.M.S; E.Z., T, T.S. and N.A.S; Visualization: M.A.S.; Supervision: D.R.I.M.S.; Project administration: E.Z., T.; Funding acquisition: All.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] D. R. I. M. Setiadi, R. R. Fratama, and N. D. A. Partiningsih, "Improved Accuracy of Vehicle Counter for Real-Time Traffic Monitoring System," *Transp. Telecommun. J.*, vol. 21, no. 2, pp. 125–133, Apr. 2020, doi: 10.2478/ttj-2020-0010.

[2] D. R. I. M. Setiadi, R. R. Fratama, N. D. A. Partiningsih, E. H. Rachmawanto, C. A. Sari, and P. N. Andono, "Real-time multiple vehicle counter using background subtraction for traffic monitoring system," in *Proceedings - 2019 International Seminar on Application for Technology of Information and Communication: Industry 4.0: Retrospect, Prospect, and Challenges, iSemantic 2019*, Sep. 2019, pp. 23–27. doi: 10.1109/ISEMANTIC.2019.8884277.

[3] N. Mirzaeian, S.-H. Cho, and A. Scheller-Wolf, "A Queueing Model and Analysis for Autonomous Vehicles on Highways," *Manage. Sci.*, vol. 67, no. 5, pp. 2904–2923, May 2021, doi: 10.1287/mnsc.2020.3692.

[4] J. E. Park, W. Byun, Y. Kim, H. Ahn, and D. K. Shin, "The Impact of Automated Vehicles on Traffic Flow and Road Capacity on Urban Road Networks," *J. Adv. Transp.*, vol. 2021, pp. 1–10, Nov. 2021, doi: 10.1155/2021/8404951.

[5] P. Rodrigues and S. Vieira, "Optimizing Agent Training with Deep Q-Learning on a Self-Driving Reinforcement Learning Environment," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec. 2020, pp. 745–752. doi: 10.1109/SSCI47803.2020.9308525.

[6] B. Thunyapoo, C. Ratchadakorntham, P. Siricharoen, and W. Susutti, "Self-Parking Car Simulation using Reinforcement Learning Approach for Moderate Complexity Parking Scenario," in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, Jun. 2020, pp. 576–579. doi: 10.1109/ECTI-CON49241.2020.9158298.

[7] Z. Cao *et al.*, "Highway Exiting Planner for Automated Vehicles Using Reinforcement Learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 990–1000, Feb. 2021, doi: 10.1109/TITS.2019.2961739.

[8] 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 Trans. Syst. Man, Cybern. Syst.*, pp. 1–14, 2019, doi: 10.1109/TSMC.2018.2870983.

[9] A. Amballa, A. P., P. Sasmal, and S. Channappayya, "Discrete Control in Real-World Driving Environments using Deep Reinforcement Learning," *arXiv*. Nov. 28, 2022. [Online]. Available: http://arxiv.org/abs/2211.15920

[10] D. Valencia *et al.*, "Comparison of Model-Based and Model-Free Reinforcement Learning for Real-World Dexterous Robotic Manipulation Tasks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 871–878. doi: 10.1109/ICRA48891.2023.10160983.

[11] P. Christodoulou, "Soft Actor-Critic for Discrete Action Settings," *arXiv*. Oct. 16, 2019. [Online]. Available: http://arxiv.org/abs/1910.07207

[12] D. Attard and J. Bajada, "Autonomous Navigation of Tractor-Trailer Vehicles through Roundabout Intersections," *arXiv*. Jan. 10, 2024. [Online]. Available: http://arxiv.org/abs/2401.04980

[13] R. Sharma and P. Garg, "Optimizing Autonomous Vehicle Navigation with DQN and PPO: A Reinforcement Learning Approach," in *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*, Jul. 2024, pp. 1–5. doi: 10.1109/APCIT62007.2024.10673440.

[14] J. Wang, C. Hu, J. Zhao, L. Zhang, and Y. Han, "Deep Q-Network-Enabled Platoon Merging Approach for Autonomous Vehicles," *Transp. Res. Rec. J. Transp. Res. Board*, vol. 2678, no. 7, pp. 17–31, Jul. 2024, doi: 10.1177/03611981231203229.

[15] A. Rizehvandi, S. Azadi, and A. Eichberger, "Enhancing Highway Driving: High Automated Vehicle Decision Making in a Complex Multi-Body Simulation Environment," *Modelling*, vol. 5, no. 3, pp. 951–968, Aug. 2024, doi: 10.3390/modelling5030050.

[16] Z. Lu and L. Lu, "Autonomous Driving Strategy for Highway Parallel-type On-ramp Merging Based on Deep Reinforcement Learning," in *2023 5th International Academic Exchange Conference on Science and Technology Innovation (IAECST)*, Dec. 2023, pp. 958–966. doi: 10.1109/IAECST60924.2023.10503333.

[17] T. Chu, J. Wang, L. Codeca, and Z. Li, "Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020, doi: 10.1109/TITS.2019.2901791.

[18] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Altruistic Maneuver Planning for Cooperative Autonomous Vehicles Using Multi-agent Advantage Actor-Critic," Jul. 2021, [Online]. Available: http://arxiv.org/abs/2107.05664

[19] Z. Peng, X. Zhou, Y. Wang, L. Zheng, M. Liu, and J. Ma, "Curriculum Proximal Policy Optimization with Stage-Decaying Clipping for Self-Driving at Unsignalized Intersections," in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, Sep. 2023, pp. 5027–5033. doi: 10.1109/ITSC57777.2023.10422594.

[20] C. Yu, L. Zhang, D. Yin, D. Peng, and H. Huang, "Proximal Policy Optimization with Future rewards," *J. Phys. Conf. Ser.*, vol. 2010, no. 1, p. 012085, Sep. 2021, doi: 10.1088/1742-6596/2010/1/012085.
[21] C.-V. Pal and F. Leon, "Brief Survey of Model-Based Reinforcement Learning Techniques," in *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, Oct. 2020, pp. 92–97. doi: 10.1109/ICSTCC50638.2020.9259716.
[22] L. Kaiser *et al.*, "Model-Based Reinforcement Learning for Atari," *arXiv*. Mar. 01, 2019. [Online]. Available: http://arxiv.org/abs/1903.00374
[23] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-Based Reinforcement Learning via Meta-Policy Optimization," in *2nd Conference on Robot Learning (CoRL 2018)*, Sep. 2018. [Online]. Available: http://arxiv.org/abs/1809.05214
[24] S. Y.-C. Chen, "Quantum Deep Q-Learning with Distributed Prioritized Experience Replay," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Sep. 2023, pp. 31–35. doi: 10.1109/QCE57702.2023.10180.
[25] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," *arXiv*. Sep. 09, 2015. [Online]. Available: http://arxiv.org/abs/1509.02971
[26] Z. Zhu, C. Hu, C. Zhu, Y. Zhu, and Y. Sheng, "An Improved Dueling Deep Double-Q Network Based on Prioritized Experience Replay for Path Planning of Unmanned Surface Vehicles," *J. Mar. Sci. Eng.*, vol. 9, no. 11, p. 1267, Nov. 2021, doi: 10.3390/jmse9111267.
[27] S. Nugroho, D. R. I. M. Setiadi, and H. M. M. Islam, "Exploring DQN-Based Reinforcement Learning in Autonomous Highway Navigation Performance Under High-Traffic Conditions," *J. Comput. Theor. Appl.*, vol. 1, no. 3, pp. 274–286, Feb. 2024, doi: 10.62411/jcta.9929.
[28] A. EL Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep Reinforcement Learning framework for Autonomous Driving," *Electron. Imaging*, vol. 29, no. 19, pp. 70–76, Jan. 2017, doi: 10.2352/ISSN.2470-1173.2017.19.AVM-023.
[29] Z. Cao and J. Yun, "Self-Awareness Safety of Deep Reinforcement Learning in Road Traffic Junction Driving," in *arXiv*, Jan. 2022. [Online]. Available: http://arxiv.org/abs/2201.08116